

STOCHASTIC SIMULATION OF PATTERNS
USING DISTANCE-BASED PATTERN MODELING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ENERGY
RESOURCES ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Mehrdad Honarkhah

May 2011

Abstract

The advent of multiple-point geostatistics (MPS) gave rise to the integration of complex subsurface geological structures and features into the model by the concept of training images. Initial algorithms generate geologically realistic realizations by using these training images to obtain conditional probabilities needed in a stochastic simulation framework. More recent pattern-based geostatistical algorithms attempt to improve the accuracy of the training image pattern reproduction. In these approaches, the training image is used to construct a pattern database. Consequently, sequential simulation will be carried out by selecting a pattern from the database and pasting it onto the simulation grid. One of the shortcomings of the present algorithms is the lack of a unifying framework for classifying and modeling the patterns from the training image. In this thesis an entirely different approach will be taken towards geostatistical modeling. A novel, principled and unified technique for pattern analysis and generation that ensures computational efficiency and enables a straightforward incorporation of domain knowledge will be presented.

In the developed methodology (called DisPAT), patterns scanned from the training image are represented as points in a Cartesian space using multi-dimensional scaling. The idea behind this mapping is to use distance functions as a tool for analyzing variability between all the patterns in a training image. These distance functions can be tailored to the application at hand. Next, by significantly reducing the dimensionality of the problem and using kernel space mapping, an improved pattern classification algorithm is obtained. Additionally, a multi-resolution approach is presented for modeling the patterns of the training image at various scales. The proposed distance-based pattern-modeling techniques are inspired by biology and the human visual system. Several examples are presented and a qualitative comparison is made with previous methods to demonstrate the capabilities

of this simple, yet powerful system. We show how the proposed methodology is much less sensitive to the user-provided parameters, and at the same time has the potential to reduce computational time significantly.

Another aspect of present MPS algorithms is their strong dependence on the algorithmic parameters that the practitioner specifies. This, not only entails tedious trial-and-error computations for tuning the parameters, but also triggers potential subjectivity in modeler's decisions. In order to obtain a systematic pattern-based approach, new techniques on learning the optimal set of parameters are introduced. A series of examples is provided to verify the competency of these approaches in, not only facilitating the modeling process, but also ensuring a rigorous simulation framework.

Furthermore, better data conditioning algorithms for both the hard data and the soft data are proposed. An improved pattern continuity and data-conditioning capability is observed in the generated realizations for both continuous and categorical variables. Some improvements in multi-scale data conditioning are also described. Overall, we demonstrate the higher conditioning capability of the proposed method in comparison with the traditional algorithms.

Finally, novel techniques on modeling non-stationary phenomena are introduced. The traditional approaches rely mainly on auxiliary variables to force the MPS algorithm into generating the desired spatial behavior; such as defining regions, constraining the facies proportions, or specifying the rotation/scaling of the features spatially. Rather in this thesis, the original MPS modeling paradigm, where a training image is the sole prerequisite for simulation, is re-established. The proposed framework conceptually embeds the spatial components of the patterns into geostatistical modeling. Various training images are used to demonstrate the capabilities of proposed approaches for modeling non-stationarity.

Acknowledgement

Even though the work done in this thesis may seem to be a work of one person, but in the end, it's the joint effort of all the people who inspired me and supported me in a scientific or non-scientific sense. First and foremost, I would like to single out the most important person, my advisor Prof. Jef Caers, for accepting me in the SCRF research group and giving me the opportunity to flourish and learn different aspects of petroleum engineering. I cannot begin to explain how much this Ph.D. has helped me increase my understanding of a variety of scientific disciplines. When I compare myself with four years ago and all the years before that, I can comfortably state that my knowledge has increased dramatically during this time. Jef provided me with the guidance and support in a way that no other professor could. He not only trusted me with my research, but also challenged me during this process. I am thankful for the fruitful discussions we have had in these years. He also gave me a degree of freedom in my research to explore different techniques and allowed me to become creative more than I could imagine. His guidance was not limited to an intellectual level. He also helped me in other aspects, such as writing and presenting. I can honestly say that all those revisions that I went through during paper submissions have provided me with a tremendous insight on how to write clearly and how to entertain my readers. He has also, unknowingly, improved my presentations skills dramatically. At the end, he is not only a good advisor, but a great mentor and a friend that anybody should be proud to have. I am grateful for all his guidance, challenges, brain stormings, availability, and understanding.

Next, I would like to thank the members of the SCRF research group, especially Prof. Tapan Mukerji, for their help during all the seminars and courses. It has been a great pleasure to have him in my group and take advantage of his ideas and suggestions. If I had to go back and choose another research group, I would have still chosen the same. Our research group has had a variety of research subjects from history-matching, to uncertainty

modeling, to geophysics. This variety allowed me to widen my knowledge in Stanford, and provided me with the ability to think out of the box. Moreover, I would like to thank my colleagues in this research group, Alejandro, Antoine, Amit, Whitney and Celine among many, who have helped me with my research and questioned my methodologies. All their inputs and discussions have given me ideas for my research. SCRF group was also a fun environment to be in. We have had many gathering and outdoor activities during these years. I have become closer to all my friends in this group throughout all these years.

Furthermore, I would like to thank my committee members. Especially I would like to thank Jef, Tapan and Lou for being the readers of this dissertation. Even though it turned out to be a long thesis, but they did not hesitate to provide me with their inputs on how to make it better. I would also like to thank Hamdi for being an evaluator, and asking the tough core questions in my oral defense.

I also would like to thank ConocoPhillips and Chevron for giving me the opportunity to work for them during the summers. I want to thank Alan Rezig, my supervisor in ConocoPhillips, to have given me two internships and helping me get a better understanding of the industry and the challenges within. I would also like to thank Curt Schneider for being the most valuable mentor anybody could have. He has helped me throughout my stay in Houston. I would also like to thank Wen Chen for providing me with the opportunity to join his team in Chevron, and most importantly, I would like to thank Yuguang Chen and Herve Gross, my two mentors in the RPP and RPS teams in Chevron. I am grateful for the meetings, their help, the fruitful discussions, and the tremendous insight they have provided me during my internship.

Most importantly, I would like to thank all my friends and colleagues in the ERE department. They have single-handedly made life in Stanford a memorable experience. Without their support and caring, these four years could not have passed in such a fast pace. All the vacations, gatherings, outdoor activities, and basically being a part of my life is something I can never forget. Especially, I would like to mention Alireza Iranshahr, Mohammad Shahvali and Mohammad Maysami for always being there for me. They helped me through both the bad times and the good times. I am blessed to have such good friends. Moreover, I want to thank Alejandro, Antoine, Guillaome, Cedric, Markus, Matthiew, Amit, Abhishek, Alex, Elnur, Rustem, Mohammad, Ernar, Navid, Taraneh and many others for giving me the support and confidence all these years, and the countless fun moments. It has been a privilege to have this crowd as my life-long friends.

Last but not least, I would like to thank my parents and brothers for believing in me. They have always loved me and encouraged me to become a better and a more successful person. My mother and father were the most influential people in my life. Without their support, I would have not accomplished this. Their unconditional caring has made my life in Stanford, away from them, an easier one. I would like to show my greatest appreciation and love towards them by dedicating them this thesis.

I shot an arrow into the air,
It fell to earth, I knew not where;
For so swiftly it flew, the sight
Could not follow it in its flight.

I breathed a song into the air,
It fell to earth, I knew not where;
For, who has sight so keen and strong
That it can follow the flight of song?

Long, long afterward, in an oak
I found the arrow, still unbroke;
And the song, from beginning to end,
I found again in the heart of a friend.

– *Henry Wadsworth Longfellow*

Contents

Abstract	iv
Acknowledgement	vi
1 Introduction	1
1.1 Why we need Geostatistics?	1
1.2 Two-Point Statistics	3
1.2.1 Kriging	3
1.2.2 Stochastic Simulation	5
1.3 Object-based Algorithms	10
1.4 Multiple-Point Geostatistics	11
1.4.1 Probabilistic-based Approaches	14
1.4.2 Iterative Approaches	16
1.4.3 Pattern-based Approaches	21
1.5 Motivation	26
1.6 Dissertation Outline	40
2 Pattern Modeling with Distances	42
2.1 Notations and Terminology	44
2.1.1 Training image	44
2.1.2 Pattern	47
2.1.3 Dissimilarity distance function	48
2.1.4 Dissimilarity distance Matrix	50
2.1.5 MDS Space	50
2.2 Distance-based Methods Overview	50

2.3	Multi-dimensional scaling	52
2.3.1	Theory	52
2.3.2	Examples	54
2.3.3	Scaling dimension	54
2.4	Pattern Classification	60
2.4.1	k -means	63
2.4.2	Number of Clusters	65
2.5	Kernel Methods	67
2.5.1	The Overall Picture	67
2.5.2	Properties of kernels	69
2.5.3	Kernel k -means	72
2.6	Conclusion	77
3	Unconditional MPS Simulation	80
3.1	Simulation Methodology	81
3.1.1	Processing of the Training Image	81
3.1.2	Classification of Pattern Database	83
3.1.3	Sequential Simulation	85
3.1.4	Distance Function	88
3.1.5	Pattern-Skipping concept	91
3.2	Comparison	96
3.2.1	Clustering Accuracy	96
3.2.2	MPS Simulation	102
3.3	Study of Model Variability	107
3.3.1	Concepts	107
3.3.2	JS-MPH Methodology with Example	109
3.4	Conclusion	115
4	Multi-Scale MPS Simulation	117
4.1	Multi-Grid Approach	118
4.1.1	Previous Work	118
4.1.2	Coarse-grid Rescaling	122
4.2	Multi-Resolution Approach	126
4.2.1	Resolution Theory	127

4.2.2	Simulation Algorithm	128
4.2.3	Examples	133
4.3	DisPAT Simulation Examples	136
4.4	Simulation Time Comparison	155
4.5	Conclusion	157
5	Parameter-free Learning	158
5.1	Its Perils	158
5.2	Template Size	160
5.2.1	Concept	161
5.2.2	Methodology	162
5.2.3	Empirical Evaluations	164
5.3	Clustering	171
5.3.1	Concept	171
5.3.2	Methodology	172
5.4	Empirical Evaluation	177
5.5	Conclusion	180
6	Data Integration	182
6.1	Hard Data Conditioning	183
6.1.1	Method	183
6.1.2	Examples	185
6.1.3	Multi-Grid Approach	191
6.1.4	Multi-Resolution Approach	204
6.2	Soft Data Conditioning	210
6.2.1	Review of Methods	210
6.2.2	Soft Distance-based Fusion	214
6.2.3	Note on Distance-Fusion	216
6.2.4	Examples and Sensitivity	223
6.3	Conclusion	226
7	Non-Stationarity	232
7.1	Overview	234
7.2	Note on Non-stationary Modeling	241

7.3	Spatial-Similarity Method (SSM)	246
7.4	Neighborhood-Radius Method (NRM)	252
7.5	Automatic-Segmentation Method (ASM)	268
7.5.1	Gabor Filters	272
7.5.2	Methodology	276
7.6	Examples	291
7.7	Conclusion	302
8	Conclusion	305
8.1	Summary of the Study	306
8.2	Future Work	310
A	Algorithmic Enhancements	329
A.1	<i>SEQ</i> -MDS Algorithm	329
A.1.1	Methodology	330
A.1.2	Condition for Accuracy	332
A.1.3	Computational Complexity	334
A.1.4	Experimentation	335
A.2	H-Ann k -means	340

List of Tables

3.1	memory requirement to store the distance matrix for different training image, with template sizes of 11×11 in 2D, and $11 \times 11 \times 5$ in 3D	93
3.2	FilterSim comparison with DisPAT in terms of sharpness index	104
4.1	Simulation parameters	140
4.2	MPH errors	151
4.3	Jensen-Shannon divergence between the multiple-point histogram of the training image with respect to each realization.	155
4.4	Time comparison for generating 100 unconditional simulation with different MPS methodologies using a two-dimensional training image (Figure 4.24(a)).	156
4.5	Time comparison for generating 100 unconditional simulation with different MPS methodologies using a three dimensional training image (Figure 4.24(b)).	156
8.1	Example of textural properties used to find the optimal template size.	317

List of Figures

1.1	An unsampled location is shown with a lag distance \mathbf{h} apart from the measured variable. The variogram defines a measure of correlation between the unknown variables with respect to the lag distance from measure ones. There variogram defined by the vector \mathbf{h} is specific to a certain spatial direction (such as azimuth angle).	4
1.2	An illustration of the kriging result, producing the best linear estimation of the variable conditioned to the available hard data, and three different realizations capturing the spatial variability with the same variogram model.	6
1.3	(Left) the two points configuration used to infer statistics in traditional variogram-based approaches, (Right) a sample of a 5-point configuration used in multiple-point geostatistics to model highly complex geological phenomena and curvilinear sedimentary features.	12
1.4	The unilateral path used in Markov mesh models for sequential simulation, where the assumption of dependence of the conditional probability of a value on its sequential neighboring cells holds.	20
1.5	Exceptional ability of the brain to decipher low-resolution images. Not only it can recognize the existence of human faces, it can also recognize those people; Bill Clinton and Hillary Clinton.	28
1.6	A conceptual training image depicting channel structures in the subsurface.	30
1.7	A representation of different steps an sketch artist would draw an eye; from a coarse layout towards adding increasingly more details.	34
2.1	Training image examples for a 2 dimensional or 3 dimensional simulations, depicting different channel systems.	46

2.2	(Left) Training image, ti , and a template \mathbf{T} on location \mathbf{u} , (Right) the pattern, $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ with template \mathbf{T}	48
2.3	(a) Training Image and (b) MDS sample result using an 8×8 pattern template. Any two patterns that are similar map as two close points in Euclidean space.	55
2.4	Eigenvalues obtained from mapping to MDS space. A dimensionality reduction to 9, seen by the elbow of this figure, can reasonably capture the complexity between the patterns.	56
2.5	(a) The correlation coefficient between the original distance matrix and the one constructed by the lower-dimensional MDS configuration is plotted with respect to the dimension of the MDS space, and (b) shows the scatter plot for the dimensionality reduction from 81 to 15, where a high correlation of 99.203% is obtained.	57
2.6	MDS dimensionality selection. (a) The scree plot, (b) the elbow of the scree plot is easily observed in the semi-log scale, (c) the profile log-likelihood of the scree plot showing 15 as the maximum.	61
2.7	Clustering illustration on three different configurations of points. Colors of black or gray represent two different clusters within the data.	62
2.8	Clustering using k -means where the cluster centers, shown in a cross-circle shape, minimize sum-squared distances.	64
2.9	Silhouette and Calinski-Harabasz (CH) indexes; indicating 23 as the best number of clusters	66
2.10	Kernel transformation, and the principal component in the linear kernel space	69
2.11	Clustering results using 10 clusters (shown by projection into $2d$ space) using the training image given in Figure 2.3. K -means clustering without applying kernel transformation (top) is compared with kernel k -means clustering (bottom).	73
2.12	(a) kernel k -means speed analysis, (b) kernel k -means speed shown in log-log scale, showing the computational complexity of the algorithm, $O(N^3)$	76
2.13	Comparison of kernel k -means with the proposed methodology of using k -means as an initializer for kernel k -means centroids.	77

3.1	A training image is scanned with a 3×3 template size (left), and all the patterns are stored in the pattern database (right).	82
3.2	MDS representation of patterns from the training image shown in Figure 3.1. As can be observed, the patterns that are similar to each other are positioned closely in this 2-dimensional MDS space.	84
3.3	A realization \mathbf{re} is shown, with a template \mathbf{T} of 3×3 used to extract the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$	86
3.4	A template \mathbf{T} of 5×5 (red) used to extract the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ or paste a pattern, and the inner patch of 3×3 (blue) used for freezing the nodes around location \mathbf{u} after pasting a pattern in order to remove those nodes from random path	87
3.5	A pattern (left), and its distance transform (middle), and the proposed proximity distance transform (right) is shown.	90
3.6	(a) distance matrix obtained using distance transform function, and (b) distance matrix obtained using proximity distance transform function. In this figure, pattern at left location is \mathbf{pat}_1 , pattern at middle location is \mathbf{pat}_2 , and pattern at right location is \mathbf{pat}_3	92
3.7	Showing training image of 9×9 , scanned with a 3×3 template resulting in all possible patterns. Concept of pattern skipping is shown for skip-sizes of 1 (None), 2 and 3. (picture after Arpat (2005))	94
3.8	Training Image used in this study having 101×101 dimensions with 9×9 template.	97
3.9	(a) MDS space of patterns, (b) 3D feature space projection using kernel PCA.	99
3.10	Kernel k -means clustering of 25 clusters in feature space.	100
3.11	Cluster prototypes for all 25 clusters, more sharpness means better clustering	101
3.12	51×51 Training Image used for comparison of DisPAT and Filtersim with a 9×9 template	102
3.13	Comparison of clustering algorithms using (a) filter-based analysis in filtersim, and (b) distance-based analysis in DisPAT.	103
3.14	51×51 Training Image used with DisPAT algorithm to generate single-grid unconditional realization with two different template sizes of 5×5 and 9×9 . The E-types of obtained from each template size over 1000 realizations is also shown next to the realizations.	105

3.15	51 × 51 Training Image used with filtersim algorithm to generate single-grid unconditional realization with two different template sizes of 5 × 5 and 9 × 9. The E-types of obtained from each template size over 100 realizations is also shown next to the realizations.	106
3.16	Comparison of the variability between the 100 realizations generated by filtersim (red) and the proposed method (black) by the first three principal coordinates of the space of uncertainty. Two dimensional views of the data are also plotted for clarity.	112
3.17	Space of variability is shown in parallel coordinates with 20 first dimensions in (a), and the 90% and 10% quantiles are shown in (b). Red lines represent the proposed method’s realizations variability and gray lines the filtersim.	113
3.18	The error between the realizations generated by the proposed method (yellow) and filtersim method (black) with respect to the training image in terms of pattern reproductivity.	115
4.1	The channel training image of size 101 × 101 is shown. One realization using the single-grid (middle), and one using multi-grid (bottom) is illustrated, where the differences in long-range connectivity is demonstrated.	119
4.2	Multigrid concept is illustrated with three levels of multiple-grid. The realization is of size 15 × 15 and the template is 3 × 3. In multi-grid 2, the visited nodes are expanded in spacings of 2, and in multi-grid 3, they are expanded in multiples of 4.	120
4.3	An example application of multi-grid concept with three levels.	123
4.4	Illustration of ”coarse-grid rescaling” concept to fill the uninformed nodes, performed only at the end of multi-grid level 2. It shows the simple interpolation scheme used for rescaling the grid G_2	124
4.5	Coarse-grid rescaling method is illustrated for channelized training image. (a) depicts the MPS methodology without coarse-grid rescaling, and (b) shows the same results with coarse-grid rescaling applied. The final realizations are shown and yellow circles are used to represent the differences between the two simulations. Clearly, the pattern connectivity in the bottom realization has improved.	125

4.6	Multi-resolution concept is illustrated with $n_r = 3$ resolutions. The original grid \mathbb{G}^1 is of size 15×15 . The second and third multi-resolution grids, \mathbb{G}^2 and \mathbb{G}^3 , are of sizes 8×8 and 5×5 dimensions, respectively. The template \mathbf{T} remains the same (3×3) in all resolutions.	129
4.7	A sample illustration for 2D of 16 neighboring nodes around location \mathbf{u} , inclusive.	131
4.8	Sample experiment with multi-resolution approach where the chessboard training image consists of only fine-scale features. Three resolutions of $r = 1, 2, 3$ are shown with their thresholded results.	134
4.9	Multi-resolution training images used for each stage of the coarse to fine-scale simulations.	135
4.10	Illustration of multi-resolution approach on channelized training image in three levels.	137
4.11	Comparison of multiple-resolution simulation with multiple-grid simulation over three generated realizations.	138
4.12	Multiple-pint histogram error distribution between 200 realizations and the training image. Red distribution illustrates multiple-grid simulation errors and the black distribution for multiple-resolution case. In both distributions the proposed method is used for the simulation. The yellow distribution shows the corresponding filtersim error distributions with a multiple-grid setting.	139
4.13	Comparison of the filtersim method with the proposed methodology for the training image shown above. Different template sizes of 9×9 and 13×13 are illustrated for comparison.	141
4.14	Simulated realizations using both the (a,c) new proposed methodology and (b,d) filtersim. There is a clear superiority in terms of spatial pattern reproductivity in the proposed method by using the same set of parameters. Two different realizations have been generated.	143
4.15	Filtersim resulting realizations with increasing number of clusters, k , in k -means clustering.	144
4.16	Simulated realizations with search template of 13×13 and inner patch of 7×7 . (a) For the proposed method with 100 clusters, (b,c) for Filtersim method with 100 and 400 clusters respectively.	145

4.17	Comparison of the filtersim method with the proposed methodology for a quasi-stationary meandering training image. Three realizations are shown for filtersim (top) and the proposed method (down).	146
4.18	Comparison of filtersim with the proposed methodology for a categorical training image. Template size of 11×11 and an inner patch of 7×7 are used for simulation. Three realizations are shown for filtersim (top) and the proposed method (down).	148
4.19	Unconditional filtersim results using the previous categorical training image but without any histogram matching. Template size of 11×11 and an inner patch of 7×7 are used for simulation. These are the realizations that should be compared with the ones generated using the proposed method.	149
4.20	Continuous training image (159×159) shown on top is chosen for comparison. Three different filtersim realizations (middle) and three different realizations using the proposed method (bottom) are shown.	150
4.21	3D unconditional simulation on a $69 \times 69 \times 39$ training image shown in top. Filtersim realization (on left) and the proposed method's realization (on right) are illustrated for comparison.	152
4.22	3 slices of the training image (on top) and two realizations, one from filtersim (on left) and the other from the proposed method (on right), are shown on horizontal planes at locations 7, 20 and 39.	153
4.23	Multiple-point histogram (MPH) of the models, considering the multiple-point template of size $3 \times 3 \times 2$. x -axis is labeled according to the index related to the multiple-point configurations, and y -axis represents the number of observed instances of each configuration.	154
4.24	The two training images used for computation time comparison of different MPS techniques.	156
5.1	The application of automatic template selection on a square-filled training image, showing mean entropy curve (lower left) and profile log-likelihood of that curve (lower right), indicating the best template to be of size 5×5 . . .	165
5.2	The training image and the original feature used in its construction are shown on top. Six different realizations using different template sizes are shown in the middle and lower parts of the figure.	167

5.3	The application of automatic template selection on a 2D training image consisting of circles (a). Mean entropy curve (b-left) and profile log-likelihood of that curve (b-right) indicate best template size of 17×17	168
5.4	The application of automatic template selection on a 2D channelized training image. The figure shows the training image (left), the mean entropy curve (middle) and the profile log-likelihood (right). The best template dimension of 13×13 has been obtained.	169
5.5	The application of automatic template selection on a non-stationary training image and the comparison with a stationary training image. The plots represent the mean entropy curve (left) and the second derivative of entropy curve (middle) and the variance of the entropy values (right) for both stationary and non-stationary cases. The plots demonstrate the non-existence of an appropriate template size for non-stationary training image.	170
5.6	Eight sample data points shown in \mathbb{R}^2 , where two clear cluster of data is noticeable.	173
5.7	Kernel matrix is shown where each element is colored according to its value. Left figure shows the original kernel matrix, and right figure show the one where rows and columns are permuted. A clear structure between the data, indicating two clusters, is evident in the right figure.	174
5.8	Computational speed of the kernel k -means algorithm with respect to the number of clusters. (a) showing the normal plot, (b) showing the smoothed version of the semi-log plot with four distinct regions, (c) the Log-Log plot showing different speed behaviors in each region.	178
5.9	Plot of $\lambda_i \{ \mathbf{1}_N^T \mathbf{u}_i \}^2$, and observing the contribution of each of these terms to the overall value. The results are interpreted at different scales of normal or logarithmic (considering the importance of the accuracy obtained with clustering).	179
5.10	Example application of finding the optimal number of clusters. Top figure is the plot of $\lambda_i \{ \mathbf{1}_N^T \mathbf{u}_i \}^2$, middle figure is the logarithm of the top figure. The bottom-left plot is the smooth version, and the bottom-right is the application of profile log-likelihood on finding the elbow.	181

6.1	Example of a hard data grid \mathbf{hd} , with two hard data one with a value of zero (indicating shale), and the other with one (indicating sand).	184
6.2	The hard data conditioning locations are shown in top. The ensemble average (E-Type) of 100 realization generated using the proposed method (left), the sgsim method (middle) and filtersim method (right) are shown. The multiple-point structures are better captured in the proposed methodology as seen in the ensemble averages. Two different MPS realizations of each method is also shown on the bottom.	187
6.3	The hard data conditioning locations are shown in top. The ensemble average (E-Type) of 100 realizations generated using the proposed method (left-bottom) and filtersim method (right-bottom) are shown. The reference case is also shown on left.	188
6.4	The neighboring hard data conditioning locations are shown in top. The ensemble average (E-Type) of 100 realizations generated using the proposed method (left), snesim method (middle) and filtersim method (right) are shown in the bottom.	189
6.5	The average of all 35×35 patterns in the training image that honor the neighboring hard data at their central nodes is shown on top-right corner. The zoomed-in version of the ensemble average (E-Type) of different methods is shown for comparison on the bottom.	190
6.6	An example on how data relocation approach will move the hard data \mathbf{u}_{hd} from the hard data grid \mathbf{hd} to the nodes on the multi-grid realization \mathbf{re}_g	191
6.7	An illustration of the first limitation in current data relocation practice, where well data in proximity to each other can be dropped and eliminated in the process.	193
6.8	An illustration of the second limitation in current data relocation practice, where a clear spatial deformity is observed in the E-type obtained through conditional snesim simulation. Rejection sampling on left shows the correct E-type that should be obtained with the algorithm.	195
6.9	Example indicating that well w_2 is better to be assigned to \mathbf{u}_2 rather than being dropped. Previous implementations would drop w_2 since w_1 was closer to \mathbf{u}_1	196

6.10	Example indicating that well w_2 should not anymore be copied on node \mathbf{u}_2 , and instead be copied over \mathbf{u}_4 due to a third well w_3 being a better candidate for relocation on node \mathbf{u}_2	197
6.11	A complicated example configuration of wells and multi-grid nodes is shown. Different rules can be elaborated by such an example.	198
6.12	Investigation of the new data relocation approach. The E-type of rejection sampling method is shown in the middle as the truth. The two lower E-types from the DisPAT, and the snesim algorithms demonstrate the improved conditioning capability obtained by the proposed data relocation approach. Simulations are done in three multi-grid levels.	202
6.13	The vibrating multi-grid shifting approach is illustrated for $n_g = 2$ multi-grid setting. There are four possible shifts in the origin, thus introducing stochasticity within hard data relocation.	204
6.14	An illustration of the smart-vibrating approach, where no spatial differences is observed in the E-type obtained through conditional DisPAT simulation and rejection sampling.	205
6.15	Data relocation by inverse-distance weighting approach for multi-resolution simulation. For node \mathbf{u}_5 , all the wells within the distance less than the size of a coarse grid-cell are included in analysis, that is wells w_1 , w_2 , and w_3 . The red lines shows the distances used for interpolation of node \mathbf{u}_5 . The gray lines for each well w_i show all nodes that w_i will be used for their interpolation.	207
6.16	Example application of data conditioning in multi-resolution simulation using inverse-distance weighting. There are 100 hard well data of sand/shale in the 2D grid size of 101×101 . Both multi-grid (bottom-left) and multi-resolution (bottom-right) E-types of DisPAT algorithm are shown. Better conditioning is obtained in multi-resolution approach.	209
6.17	Flowchart of the distance-fusion technique for integration of soft information. Data event represents the training image and the hard data. Soft data event represents the soft data only. The result is a analogous to the naive Bayesian classifier, where the optimal cluster prototype is selected.	218
6.18	An example Bayesian network showing the Bayesian classifier, where the assumption of conditional independence is enforced. The root node, \mathbf{A} , is connected to all other nodes (additional information).	219

6.19	The channel training image, used for application of soft data integration, and the reference case are shown. Four soft data are generated by applying an averaging filter with windows sizes of 2×2 , 3×3 , 4×4 , and 5×5 , indicating the resolution of seismic data.	224
6.20	Sensitivity of soft data integration approach on template sizes T . Two sample realizations are shown for each template size of 7×7 and 9×9 . The E-types obtained from 150 realizations are also shown for each template. Smaller template of 7×7 can provide more accurate data integration.	225
6.21	Two realizations are shown for each of the four soft data used used in a multi-grid simulation. The illustrated soft data differ with respect to only their resolutions.	227
6.22	The E-types obtained from 150 Earth models generated using different soft data resolutions; indicating the increase in uncertainty by a decrease in resolution. Medium confidence interval is chosen in this analysis.	228
6.23	The resulting E-types with different confidence levels of weak, medium, and strong over soft data are illustrated. Confidence levels are characterized by setting $\omega_{sd} = 0.15$, 0.35 , and 0.60 respectively. It is shown that higher confidence can lead to less uncertainty in the E-type.	229
6.24	The application of soft data integration in a multi-resolution setting is depicted. The effectiveness of methodology is observed by observing the generated E-types for each confidence level of medium and strong ($\omega_{sd} = 0.35$, and 0.60 respectively).	230
7.1	(a) Training image representing a fluvial deltaic fan reservoir, (b) one sample realization generated with MPS technique of DisPAT without accounting for non-stationarity of the training image.	233
7.2	Example application of rotation and affinity concept for non-stationary analysis. The channel training image is used to construct a search tree in snesim. Different location-specific rotation constraints or scale constraints can be applied on data event to generate non-stationary realizations (models from Zhang (2002)).	235

7.3	Example application of non-stationary modeling in simpat MPS methodology by defining 10 regions according to the morphological characteristics of rotation and scale (models from Arpat (2005)).	238
7.4	Example application of constructing five partition classes for the fracture training image that consists of three different fracture networks with three orientations. The partition classes show spatially scattered regions (models from Boucher (2007)).	240
7.5	A non-stationary training image is shown that consists of two stationary processes of E-W and N-S channels. The subregions on the other hand should define four regions in traditional approaches.	244
7.6	An example non-stationary fluvial fan training image with a realization grid with different dimensions is shown. Three different non-stationary simulation results are depicted emphasizing that the algorithm is unable to understand the modeler’s perspective.	247
7.7	The convexity or smoothness of the spatial weighting function used to integrate spatial components into non-stationarity modeling. Patterns similarities change with respect to their spatial proximity with the node being simulated.	249
7.8	Example application of spatial-similarity method (SSM) is provided. A training image and a corresponding node location \mathbf{u} on the realization grid, that is to be simulated, are shown. The two distances for pattern similarity and location similarity are visually illustrated on the original grid \mathbf{G}_{ti} . The most optimal pattern, to be pasted on the realization, is obtained from the minimum of \mathbf{d}_{ns}	251
7.9	Application of Spatial-Similarity Method (SSM) on fluvial fan-deposit training image of size 199×199 . Template size is 15×15 with a multi-grid level of 3. The stationarity level is set to $\omega_{SSM} = 0.75$. Four realizations are shown to demonstrated non-stationarity and stochasticity of the proposed algorithm.	253
7.10	Sensitivity analysis of the non-stationarity level ω_{SSM} on the realizations generated by the application of Spatial-Similarity Method (SSM) on fluvial fan-deposit training image of size 199×199 . Weight of zero indicates stationarity, weight of 0.5 provides the desired results, and weight of one forces the exact reproduction of the training image.	254

7.11	Sensitivity analysis of the E-types with respect to the non-stationary weight ω_{SSM} . The training image is the same fluvial fan-deposit shown in Figure 7.10. Higher weights lead to more non-stationarity in the resulting models. A value of zero provides no structural information due to stationary simulation, and the weight of one exactly reproduces the training image due to exclusion of pattern similarity searches.	255
7.12	The concept of neighborhood-radius defined by $\omega_{NRM} \times r_T$ used in the indicator functional of $\mathbf{I}_{\omega_{NRM}}(\mathbf{u})$. Two different spatial weights of 1 and 0.7 are illustrated by their corresponding areas around location \mathbf{u} . Only this neighborhood area will be considered in the NRM method for modeling non-stationarity.	260
7.13	The non-smooth indicator functional of neighborhood-radius method for integration of spatial components into non-stationarity modeling. Patterns similarities are included for those in the local proximity of the visited node location \mathbf{u}	260
7.14	Example application of neighborhood-radius method (NRM) is provided. A training image and a corresponding node location \mathbf{u} on the realization grid, that is to be simulated, are shown. The distance for pattern similarity and indicator function $\mathbf{I}_{\omega}(\mathbf{u})$ for location similarity are visually illustrated on the original grid \mathbf{G}_{ti} . The most optimal pattern, to be pasted on the realization, is obtained from the minimum of \mathbf{d}_{ns}	262
7.15	Application of Neighborhood-Radius Method (NRM) on fluvial fan-deposit training image of size 199×199 . Template size is 13×13 with a multi-grid level of 3. The stationarity level is set to $\omega_{NRM} = 1.5$. Four realizations are shown to demonstrate the non-stationarity and stochasticity of the proposed algorithm.	264
7.16	Sensitivity analysis of the non-stationarity level ω_{NRM} on the realizations generated by the application of Neighborhood-Radius Method (NRM) on fluvial fan-deposit training image of size 199×199 . Small weights indicate a strict non-stationarity of the training image, i.e., no locally noticeable stationary regions. As the weight increases, the stationarity assumption is reinforced. A large value of 30 is shown to produce stationary results.	265

7.17	Sensitivity analysis of the E-types with respect to the non-stationary weight ω_{NRM} . The training image is the same fluvial fan-deposit shown in Figure 7.16. A template size of 13×13 is chosen for simulation. Lower weights lead to more non-stationarity in the resulting models. A very high value of 40 demonstrates uncertainty in the location of channels due to stationary simulation, and the weight of 0.2 exactly reproduces the training image due to extremely small neighborhood-radius.	266
7.18	A non-stationary training image with three distinct features. Due to the scale differences between the features, there is no single pattern size that can be used to accurately segment/partition this training image.	270
7.19	An elementary Gabor function obtained by multiplication of a Gaussian envelope, $\exp(-\frac{1}{2} t^2)$, with a sinusoidal function, $\cos(8t)$	273
7.20	The real component of a 2D Gabor function, with parameters $\sigma_x^2 = \sigma_y^2 = 20$, $\theta = \frac{\pi}{6}$, and $f = 1$, plotted for $x, y \in [-6, +6]$	274
7.21	A bank of real-valued even-symmetric Gabor filters is shown. Eight different orientations with angular separation of 22.5° are shown horizontally, from left to right. Furthermore, nine different frequency values are shown in a descending order from top to bottom. This set of Gabor filters will be applied individually on the training image to accurately analyze its various features.	277
7.22	Application of four Gabor filters with orientation angles of $\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$, and a constant small frequency of $f = 0.17$. Each filter is distinguishing a different set of features in the training image.	279
7.23	Application of four Gabor filters with orientation angles of $\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$, and a constant small frequency of $f = 0.33$. Each filter is distinguishing a different set of features in the training image, but it is more focused on small-scale details and is good for region-boundary identifications.	280
7.24	The sigmoidal function, $\tanh(0.25t)$, is illustrated, which, upon application, will make all values positive.	281
7.25	Application of energy extraction on the filtered images shown in Figure 7.22. It involves a non-linear transformation with sigmoidal function, and a smoothing filter with a Gaussian function that has three times bigger spread than the scale of its corresponding Gabor filter.	283

7.26	Application of energy extraction on the filtered images shown in Figure 7.23. It involves a non-linear transformation with sigmoidal function, and a smoothing filter with a Gaussian function that has three times bigger spread than the scale of its corresponding Gabor filter.	284
7.27	The spatial component images for a two-dimensional training image of size 199×199 is shown. The values are normalized to $[0, 1]$ range.	285
7.28	All 88 features obtained from the non-stationary training image shown in Figure 7.1(a). The final row depicts the spatial components that were embedded to ensure adjacency of the regions (each being repeated four times due to the employed weighting scheme).	286
7.29	Flowchart of Automatic training image segmentation using a bank of Gabor filters, with the corresponding non-linearity filters and the Gaussian smoothing. The k -means clustering algorithm is used for defining the stationary regions.	287
7.30	Application of segmenting a non-stationary training image. Each run of k -means clustering algorithm might lead to different subregions, due to the random initialization. It is a desirable characteristic due to the additional stochasticity.	288
7.31	Application of Automatic-Segmentation Method (ASM) on fluvial fan-deposit training image of size 199×199 is investigated with a template size is 15×15 with a multi-grid level of three. There are six stationary regions defined for this training image. Four realizations are shown to demonstrate the non-stationarity and stochasticity of the proposed algorithm.	290
7.32	Investigation on the effect of the number of stationary regions on the generated realizations obtained by the ASM method. As can be seen, higher numbers indicate stricter non-stationarity in the results. Accordingly, choosing only one stationary region leads to stationary simulations, and hence, stationary realizations.	292

7.33	Investigation on the effect of the number of stationary regions on the generated realizations obtained by the ASM method. As can be seen, higher numbers indicate stricter non-stationarity in the results. A large number of 150 stationary realizations results in an E-type similar to the training image. Accordingly, choosing only one stationary region leads to stationary simulations, and hence, stationary realizations.	293
7.34	Application of the SSM method for non-stationary modeling of a training image consisting of elongated/rotated ellipses. Four realizations are shown by a template size of 15×15 , and a non-stationary weight of 0.5.	295
7.35	Application of the NRM method for non-stationary modeling of a training image constructed by truncating a Gaussian field according to the proportions between zero and one. Four realizations are shown by a template size of 15×15 , and a non-stationary weight factor of 3.	296
7.36	Application of the SSM method for non-stationary modeling of a fractured network. Four realizations are shown by a template size of 15×15 , and a non-stationary weight factor of 0.5.	297
7.37	Application of the ASM method for non-stationary modeling of a fractured network. Three subregions are chosen for this training image, and the segmentation result is within expectation. Four realizations are shown, where a clear stochasticity is evident in every network. Furthermore, the patterns on the region boundaries conform to the two non-stationarities on each side. .	299
7.38	Application of the ASM method for non-stationary modeling. The training image consists of two visually distinct stationary regions. A correct segmentation is obtained by the application of Gabor filters on this training image. Four realizations are shown by a template size of 15×15	300
7.39	Application of the SSM method for non-stationary modeling of a complex tidal-dominated training image of size 447×173 . The SSM method can handle any form of non-stationarity. Six realizations are shown by a template size of 15×15	301
7.40	Application of the ASM method for non-stationary modeling of a complex tidal-dominated training image of size 447×173 . The training image is divided into five stationary regions for this simulation. Six different realizations are illustrated, where the desired spatial variability of the categories is evident.	303

8.1	The application of new pattern mapping (data event) to MDS space using a subset of patterns by the proposed <i>SEQ</i> -MDS method. The patterns from the database are shown on the left, and the 12-dimensional MDS coordinates are shown on the right. Gray lines represent the entire database, black line represents the true coordinates, and red line represents the approximate reconstructed coordinate.	323
8.2	The application of incomplete pattern mapping to MDS space using a subset of patterns. Uninformed nodes are represented by yellow, and the closest pattern to the obtained MDS point is shown.	324
A.1	Simple procedure of <i>SEQ</i> -MDS method with four similar red points as anchors.	333
A.2	mapping 2209 patterns with the dimensionality of 9×9 , to a 12 dimensional MDS space, using (a) classical MDS and (b) <i>SEQ</i> -MDS algorithms. This mapping shows the similarity in the layout of MDS produced by the sequential MDS method to the classical MDS layout.	338
A.3	Speed analysis of CMDS and <i>SEQ</i> -MDS with respect to the number of sample points. The dimensionality is reduced from 81 to 12. (a) the semi-log behavior, (b) the log-log behavior, indicating the time complexity of both methods. The mapping is performed using a PC with CPU $2.66GHz$ and $2GB$ of memory.	339
A.4	(a) Speed comparison of Matlab k -means algorithm and Ann k -means algorithm, showing clear improvement of Ann k -means method, (b) Log-Log scale of the speed improvement showing the computational complexity of both methods being $O(N^{1.7})$ and $O(N^{1.1})$ for Matlab and Ann k -means respectively.	341
A.5	(a) 100000 data points used for clustering comparison, (b) Speed results for the two different methods of Ann k -means and H-Ann k -means, showing the improvement with the hierarchical approach.	343

Chapter 1

Introduction

The eye sees only what the mind is prepared to comprehend

HENRI BERGSON (1859 - 1941)

1.1 Why we need Geostatistics?

There is a tremendous amount of variability in space, or in space and time, when dealing with modeling spatio-temporal phenomena in the Earth sciences. Geostatistics provides the tools to quantify this variability. It gives the practitioners the ability to predict the behavior of the system and make informative decisions under the constraints of limited knowledge, resources, and time. Some example applications of Geostatistics are:

- Ore grades evaluation in mineral deposits (such as gold, copper, etc.).
- Pressure and temperature interpolation in atmospheric sciences.
- Permeability or porosity characterization in oil reservoirs.

These geological phenomena follow a complex set of physical rules that cannot be adequately described by simplified models, such as a constant value layer cake model. The fallacy of such simplistic models is more pronounced in petroleum industry. The governing flow equations require reservoir properties such as porosity and permeability for the prediction of oil production and maximizing recovery using various drive mechanisms. In such

situations, there is limited amount of data available to model the system with complete certainty. For example, the porosity and permeability are only obtained at sparse drilled well locations. However, to simulate the flow behavior, the governing physical equations need these reservoir properties at every spatial location. But, due to economic reasons, one cannot afford to obtain all the information necessary for an accurate deterministic modeling of the subsurface.

In all of the Earth sciences, one needs to predict the systems' behavior and make business decisions based on a variety of information at hand; such as geology, geophysics, well testing, etc. The fundamental aspect existing in all these Earth sciences is the notion of *spatial uncertainty*. In earlier years, engineers were trained to think deterministically, especially in oil industry where the managers are expected to get one estimate for oil recovery, and the presence of uncertainty in the results was a sign of unskillfulness. However, uncertainty inherent to the modeling of the geological subsurface is becoming a more acceptable practice.

Geostatistics aids in producing many plausible Earth models (also called stochastic realizations) that can capture this so called spatial uncertainty. By integrating different sources of information such as well logs, cores, geologic interpretations, or seismic data, a more realistic representation of the field under study can be obtained, as well as a more realistic representation of this type of uncertainty. However, in any engineering project, we are interested in the spatial distributions of the data, or particularly, the response of the model to an engineering design. Each flow simulation will provide us with what is called a response function. The histogram of these response functions provides a probabilistic description of the desired variable; i.e., how much oil one can recover from a particular reservoir given a certain production scenario. Since there are several alternative models, the histogram of these response variables provides us with what we are interested in, that is to say, the recovery of that particular field.

Matheron (1971) introduced the notion of *regionalized variable*, $Z(\mathbf{u})$, as the value at location \mathbf{u} of the characteristic Z of the geological phenomenon. The term regionalized refers to that fact that the variable is spread in space and exhibits a certain spatial structure. If the regionalized notion is ignored, the variables can be randomly positioned on the reference area, and hence, would not exhibit any spatial continuity. However, physical processes do not behave randomly in space, and there is spatial continuity in their attributes. For example, a channel that is shaped by a river flowing down the hill is governed by a complex physical phenomenon that forces continuity on the path of the channel, or mineral deposits

tend to concentrate in certain spatial locations. Therefore, the regionalized aspect of the variable is the main principle in geostatistics that makes it different from pure statistics. If there was no spatial continuity, the prediction would not be possible. However, the physical laws determining this spatial continuity cannot be deterministically modeled because of lack of information. To quantify spatial uncertainty, one needs a mathematical model that can describe the spatial structure of variables under study, which essentially, reflects the physics behind the deposition that created the subsurface variability. For example, the existence of spatial trends in the data, isotropic or anisotropic characteristics of the variables, the scales of continuity are some of the questions that can be answered in geostatistics by defining a model of spatial variability.

Different geostatistical techniques rely mainly on their models of spatial continuity and spatial uncertainty, and how they integrate direct data, such as cores, or indirect data, such as seismic, to generate stochastic realizations of the subsurface. In the next sections, an overview of geostatistical methodologies and their modeling paradigms will be provided.

1.2 Two-Point Statistics

1.2.1 Kriging

One of the first methods for estimating an unknown variable at an unsampled location is kriging (Krige, 1951). Kriging is a linear least-square estimation technique that relies on the interpolation of a variable from the sparsely available hard data. The method was initially applied to mineral deposits by assuming a continuous mineralization between measured mineral grades.

Kriging applies a linear combination of measured data one at a time, without any assumption on the type of their statistical distribution, to obtain the values at unsampled locations. The weights, used for these linear combinations, were determined in kriging so as to minimize the estimation variance in the least-square sense. This can be formulated as minimizing the squared error as follows,

$$SE = [z(\mathbf{u}) - z^*(\mathbf{u})]^2 \quad (1.1)$$

where Z^* denotes the estimator of the random variable Z at location \mathbf{u} . The linear estimator in kriging is computed accordingly,

$$z^*(\mathbf{u}) - m(\mathbf{u}) = \sum_{\alpha=1}^n \lambda_{\alpha} \cdot [z(\mathbf{u}_{\alpha}) - m(\mathbf{u}_{\alpha})] \quad (1.2)$$

where $m(\mathbf{u})$ is the mean value at the unsampled location \mathbf{u} , and λ_{α} , $\alpha = 1, \dots, n$ are the linear weights applied to the n measured variables.

The structural function characterizing the spatial variability of the variables or the physics of the geology in linear geostatistics is called the variogram. A variogram is the linear correlation between any two points in space. In other words, it provides the statistical correlation between two points as their separation increases in space. It is the simplest method to relate a variable at an unsampled location from its near observations. In probabilistic notation, under stationarity, a variogram is defined:

$$2\gamma(\mathbf{h}) = E \left\{ [Z(\mathbf{u}) - Z(\mathbf{u} + \mathbf{h})]^2 \right\} \quad (1.3)$$

A simple variogram is depicted in Figure 1.1. It should be mentioned that each variogram is defined for a specific spatial direction through the vector \mathbf{h} of lag distances. The spatial model of continuity defines how related or unrelated a variable at unsampled location is to the measured variables in its proximity. For example, locations close together in spatial sense, tend to have values in a similar range due to the physics behind geological phenomena.

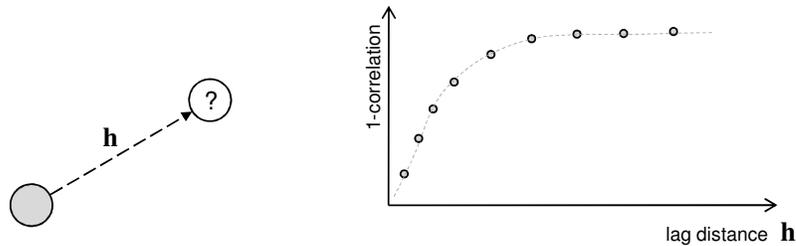


Figure 1.1: An unsampled location is shown with a lag distance \mathbf{h} apart from the measured variable. The variogram defines a measure of correlation between the unknown variables with respect to the lag distance from measure ones. There variogram defined by the vector \mathbf{h} is specific to a certain spatial direction (such as azimuth angle).

Different variations of kriging were introduced based on their underlying models (such as simple kriging, ordinary kriging, and so on) for interpolating the unknown variables (Deutsch and Journel, 1997; Goovaerts, 1997; Journel and Huijbregts, 1978). Moreover,

multivariate geostatistics was introduced as the extension of linear geostatistics to several variables. For example, cokriging is the method of linearly estimating one variable from several other variables. For example, porosity values can be estimated from a linear combination of nearby porosity measurements and corresponding acoustic impedance values. The most widely used multivariate technique of collocated cokriging estimates the primary variable of interest from the nearby measurements of primary variables and the collocated secondary variable at the unsampled location (Doyen, 1988; Xu et al., 1992).

1.2.2 Stochastic Simulation

All kriging approaches derive the ‘best unbiased linear estimator’ according to the specified variogram model. Therefore, they provide only *one* smooth representation of the modeled Earth. In reality, due to uncertainty, generation of one deterministic model is not enough. Consequently, alternative approaches for stochastic modeling of the reservoir facies that can result in a different set of property distribution models were required.

Stochastic simulation was introduced by Matheron (1973) and Journel (1974) to correct for the smoothing effect and other artifacts of kriging, which also allows the reproduction of spatial variance predicted by the variogram model. It is the process of generating several alternative models of the spatial distribution of variables under study. According to Deutsch and Journel (1997) there are two major differences between kriging and sequential simulation:

1. Kriging provides the ‘best’ linear estimator of a variable regardless of the spatial statistics of the estimates taken together. In other words, kriging provides locally accurate representations of the variable. Simulation, on the other hand, provides global representations of the variable, and instead of local accuracy it focuses on the reproduction of patterns of spatial continuity, such as a variogram model.
2. Kriging provides only local measures of accuracy, while simulation provides both the local measures of accuracy and the joint accuracy involving several locations.

Sequential simulations are particularly important for numerical modeling of spatial systems. For example, the global spatial permeability field in an oil reservoir predominantly controls the dynamic flow behavior, and as such, simulation algorithms that honor the global structures specified by the variogram are preferred over the locally accurate methods

of kriging for construction of relevant reservoir models. The smoothing effect of kriging can lead to significant biases when non-linear physical processes are involved. Sequential simulations, besides providing alternative realizations, generate realistic pictures of spatial variability of the geological phenomena. Each realization can lead to different conclusions regarding the connectivity or fluid flow of the reservoir, especially in extreme situations (see Journel and Deutsch (1993) for an example). Therefore, if we are interested in some complex non-linear response functions of the subsurface, we can compute the result for each simulation, and obtain a probabilistic description for the variable of interest.

An example application of kriging versus sequential stochastic simulation is provided in Figure 1.2. In both approaches, the models are conditioned to the 200 available hard data. One can observe that kriging can only produce one unique Earth model; representing the best local estimate of the variable. On the other hand, three different realizations, obtained through sequential simulation, are shown. These realizations provide a more realistic depiction of the fluctuations in geological processes.

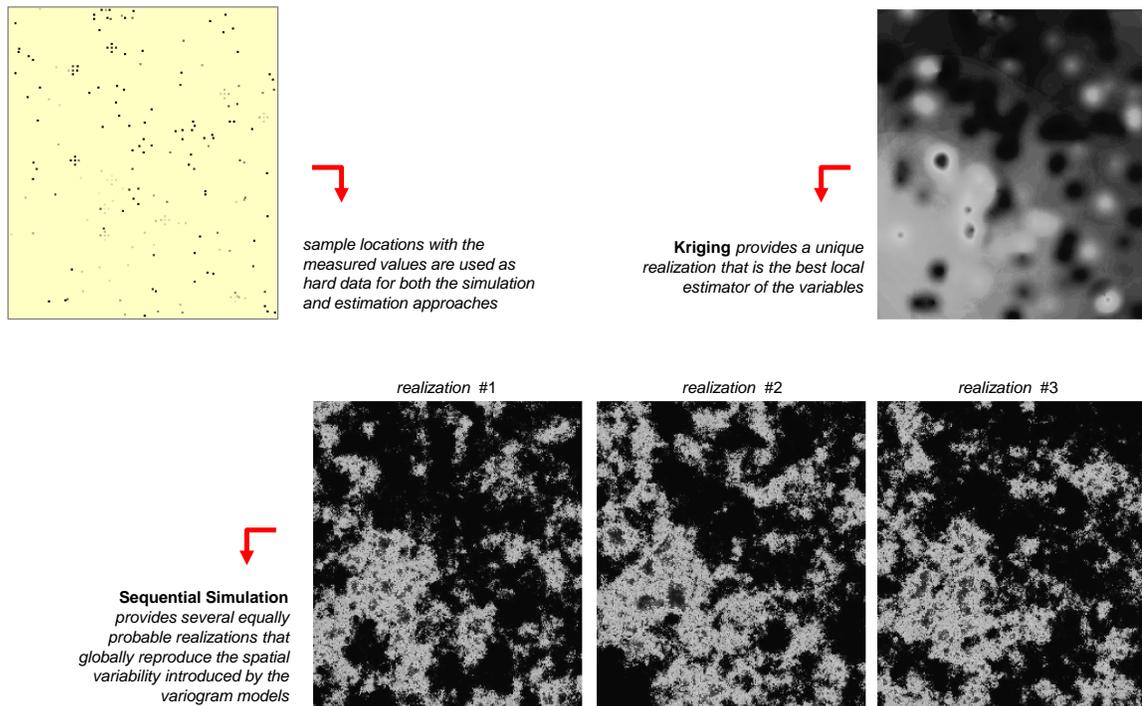


Figure 1.2: An illustration of the kriging result, producing the best linear estimation of the variable conditioned to the available hard data, and three different realizations capturing the spatial variability with the same variogram model.

The mathematical principle behind stochastic simulations is simple. It relies on the conditional probabilities of the variable $Z(\mathbf{u})$ given all previously simulated values in the neighborhood of \mathbf{u} . Consider the joint distribution of N random variables $\{Z_\alpha = Z(\mathbf{u}_\alpha), \alpha = 1, \dots, N\}$ by N available data of any type. The corresponding N -variate conditional distribution is denoted:

$$F(Z_1, Z_2, \dots, Z_N | (n)) = \text{Prob}\{Z_\alpha < z_\alpha, \alpha = 1, \dots, N | (n)\} \quad (1.4)$$

The Bayesian formalisms for conditional probabilities can be exploited for finding the N -variate distribution of the variable Z by successive applications of univariate cdf's with increasing levels of conditioning. That is to say,

$$F(Z_1, Z_2, \dots, Z_N | (n)) = F(Z_1 | (n)) \cdot F(Z_2 | (n+1)) \cdots F(Z_N | (n+N-1)) \quad (1.5)$$

Thus, in general, sequential simulation takes the following steps:

- Draw a value z_1 from the univariate conditional distribution of Z_1 given all the original (n) data
- Draw a value z_2 from the univariate conditional distribution of Z_2 given all $(n+1)$ data; the original (n) data and the previously simulated value $Z_1 = z_1$
- \vdots
- Draw a value z_N from the univariate conditional distribution of Z_N given all $(n+N-1)$ data; the original (n) and the $(N-1)$ previously simulated values

In following subsections, the two main forms of stochastic techniques of sequential Gaussian simulation and sequential indicator simulation, respectively for continuous variables like porosity and categorical variables like facies, will be briefly reviewed.

Sequential Gaussian Simulation

The first sequential algorithm, called sequential Gaussian simulation (SGS), is the most straightforward algorithm to produce realizations of a multi-Gaussian field. The popularity of this method is mostly due to the simplicity and flexibility, introduced by the Gaussian assumption. The basic idea of sequential Gaussian simulation (SGS) is very simple. Each

variable is simulated according to its normal ccdf obtained by a simple kriging method. Unlike kriging, where the mean of the variable was chosen as the estimate for the unsampled location, in SGS, a value is randomly sampled from its corresponding Gaussian distribution, taking into account both the mean and the variance. The conditioning of continuous variable $z(\mathbf{u})$ in SGS proceeds as follows:

1. Choose a stationary domain, and obtain the univariate cdf $F_Z(z)$ representative of the domain under study.
2. Transform the data Z into a normal distribution.
3. Define a random path visiting each node \mathbf{u} of the grid.
 - Search for nearby data and all previously simulated values.
 - Use simple kriging approach to obtain the mean and the variance of the conditional distribution.
 - Draw a single value from the conditional distribution.
 - Assign the simulated value to grid node \mathbf{u} .
 - Proceed to next node until all nodes of the grid are simulated.
4. Back transform from normal distribution to their original univariate cdf $F_Z(z)$.

In above SGS approach, each random path will lead to a different realization. Several realizations, all honoring the given spatial continuity model, can thus be used for uncertainty quantification.

Sequential Indicator Simulation

The application of kriging is suited to reasonably well-behaved phenomena, where the smooth realizations can depict the main geological features. But, in highly variant phenomena with long-tailed distributions, simple variograms become useless. Kriging will underestimate high values and over-estimate low values, and hence, has limited capability in modeling non-linear phenomena. In some situations, the estimation of the variable itself is not of importance, but a non-linear function of it; for example, estimating the probability of mineral grade being greater than a given economic threshold value ($Z(\mathbf{u}) > z_{th}$). Indicator kriging (Journel, 1983) was the first geostatistical technique to model such non-linear geological models.

One of the issues in sequential Gaussian simulation is its inability to honor significant spatial correlations between extreme values (Journel and Alabert, 1988; Journel and Deutsch, 1993). Sequential Gaussian simulation uses a single covariance model for characterizing the spatial geological phenomenon. In some cases, the spatial variability is different for each range of values. For example, shale barriers, having permeability values less than a specific threshold, have good spatial continuity in the subsurface, but a sequential Gaussian simulation approach will underestimate the desired continuity. In other words, values at the long tails of the distribution are spatially uncorrelated. In such cases, a non-linear and non-parametric approach of sequential indicator simulation is used. Sequential indicator simulation (SIS) is a pixel-based simulation algorithm that builds a categorical image by drawing from the categories local probability distributions that are obtained from the specific variogram of the given data threshold (Journel and Alabert, 1988; Alabert and Massonat, 1990). Indicator techniques can easily accommodate different indicator variograms for different facies and, more importantly, can integrate diverse types of soft data (Zhu and Journel, 1993; Deutsch and Journel, 1998).

In the indicator simulation framework, a set of thresholds $z_i = 1, \dots, N_r$ are used for discretizing the range of variability of random variables. Basically, the data are either categorical, such as facies, or continuous, which in this case need to be transformed to categorical variables by an indicator transformation with a given threshold z_i :

$$I(\mathbf{u}; z_i) = \begin{cases} 1, & \text{if } Z(\mathbf{u}) < z_i \\ 0, & \text{if } Z(\mathbf{u}) \geq z_i \end{cases} \quad (1.6)$$

The experimental indicator variogram can thus be inferred from the heterogeneity model corresponding to each specific threshold:

$$\gamma_I(\mathbf{h}; z_i) = E \left\{ [I(\mathbf{u}; z_i) - I(\mathbf{u} + \mathbf{h}; z_i)]^2 \right\} \quad (1.7)$$

Noticeably, the model still considers only two-point statistics, but the introduction of a variogram model for each threshold allows the reproduction of non-linear features in the geological models, such as shale barriers. The same concept of kriging for finding a linear combination of known measurements is applied within the indicator formalism. The conditional expectation of the indicator variable is computed with the indicator coded conditioning data as follows:

$$I^*(\mathbf{u} \mid z_i, (n)) = F_Z^*(\mathbf{u} \mid z_i, (n)) = \sum_{\alpha=1}^n \lambda_{\alpha}(\mathbf{u}) \cdot i(\mathbf{u}_{\alpha}; z_i) \quad (1.8)$$

The simulation process is similar to SGS approach, where the nodes are visited in a random manner, constructing the conditional distribution and sampling a value from the local cdf for that node, and proceeding to next location with the addition of previously simulated value to data conditioning. The process continues until all nodes of the simulation grid have been visited.

1.3 Object-based Algorithms

There have been several algorithms proposed for geostatistical modeling; a sequential simulation framework (Journel, 1983; Isaaks, 1990; Srivastava, 1992; Goovaerts, 1997; Chiles and Delfiner, 1999), iterative approaches (Bratvold et al., 1993; Tyler et al., 1992), or even a direct approach that is neither iterative nor sequential (Ravenne and Beucher, 1988; Rudkiewicz et al., 1990). However, all these approaches are based on two-point statistics that are defined by a histogram and a variogram model, and hence, are unable to reproduce complex spatial patterns, such as channels. In flow modeling, these spatial patterns of continuity have a significant effect in the non-linear response functions. Accurate models and realistic realizations can better quantify the spatial uncertainty, given all the available direct and indirect information. There have been some attempts to reproduce curvilinear structures and patterns of continuity. For instance, according to Deutsch and Lewis (1992), anisotropy directions of variograms can be modified locally. Even the variogram ranges could be modified to correct for additional connectivity of the geological features (Deutsch and Gringarten, 2000). However, in spite of all these methods, none could really reproduce complex patterns or shapes.

The introduction of object-based simulation algorithms amounts to building, in space, whole objects with their specific geometries, one at a time (Haldorsen and Lake, 1984; Stoyan et al., 1987; Haldorsen and Damsleth, 1990; Deutsch and Wang, 1996; Holden et al., 1998). Object-based methods distribute geometric objects according to some probability laws in space by the use of marked point processes. They drop objects on the simulation grid until the proportional constraints (such as net-to-gross ratios) are met. Conditioning to additional information basically follows a trial-and-error approach by modifying the object

sizes, moving them around, or adding/removing objects. One of the main advantages of object-based modeling is the visual appeal of the final result. Using this framework, geologists would be able to see familiar sedimentary objects and have direct control over the shapes and sizes of those features. For example, a channel can be characterized by its sinuosity (amplitude and wavelength), channel width, channel length and channel thickness. Object-based methods try to honor this information, and place the desired geometrical objects on the simulation grid by trying various combinations of parameter distributions to get a visually satisfying geological image. Despite great visual crispness of the resulting geological features, object-based modeling approaches have three main drawbacks that make them unsuitable for geostatistical modeling:

1. They are difficult to condition to local data. For example, prior proportion curves and densely packed well data will make the trial-and-error process of object positioning extremely difficult and CPU demanding. Even the advent of simulated annealing approaches for conditioning the object-based methods were not able to provide reasonable accuracy within acceptable number of iterations, if any.
2. The multivariate distribution of marked point processes are too complex, and cannot be analytically defined.
3. Different geological environments require custom algorithms for modeling the complex features deemed relevant to the study area. For instance, a fluvial reservoir setting consists of channels, levees, and crevasse sands within a floodplain shale. Several different approaches have been designed for modeling solely this fluvial reservoirs in the Norwegian North sea (Clemensten et al., 1990; Damsleth et al., 1992; Fält et al., 1991; Henriquez et al., 1990; Omre, 1992; Stanley et al., 1990). The implementation and the theory for fluvial modeling kept refining over many years (Georgsen and Omre, 1993; Hatløy, 1995; Hove et al., 1992). This shows the need for such detailed algorithms for each geological scenario.

1.4 Multiple-Point Geostatistics

There are many measures of spatial continuity. Previous two-point statistical approaches used the linear correlation between any two points in space, or simply a variogram, as their model of spatial continuity. They rely on only two points, but in fact, what we would like to

see is the correlation of many points at once. As an example, this is similar to an orchestra. Because there is that complex interaction between various instruments, one would like to listen to a symphony where all the instruments are played together, rather than listening to each instrument alone or to duets separately (the variogram). Subsurface heterogeneity is very similar to an orchestra. The physical processes responsible for the heterogeneities are too complex to be just represented by a two-point variogram model. Some geological architectures, i.e. a meandering channel system, cannot be described by traditional random function models. Therefore, multiple-point geostatistics (MPS) was proposed for modeling the highly complex and curvilinear geological features of the subsurface. These methods, unlike object-based approaches, can be more easily conditioned to available data.

The general idea is to go beyond bivariate moments, and use multiple-points in inference of unknown random variables. This is illustrated in Figure 1.3, where the traditional approach takes into account the random function model between two points, and thus, accepts linear measures of spatial continuity, but the multiple-point geostatistics infers statistics from a configuration of more than two points, which allows reproduction of non-linear spatial correlations. In the example provided in Figure 1.3, the interaction of five points with a specific configuration defined by four vectors of \mathbf{h}_1 , \mathbf{h}_2 , \mathbf{h}_3 , and \mathbf{h}_4 illustrates the concept behind multiple-point geostatistics.

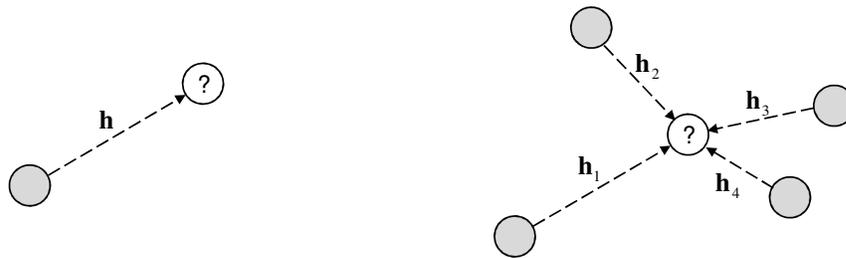


Figure 1.3: (Left) the two points configuration used to infer statistics in traditional variogram-based approaches, (Right) a sample of a 5-point configuration used in multiple-point geostatistics to model highly complex geological phenomena and curvilinear sedimentary features.

However, in reality, one has the problem of finding enough pairs of data for two-point statistics, let alone finding triplets or quadruplet for MPS in a data sparse environment. Direct inference from subsurface is thus impossible in such situations, and a suitable random function model that can describe the spatial architecture and the physical process of interest

cannot be built. Therefore, one needs to infer the multiple-point statistics from a conceptual image (possibly built based on interpretation from subsurface data) that characterizes the geometrical/geological features of the property of interest. Such a conceptual image is called a training image, \mathbf{ti} .

This training image can be built from the physics, i.e. simulation of channel deposition. From that physics, several triplets or quadruplets can be obtained for inference. Therefore, inference is not coming from the actual data anymore, but from simulated data of the phenomenon, depicted by a training image. In geology, we can simulate the physical process or even make photographs of the outcrop. This allows us to depict channels or lobes, and to infer multiple-point statistics from that photo. The measure of spatial continuity that comes from this training image is more complex and more complete than variograms because it uses many points at the same time. The training image should thus be deemed relevant to the deposit under study. However, it does not need to honor any specific data, but simply depict a conceptual image of the spatial geological features of the subsurface. Object-based algorithms are perfect candidates for generating the required training images since spatial structures and patterns can be ideally simulated by complex objects, without any need for data conditioning.

The advent of multiple-point geostatistics is seen through the work of Guardiano and Srivastava (1993). The application of multiple-point geostatistics proceeds by assuming a n -configuration template of $(0, \mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n)$, similar to the one shown in Figure 1.3. Then, the training image is scanned with this template for all replicates of the n random variables being $Z_1 = z(\mathbf{u} + \mathbf{h}_1), \dots, Z_n = z(\mathbf{u} + \mathbf{h}_n)$. This allows us to infer the empirical conditional distribution of $Z(\mathbf{u})$ at node \mathbf{u} . One can then sample a value, $z(\mathbf{u})$ from this distribution, and assign it to the simulation grid. A similar sequential approach, as in variogram-based geostatistics, is used here to randomly simulate all the nodes of the grid.

The proposed approach of Guardiano and Srivastava (1993) stayed impractical for many years due to its high computational complexity and CPU demand. However, the need for such complex geological models with highly non-linear sedimentary features necessitated the birth of many multiple-point geostatistical algorithms throughout these years. These methods, although following the same rules of inferring multiple-point statistics from the conceptual training image, can be divided into three categories of: (1) probabilistic-based, (2) optimization-based, and (3) pattern-based approaches. An overview of each of these methods is provided hereafter.

1.4.1 Probabilistic-based Approaches

Snesim

Strebelle (2000) developed the first structured multiple-point statistics algorithm of ‘single normal (extended) simulation’ (called *snesim*) for simulating categorical variables. The approach of *snesim* is based on inferring statistics from the training image, in the sense that the probability of the random variable is obtained given the nearby conditioning multiple-point data event. It is thus a probabilistic approach similar to the one proposed by Guardiano and Srivastava (1993). However, in order to reduce the computational complexity of computing the conditional probabilities, instead of scanning the training image for each conditioning data template, it stores all probabilities in a search tree by a one-time scanning of the training image. The search tree data structure allows a fast retrieval of all required conditional probabilities during the simulation.

Snesim uses a pixel-based sequential simulation approach, which makes the conditioning to well and seismic data much easier than in object-based modeling techniques. One of the shortcomings of *snesim*, however, is that the training image needs to be categorical, and it does not work with continuous variables. Furthermore, the *snesim* algorithm can only capture the stationary features of the training image. Additionally, any type of trend, such as vertical or areal proportion changes need to be explicitly enforced.

Indicator-based

Some authors attempted to integrate indicator techniques with a multiple-point probability. For example, Ortiz and Deutsch (2004) incorporated the multiple-point statistics in a Bayesian framework and under the assumption of conditional independence between sources of information by an indicator technique. In this technique, the multiple-point statistics are directly obtained from the available data. At each location on the grid, the conditional probabilities from multiple-point information are integrated with the ones obtained through indicator kriging, and used for drawing a value for that location.

Later, Ortiz and Emery (2005) proposed another approach to integrate multiple-point statistics for all traditional sequential approaches (such as indicator, multi-Gaussian, or disjunctive kriging). In addition, Hong et al. (2008) integrated multiple-point statistics with secondary data (instead of just primary data) by estimating the joint probability of the collocated secondary data and the nearby multi-point conditioning data.

It should be mentioned that the multiple-point statistics are not honored by the proposed indicator-based method. Even though some higher-order features are introduced into the generated models by locally modifying the probabilities obtained by indicator kriging, but the same visual aspect of indicator methods is evident in the final realizations. Furthermore, data integration is also dependent on the Tau model of Journel (2002), which is not only hard to get, but also location and data dependent.

Impala

Impala is a list-based approach for multiple-point simulation in a probabilistic sense. It follows the same concepts as in snesim for storing the conditional probabilities derived from the training image, and later, using them during simulation. However, instead of the search tree of snesim, Impala uses a list structure, for storing the multiple-point statistics inferred from the training image (Straubhaar et al., 2011). The advantage is the reduction in the amount of memory required for MPS algorithm. In addition, it provides the means for paralleling the retrieval of conditional probabilities. In spite of the memory and CPU advantages of using lists, the methodology behaves exactly similar to snesim algorithm.

Cumulants

Cumulants are combinations of statistical moments, such as mean or variance, that allow the characterization of non-Gaussian random variables (Rosenblatt, 1985). The concept of cumulants was first used by Dimitrakopoulos et al. (2009) in the spatial context to characterize non-linear stationary and ergodic spatial random fields. Higher-order spatial cumulants can capture the complex geological features and geometrical shapes of the physical phenomena.

A covariance being a measure of correlation between pairs of points distant by a given vector \mathbf{h} , spatial cumulants of higher-order are similarly the measure of correlation but in the direction of a pre-defined spatial template. Dimitrakopoulos et al. (2009) showed that high-order cumulants can characterize spatial pattern redundancy and are correlated to the orientation of the spatial template. Therefore, each geological process requires its own choice of cumulants for optimal pattern analysis. It has also been shown that cumulants up to and including fifth-order are enough for efficiently characterizing the spatial geometries in training images.

Next, Mustapha and Dimitrakopoulos (2010b) provided a computer code for calculating

higher-order spatial cumulants. Subsequently, the previous exploration of spatial cumulants was used as the basis for the simulation of complex geological phenomena by Mustapha and Dimitrakopoulos (2010a). The simulation takes advantage of spatial cumulants in the high-dimensional space of Legendre polynomials in a sequential framework, called ‘hosim’. It proceeds by randomly choosing a spatial node \mathbf{u} , estimating the conditional probability of the random variable given the neighboring data and previously simulated nodes, and finally, drawing a value for that node from the distribution. The process repeats until all the nodes of the grid have been visited. The only difference is the method of deriving the analytical expressions for the local probability density functions. In hosim, Legendre coefficients are obtained from the multiple-point templates, and used to derive the expressions for multivariate conditional distributions. The proposed cumulant-based method is assumed to be less dependent on the training image statistics than snesim method, but more data-driven; in a sense that it first tries to find the multiple-point statistics from the data, and only if not enough replicates could be found, the training image will be used for inference.

Mustapha and Dimitrakopoulos (2010a) claimed perfect reproduction of the histogram, variogram, and higher point statistics of the sample data and the training image by the method of ‘hosim’. However, the generated realizations had much less visual appeal for such a reproduction, especially for thin features in the training image. And most importantly, the choice on the spatial cumulants for each conceptual training image, having its own geometrical features and connectivities, remains an open and challenging problem.

1.4.2 Iterative Approaches

Simulated Annealing

Simulated annealing is a global optimization algorithm that can reduce the specified objective function. One of the first applications of simulated annealing in the spatial context was proposed by Deutsch (1992). The algorithm generates models by iteratively minimizing an objective function that characterizes the mismatch between the desired statistics of the realization and the prior model of spatial continuity.

The process starts by a random spatial distribution of values on a simulation grid. In each iteration, a node value on the grid is perturbed and the updated mismatch between current statistics and the target ones are quantified. The perturbation is kept only if it reduces the mismatch, i.e. lowers the objective function. However, in order to reach the

global minimum and not get trapped in local minima, simulated annealing approaches may sometimes reject perturbations with lower mismatches, or even accept unfavorable ones according to what is called an ‘annealing schedule’. The process continues to iteratively refine the realization grid so that the multiple-point statistics of the training images are reproduced.

This iterative approach has been applied to simulation of categorical variables (Goovaerts, 1996), or continuous petrophysical properties (Fang and Wang, 1997). The dependency on simulated annealing for optimizing the objective function calls for a subjective choice on optimization parameters and perturbation mechanisms for an efficient simulation. There have been many attempts regarding the choice on the initial temperature in annealing optimization (Norrena and Deutsch, 2000), or the update in the objective function and the perturbation mechanism (Deutsch and Wen, 2001; Deutsch, 2002). But generally, these parameters strongly depend upon the model of spatial continuity and need to be set empirically by trial-and-error.

Later, Peredo and Ortiz (2010) applied simulated annealing in the context of parallel computing to relieve the computational cost. Despite all these efforts, a simple visual comparison of geometrical features in realizations demonstrates that considering only the low-order statistics of the spatial model (due to computational limitations in computing the objective function) is not sufficient for construction of desirable Earth models. The iterative approach of simulated annealing also tolerates some amount of inconsistency in the realizations. Furthermore, CPU and memory related issues arise by an increase in the number of categories or the order of multi-point statistics. The choice on which multiple-point statistics to be retained in the objective function is also an open to question in simulated annealing approaches.

Neural Networks

Neural networks are based on ways the brain performs computations, and are used for fitting non-linear functions and recognizing patterns within the data. The first application of neural networks for multiple-point geostatistical modeling was proposed by Caers and Journel (1998).

The neural network is trained by the training image to determine the local conditional probabilities. A neighborhood template is chosen and all the multiple-point neighboring

values are mapped by the network to a one dimensional output, representing the conditional probability of the node centering the template. In the learning phase, training image statistics are used to adjust network parameters.

Similar to all iterative approaches, the simulation starts by a random initialization of the realization grid. A random path is defined on the grid, and at each location \mathbf{u} , the conditional probability of the variable $Z(\mathbf{u})$ is retrieved from the trained neural network. Simulation proceeds by cycling through the entire field until some criterion of convergence is reached.

Several examples of unconditional simulation were shown by Caers and Journel (1998). Neural network approach was also applied in data integration for reservoir modeling. Caers and Ma (2002) proposed an approach for modeling the conditional probability of facies given the additional seismic data.

The neural network architecture consists of several internal nodes, network layers, inputs, outputs and how all these are linked together. Determining the correct parameters for an effective simulation depends on the specific spatial model under study, and requires modelers' expertise. Additionally, similar to simulated annealing approach, a reliable convergence criterion is hard to be quantified, and the algorithm does not know when to stop cycling on the entire image. The learning phase requires the same attention, i.e. when to stop the learning to avoid over-fitting.

MRF Models

Markov random fields (MRF) have been used as the spatial continuity model of sedimentary facies in the subsurface (Besag, 1974). By including the higher-order interaction terms, one can theoretically obtain models with realistic physical properties. These approaches explicitly state a model, which depends on hundreds of parameters, for multivariate distribution of random variables.

Initially, Tjelmeland (1996) proposed an iterative approach based on a simple Metropolis-Hastings algorithm (Metropolis et al., 1953). In each iteration of a K categorical model, first, a potential new data template is obtained by replacing the current facies at node \mathbf{u} with a randomly selected one out of $K - 1$ other facies. Second, the new facies will be accepted according to a probability defined through Metropolis-Hastings algorithm. This Markov chain process should, with large enough iterations, converge to the specified MRF.

In addition, Daly (2005) chose a unilateral path with a MCMC in a model that is related

to MRF and termed Markov Mesh model (MMM). It is an approximate simulation, which is improved by several iterations of MCMC algorithm. However, it could only give good results for seemingly non-symmetric models.

Tjelmeland and Eidsvik (2004) also proposed a method based on the Metropolis-Hastings (MH) algorithm for simulating a Markov chain that converges to the desired posterior distribution. The simple sampling mechanism is to propose a new value for the random variable, and then, accept/reject the new variable with a certain probability. In their algorithm, however, a directional MH algorithm was adapted where blocks of the spatial variable were updated at each iteration. In other words, instead of drawing a new random variable at a specific location, they first generate an auxiliary random variable defining a direction, and afterwards, propose a value on the line specified by the current node and the auxiliary direction. The proposed approach still assumes a Gaussian random field prior model, but uses non-linear likelihood functions to explore the original non-linear posterior.

Resulting models in all these approaches suffer from slow convergence of the Markov chain simulations. Moreover, they cannot control the reproduction of large-scale structures in the realizations. For each specific depositional environment, one needs to find suitable MRF models. Estimating the parameters of a MRF model is difficult and CPU demanding. Moreover, the model specification, namely the choice of which higher-order statistics (or cliques in MRF terminology) to model in the MRF model, is still an open question. This makes modeling of new geological environments a daunting task, particularly in 3D where no success with this technique has been reported. Due to slow convergence of these algorithms, an optimal number of iterations is also required prior to the simulation. For example, stopping prematurely will result in images that are not at all representative of the specific model. In addition, model parameters in these methods have no clear geometrical interpretation, which makes parameter estimation dependent upon the computationally expensive MLE (maximum likelihood estimation) techniques with analytical expressions that are, if not impossible, hard to derive.

Markov-Mesh Models

Markov mesh models constitute a subclass of Markov Random Fields that are defined through a unilateral scan path of a regular grid (Daly, 2005). However, conditioning to hard and soft data that are located along the future simulation path is difficult and problematic. The conditional probabilities are defined only by the cell values in a sequential neighborhood.

This simple neighborhood is illustrated in Figure 1.4.

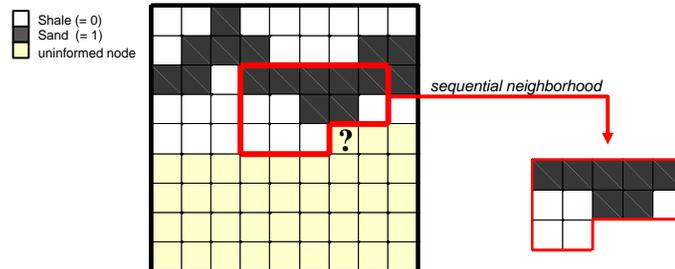


Figure 1.4: The unilateral path used in Markov mesh models for sequential simulation, where the assumption of dependence of the conditional probability of a value on its sequential neighboring cells holds.

A similar idea of texture synthesis seen in computer graphics literatures, where Markov models are combined with sequential simulation, has also been investigated for multiple-point geostatistical simulation (Daly and Knudby, 2007; Parra and Ortiz, 2009).

Due to the drawbacks originating from the inability of unilateral simulation to condition to data located along the future path of each cell, and hence, inconsistencies in ccdfs, Kjøsberg and Kolbjørnsen (2008) proposed to use probabilities obtained from indicator kriging to modify the probabilities from Markov mesh models. However, they still use iterative approaches on the modified probabilities to update variables in a MH algorithm to wash away the undesirable kriging effects while honoring the prior model. Although the algorithm is capable of generating models with proper conditioning, their results show some internal inconsistencies with the rejection sampling approach.

Later, Stien and Kolbjørnsen (2011) focused on model formulation for simulating unconditional realizations. Generalized linear models (GLM) were used for parameterizing the conditional probabilities. The distribution of random variables in their model depends on a linear combination of neighboring variables through a non-linear exponential function. However, the method cannot reproduce the desired volume fractions of the training image, and an iterative approach of tuning the model parameters has to be adapted for a correct reproduction of statistics. The issue of conditioning still remains in Markov mesh models.

Gibbs Sampler

Gibbs sampler is a statistical method proposed initially by Geman and Geman (1984). This method is also a special case of Metropolis algorithm for drawing samples from complex

joint/marginal distributions. Lyster and Deutsch (2008) applied the Gibbs sampling MCMC approach for multiple-point simulation of geological phenomena.

The process is similar to Markov-based approaches where an initial random image is iteratively refined by resampling node values from the prior multiple-point statistics of the training image. The main challenge is to find the conditional probabilities of facies at each point. In the method proposed by Lyster and Deutsch (2008), a technique based on linear combinations of observed data was adapted. However, instead of the random variables themselves, it uses multiple-point events (MPE). The multiple-point event is a pre-defined configuration of points that the practitioner deems appropriate for statistics of the training image. For example, it could consist of different spatial configurations of four points. Solving the Gibbs sampler system involves calculating the weights for each of the chosen multiple-point events from the training image.

Again, the iterative method of Gibbs sampling requires a stopping criterion, compromising between the speed of the algorithm and statistical reproduction of patterns of the training image. There are some cases that excessive iterations may even lead to artifacts in the generated realizations (Lyster and Deutsch, 2007). Furthermore, the appropriate choice on multiple-point events for any specific training image is unknown. This has a direct consequence on either the quality of complex geometrical features in generated realizations if small MPEs are chosen, or the speed of the iterative algorithm if spatially large data events are selected.

1.4.3 Pattern-based Approaches

Simpat

The probabilistic paradigm in multiple-point geostatistics, such as *snesim*, was substituted with a pattern-based approach by Arpat (2005); Arpat et al. (2007) to circumvent the limitations seen in probabilistic approaches (mostly the training image pattern reproduction or lack thereof). The pattern-based MPS method, called *simpat* (simulation of patterns), introduced an alternative approach for multiple-point conditional probability inference of random variables. The method, inherently, takes probabilities of whole multiple-point patterns conditioned to the same multiple-point data event from the training image.

The concept of ‘pattern’ represents a set of values defined by a specific multiple-point template. For example, in 2D, it could be the multiple-point values extracted with a 5×5

square over the training image grid. The proposed method also follows the same sequential approach of previous MPS methods. At each node location, it extracts the data event defined by a pattern template, and then, searches through all the patterns in the training image to find the most similar one. The most similar pattern is then pasted on the realization grid. The process continues until all the nodes of the simulation grid have been visited. The similarity search of the data event with the patterns basically takes into account the conditioning neighboring data values for the retrieval of multiple-point patterns. In other words, instead of local conditional distributions, a local pattern similarity criterion is honored.

Stochastic simulation is thus similar to an image construction problem as if one tries to put the puzzle pieces together. Simpat is shown to provide visually-appealing realizations. However, one of the main issues concerning this approach is the huge computational complexity associated with similarity searches; i.e., comparing the data event with all the patterns in a training image. This leads to poor CPU performance in comparison to other geostatistical algorithms.

Direct Sampling

The direct sampling method was proposed by Mariethoz and Renard (2010). This approach is a pattern-based approach similar to simpat. It follows the same concept of patterns to reproduce the same conditional distributions inferred from training image statistics.

In contrary to simpat, Direct Sampling method claims to be independent from a pre-defined template size. Simulation involves sequentially visiting all the nodes on the grid. At each node location, it retrieves n neighboring informed nodes of the grid within a specified search radius l . It then defines the data event according to the lag vectors obtained from the neighboring informed node locations, and searches the training image for this data event. However, unlike simpat, some stopping criteria are enforced. It stops the search in two situations: whenever an exact match is obtained, or the dissimilarity falls below a given threshold. If no such cases are found in the entire training image, it reverts back to simpat idea on choosing the configuration with minimum dissimilarity distance. However, the multi-grid concept, used in previous MPS technique for reproducing large-scale patterns of the training image, is inherent to this approach. At the beginning of simulation, it has large spatial coverage of the realization grid, but as the simulation proceeds the n -point data event can be easily retrieved from a smaller spatial radius (Mariethoz et al., 2010).

The direct sampling method will directly search the training image for the data event at hand. Therefore, it is very much like the *simpat* algorithm. The differences can be attributed as follows: (1) the search will stop upon one exact match, (2) there is another stopping criterion if the distance is below a threshold, and most importantly (3) the data event does not conform to any pre-defined shape. The final difference is however analogous to *simpat* and other pattern-based approaches. The spatial search radius, l , is similar to the spatial coverage of the pattern template of *simpat*. In other words, specifying the size of the pattern template in *simpat* is equivalent to defining a spatial search area in the direct sampling method. Moreover, the same configuration of informed nodes in the direct sampling method can be captured through the pattern template in *simpat*. And since *simpat* gives a weight of zero to the uninformed nodes on the pattern template, the similarity search is performed solely for the informed nodes. This is similar to the direct sampling method which searches the training image for the neighboring informed nodes.

One of the main advantages over *simpat* is the reduction in computational complexity. However, there are several parameters (more than in previous pattern-based approaches) that require tuning for an efficient simulation in terms of both the speed of the algorithm, and the quality of multiple-point statistical reproduction of the training image patterns.

Filtersim

Simpat requires very large and rich (varied) training images and the simulated realizations are merely local shuffles of the training image. Also, it is a very time-consuming and CPU-demanding algorithm. Thus, a new algorithm, called *filtersim*, was introduced with the idea of summarizing multiple-point spatial patterns using a few general linear filters (Zhang, 2006; Wu, 2007). The multiple-point pattern-based approach of *filtersim* is summarized as follows:

- The patterns of the training image are characterized by some linear filters (i.e. six filters for two dimensional patterns)
- The patterns are classified into bins according to some similarity/distance criterion with respect to their filter scores. Each bin is then characterized by an average pattern called prototype.
- All nodes of the simulation grid are randomly visited, similar to any sequential simulation algorithm.
- At each node location, the multiple-point conditioning data event is retrieved.

- The most similar training prototype to the data event is obtained, and a random pattern from that group is then pasted on the simulation grid
- The process continues until all the nodes have been visited.

Because of the dimensionality reduction brought by the filter summaries of the patterns, and because all bins have been pre-classified into a data tree structure, the algorithm is potentially fast and reasonable in terms of RAM demand.

However, the dependence on linear filters for pattern classification is one of the main issues in filtersim algorithm. Different training images demand for different linear filters. The capability of the filters to correctly distinguish between different patterns depends mostly on the structures and geometries of patterns themselves. For example, the filters may provide reasonably well classification with continuous variables, but low classification accuracy for binary images. Besides the limitation on the filters definition, there is no prior knowledge on the appropriate choice of filters for each training image structure. No theoretical or experimental link exists between the classification accuracy and the filters themselves. Additionally, similar to some other MPS algorithms, several parameters have to be optimally tuned to obtain visually-appealing realizations that honor the statistical properties of the training image. It thus calls for arduous trial-and-error experiments by the practitioner (refer to Dujardin et al. (2006) for such an experiment).

Wavelet-based

The use of wavelet analysis for pattern-based simulation of categorical and continuous variables was first proposed by Chatterjee et al. (2011). Wavelet analysis can decompose a training image into different frequency components (Mallat, 1999). The technique is similar to filtersim algorithm. Instead of using filters for dimensionality reduction of patterns of the training image, it uses wavelet decomposition at certain scales. The wavelet sub-bands can capture most of the pattern variability, and reduce the dimensionality of the patterns. The resulting approximate sub-band coefficients of patterns are then used for pattern classification. During the simulation, the same similarity search is performed between the conditioning data event and the patterns of the training image.

The technique integrates the conditional distribution function of each class into simulation process for categorical variables. The cdf function is generated based on the frequency of the individual categories at the central node of the template. So, instead of randomly

selecting a pattern from the most similar class (to the conditioning data event), the cdf of the class is used to draw a pattern, similar to Monte Carlo methods.

The proposed method based on wavelets does not bring any improvements over previous pattern-based approaches, neither methodological, nor practical, nor in quality of the results. For example, the incorporation of each class cdf function in MPS simulation is claimed to provide better statistical reproduction of the training images. However, by randomly selecting a pattern from each class as in *filtersim*, the probabilities of the central node (i.e. being sand or shale in binary images) is inherently considered during MPS simulation. So, *filtersim* is also following the same cdf simply by random sampling of the patterns.

One of the shortcomings, however, is the amount of complexity this technique adds to the entire algorithm; for example, the size of the sub-band (or in simple terms, the amount of dimensionality reduction of patterns). In this method, the modeler has to tune this value to obtain, as a compromise between goodness of final realization and the computational time of the algorithm, the best dimensionality reduction appropriate for the specific training image at hand.

Another similar but important shortcoming of this approach is the wavelet function and its decomposition levels. It is critical to know the best decomposition level for classifying the patterns. In other words, the decomposition level is not linearly related to classification accuracy. Increasing the decomposition level does not, in a linear sense, guarantee more accurate clustering. For example, two decomposition levels may provide better results than using three decomposition levels. Including all levels (which is equivalent to ignoring wavelets decomposition and just working with the original patterns) may even lead to worse classification (according to the arguments provided by Arpat (2005) on using proximity transforms for better similarity comparison). The choice on the decomposition level, prior to the algorithm, is still an open problem. Other factors, related to the wavelet transform, could also be considered as the source of error/uncertainty.

1.5 Motivation

Why do we need patterns, and why is it important? In an unconventional sense, pattern is what makes life beautiful; pattern is what makes life entertaining and pattern is what makes this wonderful mystery a reality. Pattern is what makes Beethoven's symphony truly exceptional, Leonardo da Vinci's arts a masterpiece, and facades of la Sagrada Familia in Barcelona so attractive. The exact form or pattern of our solar system is what allows us to walk, eat and survive. If Earth was positioned differently, water would not be formed, temperatures would not be ideal, and feet would not keep us upright. If Earth was not spherical, gravity would not be evenly distributed, seasons would not exist, and life would not have developed.

Patterns are all around us; the formation of clouds known as von Karman vortices, or the texture and formation of different mineral rocks. The way cells grow in our bodies, or the way plants form. Fluids such as water tend to move in certain patterns of branching out to smaller and smaller streams. The locations that crevasse splays are formed next to a river. The way shale layers divide geological structures into distinguishing strata. And most importantly, the way it allows us to acquire knowledge about our environments and distinguish the objects around us. Thus, patterns become a major part of physical systems and visual processings, around which we will build the main core of our geostatistical modeling.

This dissertation is exactly trying to do this: *“it uses patterns to model the same complex physical processes that shaped the patterns of the subsurface”*. The ultimate goal is to develop an algorithm that allows the computer to truly grasp and model the subsurface; to unravel the spatial model depicting the geological processes. We would like to obtain a general and unified framework for pattern analysis, and formulate geostatistical simulations to obtain better spatial understanding of the conceptual geological model and its underlying patterns.

In previous sections, several multiple-point geostatistical algorithms within three categories of probabilistic-based, iterative-based, and pattern-based approach were reviewed. Every algorithm has its own limitations or drawbacks. There is yet to find an 'ideal' algorithm for MPS modeling. However, we may never attain such an algorithm. What we plan on achieving is taking small steps towards this ultimate goal - *the ideal algorithm*.

The motivations in this dissertation can be organized according to the three phases of a geostatistical routine: learning, learning/simulating, and simulating. We will elaborate on

each of these three aspects hereafter.

Learning

In every multiple-point algorithm, one has a learning phase, where the spatial model depicting reality (the training image) is analyzed. In probabilistic approaches, such as snesim, one needs to construct a search tree that defines the statistical model of the random variable under study. By analyzing the training image statistics, the conditional probabilities of different data events are extracted. This learning phase is thus building a probabilistic model of the subsurface. The same is true for the method of cumulants. The use of cumulants and the selection of the corresponding multiple-point templates, which are deemed capable of describing the specific features of the training image, is another example of constructing a statistical model. These probabilistic algorithms learn/construct a new model from the original spatial model that is provided to them.

Similarly, in iterative approaches, the algorithm has to make some subjective assumptions on the probability density functions of the subsurface, or trains a descriptive model of the subsurface. How can we be certain that the subjective model, intrinsically defined by the assumptions of the algorithm, can correctly learn the complex patterns or features of the subsurface? Moreover, how to choose the model parameters to ensure an accurate representation of the subsurface? Different sets of parameters for the model can significantly affect the performance of the entire geostatistical simulation. As mentioned in previous sections, the convergence criteria, the stopping point, the number of layers of a neural network, or the parametric function in a MRF are examples of such parameters that cannot be objectively defined.

These methods, all have one notion in common: to generate another descriptive model from the original spatial model (the training image). In all these techniques, a statistical relation is hypothesized. The introduction of a new model entails an extra set of parameters. In order to provide a better algorithm, one needs to eliminate, as much as possible, the subjectivity that is induced into modeling by this extra set of parameters. Moreover, the algorithmic decisions that are required in these MPS methodologies are simply instruments that facilitate the task at hand. The algorithmic (subjective) decision in snesim to drop the farthest nodes to enable the extraction of conditional probabilities, or the need for indicator probabilities prior to a Markov mesh model are examples of the instruments whose sole purpose is to *make things work*.

However, if we are better off removing the dependence on extra parameters for characterizing the input spatial model, then, where should the algorithmic inspirations come from? In traditional geostatistics, i.e. kriging, the spatial model was provided by a variogram. The model is simply characterized by a curve relating the correlation of pairs of random variables with respect to their lag distances. This spatial model was the only one used in the algorithm. There was no need to build an additional model, on top of it, to generate realizations. If one provides the spatial model, without any subjective algorithmic decisions, kriging would provide one deterministic result. The resulting realization would honor all the input statistics that were provided by the histogram and variograms. Unfortunately, in multiple-point geostatistics, the complex geometries and curvilinear features within a rich training image undoubtedly called for intricate algorithms. But similar to the kriging approach, we need algorithmic inspirations that can eliminate this extra dependency on tuning complex algorithmic parameters.

In order to obtain inspirations for an ‘intelligent’ algorithm that models complex geological phenomena, we need to understand our own visual system and how we would perform the intended task. For that matter, we attempt to draw inspirations from biology. We believe that our exceptionally powerful visual system has been optimized by evolution to learn the phenomena and find the optimal solution. It is taken as the de-facto standard. For example, observe the Figure 1.5 showing a very low-resolution image. This image does not provide any meaning for a computer, but the human visual system not only can recognize the faces, but may also identify who they are. This demonstrates our powerful cognitive abilities, and the reason why searching for such a biological inspiration would be beneficial.



Figure 1.5: Exceptional ability of the brain to decipher low-resolution images. Not only it can recognize the existence of human faces, it can also recognize those people; Bill Clinton and Hillary Clinton.

Thus, developing a basic understanding of the human visual system is the first step towards our ultimate goal. The brain searches instinctively for “patterns” and makes sense of the ones that it recognizes. This ability to recognize patterns is the key element in many disciplines.

According to Gardner (1993) our visual intelligence is derived from a combination of several intelligences; the most notable one being “patterning”. It is characterized by our ability to see patterns in things. Patterning refers to the meaningful organization and categorization of patterns (Caine and Caine, 1994). It is also an attribute described as the “inner logic” of perception in visual problem-solving (Gregory, 1974). This ability to see patterns is the key to perception, and deriving meaning out of different elements to form a unified image of the way things work. Therefore, one needs to comprehend the pattern of forces that govern the subsurface geometries and structures.

We thus adapt the pattern-based approach for multiple-point geostatistical simulations. The way the brain perceives a geological concept is not by constructing a statistical or a Markovian model of the subsurface, but by observing the complex patterns and their organization within the image. By analyzing the patterns of the training image, we would be able to replicate the inner logics of our own perception.

Simpat and filtersim are two examples of such pattern-based algorithms. However, as mentioned in previous sections, each has their own limitations. Simpat, besides being extremely computational-intensive, lacks the proper link to human perception. By looking at the stationary training image of channels shown in Figure 1.6, we would not store all the patterns in our mind, but categorize them according to visual features. For example, we only see patterns representing the curvature of the channels, their orientations, the junctions dividing a channel into two streams, and so on. We classify different possible patterns, and only use those to gain a better understanding. This is similar to reading numbers or alphabets. We do not memorize different possible handwritings, but retain only a general shape/geometry (a prototype) of numbers or alphabets. The representation provided by the prototype is enough for a correct perceptual recognition in the future.

Filtersim follows the same methodological concept, but for different reasons. It tries to reduce the pattern database to make the computationally intensive simpat algorithm feasible. We adapt the same idea of pattern classification of filtersim, but for another reason: the similarity to human perception of the underlying process of acquiring, interpreting, selecting, and organizing visual information.

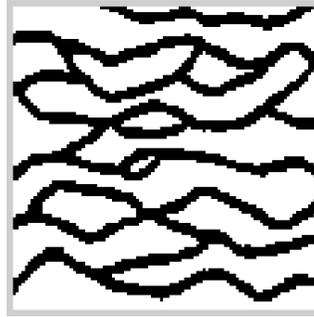


Figure 1.6: A conceptual training image depicting channel structures in the subsurface.

One of the issues of *filtersim*, similar to probabilistic and iterative MPS approaches, is the dependence on extra parameters or linear filters for pattern classification. This learning phase in *filtersim* is highly sensitive to the filters, and the recognition capability of the linear filters used in *filtersim* is poor and strongly depends on the patterns themselves. One needs an approach that improves on the limitations inherent to the use of spatial filters in *filtersim*; a unified method that does not depend on any filter specification. One solution would be to apply more filters on each pattern, and as a consequence, increase the dimensionality of the problem. While this approach is feasible with increasing CPU-speed, a more robust technique that can capture the inherent structure in the patterns, at the same time prescribing the simplest possible model, is sought. Such a model should be inspired from biology. Unlike *filtersim*, our visual system does not extract some linear features to differentiate the patterns, but it compares them to each other. One example of this comparison is the way humans quantify or evaluate beauty. Besides the cultural or evolutionary traits that define people's preferences, individually, evaluation is performed by comparing faces with each other. One face is beautiful when it is compared with other faces, or vice versa. Without a pool of subjects, this evaluation is not possible. In our mind, we basically find a distance that measures how similar one face is to another. This emergent structure obtained through similarity or dissimilarity measurements is what empowers us with such abilities.

Thus, to analyze and classify patterns, a simple distance-based framework that can model pattern complexity will be introduced. In this framework, the pairwise pattern similarities are used to formulate a mathematical representation of pattern variability. This distance-based model provides a unique set of tools that, while minimizing the need for model parameter selection, maximizes the capability of handling the inherent complexities

among patterns.

Learning/Simulating

In any geostatistical algorithm, there exists a learning phase and a simulation phase. In the learning phase, the spatial model of continuity is analyzed, and is later used in the simulation phase for generating stochastic Earth models. Both the learning phase and the simulation phase are interconnected. In other words, the method of learning dictates the method of simulation. If the learning is based on a probabilistic model, the simulation is performed according to the same probabilistic notion. For example, snesim algorithm generates a search tree in the learning phase, and uses it for simulation as well.

The learning phase demands different algorithmic complexities according to the spatial model. In kriging, the spatial model is mainly given through a variogram. It is a simple model expressing two-point continuity of random variables. The connection between the learning phase and simulation algorithm is the solution of kriging equations. The learning phase is manifested through the mathematical formulations that are derived for kriging. On the other hand, solving these equations for unknown samples constitutes the use of this learnt model for simulation. However, kriging relies on a simple variogram model. But, the training image is a very complex depiction of subsurface geological structures. Therefore, the learning complexity in multiple-point geostatistics can increase dramatically.

Despite the subjectivity-related issues brought forward by a complex MPS learning phase, the simulation phase should be “associative”. An associative simulation model is the one that directly links the training image and the learnt model with its simulating algorithm. In other words, the sole purpose of the simulation phase is to use the previously learnt model; no more subjectivity, assumptions, or adaptations should be allowed in this phase. For example, in the simulation phase of snesim algorithm, if the data event, which is extracted with the initial data template, is not found in the search tree, it will keep dropping the furthest nodes (inherently changing data template) until the combination is found. This act of dropping nodes is a subjective adaptation that the simulation phase of snesim takes to enable the retrieval of conditional probabilities. Without it, the algorithm would be unable to draw a value for the unsampled location. The subjectivity lies in the assumption of the algorithm on dropping the furthest nodes. What if those nodes are statistically more important than closer ones? What if nodes located at a specific spatial direction have greater significance for drawing the multiple-point conditional probabilities

than the one at another direction? And finally, what if the search tree would be better off with farther template nodes (i.e., at spatial locations with high nugget effect or high local noise)? This algorithmic assumption in simulation phase of *snesim* can thus be considered subjective. Likewise, in iterative approaches, the algorithmic decision on when to stop the iterations is a subjective one. These extra assumptions won't allow the simulation phase to be purely associative.

Therefore, an associative simulation phase should not introduce additional assumptions or tuning parameters, more than what the learning phase has. A good analogy would be a car, where the simulation phase corresponds to its engine, and the learning phase is the gas tank. One would expect the simulation engine to simply run on the fuel provided by the learning tank, no more injection of additional information, and no more dilution with subjectivism. The learning mathematical construct should be able to correctly describe the observed phenomena; that is, solely and precisely, it is expected to work as an all-pervasive tool. Pattern-based modeling provides us with a framework, where the simulation engine is directly interlinked with the learning: "learn patterns, then use patterns". Less subjectivity is required for the simulation phase. The pattern-based simulation that is introduced in this dissertation is a step towards an objective simulation algorithm; no more dropping of the nodes, and no more stopping criteria.

Another aspect that comes into play in this learning/simulation phase of MPS methodology is the quality of final realizations. One would expect the multiple-point information to be preserved within both small-scale and long-range structural patterns. It has been observed in previous MPS algorithms that statistics are not enough to capture the long-range continuity of subsurface features. Including longer-range statistics, besides increasing the computational time, reduces the uncertainty within the generated realizations. This is undesirable. One of the solutions on preserving the long-range spatial continuity in simulation is the multi-grid concept. According to this concept, both the training image and the simulation grids are modeled with sparser, but intrinsically larger, templates. The multi-grid approach is shown to provide better long-range reproduction of training image features. It is widely used in all MPS methods.

However, the multi-grid concept does not have a biological interpretation, nor a physical one. We know it works just because the simulation grid is populated using a larger template. There is no physical meaning or a theoretical background that supports such an approach. Moreover, the realizations in each multi-grid level contain many uninformed nodes. For

example, at the finest multi-grid level, every other node on the grid is uninformed. This leads to structures/geometries that can be easily perceived by human eye, but not so easily understood by the algorithm. For example, one may notice the channel structures even if half of the nodes on the grid are regularly uninformed, but the methodology may not be capable of accurately deciphering such geometrical relationships. We would propose an approach inspired by biology that can improve upon the pattern reproduction and provide more geologically realistic realizations.

According to the scale-space theory of vision, humans see the real-world objects at different scales. Each object is composed of different structures, and one needs a system that, without any prior assumption on its size, can capture and understand them. For example, the ability to see a small mosquito in the air, or a big truck on the road is the scale-space ability of humans to consider all scales simultaneously. There is no way for an MPS algorithm of knowing *a priori* the appropriate scale of the structures of the unknown training image. The reasonable approach is to follow the same biologically-inspired methodology, of considering the training image at multiple scales. We will introduce a technique, called multi-resolution, which considers the training image and the simulation grid representations at multiple scales/resolutions. This simulation framework is analogous to how humans would solve such a problem. For example, if one is provided with the channel training image of Figure 1.6, and is expected to generate realizations, one will start by sketching the big picture representation of channels and their pathways. When this initial coarse-resolution sketch is finished, one will switch to refining that initial sketch with more specific details; such as refining the boundaries of channels, or refining the sinuosity of them. Upon reaching the final results, one fine-tunes the geometrical shapes by comparing the patterns of the training image with the ones in the drawing. This process can be conceptualized as an ‘image evolution’, where an initial coarse sketch is gradually evolved to finer images. In order to comprehend such an evolution, an example on the way an artist paints a face can be envisioned. The artist starts from a rough sketch of the overall shape/outline of the head and the location of the nose, eyes, mouth, and ears. And it is only when this initial coarse-resolution image is obtained that the artist goes for finer drawings of the evocative features of the face; such as eyes, pupils, or eyebrows. Figure 1.7 shows an example of drawing an eye, with the steps of highlighting, shadowing, or other finer details that an artist would add in each level.

Therefore, the incorporation of distance-based pattern modeling and multi-resolution

simulation framework in the proposed MPS method will provide an instrument that mimics the biological/theoretical functioning of human front-end visual system.



Figure 1.7: A representation of different steps an sketch artist would draw an eye; from a coarse layout towards adding increasingly more details.

Simulating

The use of complex geological models to describe real physical phenomena is a new geo-statistical practice. In order to better understand the goals of such a task, we need to emphasize on the true rationale behind MPS simulations. Let us first recall on some of the commonly heard viewpoints:

- Why is this realization so similar to the training image?
- Why there is not much variability in your models?
- Why do models not adapt themselves to hard well data?

We believe that these are wrong questions to ask. Yet, engineers keep asking them, expressing their dogmatic expectations from MPS algorithms. We would like to go back to the basics, and clarify what is meant by simulation, what should be expected from the algorithm, and what is the purpose of MPS modeling.

The first expectation concerns the capability of the MPS algorithm to generate models that are substantially different than what the training image depicts. For example, the main concern in *simpat* algorithm was the similarity of the realizations to the training image. The *filtersim* algorithm tried to eliminate this concern by grouping patterns together and introducing a random pattern selection into the methodology.

All these issues, regarding similarity of realizations to the training image and the lack of variability between the realizations, are based on a subjective, and solely perceptual, evaluation of models. There is no mathematical basis (or a function) that can quantify this

variability and validate, or invalidate, these beliefs. Moreover, the fundamental question whether this ‘illusory sense of similarity’ is an undesirable outcome or something within expectations, should be analyzed.

First, let us clarify our expectations from an MPS algorithm. The modeler is to provide the algorithm with a training image as the spatial model that depicts the geological features of the subsurface. Without this model, the algorithm is incapable of generating any realizations. This spatial model should be the *only* input and source of knowledge that the algorithm is allowed to use. Thus, the algorithm must depend on this spatial model. If the multiple-point statistical properties of the spatial model follow a certain probabilistic distribution, then the realization should also follow the same distribution. This is what is expected from an MPS algorithm. A simple visual comparison of realizations, made by the engineer, should be completely avoided as a judgement criterion. One needs to look at the multiple-point statistics of realizations and the training image, and only then, can evaluate the performance of the algorithm. For example, if *snemsim* realizations show greater variability, or *filtersim* realizations provide larger uncertainty, it should not be a measure of good performance of those algorithms, but instead, it may be attributed to algorithmic internal inconsistencies (such as an unknown but unreliable multiple-point behavior, or ignorance of higher-order statistics), or deviation from that training image.

Thus, we believe that the MPS algorithm *must* honor the input statistics that are provided by the training image, in the same way that sequential Gaussian simulations honors the input variogram model. Any statistics beyond what the spatial model delivers is undesirable, and any drift beyond ergodic fluctuations should be avoided. The apparent similarities between the realizations and the training image could simply be attributed to retaining exceptionally higher multiple-point information from the training image. For example, instead of relying on 20-point statistics, one can simply get similar results with a 7-point statistics (depending on the training image). Therefore, any possible (visual) similarity between the realizations could be an effect of the way our brain perceives the patterns within the models. It could, on the other hand, be a desirable outcome of any MPS algorithm. Statistically speaking, if the realizations are stochastic in nature, that is their ensemble average is a constant field (for unconditional simulations), the models should not be evaluated based on their visual appeal, but based on the reproduction of the initial multiple-point information.

The other dogmatic expectation concerns the tendency of modelers to ask for a data-driven approach, instead of a TI-driven one. It should be stressed that a data-driven methodology is based on the analysis of all the data that characterize the subsurface geological system. According to this general definition, all MPS algorithms are data-driven; that is they try to incorporate all the system variables (training image, hard data, soft data, and user-input parameters) in order to generate Earth models. This is contrary to a knowledge-driven approach that directly describes the behavior of the geological system. So in a general definition, all MPS methods are data-driven. However, in here, a data-driven approach refers to a slightly different definition. A data-driven MPS method is the one that bases the simulation more on the hard and soft data, in comparison to a TI-driven approach which is dependent more on the training image.

Engineers expect MPS simulations to be a data-driven approach, and less of a TI-driven one. There is a great deal of implication with such an expectation. The problem arises when one uses the data (i.e. well data) to correct the spatial model (the training image). For example, if a training image represents channels flowing from east to west, but the data are coming from a geological setting with north-south channel directions, then the algorithm is expected to somehow ignore the spatial model and position the channels with a correct direction. One can clearly see the implications of such aspirations. First of all, geostatistical algorithms always need a spatial model which controls the entire process. Any geostatistical method, either implicitly or explicitly, follows a spatial model. The inverse-distance weighting is the model for one of the simplest spatial interpolation techniques. So, there is no escape from the spatial model, or the training image. Any modification from the given spatial model intrinsically means that the algorithm is adhering to another unknown spatial model. For example, changing the direction of channels is basically changing the training image. This is similar to going to a barbershop and expecting him to know a priori what kind of a haircut we want. No algorithm can objectively figure out the spatial model of the subsurface. We have seen examples of completely different realizations with the same spatial continuity assumptions (such as two realizations from a channel training image and SGS both having the same variogram and histogram). Therefore, if an MPS algorithm is somehow allowed to be only data-driven instead of TI-driven, it is definitely assuming some unknown spatial model for inference. This subjective certainty that is internally produced by the algorithm is undesirable, and can take the control away from the modeler. In addition, it will contradict the modeler's prior assumptions that were provided with the

model of spatial continuity. Any inconsistency between data and prior (training image) should be resolved before-hand, not fixed a-posteriori.

If there are evident structures or information within the data (hard or soft), it is the task of the modeler to interpret them. One should not expect the algorithm to accurately interpret the data, and provide models accordingly. Such an approach has dire consequences; such as, reduction of uncertainty due to internal assumptions of the algorithm about the spatial model, and the inability to control the outcome of simulation. Any interpretation, made by the algorithm from the input hard and soft data, is unreliable and can lead to unrealistic models. If an engineer or a geologist is incapable of such an interpretation, how could we expect the algorithm that is formulated by the same person to correctly understand the phenomena.

Therefore, we should distinguish different aspects of modeling before setting expectations. If there are uncertainties in the geological settings of the subsurface under study, then it is our task to provide several training images that can capture the inherent variability in the subsurface. We cannot expect everything to be sharply defined, and definitely, cannot expect the algorithm to define them for us. As an analogy, let us assume that we would like to create several three dimensional palm trees. The modeler thus provides an image of a palm tree. Clearly, the algorithm would be incapable of creating maple trees, which have different structures than a palm tree. If maple trees are also deemed relevant for the task at hand, the modeler should provide them as the training image. So there are two aspects in modeling; one regarding the uncertainty in our input training image, and the other regarding the MPS simulation algorithm. In here, we are only focusing on the latter, and believe that these two aspects should never be misinterpreted. Therefore, the sole purpose of the MPS method should be to exactly follow the spatial model that is provided to it, and exclude any additional algorithmic assumptions.

So far, we have explained the misconceptions regarding MPS methodologies in general, and advocated a specific simulation framework. The main focus, however, has been on stationary modeling, where a decision of stationarity was required for the training image. Stationarity refers to a situation where the statistical properties of the model are independent of the spatial location. In other words, the patterns of the training image are being repeated everywhere on the grid. In a non-stationary model, on the other hand, the statistical properties change from one location to the other. For example, a simple drift in the

proportions of the random variable from one side of a grid to the other is considered a non-stationary behavior. Recently, there has been much attention to modeling non-stationary geological phenomena.

In traditional multiple-point modeling (MPS), most ways of dealing with non-stationarity are either based on defining stationary regions or providing auxiliary non-stationary probability fields. In the region-based approach, the non-stationary image is subdivided into stationary regions. Each region is then assigned to its own stationary training image, and simulation is sequentially performed for each region. In the other approach, an auxiliary variable, such as a probability field, is provided to guide non-stationarity. For example, if the orientation of the channels within the field is spatially changing, then the auxiliary variable is indicating the desired direction of the channels. The simulation algorithm will start with a stationary training image of channels, and rotate them accordingly on the realization grid. These approaches have one thing in common: they both expect the modeler to deterministically state the nature of the non-stationarity within each simulated realization. One can realize that these methods are *forcing* the algorithm to behave in an expected way. This not only reduces uncertainty, i.e. by designating fixed region boundaries or a fixed auxiliary variable, but also takes away simplicity of MPS modeling. In addition, only those non-stationary models that can either be subdivided into stationary regions, or characterized with an auxiliary variable, can be simulated. Furthermore, even if the non-stationarity is manageable by these two approaches, there are no tools, nor a simple solution, to construct these auxiliary variables or regions.

Similar to previous arguments on what is expected from an MPS algorithm, non-stationarity should pose no difference. The simple objective is to provide a non-stationary training image to the algorithm and get back several non-stationary realizations, without the need for any additional source of information. A training image is depicting the geological settings of the subsurface, whether it is stationary or not, and the MPS algorithm should be capable of handling such a simulation. The region-based and probability field approaches are much similar to a training image generator tool, where one forces the algorithm to produce the desired non-stationary effects. However, the purpose of simulation is not to generate training images, but realizations. Therefore, in this dissertation, we will provide the necessary tools to handle non-stationary training images, and unlike previous approaches, there would be no need for auxiliary data, nor will the MPS method be forced towards *decoration* instead of *simulation*.

Final Remarks

We hope to demonstrate that the framework in this dissertation is less subjective than other MPS modeling approaches. The rigorous theory of pattern-based modeling makes it rather difficult to insert modeler's subjective expectations or algorithmic assumptions in the model; and therefore, the results are closer to the objective geological reality. The proposed distance-based pattern modeling algorithm, called DisPAT, is greatly inspired from biology and our own visual system. The purpose of the proposed approach can be summarized as follows:

1. Provide a simple pattern-based modeling framework.
2. Improve pattern reproduction in generated realizations.
3. Decrease subjective algorithmic assumptions.
4. Decrease modeler's subjective assumptions by reducing the user-input parameters.
5. Provide significant speed-up in MPS modeling.

But all these ideas, all these cognitive approaches, all this distance-based framework, all this objective modeling framework, and these elaborate parameter estimations, everything is simply a step towards a general and unifying framework for multiple-point geostatistical modeling. We are in search of a framework upon which an *ideal algorithm* is obtained. This ideal MPS algorithm is the one that merely requires the training image as input, and without any additional assumption or parameters, generates statistically correct Earth models.

If scientific geostatistical progress is envisioned as a mountain, the ideal algorithm lies at the summit. The previous approaches on multi-point geostatistical modeling were simply navigating around the base of the mountain in search of the highest foothills, trying every measure to get visually-appealing results. Instead, in this dissertation, we are trying to climb the mountain. We are trying to take some steps upward towards the summit of the mountain. Even though we might be far away from the summit at the end, but at least, we have provided the necessary framework to climb up. It may not be the most optimal route, but it is an upward road where we can gain new and wider views. Overcoming the obstacles on this adventurous way up will hopefully guide us to better MPS algorithms.

1.6 Dissertation Outline

The proposed distance-based pattern modeling framework will be explained entirely, on an algorithmic level, a computational level, and a software implementation level. The thesis is organized as follows:

- Chapter 2, *Pattern Modeling with Distances*, introduces the concepts behind distance-based pattern modeling. A review of previous distance-based application will be provided. It proceeds with the notations and definitions of multiple-point geostatistical terms. The techniques for reducing the inherent complexity of patterns using multi-dimensional scaling will be given next. We will demonstrate how patterns can be represented as points in a Cartesian space. Subsequently, kernel methods, as non-linear transformation functions, will be described. The advantages of kernel transformations for linearizing the data will be explained. Finally, classification algorithms such as k -means, but in higher dimensional kernel space, are described with experimental examples.
- Chapter 3, *Unconditional MPS Simulation*, provides the general simulation framework for MPS modeling. The previous concepts of distances and kernel transformations will be used to offer a powerful pattern classification framework that not only leads to better simulation results, but also brings robustness and accuracy. Some examples will be provided for clustering quality/accuracy, and the results are compared with the filtersim algorithm. Finally, model variability of the proposed pattern-based algorithm will be analyzed. Internal inconsistency of different algorithms will be quantified and put for comparison.
- Chapter 4, *Multi-Scale MPS Simulation*, introduces a technique for ensuring the long-range continuity of the spatial structures in generated realizations. Previous techniques for multi-scale simulation will be evaluated within the DisPAT framework, and some extensions will be provided. Next, another multi-scale pattern modeling approach that is inspired from human perception is introduced. The technique, called multi-resolution approach, will be compared with the previous methodology both visually and mathematically. Finally, several examples of binary, categorical, and continuous variables will be provided with comparisons.

- Chapter 5, *Parameter-free Learning*, attempts to reduce the number of user-set parameters. The traditional cumbersome trial-and-error procedures will be eliminated by introducing algorithms that can objectively find the most optimal parameter. Two main parameters are (1) the size of the template or the patterns of the training image, and (2) the number of clusters in classification algorithms. These parameters can significantly affect the final realizations both in terms of quality and computations. The proposed objective assessment of these parameters will aim in reducing the dependency on modeler's subjective decisions.
- Chapter 6, *Data Integration*, will provide the techniques needed for integrating additional sources of information into geostatistical modeling. Two different source of information, hard data that come from wells and soft data that come from seismic, will be investigated. The conditioning algorithms for hard and soft data will be explained in both multi-grid and multi-resolution setting. Many examples will be provided to show the capabilities of the proposed approach in conditioning the models to available data.
- Chapter 7, *Non-Stationarity*, will review the previous methods for modeling non-stationary training images, and clarifies their inherent issues. Subsequently, three different approaches will be introduced. All the techniques are based on a simple notion of embedding spatial components into pattern modeling. The methods demonstrate the applicability of non-stationary modeling for different training images and geological structures. Several examples and sensitivity analysis will be performed for each technique.
- Chapter 8, *Conclusion*, summarizes the work accomplished in this dissertation. In addition, it will provide some insights into future works for pattern-based modeling, including but not limited to different simulation frameworks, or parameter-free learning approaches. *Veni, vidi, vici!*

Chapter 2

Pattern Modeling with Distances

It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience

ALBERT EINSTEIN (1879 - 1955)

How would you describe,

- The breaking of a window?
- Spread of oil spill in ocean?
- Impact of earthquake in an urban city?
- River flow through a valley?
- Face tracking in a motion video?
- Flow through porous media?
- The texture of a stone?

There is no simple answer for all these questions. A solution that can solve all these problems cannot exist. There are many details, descriptions, properties, and features associated with each of them; and choosing among them would depend on the objective and the task at hand. Therefore, modeling each of these phenomena needs separate attention. For example, if one seeks to obtain a mathematical model for flow of water down the valley, one could use a

numerical model that solves the physical laws governing the flow (such as gravity, friction, accumulation, dip). This allows the model to replicate the real phenomenon. However, finding good parameters for the model is as hard as building the model itself. For example, in modeling the spread of oil in the ocean, not only the physical model of concentration and transport phenomenon are important, but other parameters such as the initial conditions, volume of oil, and viscosity need to be set a priori for successful modeling.

In addition, to build a model one needs to make some decisions, either explicitly or implicitly, on how to describe it. There is no universal way to make these choices. Basically, the *best model* does not exist, therefore, the fundamental principle underlying our decisions should be ‘simplicity’. This is a version of Occam’s Razor, that the simplest model that explains data is the one to be preferred (in terms of both the form of the model, and the values of the parameters) (Gershenfeld, 1999). The same argument holds in the context of pattern modeling. How would one model patterns seen through data? One would demand an algorithm that can capture the inherent structure in the data, while at the same time, prescribe the simplest model. In this chapter, a distance-based modeling framework that can model the complexity of patterns in a simple paradigm will be introduced. This framework provides a unique set of mathematical tools, which minimize the necessity to decide on model parameters, and maximize the capabilities of handling inherent complexities among patterns.

One of the new tools in modeling, in general, are the distance-based methods (Caers and Park, 2008; Caers, 2008a; Park et al., 2008b; Suzuki and Caers, 2008; Suzuki et al., 2008). While these techniques have been used so far to model variability between realizations (Scheidt and Caers, 2009b,a), the same techniques are used here at the most fundamental level: the pattern level (patterns within a training image). The first step towards analyzing this set of patterns is to endow it with a metric, a numerical measure of the difference between any two patterns. The idea behind the distance-based approach is that by applying a metric, called distance function, to any pair of patterns, a measure of their similarity is obtained. It is a function that requires two patterns as input and returns a scalar value such that similar patterns result in a small value, and different patterns result in a larger value. Having obtained the pair-wise distances between all patterns within a training image, intuitively speaking, one can gain tremendous insight into the underlying structure of variability between the patterns. A technique, called multi-dimensional scaling, can reveal these hidden structures by knowledge of pair-wise distances only.

What follows in this chapter is, first, a set of notations and their definitions to help better understand the proposed methodologies. Next, an overview of distance-based methods will be provided with strong examples of its applications in modeling subsurface flow and quantifying uncertainty. Afterwards, the concepts related to distance-based pattern modeling (such as multi-dimensional scaling, pattern classification, and kernel methods) will be thoroughly explained. Finally, the chapter concludes with an overview of the advantages of the proposed methodology in terms of delivering a simple, yet powerful, mathematical framework.

2.1 Notations and Terminology

Terminology is key to understanding the principles of the proposed methodology. The terms in this dissertation are consistent with the relevant literatures on multiple-point geostatistics (MPS). In this section, some required distance-based notations and MPS terminologies are provided.

2.1.1 Training image

A training image, according to Journel and Zhang (2006), is a conceptual description of a random process. In reservoir characterization, it is a three-dimensional image describing the patterns of variability of the subsurface. The training image is purely a conceptual geological model of the subsurface. For example, a training image is used solely to represent the geometrical characteristics of facies such as the sinuosity of channels in a fluvial reservoir, or the complex spatial relationships among multiple subsurface features. The training image is thus not conditioned to any data, and does not contain any location-specific information. In traditional geostatistical techniques, these structural and spatial characteristics of the random variables were defined by some kriging process built on the variogram model. The fundamental difference is that training images provide statistics at multiple points and within multiple scales, as opposed to the traditional variogram-based methods that rely merely on 2-point statistics.

Multiple-point geostatistical approaches (MPS) do not try to reproduce the training image itself, but they consist of extracting statistics of the variables from the training image statistics at multiple locations, and subsequently, using them to reproduce similar features

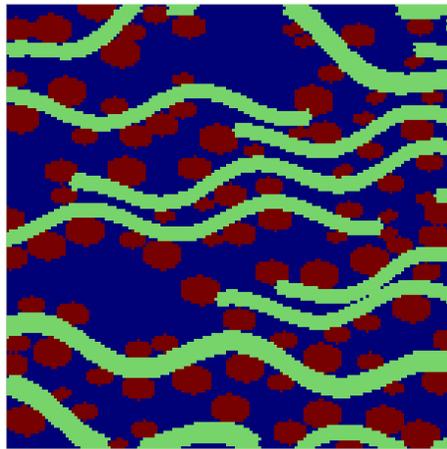
to the training image. Training images have a great impact on what patterns are reproduced. They are analogous to a set of face images used, for example, in a face recognition algorithm. However, the ultimate goal is using the training images to generate visually-appealing realizations or models of subsurface that honor all the available information at hand.

Training images can come from several different sources such as interpreted outcrop photographs or simply a geologist's sketch (properly digitized and extended to 3D), or even an unconditional object-based or process-based simulation. Different sources that a modeler can use in order to construct training images of a hydrocarbon reservoir can be categorized as follows:

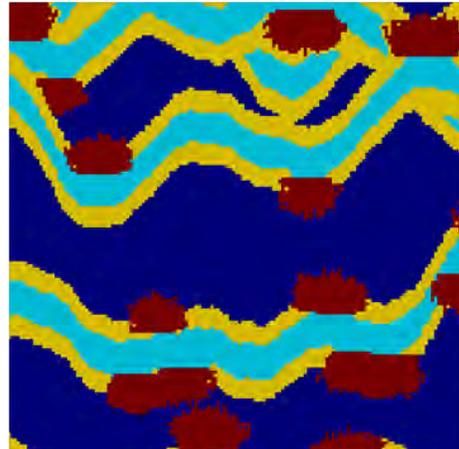
- *Geological analogs*, such as outcrops, modern depositional environments that display patterns deemed present in the reservoir under study.
- *Sequence stratigraphy studies*, used to identify different depositional system tracts, leading to different training images (Van Wagoner et al., 1990).
- *Object-based algorithms* (Haldorsen and Damsleth, 1990; Holden et al., 1998; Lantuejoul, 2002). Object-based algorithms have the capability to generate realistic shapes for geological bodies and their spatial distribution. Even though they are hard to condition to dense well locations, object-based algorithms are flexible and well suited to create non-conditional simulations that can be used as training images.
- *Process-based models* (Harbaugh and Bonham-Carter, 1970), which are generated by forward modeling the geological processes through the physical laws that govern the transportation of source materials, the deposition and compaction of rocks, their erosion, redeposition, etc. Similar to object-based models, conditioning process-based models to dense observations is difficult. But process-based realizations can be used as starting models to build a conceptual geological depositional model fine-tuned to the actual deposit under study.

Some 2D and 3D training image examples are shown in Figure 2.1.

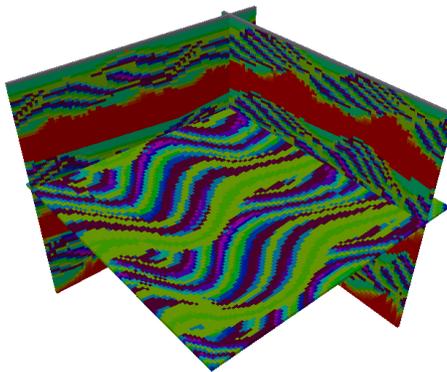
Define $ti(\mathbf{u})$ as a value of the training image \mathbf{ti} where $\mathbf{u} \in \mathbb{G}_{ti}$ and \mathbb{G}_{ti} is the rectangular Cartesian grid discretizing the training image. The training image of interest, for now is a binary (e.g. sand/shale) system. So, an indicator notation for $ti(\mathbf{u})$ is defined as follows:



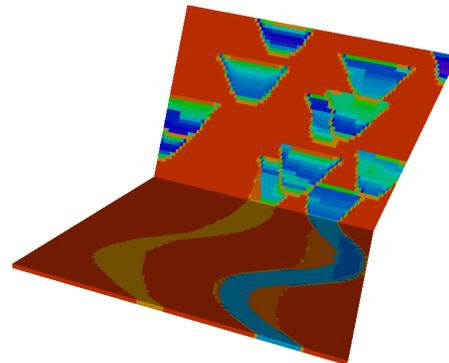
(a) 2-D Training Image



(b) 2-D Training Image



(c) 3-D Training Image



(d) 3-D Training Image

Figure 2.1: Training image examples for a 2 dimensional or 3 dimensional simulations, depicting different channel systems.

$$ti(\mathbf{u}) = \begin{cases} 0 & \text{if at } \mathbf{u}, ti \text{ containing shale} \\ 1 & \text{if at } \mathbf{u}, ti \text{ containing sand} \end{cases} \quad (2.1)$$

For some future algorithms, it is advantageous to also use a more traditional matrix notation to denote a training image and its elements. For example, a 2D training image of size $M \times N$, can be represented as follows:

$$\mathbf{TI} = \begin{bmatrix} ti_{0,0} & ti_{0,1} & \cdots & ti_{0,N-1} \\ ti_{1,0} & ti_{1,1} & \cdots & ti_{1,N-1} \\ \vdots & \vdots & & \vdots \\ ti_{M-1,0} & ti_{M-1,1} & \cdots & ti_{M-1,N-1} \end{bmatrix} \quad (2.2)$$

where each $ti_{i,j}$ represent the training image value at location $\mathbf{u} = (i, j)$, or simply, $ti_{i,j} = \{ti(\mathbf{u}) \mid \mathbf{u} = (i, j)\}$. In Equation 2.2, the right side is a matrix of real numbers. Each element of this matrix is called a node or a pixel. We will use this representation for more involved mathematical operations on training images.

2.1.2 Pattern

A training image, which consists of geometrical shapes and structures, can be assumed to be fabricated by a set of patterns tiled next to each other. For example, in a 2D training image, a pattern is represented by a small rectangular domain (a template) overlain at a specific location on the training image grid. The concept of a pattern and a template is depicted in Figure 2.2. Accordingly, we define $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ as a pattern of the training image; a specific multiple-point vector of $\mathbf{ti}(\mathbf{u})$ values within a template \mathbf{T} centered at node \mathbf{u} , i.e., $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ is the vector:

$$\mathbf{ti}_{\mathbf{T}}(\mathbf{u}) = \{ti(\mathbf{u} + \mathbf{h}_1), ti(\mathbf{u} + \mathbf{h}_2), \dots, ti(\mathbf{u} + \mathbf{h}_\alpha), \dots, ti(\mathbf{u} + \mathbf{h}_{n_T})\} \quad (2.3)$$

Where \mathbf{h}_α vectors define the geometry of the nodes of template \mathbf{T} and $\alpha = 1, \dots, n_T$. Here, n_T represents the size of the template. For example, a two dimensional template of size $M_{\mathbf{T}} \times N_{\mathbf{T}}$ has $n_T = M_{\mathbf{T}} \times N_{\mathbf{T}}$ nodes. Size of the template is one of the key parameters in pattern-based modeling. Therefore, selecting a template size that can capture the inherent patterns within a training image is of crucial importance to MPS modeling process.

Processing of the training image \mathbf{ti} is initiated by scanning the training image using

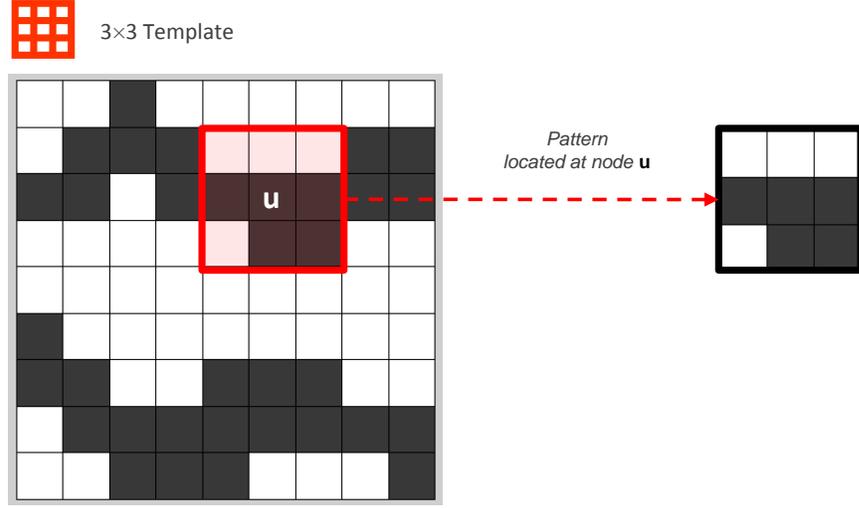


Figure 2.2: (Left) Training image, t_i , and a template \mathbf{T} on location \mathbf{u} , (Right) the pattern, $\mathbf{t}_{\mathbf{T}}(\mathbf{u})$ with template \mathbf{T}

a template \mathbf{T} and storing the corresponding multiple-point $\mathbf{t}_{\mathbf{T}}(\mathbf{u})$ vectors in a database. Each such $\mathbf{t}_{\mathbf{T}}(\mathbf{u})$ is called a pattern of the training image and the database is called the pattern database and is denoted by $\mathbf{patdb}_{\mathbf{T}}$. Once the patterns are stored in the pattern database, they are considered to be location independent, i.e. the database does not store the location $\mathbf{u} \in \mathbb{G}_{t_i}$ of a pattern; only the content of the pattern is stored. Hence, k^{th} pattern can be denoted as follows:

$$\mathbf{pat}_{\mathbf{T}}^k = \left\{ \mathit{pat}_{\mathbf{T}}^k(\mathbf{h}_1), \mathit{pat}_{\mathbf{T}}^k(\mathbf{h}_2), \dots, \mathit{pat}_{\mathbf{T}}^k(\mathbf{h}_\alpha), \dots, \mathit{pat}_{\mathbf{T}}^k(\mathbf{h}_{n_T}) \right\} \quad (2.4)$$

where $\{k = 1, \dots, n_{Pat_{\mathbf{T}}}\}$, and all $n_{Pat_{\mathbf{T}}}$ patterns are defined on the same template \mathbf{T} . Here, $n_{Pat_{\mathbf{T}}}$ is the number of patterns in the pattern database. For example, a 2D training image of size $M \times N$, with a template \mathbf{T} of size $M_{\mathbf{T}} \times N_{\mathbf{T}}$ has $n_{Pat_{\mathbf{T}}} = (M + M_{\mathbf{T}} - 1) \times (N + N_{\mathbf{T}} - 1)$ patterns in the pattern database $\mathbf{patdb}_{\mathbf{T}}$.

2.1.3 Dissimilarity distance function

Dissimilarity distance function is a measure of similarity. For example, it is a function which calculates the dissimilarity between a pair of patterns. In this context, the function indicates how far pairs of patterns are from each other in pattern space, similar to Euclidean distances between two points in a Cartesian space. Therefore, this dissimilarity distance

can provide a measure on how much two patterns look alike; i.e. the smaller dissimilarity between two patterns indicates more similarity between the patterns in a visual sense. In a space representing the patterns, similar patterns would be clustered near each other, and the dissimilar ones would be located far away from each other. This parametrization obtained by the *dissimilarity distance function* has many advantages; such as, in visualization, classification, or statistical inferences of data. In this chapter, we will exploit this functionality to differentiate the patterns, which in turn, results in accurate clustering or grouping of similar patterns with each other.

The concept of *dissimilarity distance function* was first introduced by Arpat (2005); Suzuki and Caers (2006). Arpat (2005) defined this function within the context of pattern recognition and template matching, and introduced a multiple-point geostatistical algorithm called *simpat*. The core of his simulation algorithm was to use the dissimilarity distance function to search for the most similar pattern, within the pattern database, to an input data event. Later, Suzuki and Caers (2006) used the Hausdorff distance (Dubuisson and Jain, 1994) as a measure of dissimilarity between subsurface realizations, and used it for history-matching production data. The function provided a parametrization in the model space, which facilitated the optimization and resulted in an effective history-matching process.

In principle, the dissimilarity distance measure should have two properties:

1. It should have a large discriminatory power.
2. Its value should increase with the amount of difference between the two patterns.

One simple example of a distance function is the Euclidean distance. The definition of this function is provided below:

Euclidean distance: Among all distance metrics, the Euclidean distance is the most commonly used one due to its simplicity, as well as its convenient mathematical properties. Let $\mathbf{pat}_T^m(\mathbf{u})$, $\mathbf{pat}_T^n(\mathbf{u})$ be two patterns from the pattern database \mathbf{patdb}_T . The Euclidean distance $d_E(\mathbf{x}, \mathbf{y})$ is given by:

$$d_E \langle \mathbf{pat}_T^m(\mathbf{u}), \mathbf{pat}_T^n(\mathbf{u}) \rangle = \sum_{\alpha=1}^{n_T} (\mathit{pat}_T^m(\mathbf{h}_\alpha) - \mathit{pat}_T^n(\mathbf{h}_\alpha))^2 \quad (2.5)$$

As seen in formula above, the Euclidean distance is only a summation of the pixel-wise differences between two patterns. However, this distance function may not provide much

discriminative power for complex patterns. Another distance function called *Proximity Transform* has been used as a dissimilarity distance function (see Section 3.1.4).

2.1.4 Dissimilarity distance Matrix

One important distance-based definition that will be used a lot in this dissertation is that of a dissimilarity distance matrix. Having a pattern database \mathbf{patdb}_T of a training image, the dissimilarity distance matrix can be obtained by calculating the pair-wise distances between all available patterns. If we have n_{Pat_T} patterns in a training image, dissimilarity matrix D will be a $n_{Pat_T} \times n_{Pat_T}$ size matrix, where each element d_{ij} of matrix D will represent the distance between i^{th} and j^{th} pattern. This distance is calculated using a dissimilarity distance function. According to this definition, the dissimilarity matrix should be symmetric with zero-diagonal elements.

2.1.5 MDS Space

MDS space is the space of models or patterns that is obtained from the dissimilarity distance matrix. It is a lower-dimensional representation of data in a Cartesian space, called MDS space. Section 2.3 explains the mathematical details of this procedure. For clarity, some notations relating to the MDS space is provided below:

- d : MDS dimension ($d \leq n_T$).
- \mathbb{R}^d : d -dimensional MDS space.
- \mathbf{x}_k : location of each pattern in a d -dimensional row vector in MDS space, as follows:

$$\mathbf{x}_k = \{x_{k1}, x_{k2}, \dots, x_{kd}\}, \quad \mathbf{x} \in \mathbb{R}^d \quad (2.6)$$

- X_d : A set of $N(= n_{Pat_T})$ points is represented by matrix $X = \{\mathbf{x}_k \mid k = 1, 2, \dots, N\}$, a $N \times d$ matrix where each row represents the coordinates of points in MDS space.

2.2 Distance-based Methods Overview

Distance-based methods have provided a new avenue in uncertainty modeling and history-matching. The concept of metric space uncertainty modeling was first introduced by Caers

(2008a). A metric space is a space that has no axis, origin, or direction. It is only defined by distances. So points in this space are purely represented by their distances to each other and there are no coordinates assigned to them. According to Caers (2010), the complexity and dimensionality of the input data and the models are far greater than the complexity and dimensionality of the output responses or final decisions. Therefore, the uncertainty of a set of Earth models can be represented in a simpler way by the use of metric space. Reducing the inherent complexity of a process is thus of utmost importance, and can yield to effective and robust algorithms for uncertainty quantification and modeling.

First, Suzuki and Caers (2008) used distance-based methods in the context of inverse modeling by parameterizing the prior model. However, instead of parameterizing the complex architectures by a set of model parameters, a metric space that accommodates a large set of realizations was defined. It was demonstrated that the inverse solutions could be efficiently found in this metric space.

Scheidt and Caers (2009b) used the same concept to represent spatial uncertainty. They used this tool to select a subset of geostatistical realizations, containing similar properties to a larger set, by parameterizing the spatial uncertainty of the larger set. They were then able to effectively quantify uncertainty and obtain the P10, P50, and P90 quantiles of model responses with much fewer realizations. Additionally, Scheidt and Caers (2009a) applied the uncertainty modeling concept to an architecturally complex deepwater turbidite offshore reservoir, having large uncertainty in the type of depositional system. They showed that a subset of reservoir models can reflect the same uncertainty in flow responses as the full set. Moreover, it provided a simple framework for sensitivity analysis of model responses with respect to input depositional parameters (Scheidt et al., 2008).

Park et al. (2008a) applied the distance-based method for the optimization algorithm of ensemble Kalman filter (Evensen, 1994). They explicitly visualized the process of updating models by the projections of metric space to a two or three-dimensional MDS space. Moreover, in the context of history-matching, by assuming that oil and gas productions are mainly controlled by the connectivity between injectors and producers, they defined a connectivity distance to not only select a representative set of models from all realizations, but also generate additional models from the same prior distribution (Park et al., 2008b; Caers, 2008b).

For a detailed summary of all these techniques please refer to Caers et al. (2010). In this chapter, the distance-based methodology will be applied in the most fundamental level

of patterns. The following sections will introduce the mathematical concepts and challenges related to distance-based pattern modeling. Afterwards, Chapter 3 introduces the stochastic simulation of patterns within a distance-based framework used for the proposed MPS algorithm.

2.3 Multi-dimensional scaling

2.3.1 Theory

In order to obtain a mathematical framework for statistical modeling, we need to use a distance function to map the patterns as a set of points in Cartesian space. Multi-dimensional scaling (MDS) allows gaining insight in the complex relationship among patterns by providing a geometrical representation of their similarities (Cox and Cox, 2001). It places patterns as points in a multi-dimensional space where the dissimilarity between the patterns is related to the distances of the corresponding points in that space. Mathematically speaking, multi-dimensional scaling (MDS), graphically portrays the dissimilarity distance matrix with a configuration of points in a Euclidean space \mathbb{R}^d (Borg and Groenen, 2005). This dissimilarity matrix is calculated using a dissimilarity distance function such as Euclidean distance function, or a more elegant one being proximity transform distance function (Russ, 2002). MDS maps the high-dimensional patterns in a lower-dimensional MDS space such that the distances between points are as close as possible to the original pattern similarities.

In mathematical term, the goal of MDS is to map the objects $i = 1, \dots, N$ to points $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$ in such a way that the given dissimilarities $d_{i,j}$ are well-approximated by the distances $\|\mathbf{x}_i - \mathbf{x}_j\|$. This can be shown as follows:

$$X_{n_T} \in \mathbb{R}^{N \times n_T} \mapsto X_m \in \mathbb{R}^{N \times d} \quad \text{s.t.} \quad d(\mathbf{x}_i, \mathbf{x}_j) \cong \sqrt{(\mathbf{x}_{i,m} - \mathbf{x}_{j,m})^\top (\mathbf{x}_{i,m} - \mathbf{x}_{j,m})} \quad (2.7)$$

where, X_d is defined in Section 2.1.5. Here, as a reminder, N is the number of different patterns (objects) in \mathbf{patdb}_T , and the dissimilarity between pattern i and j is denoted by d_{ij} . The resulting coordinates are gathered in an $N \times d$ matrix X_d , where d is the dimensionality of the MDS solution. Thus, row i from X_d gives the point coordinates, representing pattern i , in dimensionality-reduced MDS space.

MDS is simply done by an eigenvalue decomposition, and only retaining an *optimal* number of largest positive eigenvalues, as shown in Equations 2.8 to 2.12. *Optimal* number of eigenvalues means large enough to capture the variation of patterns in metric space but small enough to reduce the dimensionality. The first step in MDS mapping is to center the data represented in the metric space through the distance matrix D . We define matrix A , where (i, j) -th element is calculated by:

$$a_{ij} = -\frac{1}{2}d_{ij}^2 \quad (2.8)$$

Then centering the distance matrix is performed as follows:

$$B = HAH \quad (2.9)$$

where, H represents the centering matrix:

$$H = I - \frac{1}{n_T} \mathbf{1}\mathbf{1}^\top \quad \in \mathbb{R}^{n_T \times n_T} \quad (2.10)$$

with I the identity matrix and $\mathbf{1}$ a column vector of n_T ones ($\mathbf{1} = [1 \ 1 \ \dots \ 1]^\top \in \mathbb{R}^{n_T \times 1}$). Next, the eigenvalue decomposition of B is:

$$B = V_B \Lambda_B V_B^\top \quad (2.11)$$

where, V_B denotes the collection of eigenvectors of B and Λ_B the diagonal matrix of eigenvalues of B . In order to reduce the dimensionality of patterns from $n_T \rightarrow d$, we retain the d largest eigenvalues and the corresponding eigenvectors, and construct a small eigenvalue matrix $\Lambda_{B,d}$ and eigenvector matrix $V_{B,d}$. The projection of the metric space of models X_d is obtained by:

$$X_d = V_{B,d} \Lambda_{B,d}^{1/2} \quad (2.12)$$

As can be noted in equations above, metric multi dimensional scaling (MDS) is similar to principal component analysis (PCA) in the context of dimensionality reduction. When the data are shown through a distance matrix, MDS creates a new configuration of points by only retaining the first few dimensions of those points. Thus, the application of MDS with formulations shown above provides an identical result to PCA, up to a change in sign.

2.3.2 Examples

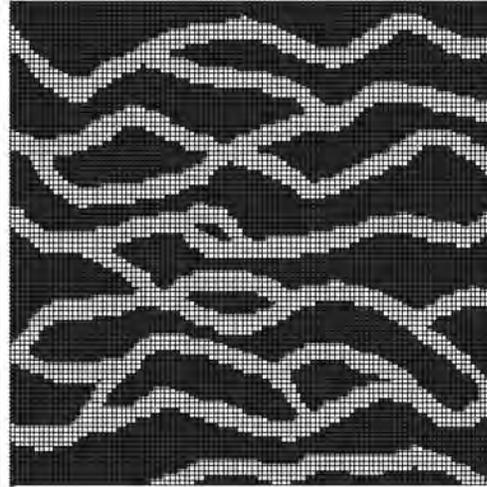
An example application of classical MDS is represented in Figure 2.3. For illustration purposes, a 2D Euclidean space, \mathbb{R}^2 , was chosen for mapping all the patterns of a training image. Assuming the distances to be a good representation of the dissimilarities between patterns, points close to each other in 2D Euclidean space coincide with similar patterns. This can be clearly observed in Figure 2.3.

One of the main advantages put forward by using distance-based methods is the ability of mapping a pattern as a point in a lower-dimensional space (MDS space). Working in MDS space gives the advantage of working with the same high-dimensional patterns, but in a lower-dimensional system. For example, by mapping a 9×9 pattern to a 9-dimensional MDS space, a reduction factor of 9 in the dimensionality of the data is obtained. In order to analyze the dimensionality reduction, we can look at the eigenvalues obtained within MDS algorithm for the training image shown in Figure 2.3. The eigenvalues are plotted in descending order in Figure 2.4. It is observed that almost all the variability between the patterns can be represented by the first nine dimensions. In Section 2.3.3, a general method on how to select this dimensionality will be introduced.

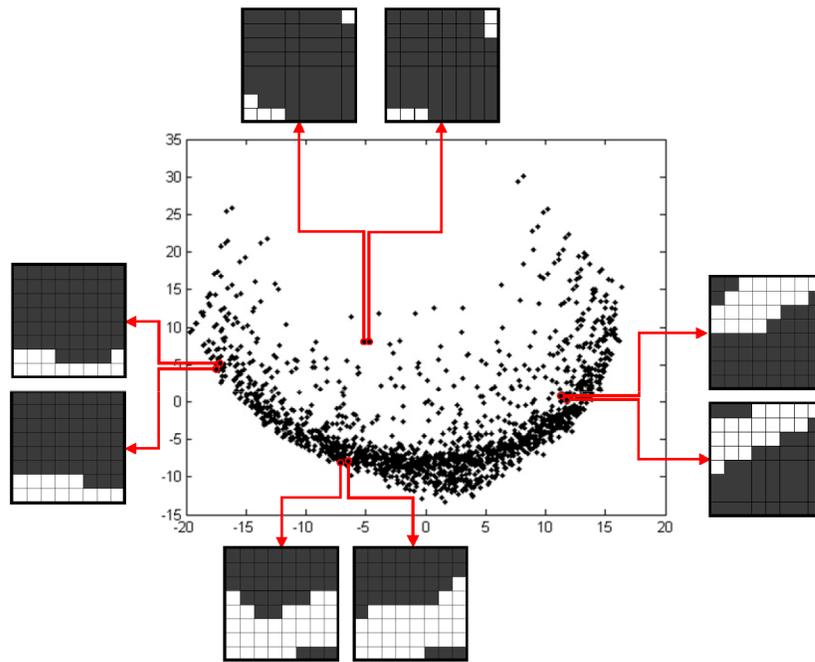
One of the challenges that might make this method inapplicable is the computational cost associated with the MDS mapping. Improving the multi-dimensional scaling algorithm is essential for achieving reasonable run-times in MPS simulations. Moreover, the huge number of patterns in a 3D training image renders the MDS algorithm extremely memory-demanding. The matrix calculations involved in this mapping, which are at least quadratic in nature, have an adverse effect on the computational efficiency of MPS methods. Therefore, the number of analyzed patterns is restricted by the high computational complexity of MDS. In order to make metric MDS applicable to large pattern databases, a new sequential metric MDS method, called *SEQ*-MDS, with a low computational complexity is introduced in Section A.1 of Appendix A.

2.3.3 Scaling dimension

A pattern can be seen as a high-dimensional object. For example, a pattern template of $9 \times 9 \times 5$ in a three-dimensional training image has 405 dimensions. MDS is therefore a distance-preserving dimensionality reduction technique which can further speed up the subsequent mathematical computations. This reduction in dimensionality is crucial for



(a) Training Image



(b) 2-d MDS space

Figure 2.3: (a) Training Image and (b) MDS sample result using an 8×8 pattern template. Any two patterns that are similar map as two close points in Euclidean space.

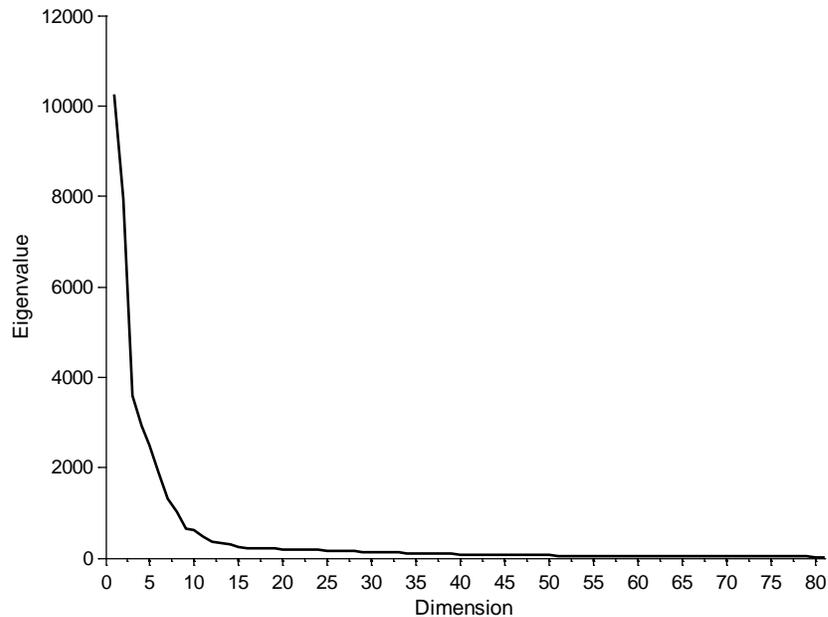


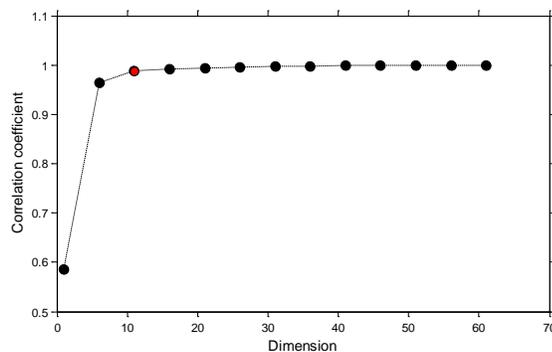
Figure 2.4: Eigenvalues obtained from mapping to MDS space. A dimensionality reduction to 9, seen by the elbow of this figure, can reasonably capture the complexity between the patterns.

analyzing and revealing the genuine structures hidden in high-dimensional data, such as patterns.

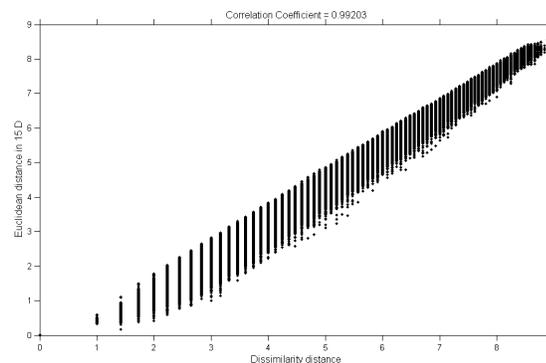
An important issue in MDS mapping is the choice on the number of dimensions for the scaling solution (e.g. two dimensions as seen in Figure 2.3). A configuration with a high number of dimensions achieves very small errors, but it can hurt computational time. On the other hand, a solution with too few dimensions might not reveal enough of the structures in the data. The principle of parsimony advocates obtaining the maximum dimensionality reduction while having minimum representation error. One needs to find an automatic procedure that can compromise between these two extremes and decide on a *reasonable* number of coordinates to be retained.

First attempt on a methodology for selecting a *reasonable* number of dimensions for MDS space was made by Honarkhah and Caers (2008). In that methodology the dimensionality was selected through the correlation between the inter-point distances in the MDS space and their corresponding dissimilarity distance values obtained from the original high-dimensional space. Then, the correlation coefficient would be plotted against the dimension. Ideally, the choice of dimensionality gets visually obvious from the “elbow” in the plot, where after

a certain number of dimensions the correlation would stabilize. An example for the training image of Figure 2.3(a) for that workflow is depicted in Figure 2.5, where 81 dimensional patterns (9×9 pattern template) are analyzed, and a lower 15-dimensional representation is determined with a correlation coefficient of 0.992. This means that the original dissimilarity distance can be very accurately represented by 15D Euclidean distances.



(a) Correlation coefficient with respect to dimension



(b) Scatter plot for 15 dimensional MDS space

Figure 2.5: (a) The correlation coefficient between the original distance matrix and the one constructed by the lower-dimensional MDS configuration is plotted with respect to the dimension of the MDS space, and (b) shows the scatter plot for the dimensionality reduction from 81 to 15, where a high correlation of 99.203% is obtained.

This procedure can also be automated by assuming a certain threshold for the correlation coefficient value. For instance, assuming 98% for the correlation coefficient will be a reasonable value.

One of the shortcomings in this method is the computational burden associated with

it. For each dimension, a distance matrix corresponding to the MDS space points needs to be constructed and compared with the original distance matrix. The more dimensions one retains, the more these two distance matrices would be correlated. Constructing the distance matrix for all dimensions is extremely time-consuming. A pattern database with N patterns in MDS space of d dimensions needs $\frac{N(N-1)}{2} \times d$ CPU calculations. In order to obtain the correlation–dimension plot for this method, one needs $\sum_{d=1, \dots, n_T} \left(\frac{N(N-1)}{2} \times d \right) = \frac{N(N-1)}{2} \cdot \frac{n_T(n_T-1)}{2}$ calculations. Therefore, another approximate technique needs to be developed.

The proposed solution is to use Maximum Likelihood Estimation (MLE) on the dimensionality of the MDS space. The method was first introduced by Zhu and Ghodsi (2006). A modified version was adapted to our analysis. Generally speaking, an appropriate choice for the dimension is made by considering the magnitudes of the eigenvalues, which provide a measure of how much variation is being captured in each dimension. The analysis has been simplified by an initial application of principal component analysis (PCA) on the patterns, which can be performed extremely fast. Plotting the eigenvalues obtained through PCA approach in a descending order provides us the means for finding the minimum dimension that can capture the maximum energy of the eigenvalues. Similar to the previous technique, this resulting scree plot has an elbow that determines the *reasonable* number of dimensions. The dimensionality, represented by the elbow of the scree plot, is found by using the maximum of a log-likelihood function. It is simply a technique to find the elbow of a plot. The assumption behind this technique is that the eigenvalues, considered as random variables, are drawn from two different distributions, one for significant components and one for noise components. A compact explanation of the methodology is provided hereafter in order to point out the differences with the original method proposed by Zhu and Ghodsi (2006).

Let $d_1 \geq d_2 \geq \dots \geq d_{n_T} > 0$ be the eigenvalues in a descending order obtained from principal component analysis. We split this set of eigenvalues into two smaller subsets, φ_1 and φ_2 , according to a splitting point, q , in such a way that $\varphi_1 = \{d_1, d_2, \dots, d_q\}$ and $\varphi_2 = \{d_{q+1}, d_{q+2}, \dots, d_{n_T}\}$. Here, n_T is the dimensionality of the patterns, which basically corresponds to the maximum number of positive eigenvalues obtained through PCA. If we plot the eigenvalues in their descending order, we would obtain a plot called scree plot. One of the characteristics observed in a scree plot is an elbow. For example, a sharp elbow in the scree plot can be conjured up with a sharp decrease in eigenvalues up to the elbow location, and then a nearly flattening behavior for the remaining points. If we split the data at this elbow location, we can observe two subsets with different distributions, i.e.,

$f(d; \mu, \sigma^2)$ with different means (μ), and different variances (σ^2). Therefore, one needs to find a location, q , that will provide two distinct distributions. In other words, one needs to define a likelihood function such that q denotes its maximum. Under these assumptions, we assume the log-likelihood function to be a Gaussian distribution as follows:

$$l_q(q) = \sum_{i=1}^q \log f(d_i; \mu_1, \sigma_1^2) + \sum_{i=q+1}^{n_T} \log f(d_i; \mu_2, \sigma_2^2) \quad (2.13)$$

Here, $\{\mu_1, \sigma_1\}$ and $\{\mu_2, \sigma_2\}$ depend on φ_1 and φ_2 respectively. An estimate on q can then be obtained by maximizing the log-likelihood function above. An empirical search for the maximum of this function would provide us with the dimensionality of MDS space.

$$\hat{q} = \arg \max_{k=2,3,\dots,n_T-1} l_q(k) \quad (2.14)$$

After substituting the Gaussian distribution for f , we would get:

$$l_q(q) = -q \log \left(\frac{1}{\sqrt{2\pi\sigma_1^2}} \right) + \sum_{i=1}^q \frac{(d_i - \mu_1)^2}{2\sigma_1^2} + (q - n_T) \log \left(\frac{1}{\sqrt{2\pi\sigma_2^2}} \right) + \sum_{i=q+1}^{n_T} \frac{(d_i - \mu_2)^2}{2\sigma_2^2} \quad (2.15)$$

where, μ_1 and μ_2 are the means, and σ_1 and σ_2 are the variances of φ_1 and φ_2 , respectively.

The details of maximum log-likelihood elbow detection method is summarized in Algorithm 1 as a reference.

Algorithm 1 Maximum Log-likelihood Elbow Detection

Require: Set $\varphi = \{d_1, d_2, \dots, d_{n_T}\}$ off all the inputs in descending order

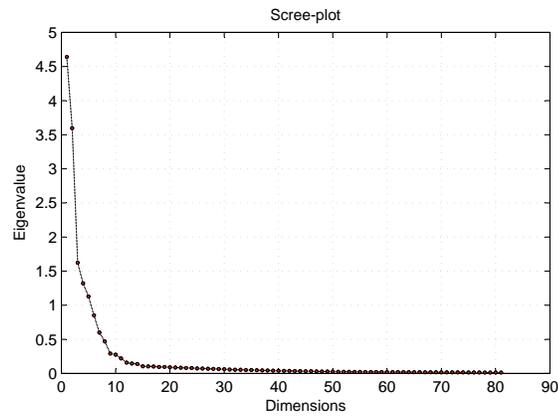
- 1: $\varphi \leftarrow \frac{d^2\varphi}{di^2}$, where $\varphi(i) = d_i$ (second derivative amplifies the elbow)
 - 2: **for** $q = 2, \dots, n_T - 1$ **do**
 - 3: $\varphi_1 : \{d_1, d_2, \dots, d_q\}$
 - 4: $\varphi_2 : \{d_{q+1}, d_{q+2}, \dots, d_{n_T}\}$
 - 5: $\mu_1 \leftarrow E(\varphi_1)$
 - 6: $\mu_2 \leftarrow E(\varphi_2)$
 - 7: $\sigma_1^2 \leftarrow E((\varphi_1 - \mu_1)^2)$
 - 8: $\sigma_2^2 \leftarrow E((\varphi_2 - \mu_2)^2)$
 - 9: $l(q) = f(\varphi_1; \mu_1, \sigma_1^2) + f(\varphi_2; \mu_2, \sigma_2^2)$
 - 10: **end for**
 - 11: **return** $\hat{q} = \arg \max_q l(q)$ as the elbow
-

In order to test the dimensionality selection, the same training image with the same pattern template as before is chosen. 2209 patterns are used for this analysis. The scree plot, obtained from the eigenvalues of the covariance matrix is shown in Figure 2.6(a). In order to find the elbow of this scree plot, the eigenvalues are shown in logarithmic scale in Figure 2.6(b). As can be seen, the best choice for the dimensionality is 15 as the slope changes significantly. The profile likelihood was calculated for different dimensions as shown in Figure 2.6(c). The maximum of the log-likelihood, 15, is chosen as the MDS dimension, which is in perfect agreement with the visual inspection. Satisfactorily, the correlation coefficient of the inter-point distances between MDS points and the original points is 0.992. In conclusion, this profile log-likelihood method of obtaining the dimensionality merely formalizes and automates what can be easily accomplished by a visual analysis, and provides the best dimension for MDS space. We will refer to this technique throughout the dissertation for finding the elbow of any plot.

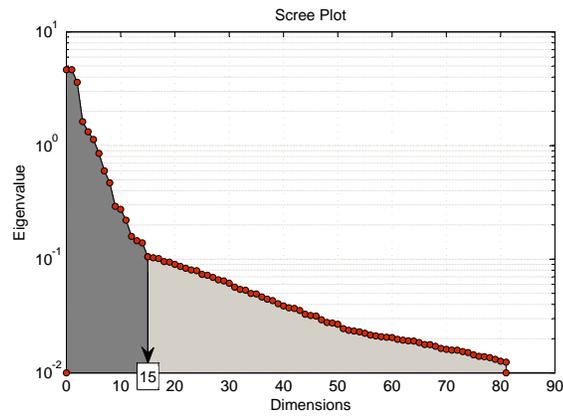
2.4 Pattern Classification

One of the goals of this novel distance-based approach is to develop a technique that removes the need for spatial filters used in *filtersim*, and also to have an avenue for new pattern reproduction concepts. Instead of some pre-defined filters, a more universal classifier is sought. Another important goal of pattern classification in the distance-based method stems from the inspirations obtained from our cognitive visual abilities. If we are provided with a set of patterns, and asked to fill a partially filled area with the provided patterns, we would not be looking for each pattern individually to check if it fits, but we would organize the patterns in our mind by classifying them into similar groups. For example, patterns representing one direction are put in one group than the ones having another direction. This way, when we are to find the most suitable pattern for a partially filled area, we would first, in our mind, search for the most similar group of patterns (i.e. find the ones having the desired direction), and then, use one of those patterns to fill that specific area.

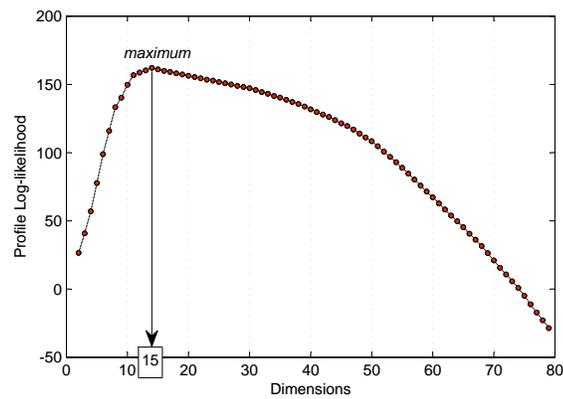
One of the most important methods in unsupervised learning is clustering. Clustering techniques organize the collection of patterns in the pattern database (represented by points in a multi-dimensional space) into clusters based on similarity. Intuitively, after clustering, patterns within a cluster will be more similar to each other than patterns belonging to a different cluster. Therefore, one can see pattern classification as a method that can find the



(a) Scree-plot



(b) Semilog Scree-plot



(c) Profile Likelihood

Figure 2.6: MDS dimensionality selection. (a) The scree plot, (b) the elbow of the scree plot is easily observed in the semi-log scale, (c) the profile log-likelihood of the scree plot showing 15 as the maximum.

internal organization of the pattern database by finding the structures within the points in the MDS space. The clustering methods are widely used in machine learning and data mining (Fayyad et al., 1996), data compression and vector quantization (Gersho and Gray, 1991), and pattern recognition and pattern classification (Duda and Hart, 1973).

Up to this point, each pattern is mapped as a point in a low-dimensional space where the inner-point distances represent the dissimilarity between the patterns. Now, we can easily classify the patterns by organizing the points into clusters. An illustrative example of clustering is depicted in Figure 2.7. The input patterns (represented by points) are shown, and their cluster membership (1 or 2) is colored black or gray. As can be observed, points belonging to the same cluster are given the same label. This approach eliminates the need for any filters and results in better classifications within any set of patterns.

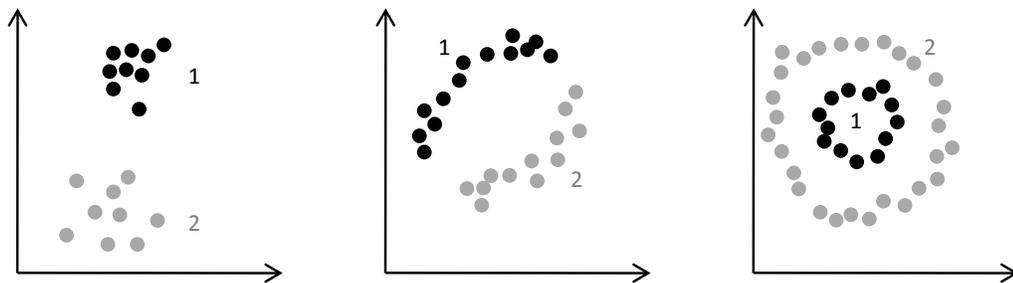


Figure 2.7: Clustering illustration on three different configurations of points. Colors of black or gray represent two different clusters within the data.

Pattern classification is thus one of the main tools that can be used for understanding the structures and reducing the complexity within the data. First, it helps in understanding the pattern variability by breaking them down into different subsets which have more similar spatial features. This can define different prototypes over the data, where actions can be applied accordingly. Secondly, it forms a basis for a more complex analysis. For example, instead of searching over all the patterns in the database, one can apply the search in two steps; one on the cluster representatives, and next, on the cluster members. So patterns are now simpler to describe; where an unknown data can be represented by its relation to a cluster.

In following section, we will introduce a simple and computationally efficient k -means algorithm for clustering. We would then introduce a variant of k -means that guarantees a faster convergence to a clustering solution. And finally, we conclude with an overview

of traditional techniques on how to choose the optimal number of clusters for any dataset. Clustering is a major component of the proposed pattern-based methodology, and thus, the concepts introduced in this section will be later used for classifying patterns of the training image, and generating multiple-point geostatistical Earth models.

2.4.1 k -means

One popular methods for clustering is k -means algorithm (Forgy, 1965; MacQueen, 1967). It is the most widely used and studied algorithm based on a simple iterative scheme for finding the minimal solution to an objective function. For example, Zhang (2006) used it in the context of multiple-point geostatistics for filter-score partitioning.

In this method, a class label l_i will be assigned to each data point \mathbf{x}_i , identifying the class that the corresponding pattern belongs to. The set of all labels for a pattern database of size N is $\mathcal{L} = \{l_1, \dots, l_N\}$, where $l_i \in \{1, \dots, k\}$ and k is the number of clusters. The objective of this method is to determine a set of k points in \mathbb{R}^{n_T} , called centers or prototypes, so as to minimize the mean squared distance from each data point to its nearest center. This allocates each pattern (data point) to one of the k clusters by minimizing the within-cluster sum-of-squares:

$$\text{minimize} \quad \sum_{i=1}^k \sum_{p \in l_i} \|\mathbf{x}_p - \mathbf{v}_i\|_2 \quad (2.16)$$

where \mathcal{A}_i is a set of patterns (data points) in the i^{th} cluster, and \mathbf{v}_i is the mean of those points over cluster i . In k -means clustering methodology \mathbf{v}_i is called the cluster prototype, i.e. the cluster centers:

$$\mathbf{v}_i = \frac{1}{N_i} \sum_{p=1}^{N_i} \mathbf{x}_p \quad , \mathbf{x}_p \in l_i \quad (2.17)$$

where N_i is the number of patterns in \mathcal{A}_i . It should be noted that the reason behind choosing the Euclidean norm in the minimization phase of the algorithm is simply because of the specific metric distance used in multi-dimensional scaling. In mapping the dissimilarity distances between patterns to MDS space, Euclidean distance was used as the metric measure for inter-point distances. Hence, the same Euclidean norm would be the correct representation of the dissimilarities between patterns. Eventually, by using k -means algorithm, patterns can be classified in different clusters.

An example of k -means clustering result is illustrated in Figure 2.8. The choice of centroids, shown in this figure, minimizes the within-cluster distances, and at the same time, maximizes the between-cluster distances. The iterative algorithm used in k -means to reach this final solution is as follows. First, a set of initial centroids is chosen randomly from the dataset. Next, each point is compared with all the centroids to find the closest one. After assigning all the points to their corresponding closest centroids, the algorithm will update the centroids by calculating the mean of all the points within each cluster. This procedure will repeat until there is no more change in the cluster centroids. As one can see, there is always a limitation within the general k -means algorithm on initialization. Different initializations may converge to different local minima. So the initialization is important since it can have detrimental effects in the final clustering result. For example, in the dataset shown in Figure 2.8, if the initial centroids were both located within one of the clusters, they would never be able to escape the local minima trap, and the final result would cluster the data incorrectly by only splitting that group of points into two. This is an intrinsic limitation of clustering algorithms in general. A variant of this algorithm is developed that hierarchically finds the clustering. This algorithm, described in Section A.2 of Appendix A, will not only improve on this limitation, but will also provide a computationally faster clustering algorithm. Other limitations of k -means algorithm will be discussed in Section 2.5 on kernel methods.

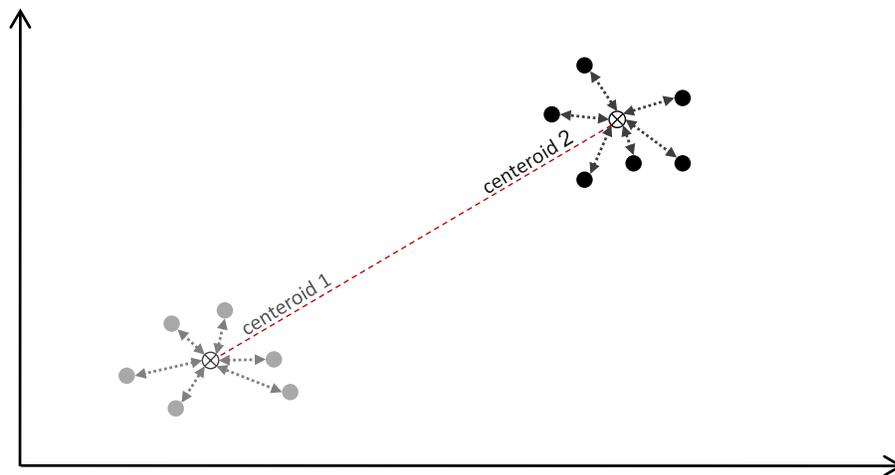


Figure 2.8: Clustering using k -means where the cluster centers, shown in a cross-circle shape, minimize sum-squared distances.

2.4.2 Number of Clusters

Given an input set of data $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, k -means clustering tries to find an assignment, f , for each point to k different clusters $f : \mathcal{S} \rightarrow \{1, 2, \dots, N\}$. This grouping of data should be chosen in such a way as to minimize an objective function which measures the quality of clustering. One of the main issues in any classification is the decision that should be made by the user on the number of clusters, k , prior to the algorithm. In order to automate this phase, and most importantly, to find an appropriate number of clusters, two existing methods will be introduced hereafter. In both methods, an index is used to validate the best number of clusters. This index is nothing but a measure of the quality of cluster assignments.

In the first method, it is assumed that “good” clusters are those whose patterns are close to each other compared to the next closest cluster. Accordingly, an index called ‘silhouette’ is defined as follows:

$$S(i) = \frac{b_i - a_i}{\max(a_i, b_i)} \quad (2.18)$$

where a_i is the average distance from the i^{th} point to the other points in its cluster, and b_i is the minimum (over all clusters) of the average distances from the i^{th} point to the points in another cluster. The silhouette value for each point is a measure of how similar that point is to points in its own cluster compared to points in other clusters, and ranges from -1 to $+1$. The average of this index for all clusters is called ‘mean silhouette index’. As a result, the maximum value in ‘mean silhouette plot’ will point out the appropriate number of clusters.

The second method stems from the fact that a good clustering yields clusters where patterns have small within-cluster sum-of-squares (SSW), and high between-cluster sum-of-squares (SSB). These statistics measure the dispersion of the data points in a cluster and between the clusters, respectively. According to that, ‘Calinski and Harabasz’ index is one of the best candidates which provides excellent recovery in terms of selection of the number of clusters (Milligan and Cooper, 1985).

$$\text{Calinski Harabasz} = CH = \left(\frac{\text{SSB}}{N-1} \right) \left(\frac{\text{SSW}}{N-k} \right)^{-1} \quad (2.19)$$

where N is the number of data points and k is the number of clusters. Here, the maximum

value determines the proposed number of clusters (Calinski and Harabasz, 1974).

A sample illustration of these formulations is given in Figure 2.9 for a 30×30 TI, and a 8×8 pattern template. As can be seen, both indices provide 23 as the optimum number of clusters. In this analysis, a maximum value should be assigned for the number of clusters such that the search could be limited to that constraint.

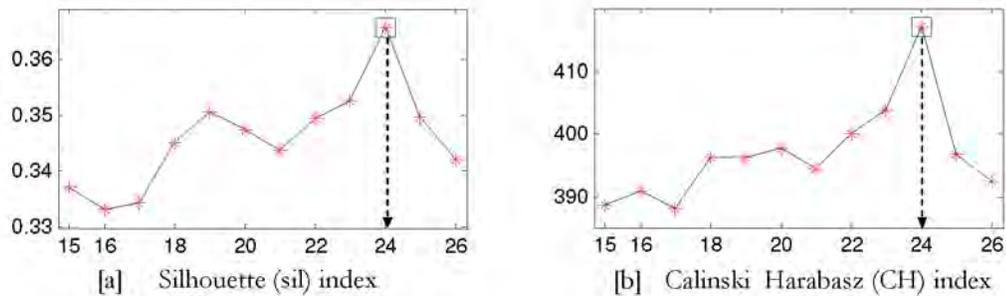


Figure 2.9: Silhouette and Calinski-Harabasz (CH) indexes; indicating 23 as the best number of clusters

These methods of cluster index analysis have been widely used in literature. The advantage of these two methods is their simplicity. They rely on the quality of clustering as a measure to find the optimal number of clusters. However, they all suffer from a limitation; that is, in order to find the number of clusters, k -means algorithm needs to be applied several times for each number of cluster. For example, if one wishes to use mean silhouette plot, the silhouette for all possible number of clusters needs to be obtained first. This can be time-consuming. One seeks a measure that can provide the optimal number of clusters without the need to run the clustering algorithm. In other words, one needs to be able to understand the structure of the data and decide on a reasonable number of clusters; one that can simplify the complexity inherent in the data. In order to have such an algorithm, we can take advantage of the distance-based framework. Having constructed the distance matrix, there would be a wealth of information provided through this proximity information. Therefore, a method that analyzes the structure of the distance matrix is developed and given in Section 5.3.

Up until now, the patterns of the training image are mapped into a lower-dimensional MDS space, where each point represents a pattern. There are numerous advantages to this representation. It provides a framework for further analysis of patterns, and reduces the inherent complexities of the patterns in a training image. In this section, we demonstrated

how one can take advantage of this mathematical framework for pattern modeling by the application of k -means clustering. Grouping similar patterns into same clusters will aid in better understanding the structures of the patterns. However, k -means clustering algorithm is a linear technique; thus, it cannot reveal the non-linearities among patterns. Next section will introduce a non-linear transformation, kernel mapping, to alleviate the intrinsic weaknesses existing in all linear algorithms.

2.5 Kernel Methods

The variability in the set of patterns, obtained from a training image, can be mathematically rather complicated. There are abundant algorithms that can efficiently find the linear relationship among the data, while non-linear data are handled in a less principled way. The method, described in this section, will combine the theoretically well-founded methodologies for linear processes with the non-linear ones. It allows forming extraordinarily powerful and robust class of pattern analysis techniques by taking advantage of the flexibility and applicability of non-linear processes. In this section, the definition of kernel transformation and feature space as the means to capture the non-linearities inherent in the patterns of a training image is introduced. Next, the feature space counterpart of the clustering algorithm, which will be beneficial for geostatistical simulation of patterns, is explained.

2.5.1 The Overall Picture

Pattern modeling deals with the automatic detection of patterns within the training image. By understanding the relations and structures between the patterns, we can implement a system that can make predictions from the data. For example, we can use this system to generate multiple-point statistical realizations that honor the structures learned through the input patterns. However, the learning power of the previously introduced algorithms, such as k -means, does not generalize satisfactorily. One seeks a method that can capture the non-linearities that exist in any complex training image. The structures within the patterns may consist of sinusoidal channels, lobes, mounds, and many more that are not recognizable by simple linear techniques. However, by recoding the data into another representation, one can increase the chances of identifying different complex features within the patterns. Kernel approaches embed the input data into a higher-dimensional feature space, where

the relation among the data becomes more linear. Therefore, in order for this method to work, one needs two steps. The first step is to have the tools to map the data into another so called feature space, and the second step is to apply linear learning algorithms over the data but in the feature space. In order to better understand the need for kernel mapping, we will show an example of the limitations in a linear algorithm such as k -means.

Considering the amount of available patterns within the pattern database of a training image, the simple, fast and efficient clustering algorithms, such as k -means method, offer a solution for classification. However, k -means suffers from several drawbacks. In the case where data exhibit a complex structure (e.g. data are non-linearly separable), a direct application of k -means is not suitable because of its tendency to group data into globe-shaped clusters (Mackay, 2003). This misclassification has a direct effect on the pattern recognition capability in the proposed methodology. In order to solve this problem, data will be mapped by a transformation Φ into a new feature space F where samples become linearly separable (Shawe-Taylor and Cristianini, 2004). Separation is easier in higher-dimensional space. This change of coordinates maintains the presence of regularities in the data, but changes their representation; which then makes the regularities easier to detect. In other words, by the non-linear mapping of data points to that higher-dimensional space F , the non-linear relationship among the information provided by the data can be captured more efficiently. A sample representation of this process is shown in Figure 2.10, where a non-linear dataset is linearized in the feature space (Schölkopf and Smola, 2001). In this feature space, classical clustering algorithms such as k -means would have a better performance, and consequently, a stronger and more accurate pattern classification will result. The mathematical formulations for this non-linear transformation are explained extensively in Section 2.5.2.

The combination of multi-dimensional scaling (MDS) and kernel mapping enables the application of any distance function in pattern analysis. For example, using Hausdorff distance for calculation of similarities between the patterns will not result in a positive semi-definite kernel matrix, and hence will not be a plausible function. On the other hand, upon embedding the patterns into a metric space, with Hausdorff as the distance function, kernel transformation would become feasible (since it is the Euclidean distance which will be used in the lower-dimensional space as opposed to the Hausdorff distance). Therefore, if we need the freedom in the choice of our distance function, MDS must inevitably be a part of the simulation algorithm.

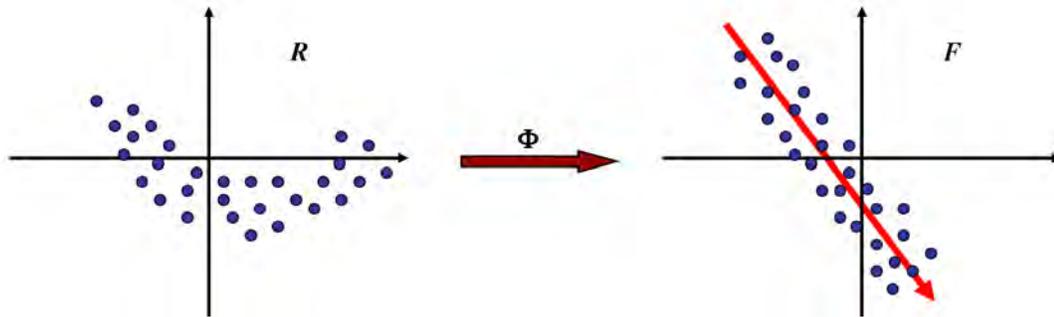


Figure 2.10: Kernel transformation, and the principal component in the linear kernel space

2.5.2 Properties of kernels

Generally, kernel transformation needs to preserve the similarity between the data in the feature space. One particularly simple yet surprisingly useful notion of this similarity - the one that is used in this study - is derived from embedding the patterns into a Euclidean space, where simple geometrical concepts could be used. Likewise, this simple dissimilarity can also be measured by their dot product in the high-dimensional feature space F obtained by kernel transformation. This leads to one of the crucial ingredients of this formulation, *the kernel trick*, for the computation of this dot product in the feature space through the application of simple functions defined on pairs of input patterns (Shawe-Taylor and Cristianini, 2004). Therefore, the patterns are first mapped into F using the following embedding:

$$\Phi : \mathbf{x} \in \mathbb{R}^d \mapsto \Phi(\mathbf{x}) \in F \subseteq \mathbb{R}^N \quad (2.20)$$

where $N > d$, and then compared using a dot product $\langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l) \rangle$. However, to avoid working in potentially high-dimensional feature space F , one tries to pick a feature space in which the dot product can be evaluated directly using a non-linear function $\kappa(\cdot)$ in the input space, i.e. by the kernel trick.

$$\kappa(\mathbf{x}_k, \mathbf{x}_l) = \langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l) \rangle \quad (2.21)$$

The success of this approach is related to the fact that using a kernel is equivalent to defining a feature space transform. On top of that, the advantage of the kernel trick is

that the coordinates of the feature space data are not explicitly computed, and the distance calculations in F are efficiently performed through a kernel function. Hence, $\kappa(\mathbf{x}_k, \mathbf{x}_l)$ is the inner product not of the coordinate vectors \mathbf{x}_k and \mathbf{x}_l in \mathbb{R}^d but of vectors $\Phi(\mathbf{x}_k)$ and $\Phi(\mathbf{x}_l)$ in higher dimensions. The kernel trick allows the application of linear algorithms on non-linear data only when they are cast in terms of dot products; k -means and PCA being the most prominent examples.

In order to define the kernel function, it should be noted that the computational procedure depends only on the inner products $\langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l) \rangle$ in the feature space. Thus, a linear algorithm can be easily transformed into a non-linear one. Not having the kernel trick, it would be impossible to apply any linear algorithm on infinite-dimensional feature space. Therefore, kernel functions can improve upon computational complexities of calculating the inner products in the feature space. Due to these advantages, the solution to a better pattern classification can be obtained by using the *Kernel K-means Algorithm* (Maitre et al., 2008) as an alternative to k -means.

The most familiar kernel function, as used in this study, is the Gaussian radial basis function, which is given by:

$$\kappa(\mathbf{x}_k, \mathbf{x}_l) = \exp\left(-\frac{\|\mathbf{x}_k - \mathbf{x}_l\|^2}{2\sigma^2}\right) \quad (2.22)$$

One of the properties of the Gaussian radial basis function is that it maps the data to an *infinite* dimensional feature space. It should be noted that besides the Gaussian kernel, a majority of other admissible kernels exist (for example: polynomial kernel, hyperbolic kernels, covariance kernel). The choice of the kernel function used in our pattern modeling framework comes from the simplicity and applicability of Gaussian function, and more importantly, its psychological foundation. In terms of generality, it has performed reasonably well on a wide range of problems in many different application areas; such as face recognition, hand recognition, genome detection, multivariate calibration, etc. Despite its generalities, psychologically speaking, it also has very interesting properties. According to Shepard (1987), the best measures of similarity are generalization gradients. According to his studies, a non-linear relationship exists between the similarities in psychological space and the measured ones, which is monotonic and concave upwards. He demonstrated that the exponential function is the link between these two similarities for an optimal classification, and used it as the *universal law of generalization* (Shepard, 1987; Tenenbaum and

Griffiths, 2001; Chater and Vitányi, 2001). We have adapted the same Gaussian function for pattern recognition, due to both the proven capabilities and the psychological foundations.

Having defined the kernel function, one can transform the data into feature space by the means of constructing a kernel matrix. The kernel matrix is a positive semi-definite matrix that holds the results of inner products of the input data in a feature space with feature map Φ . Therefore, the kernel matrix, \mathbf{K} , can be defined as follows:

$$\mathbf{K} = \begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \kappa(\mathbf{x}_N, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix} \quad (2.23)$$

This kernel matrix plays an important role in subsequent algorithms. It holds all the information needed for linear algorithms in the feature space. Moreover, it transforms the information about the input data in feature space and acts as an interface for linear algorithms. One prominent example of such a role is taking advantage of the structure of the kernel matrix in order to obtain a better understanding of the regularities among the patterns.

However, in order to construct the kernel matrix for Gaussian radial basis function, one needs to define its bandwidth, σ . This parameter depends on our prior knowledge of the pattern domain. Instead, in this analysis we will use some heuristic algorithms to define the bandwidth that is appropriate for any distribution of patterns. In order to understand the effect of bandwidth on the mappings in Gaussian radial basis functions, we will analyze two extreme values of $\sigma = 0$ and $\sigma = \infty$. Generally speaking, a Gaussian kernel with a very large bandwidth ($\sigma \rightarrow \infty$) will not be able to differentiate between the input data. This is due to the kernel function resulting in a value of 1 for all the input data: $\kappa(\mathbf{x}_i, \mathbf{x}_j) \simeq 1, \forall \mathbf{x}_i, \mathbf{x}_j$. Since kernel function reflects the inner product of points in feature space, $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \simeq 1, \forall \mathbf{x}_i, \mathbf{x}_j$, they would all tend to become similar $\Phi(\mathbf{x}_1) = \Phi(\mathbf{x}_2) = \cdots = \Phi(\mathbf{x}_N)$. Intuitively, it results in an under-fitting of the data, and can be visualized in 2D by assuming large Gaussian surfaces overlain on top of a small bivariate distribution; where no variability between the data can be expressed by the surfaces. On the other hand, a Gaussian kernel with a very small bandwidth ($\sigma \rightarrow 0$) is not able to understand the non-linear relationship between the data. All the patterns would thus become dissimilar to each other; which results in an over-fitting to the data. Therefore,

the optimal σ is a tradeoff between these two under-fitting and over-fitting characteristics.

In order to find the optimal bandwidth for our kernel function, we use a heuristic approach on the input data (Lampert, 2009). For the Gaussian kernel of the form

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\gamma d^2(\mathbf{x}, \mathbf{x}')) \quad (2.24)$$

where d is a distance function between the input data (i.e., Euclidean distance function), a rule of thumb for setting γ is:

$$\frac{1}{\gamma} \approx \text{median}_{i,j=1,\dots,N} d(\mathbf{x}_i, \mathbf{x}_j) \quad (2.25)$$

The concept behind this choice is the shape of the Gaussian kernel. We would like to choose the bandwidth in such a way that equal number of similar and dissimilar pattern pairs be included in the mapping. Since the Gaussian function has its steepest decline at the exponent of -1 , we choose γ such that the exponent becomes -1 . The median of d will provide an equal splitting radius for similar and dissimilar input patterns.

In summary, kernel functions compute the inner products of the projections of input data into feature space. This mapping of the data into higher-dimensional feature space will enable the use of existing linear algorithms to discover non-linear structures among patterns. In the next section, Kernel k -means as a non-linear variant of k -means will be introduced with an example.

2.5.3 Kernel k -means

As said earlier, kernel function can be viewed as a non-linear transformation that increases the separability of the input data. The kernel trick allows formulating non-linear variants of any algorithm that can be cast in terms of dot products. Hence, the solution to a better pattern classification can be obtained by using the *kernel k -means* algorithm. This enables the k -means algorithm to explore the patterns in the new feature space F , and hence, result in a powerful clustering method. An example comparison of the beneficial results of clustering using kernel k -means (in linear feature space) as opposed to k -means (in non-linear Euclidean space) is depicted in Figure 2.11.

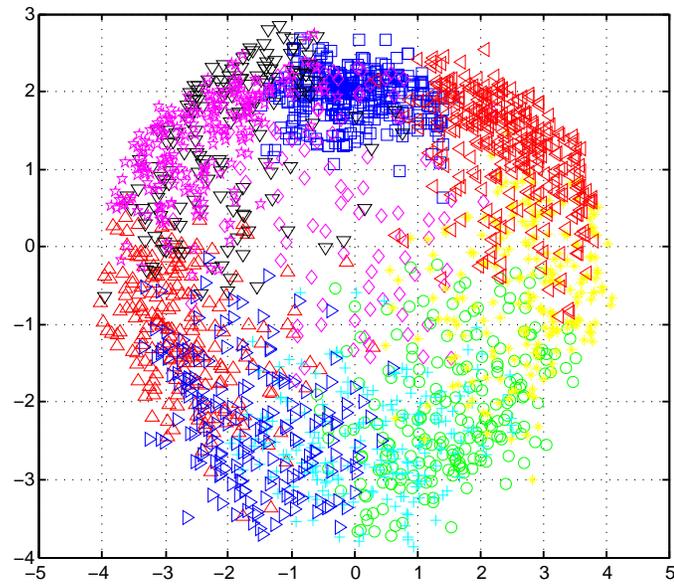
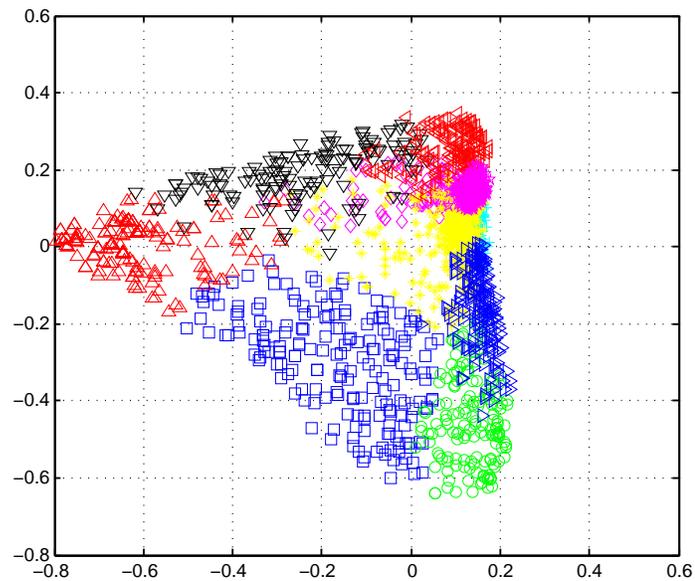
(a) k -means clustering(b) kernel k -means clustering

Figure 2.11: Clustering results using 10 clusters (shown by projection into $2d$ space) using the training image given in Figure 2.3. K -means clustering without applying kernel transformation (top) is compared with kernel k -means clustering (bottom).

In order to formulate k -means algorithm for kernel-induced feature space, we need to find a dual representation of all calculations in terms of dot products. The main step in k -means algorithm is the definition of Euclidean distance between two points. It can be calculated as follows in the feature space. Assuming two vectors, \mathbf{x} and \mathbf{z} , in the input space, the distance between their feature space representations, $\Phi(\mathbf{x})$ and $\Phi(\mathbf{z})$, can be calculated by:

$$\begin{aligned} \|\Phi(\mathbf{x}) - \Phi(\mathbf{z})\|^2 &= \langle \Phi(\mathbf{x}) - \Phi(\mathbf{z}), \Phi(\mathbf{x}) - \Phi(\mathbf{z}) \rangle \\ &= \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}) \rangle - 2\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle + \langle \Phi(\mathbf{z}), \Phi(\mathbf{z}) \rangle \\ &= \kappa(\mathbf{x}, \mathbf{x}) - 2\kappa(\mathbf{x}, \mathbf{z}) + \kappa(\mathbf{z}, \mathbf{z}) \end{aligned} \quad (2.26)$$

which can be further simplified for the Gaussian function (since $\kappa(\mathbf{x}, \mathbf{x}) = 1$), as follows:

$$\|\Phi(\mathbf{x}) - \Phi(\mathbf{z})\|^2 = 2 - 2\kappa(\mathbf{x}, \mathbf{z}) \quad (2.27)$$

It should be noted that with this new metric the distances are bounded from above. The maximum distance between two points in the feature space is $\sqrt{2}$. Physiologically speaking, this is an appealing property. It states that two patterns that are very different from each other cannot become more different. A perceptual comparison of patterns will indicate that only local similarity measurements can be made, and large differences are not as cognitively discernible. For example, a human subject can easily obtain a similarity measure for two similar patterns, but if asked to quantify two very different patterns, he would just respond that they are “totally different” (Indow, 1994). This is the same notion of similarity that is carried by inner products in the kernel-induced feature space of Gaussian functions.

Another important step during the iterations of k -means algorithm is to find the distance between a point and a cluster centroid. However, we neither know the coordinates of the points, nor the coordinates of cluster centroids in feature space. We can reformulate the distance between a point and a cluster only by means of inner products. Let us assume that the s^{th} cluster consists of ℓ points as follows: $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell\}$. Then, the centroid of this cluster can be shown:

$$\Phi_{\mathcal{S}} = \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(\mathbf{x}_i) \quad (2.28)$$

The distance between an input point \mathbf{z} and the centroid of s^{th} cluster in feature space can

be computed as follows:

$$\begin{aligned}
\|\Phi(\mathbf{z}) - \Phi_S\|^2 &= \langle \Phi(\mathbf{z}), \Phi(\mathbf{z}) \rangle + \langle \Phi_S, \Phi_S \rangle - 2\langle \Phi(\mathbf{z}), \Phi_S \rangle \\
&= \kappa(\mathbf{z}, \mathbf{z}) + \left\langle \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(\mathbf{x}_i), \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(\mathbf{x}_i) \right\rangle - 2 \left\langle \Phi(\mathbf{z}), \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(\mathbf{x}_i) \right\rangle \\
&= \kappa(\mathbf{z}, \mathbf{z}) + \frac{1}{\ell^2} \sum_{i,j=1}^{\ell} \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle - \frac{2}{\ell} \sum_{i=1}^{\ell} \langle \Phi(\mathbf{z}), \Phi(\mathbf{x}_i) \rangle \\
&= \kappa(\mathbf{z}, \mathbf{z}) + \frac{1}{\ell^2} \sum_{i,j=1}^{\ell} \kappa(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{\ell} \sum_{i=1}^{\ell} \kappa(\mathbf{z}, \mathbf{x}_i)
\end{aligned} \tag{2.29}$$

The rest of the procedure is the same as k -means algorithm, where it alternates between two steps of assignment of points to the clusters and updating the clusters in each iteration.

Similar to the previous algorithms, kernel k -means is prone to local minima. This is due to the non-convex iterative optimization, and the fact that the choice on initial centroids can have significant effect on the quality of the clustering solution. Therefore, we are proposing another variant of the kernel k -means clustering that reduces the sensitivity of the algorithm to the initial centroids. As said earlier, the kernel function can be viewed as a non-linear transformation that increases the separability of the input data by mapping them to a new high-dimensional space. The incorporation of the kernel function enables the k -means algorithm to explore the inherent data pattern in the new space, and hence, is a very powerful clustering method. However, the recent application of kernel k -means algorithm is confined to small datasets due to its expensive computations and storage costs. We will be analyzing the computational complexity of kernel k -means, and then, will provide a simple improvement on its efficiency.

A dataset of training images with 2209 patterns is chosen as an example for this analysis. The dimensionality of the patterns (in MDS space) is 12. First, the patterns are mapped into kernel space by calculating the kernel matrix. Since, the mapped data are of infinite-dimensions, the kernel trick is used for k -means clustering. The results, using the improved and efficient kernel k -means algorithm, is provided in Figure 2.12. As observed, kernel k -means algorithm has a computational complexity of $O(N^3)$. This is one of the most computationally prohibitive parts of the entire methodology.

However, one can think of a better initialization for the centroids, which has a direct

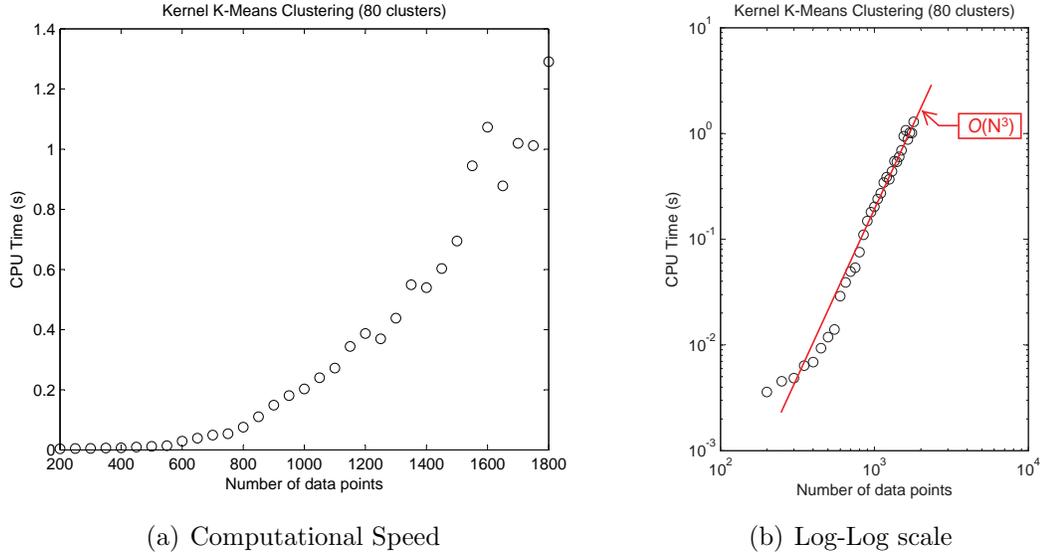


Figure 2.12: (a) kernel k -means speed analysis, (b) kernel k -means speed shown in log-log scale, showing the computational complexity of the algorithm, $O(N^3)$.

effect on the convergence rate of the k -means algorithm. It has been shown that the k -means algorithm on a set of non-linear data may produce incorrect globe-shaped clusters. However, in many instances there are some linearities inherent to different subspaces of the data. Assuming that these linear subspaces dominate the entire structure of the non-linear input data, we can directly apply k -means clustering on the input data. Knowing the general non-linear behavior in the patterns, one expects some errors to be introduced in the clustering result. In order to reduce the errors and obtain more accurate classification, kernel k -means is applied on the input data, where the initial centroids are assigned according to the final clustering result of the previous k -means implementation. On the assumption that kernel k -means algorithm will find the correct set of non-linear clusters, it will converge to the true solution no matter how the initial centroids are chosen. The gain in speed is due to the better initialization obtained by k -means algorithm. The initial centroids are correct to the degree of the linearity in subspaces of the samples. It should be mentioned that the final clustering results are still approximate due to the optimization routine of the k -means algorithm, which makes it susceptible to local optimum traps. This may be more pronounced here, since the initial centroids depend on the previous linear clustering results. The speed improvement of this method is shown in Figure 2.13. The larger the pattern

database or the training image, the greater is the improvement gained using the proposed methodology.

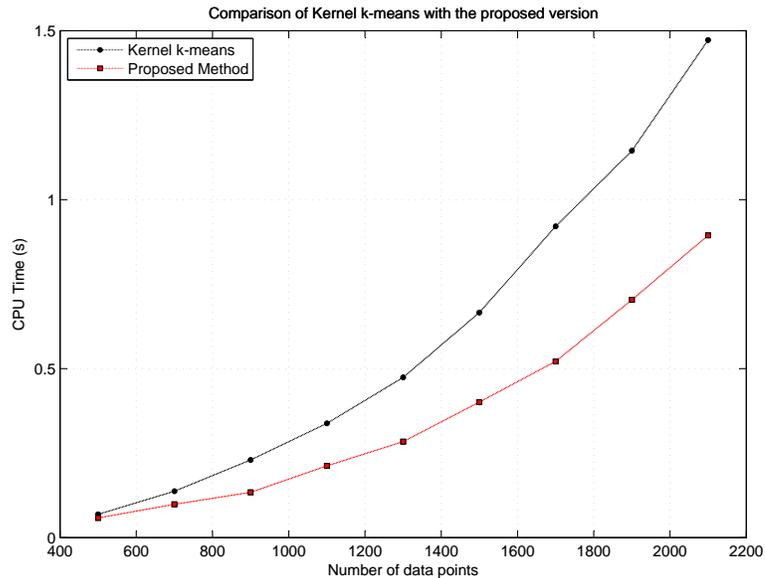


Figure 2.13: Comparison of kernel k -means with the proposed methodology of using k -means as an initializer for kernel k -means centroids.

2.6 Conclusion

One of the main goals in pattern-based multiple-point geostatistics is to have a general tool for modeling patterns. This tool should be able to learn the complex relationship that exists between the input data (patterns within the training image), and provide accurate classifications. The main interest is to understand, perceptually, what are the critical features within the patterns. In other words, from an information-theoretical point of view, what features should one take? and how can one come up with a pattern recognition system that is based on those features? For this matter, in this chapter, we tried to provide a modeling class that is as general as possible for pattern analysis.

There is considerable complexity among the patterns of a training image. However, in principle, there is just a simple model which generated the training image. More complex patterns do not necessarily need more parameters nor do they require more complex models for their representation. In this chapter, we tried to introduce the techniques that can

capture this complex behavior and provide a simple, yet powerful, framework for pattern modeling. Distance-based modeling allowed us to reduce the complexities among those patterns.

Distance or similarity was shown to be a useful notion to represent patterns. One does not need to store the complex patterns themselves, but only their interaction and the similarities between each other. Psychologically, this is how a human observer would differentiate between the objects. For example, it would be much easier for a human being to measure the beauty of a face by comparison with other faces, than just individual assessments. The same is true with different objects. Patterns can be represented in metric space by defining a distance function between them. This distance function is purpose-driven and depends on the decision at hand. In MPS modeling, the distances define a measure of similarity between patterns, i.e., Euclidean distance as the simplest one.

Having these distances, a metric space is defined, and represented mathematically by the distance matrix. Next, multi-dimensional scaling can be applied on this distance matrix to obtain a lower-dimensional representation of patterns in a Cartesian space, called MDS space. Patterns are thus represented by points in MDS space, where similar patterns are located close to each other, and dissimilar patterns are far apart. Using multi-dimensional scaling, we have been able to reduce the complex relationship between the patterns and visualized their interaction in MDS space. This provided us with two advantages; being computational speed, and model simplicity. As a result, we have obtained a mathematical framework for pattern modeling, which facilitates the application of several different algorithms on these patterns.

However, since most of the algorithms, such as classification algorithm of k -means, are linear techniques, they can be applied only on linear data. Let's assume that one is allowed to use a hyperplane to split a set of data into two clusters. A linear algorithm would only be able to fit a linear hyperplane as the classification boundary between the two different clusters. Patterns may have non-linear features among them that could not be captured by a linear algorithm. Therefore, we introduced the concept of kernels and feature space to resolve such situations. According to the kernel transformation, one can map the data to a higher-dimensional feature space, where the possibility of linear separability of data increases. According to Cover's Theorem (Cover, 1965):

A complex pattern-classification problem, cast in a high-dimensional space non-linearly, is more likely to be linearly separable than in a low-dimensional space,

provided that the space is not densely populated.

Therefore, by mapping the input patterns from MDS space into feature space, one increases the possibility of linear separability between the data, or in other words, in the case of Gaussian radial basis functions where data get mapped into an infinite-dimensional space, the patterns become more linearly separable. Thus, kernel k -means, as the non-linear variant of k -means in the feature space, can better classify the patterns into different clusters. This learning approach of using kernel transformation provides a more accurate system for modeling the patterns. We will use this distance-based pattern modeling framework in the next chapter, and will show how the multiple-point geostatistical algorithm can take advantage of the capabilities introduced by the concept of distances and kernels.

Chapter 3

Unconditional MPS Simulation

What an interior strength a man can summon if he devotes himself entirely to knowledge and creation, rather than to a vain search for honors and celebrity! What a lesson!

BERTRAND RUSSELL (1872 - 1970)

According to Journel (2005) the training image is a conceptual model of the random process. Multiple-point statistical simulations are tools to generate complex subsurface models by honoring the statistics provided through a training image. It can be considered a quantitative model in the same way as a variogram quantifies the spatial variability in the Earth model. For example, in kriging or sequential Gaussian simulation, the parameters such as mean, variogram range, nugget effect, or the variogram model define the spatial statistics deemed representative of the field under study. The same is true in multiple-point geostatistics, where a training image explicitly quantifies the relevant statistics of the subsurface. These statistics can be captured by a direct probabilistic approach, such as in *snesim*, or by a pattern-based approach, as proposed in this dissertation.

Pattern-based modeling allows reproduction of the same statistics provided through a stationary training image in order to generate realizations. Therefore, the main ingredient in this approach is to have a robust and general framework for modeling the patterns of the training image. In the previous chapter, we introduced a distance-based modeling framework for pattern analysis. However, none of the previous algorithms would be beneficial if there is no simulation algorithm that can properly utilize this framework.

In this chapter, a simulation algorithm, called DisPAT, that operates on this framework will be presented with examples. In the first section, the pattern-based MPS methodology will be explained. Afterwards, a comparison with previous MPS techniques in terms of classification performance and pattern reproduction will be made. The chapter then concludes by comparing the variability between the generated models, and a discussion on the uncertainty space manifested through MPS simulations.

3.1 Simulation Methodology

The purpose of MPS simulation is to use the conceptual training image, \mathbf{ti} , to generate many realizations, \mathbf{re} , in such a way that the statistics conveyed through a training image is honored. Here, each realization represents an Earth model that can subsequently be used for flow simulation and decision making purposes. Assuming our training image to accurately conceptualize our geological models, MPS simulation generates several realizations that will eventually help in quantifying the uncertainty range for the desired output variable. In the petroleum industry, the output variables can be the ultimate oil recovery, or the net present value of the project under study. Quantifying their uncertainty is of great advantage for making economic decisions on how to operate the field and produce the hydrocarbons. In this section, DisPAT, as a pattern-based multiple-point geostatistical algorithm, which can generate the required models for spatial uncertainty quantification, will be explained. The steps required for generating such realizations are provided successively in following subsections.

3.1.1 Processing of the Training Image

The first step of the methodology is to process the training image. Processing of the training image, \mathbf{ti} , is performed by scanning the training image using a template \mathbf{T} and storing the corresponding multiple-point $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ vectors in a database. Each such $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ is called a pattern of the training image and the database is called the pattern database and is denoted by $\mathbf{patdb}_{\mathbf{T}}$. According to the template \mathbf{T} and since the training image has a finite volume, the number of patterns that can be stored in the pattern database are limited. The number of patterns in the $\mathbf{patdb}_{\mathbf{T}}$ is N , where each pattern is a n_T dimensional vector represented by:

$$\mathbf{pat}_T^k = \{pat_T^k(\mathbf{h}_1), pat_T^k(\mathbf{h}_2), \dots, pat_T^k(\mathbf{h}_\alpha), \dots, pat_T^k(\mathbf{h}_{n_T})\} \quad (3.1)$$

where k represents the index of the pattern in pattern database.

An example application of processing a simple training image, with a 3×3 template size, is shown in Figure 3.1.

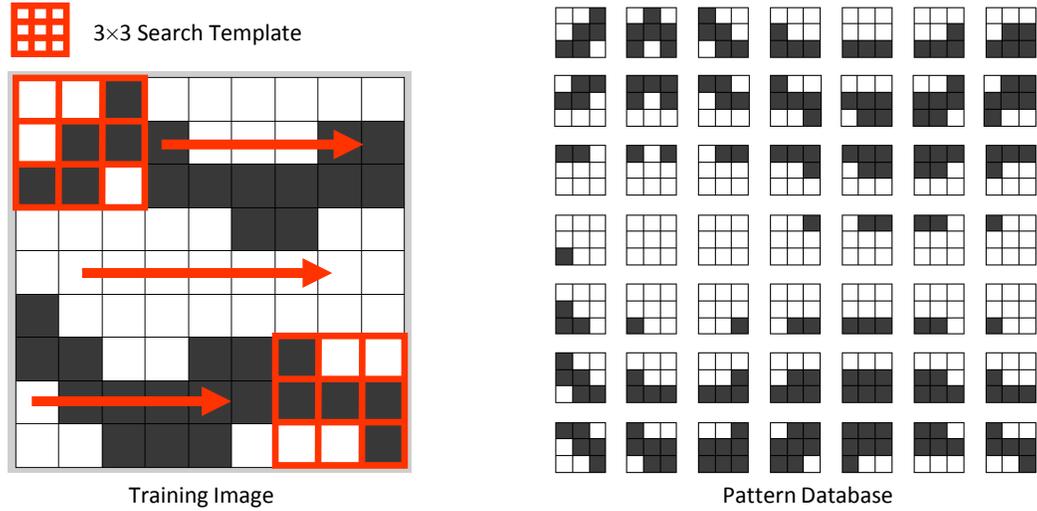


Figure 3.1: A training image is scanned with a 3×3 template size (left), and all the patterns are stored in the pattern database (right).

The pattern database can be seen as a $N \times n_T$ matrix, where each row represents one pattern ($N = n_{Pat_T}$). The order of patterns within this database is arbitrary. In other words, the patterns are now location-independent. A stationary training image can thus be fully represented by the pattern database up to the size of the template, n_T . This is the first step of the proposed pattern-based approach, where a pattern database matrix is constructed as follows:

$$\mathbf{patdb}_{\mathbf{T}} = \begin{bmatrix} pat_{\mathbf{T}}^1(\mathbf{h}_1) & pat_{\mathbf{T}}^1(\mathbf{h}_2) & \cdots & pat_{\mathbf{T}}^1(\mathbf{h}_{n_T}) \\ pat_{\mathbf{T}}^2(\mathbf{h}_1) & pat_{\mathbf{T}}^2(\mathbf{h}_2) & \cdots & pat_{\mathbf{T}}^2(\mathbf{h}_{n_T}) \\ pat_{\mathbf{T}}^3(\mathbf{h}_1) & pat_{\mathbf{T}}^3(\mathbf{h}_2) & \cdots & pat_{\mathbf{T}}^3(\mathbf{h}_{n_T}) \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ pat_{\mathbf{T}}^N(\mathbf{h}_1) & pat_{\mathbf{T}}^N(\mathbf{h}_2) & \cdots & pat_{\mathbf{T}}^N(\mathbf{h}_{n_T}) \end{bmatrix} \quad (3.2)$$

3.1.2 Classification of Pattern Database

Having obtained the pattern database, we can now apply the methods introduced in the previous chapter, namely, distances and kernels, to classify the patterns into different clusters. The first step towards classification is to define a distance function between two patterns. The function used in the proposed algorithm is defined in Section 3.1.4. As explained in previous chapter, a distance matrix can be constructed by calculating the pair-wise distances between all the patterns in the database. This distance matrix, \mathbf{D} , represents a metric space of similarities.

Having obtained this metric space of patterns, we apply multi-dimensional scaling using \mathbf{D} as the distance matrix. The mapping obtained through MDS would provide a configuration of points in a lower dimensional Cartesian space, \mathbf{X}_d . An example application, where the patterns are represented in a two dimensional MDS space, is depicted in Figure 3.2. One can observe that similar patterns are located close to each other in MDS space, and vice versa. This representation of patterns is one of the advantages of distance-based pattern modeling. It captures the complex interactions among the patterns, and provides a simple two dimensional projection.

The final step towards pattern classification is to apply the kernel k -means clustering algorithm by mapping the data into a higher-dimensional feature space, F . With this projection, the patterns become more linearly separable. Therefore, clustering in the feature space would provide a very high classification accuracy. Then, by mapping the points back into the original MDS space, we would have obtained a very good clustering. The quality of the clustering obtained with this procedure is superior to the one where k -means algorithm is applied in the original pattern space (or MDS space).

Pattern classification is perceivably a learning algorithm. Previously introduced techniques simply aid in learning the relationship between patterns and grouping them according

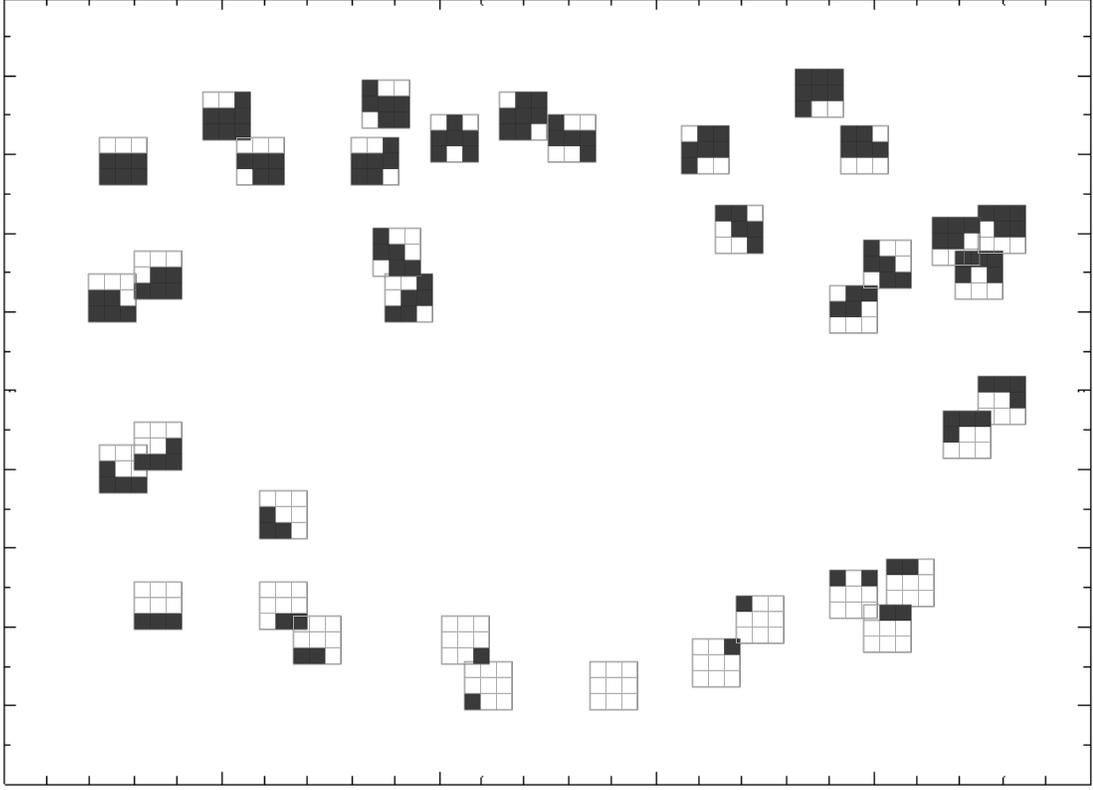


Figure 3.2: MDS representation of patterns from the training image shown in Figure 3.1. As can be observed, the patterns that are similar to each other are positioned closely in this 2-dimensional MDS space.

to their complex, non-linear features. Each cluster is then represented by the average of all its patterns, namely a prototype. A cluster prototype is defined as the pixel-wise arithmetic average (ensemble average) of all training patterns in a cluster. The prototype represents a class of patterns. For example, let's assume that set \mathcal{S}_i ,

$$\mathcal{S}_i = \{\mathbf{pat}_{\mathbf{T}}^{C_{i1}}, \mathbf{pat}_{\mathbf{T}}^{C_{i2}}, \dots, \mathbf{pat}_{\mathbf{T}}^{C_{i\ell}}\} \quad (3.3)$$

represents all the patterns that belong to i^{th} cluster; where C_i represents the set of indices of ℓ patterns existing in cluster i .

$$C_i = \{C_{i1}, C_{i2}, \dots, C_{i\ell}\}, \quad \text{s.t.} \quad \forall j \in [1, \ell] : \mathbf{pat}_{\mathbf{T}}^{C_{ij}} \in i^{\text{th}} \text{ cluster} \quad (3.4)$$

Then the prototype for i^{th} cluster associated with the cluster labels C_i can be computed as

follows:

$$\mathbf{prot}_i = \frac{1}{\ell} \sum_{j=1}^{\ell} \mathbf{pat}_{\mathbf{T}}^{C_{ij}} \quad (3.5)$$

Having efficiently analyzed the patterns, the simulation algorithm will be described in the next subsection.

3.1.3 Sequential Simulation

Once the patterns in the pattern database $\mathbf{patdb}_{\mathbf{T}}$ have been classified, DisPAT proceeds with pattern simulation on the realization grid \mathbf{re} . Initially, all the nodes in \mathbf{re} are uninformed. The classical simulation algorithm paradigm of Deutsch and Journel (1998) is used for the DisPAT algorithm. First, a random path over the realization grid \mathbf{re} is constructed. Next, at each node \mathbf{u} along this random path, the search template \mathbf{T} is used to extract the corresponding data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$. A data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ is defined as the set of data values and their mutual spatial configuration of any hard data and previously simulated values, neighboring \mathbf{u} with the template \mathbf{T} .

$$\mathbf{dev}_{\mathbf{T}}(\mathbf{u}) = \{dev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_1), dev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_2), \dots, dev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_{n_T})\} \quad (3.6)$$

where $dev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_\alpha) = re(\mathbf{u} + \mathbf{h}_\alpha)$. Therefore the data event might have some uninformed nodes. Next a prototype closest to this data event is found. However, different nodes may get different weights according to their status. A node \mathbf{u} can have three statuses:

1. **informed node**: a node in realization \mathbf{re} that has been set by previous visit to another location \mathbf{u}' .
2. **frozen node**: a node in realization \mathbf{re} that has been frozen by previous visit to another location \mathbf{u}' and has been removed from the random path.
3. **hard data node**: a node in realization \mathbf{re} that has initially been set as a hard data for conditioning.

Therefore, the distance from a data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ to a prototype \mathbf{prot}_i can be calculated according to different weights for each node as follows:

$$d\langle \mathbf{dev}_{\mathbf{T}}(\mathbf{u}), \mathbf{prot}_i \rangle = \sum_{j=1}^{n_T} \omega_j \cdot |dev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_j) - \mathbf{prot}_i(j)| \quad (3.7)$$

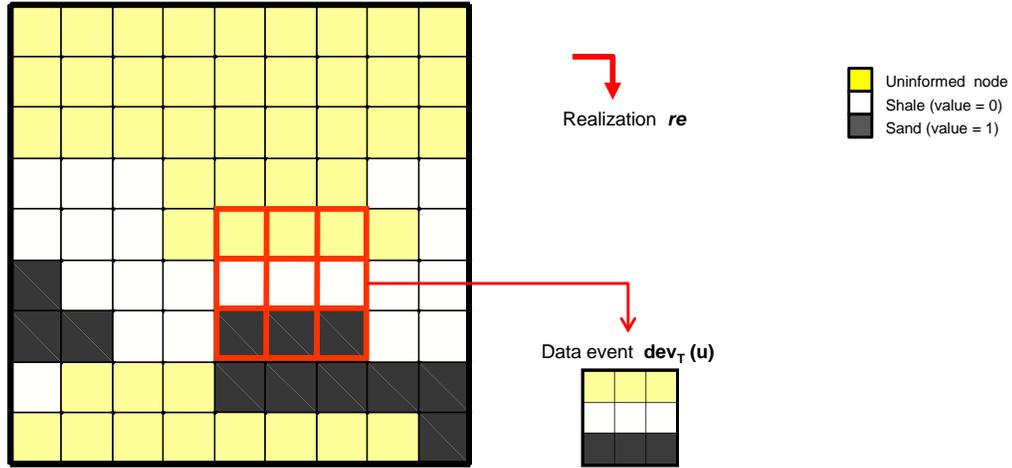


Figure 3.3: A realization re is shown, with a template T of 3×3 used to extract the data event $dev_T(\mathbf{u})$

Afterwards, the closest prototype to the data event $dev_T(\mathbf{u})$ will be chosen, and one pattern will be randomly selected from that cluster and will be pasted on the simulation grid. This pattern has already been conditioned to the previously informed nodes on the realization re according to different weights for each node. However, after pasting the pattern, the nodes located in the inner part of the template T , called inner patch, will be set to frozen nodes. The term *inner patch* refers to the inner part of the pasted pattern, which will be frozen and not revisited during simulation (see Zhang (2006)). The inner patch at node \mathbf{u} is basically a smaller template of size $n_{T'} < n_T$ which is centered at node \mathbf{u} . If instead of freezing the inner patch, on the other hand, the entire pattern is frozen, then the stochasticity of the final realizations would be reduced. This is due to the greater similarities between the patterns of the training image and the ones in the realization grid. By fixing only the inner part, one would be able to introduce more stochasticity, and at the same time, reduce the computational time.

The concept of data event for an arbitrary realization is depicted in Figure 3.3. The concept of pasting a pattern on realization re , where the same template T is used for extracting or pasting patterns and the inner patch is used for removing those nodes from the random path, is also illustrated in Figure 3.4.

In summary, a general description of the MPS simulation within the context of distance-based pattern modeling is provided hereafter. The procedure is also outlined in Algorithm 2.

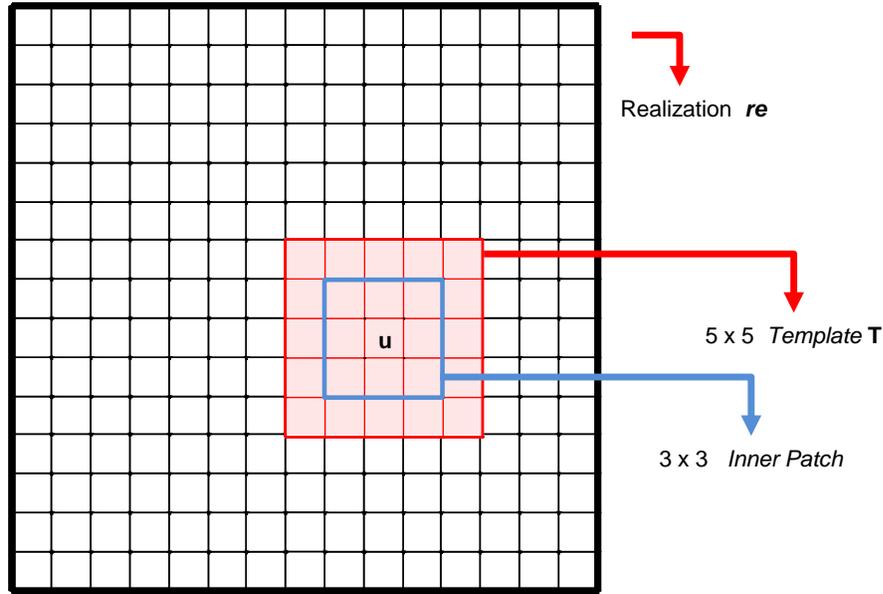


Figure 3.4: A template \mathbf{T} of 5×5 (red) used to extract the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ or paste a pattern, and the inner patch of 3×3 (blue) used for freezing the nodes around location \mathbf{u} after pasting a pattern in order to remove those nodes from random path

This algorithm is similar to the filtersim algorithm of Zhang (2006), except that the modeling of patterns is changed.

- Analyze the training image to obtain the optimal template \mathbf{T} .
- Store all patterns in the pattern database, $\mathbf{patdb}_{\mathbf{T}}$.
- Using the distance-based method map the patterns to points in MDS space, \mathbb{R}^d .
- Map points in \mathbb{R}^d to kernel space F (implicit in algorithm).
- Apply kernel k -means clustering on dataset in kernel space for pattern classification.
- Define a random path on the grid \mathbb{G}_{re} of realization.
- At every node \mathbf{u} along the random path
 - Extract the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ from realization \mathbf{re} .
 - Find the cluster prototype \mathbf{prot}^* closest to $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$.

- Sample a pattern from the prototype class \mathbf{prot}^* .
 - Paste the pattern to the realization \mathbf{re} and freeze the nodes within a central inner patch.
- Move to the next node of the random path and repeat the above steps until all the grid nodes along the random path are exhausted.

Algorithm 2 DisPAT Unconditional Simulation

Require: Set template size \mathbf{T}

- 1: Construct pattern database $\mathbf{patdb}_{\mathbf{T}}$.
 - 2: Construct distance matrix D from $\mathbf{patdb}_{\mathbf{T}}$.
 - 3: Obtain X_d by applying MDS on matrix D .
 - 4: Apply kernel k -means clustering on X_d with k clusters.
 - 5: Calculate cluster prototypes \mathbf{prot}_i for $i = 1, \dots, k$.
 - 6: Define a random path on the grid G_{re} of realization.
 - 7: **for** each node \mathbf{u} along the random path **do**
 - 8: **if** $\mathbf{u} \neq$ frozen **then**
 - 9: Extract the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ from realization \mathbf{re} .
 - 10: **if** all nodes $\in \mathbf{dev}_{\mathbf{T}}(\mathbf{u}) =$ uninformed **then**
 - 11: Randomly select a pattern $\mathbf{pat}_{\mathbf{T}}^*$ from pattern database $\mathbf{patdb}_{\mathbf{T}}$
 - 12: **else**
 - 13: Find prototype \mathbf{prot}^* that minimizes $d\langle \mathbf{prot}_i, \mathbf{dev}_{\mathbf{T}}(\mathbf{u}) \rangle$ for $i = 1, \dots, k$.
 - 14: Randomly sample a pattern $\mathbf{pat}_{\mathbf{T}}^*$ from the prototype class \mathbf{prot}^* .
 - 15: **end if**
 - 16: Paste the pattern $\mathbf{pat}_{\mathbf{T}}^*$ on the realization \mathbf{re} .
 - 17: Freeze the nodes within the central inner patch neighboring location \mathbf{u} .
 - 18: **end if**
 - 19: **end for**
 - 20: **return** realization \mathbf{re} .
-

3.1.4 Distance Function

Distance functions are the core of the distance-based modeling approach. The distance function is purpose-driven and can be tailored to the application at hand. For example, in the context of history-matching, the distance function has been defined in such a way as to measure the similarity between the outputs, i.e., the oil production responses of wells (Caers and Park, 2008). In other applications, a connectivity distance has been used to

differentiate between Earth model responses (Park and Caers, 2007). Hausdorff distance has also been used for parameterizations of geological models (Suzuki and Caers, 2008).

The concept of distance in pattern modeling is not new. Previous MPS algorithms have used distances for pattern similarity searches at different phases of the algorithm. Simpat uses Euclidean distance in order to find the most similar pattern to data event. Filtersim uses Manhattan distance as a measure to find the closest prototype to the data event. In the proposed algorithm, we need to define a distance function as well. However, two different distances are required, one for constructing the distance matrix, and one for finding the closest prototype to the data event.

For constructing the distance matrix, we would need a function that can provide a similarity measure as close as possible to the one obtained from human perception. Pattern-based modeling, similar to many image analysis applications, requires the measurement of patterns, the components of patterns or the relationship between patterns.

One of the mostly used dissimilarity function is the Euclidean distance, or similarly, the Manhattan distance. According to the analysis made by Arpat (2005), dissimilarity measurements using the Manhattan distance gives radically different results from the perception of a human expert. In other words, it finds more similarity between somewhat fewer dissimilar patterns (for more examples regarding this issue refer to Arpat (2005)). Hence, in the proposed methodology, for constructing the distance matrix, we will only use Manhattan distance for continuous cases. For binary images, on the other hand, one technique that may be used in a wide variety of applications is the distance transform or Euclidean distance map (Russ, 2002). The aim of distance transform is to compute the distance of each point of a pattern to a given subset of it. In a binary image, consisting of only object (value of 1) and a background (value of zero), the distance transform assigns each pixel its distance to the closest object (distance to the closest node with value of 1). This process of converting a binary image to an approximate distance image is called distance transformation (DT). ‘‘Chamfer 3-4 algorithm’’ is used to easily and efficiently calculate this distance transform by approximating the Euclidean metric (Borgefores, 1984, 1986).

However, in order to find a similarity measure for constructing the distance matrix, the proximity transform (which is just an inverse normalized transformation of the distance transform) will be used for binary images. After this transformation, each node \mathbf{u} of the pattern holds additional information about the values of the neighboring nodes $\mathbf{u} + \mathbf{h}_\alpha$. A sample illustration of these two transformations is given in Figure 3.5.

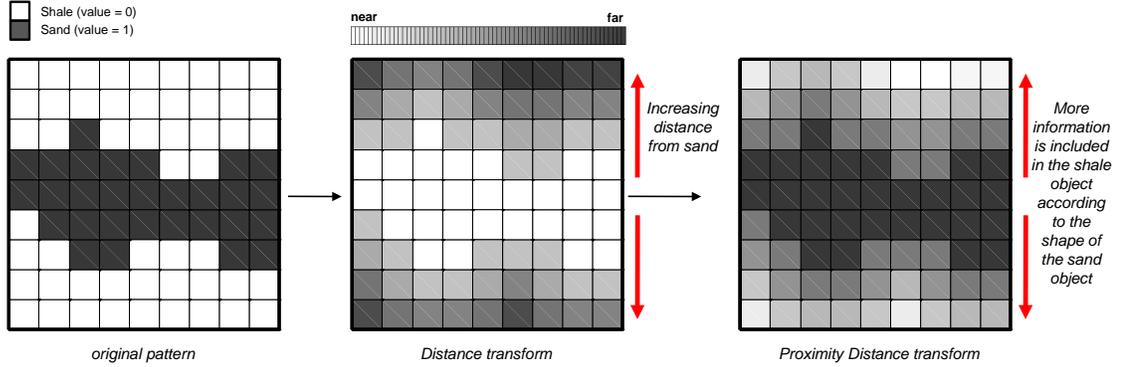


Figure 3.5: A pattern (left), and its distance transform (middle), and the proposed proximity distance transform (right) is shown.

In order to find the dissimilarity between two patterns in this method, the Euclidean distance between the transformed patterns (instead of the patterns themselves) will be measured. By doing so, more information on the neighboring nodes will be included in the distance measure, and a much closer assessment to human perception will be obtained.

It should be noted that the distance transform method, using chamfer’s algorithm, will assign infinity to every pixel in situations where the pattern consists of only a background. In those situations, instead of infinity, the maximum possible value of distance transform, $\sqrt{2}(\sqrt{n_T} - 1)$, is manually assigned to all the pixels of the pattern.

Therefore, for binary images, we have a choice of distance transform, or newly developed proximity distance transform. In order to assess their ability to provide a similarity measure that is more inline with human perception, we will conduct a simple visual analysis on patterns of size 9×9 . Some of the results of dissimilarity measurements using proximity transform and distance transform are shown in Figure 3.6, with their respective notations. Visually speaking, a human expert would categorize pairs of patterns with decreasing similarity as follows:

$$s(\mathbf{pat}_1, \mathbf{pat}_3) > s(\mathbf{pat}_2, \mathbf{pat}_3) > s(\mathbf{pat}_1, \mathbf{pat}_2) \quad (3.8)$$

where s denotes similarity. However as visually observed in Figure 3.6, the distance transform results are counter-intuitive to the human expert’s choice. On the other hand, the proximity distance transform provides an evaluation more in line with the psychovisual model of the human optical system. This is clearly seen in the dissimilarity function values

obtained with these two methods.

The same analysis was also made for Hough and Radon transform distance functions. For both, there was a mismatch with visual perception. However, their results are somewhat accurate as the dissimilarity achieved was:

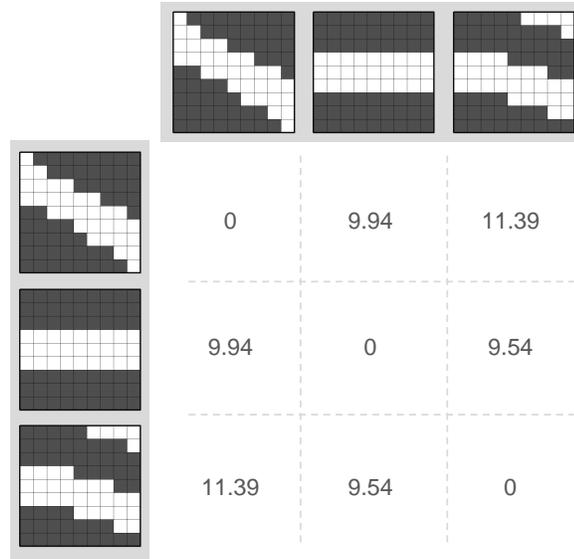
$$s\langle \mathbf{pat}_1, \mathbf{pat}_2 \rangle > s\langle \mathbf{pat}_1, \mathbf{pat}_3 \rangle > s\langle \mathbf{pat}_2, \mathbf{pat}_3 \rangle \quad (3.9)$$

which is a slightly better result than the distance transform's comparison. On these accounts, in cases where there is no prior knowledge on the optimal distance function, we choose proximity distance transform as the one for constructing the distance matrix and eventually performing classification. However, as mentioned earlier, for continuous training images, we would resort back to Manhattan distance since distance transformation only works on binary patterns.

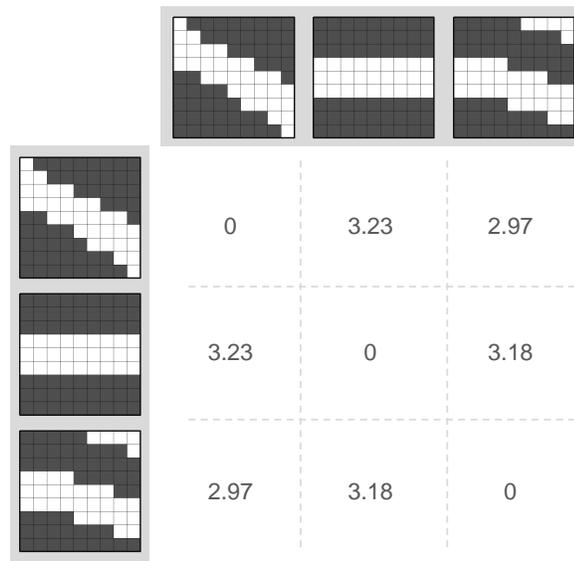
Besides the distance function used for constructing the distance matrix, the proposed MPS algorithm requires another distance definition during the simulation, where a closest match of a data event to the prototypes is sought. In these situations, we would not be able to use the proximity distance function, because the data events may contain uninformed nodes. In case of the existence of uninformed nodes, the algorithm may favor target objects (i.e. sand), or background objects (i.e. shale) inequitably. On the other hand, since the prototypes are already pixel-wise averages of all their patterns within the cluster, they are similar to the proximity distance transform. Averaging a set of patterns will provide a perceptually smooth gradient in prototypes, similar to proximity transform results. Therefore, for the task of finding the most similar prototype \mathbf{prot}^* to the data event $\mathbf{dev}_T(\mathbf{u})$, we will use Manhattan distance for simplicity.

3.1.5 Pattern-Skipping concept

Due to a large amount of patterns in a training image, issues of memory and computational burden may arise. This is mainly caused by the size of the pattern database, n_{Pat_T} , that grows according to the size of training image and the template. Furthermore, the distance calculations between all the patterns imply at least $\frac{1}{2}n_{Pat_T}^2$ similarity calculations, and similarly for their storage. For example, table 3.1 shows the memory needed to store the distance matrix for different sizes of a training image, but with same template size of 11×11 in 2D, or $11 \times 11 \times 5$ in 3D. Clearly, there is memory limitation for storing the



(a) Distance Transform



(b) Proximity Distance Transform

Figure 3.6: (a) distance matrix obtained using distance transform function, and (b) distance matrix obtained using proximity distance transform function. In this figure, pattern at left location is pat_1 , pattern at middle location is pat_2 , and pattern at right location is pat_3 .

Table 3.1: memory requirement to store the distance matrix for different training image, with template sizes of 11×11 in 2D, and $11 \times 11 \times 5$ in 3D

training image	memory storage
100×100	131 MB
250×250	6.64 GB
$70 \times 70 \times 30$	20.2 GB
$100 \times 100 \times 50$	302 GB

distance matrix. Indeed, this limitation can also be perceived simply as a pragmatic one. In other words, one is not obliged to store the distance matrix, D , in the memory, but can access each element of it, d_{ij} , on the fly by re-calculating its value from the patterns $d_{ij} = d\langle \mathbf{pat}_{\mathbf{T}}^i, \mathbf{pat}_{\mathbf{T}}^j \rangle$. However, on the fly access to matrix elements is very CPU-intensive. Therefore, two measures to circumvent this limitation will be provided next.

The first measure to avoid the previously mentioned drawback is the idea of pattern skipping introduced by Arpat (2005). Pattern skipping is the process of omitting some patterns from the pattern database. This task is done by skipping the neighborhood patterns next to the one being accepted into the pattern database. For instance, by having a skip size of two lags, every other pattern is skipped. That is, if one pattern is located at location \mathbf{u} as its center point, the next pattern to be considered will be located at $\mathbf{u} + 2\mathbf{h}$; (\mathbf{h} being the distance between grid cells). An illustration of pattern skipping concept is given in Figure 3.7. By this method, the size of the pattern database, and also, the dimensions of the dissimilarity distance matrix will be reduced by a factor of $(\text{skip} - \text{size})^2$, which will significantly reduce the computational time. This method does not pose any problem in pattern reproduction, because patterns are usually just a translation of their neighboring patterns, hence very similar to each other. By skipping similar existing patterns, the database still covers nearly every possible pattern.

However, using the pattern skipping concept, in a large training image (i.e., $100 \times 100 \times 50$) the skip size becomes very large, even larger than the template size itself. There is a significant amount of patterns that will be dropped by having such a large spatial lag between the stored patterns. It can pose undesirable effects during MPS simulation. Two issues may arise because of this:

- Multiple-point statistics of the training image that is expressed through the pattern

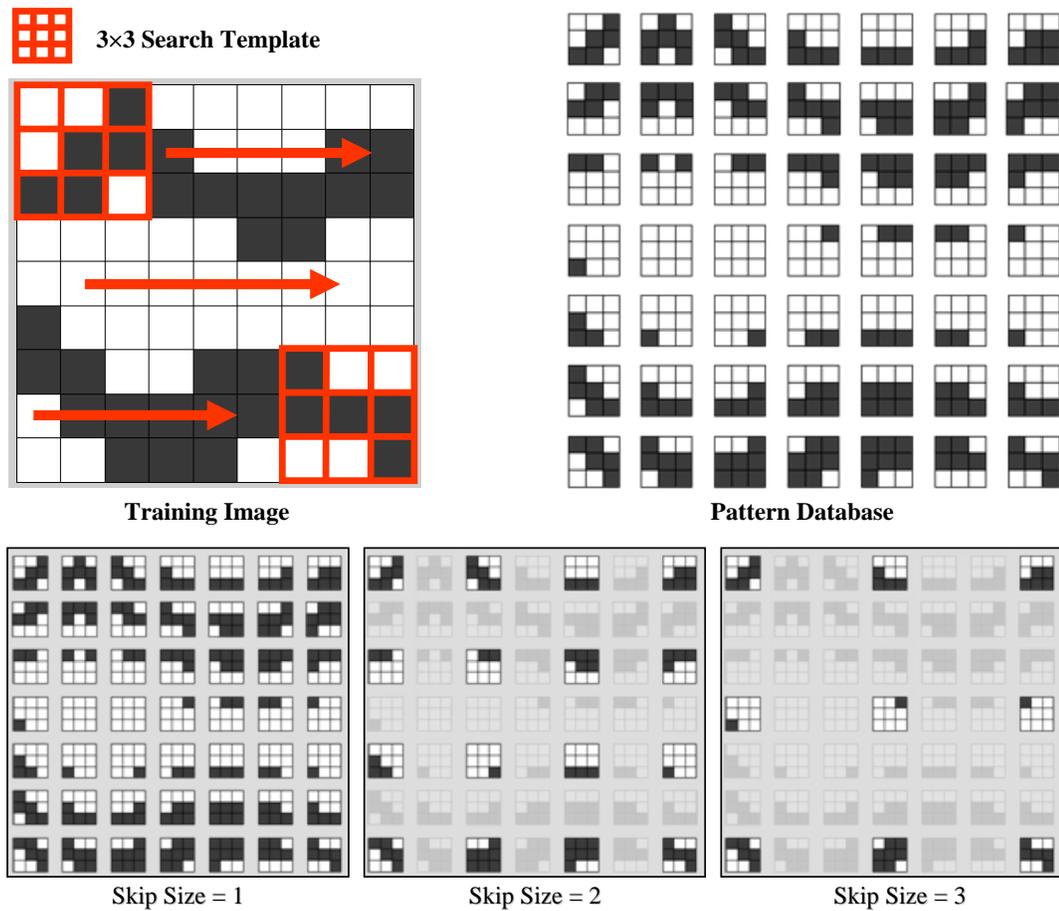


Figure 3.7: Showing training image of 9×9 , scanned with a 3×3 template resulting in all possible patterns. Concept of pattern skipping is shown for skip-sizes of 1 (None), 2 and 3. (picture after Arpat (2005))

database is no longer honored. For example, there would not be enough samples to accurately model the one-point probability distribution function, histogram.

- Search for the most similar patterns or prototypes to the data event would not deliver the desired similarity for the optimal pattern, and hence, quality of pattern reproduction in generated realizations degrades.

Therefore, relying only on the pattern skipping concept to circumvent memory issues has its own drawbacks. A new measure to mitigate these issues, called "smart skipping" concept, is explained hereafter.

The smart skipping concept attempts to provide a more accurate performance. Unlike the previous pattern skipping approach, it will not drop the patterns according to their locations. In this method, patterns are analyzed and selected according to their similarities. For example, if the algorithm can only retain a handful of patterns for further analysis, then the smart skipping algorithm is required to select the most representative set of patterns from the entire database. This way, one can ensure that no memory issues will surface for constructing the distance matrix.

The smart skipping concept will need to select the most representative set of patterns, i.e., selecting 8000 patterns when the database has 500000 patterns. One cannot use the previous distance-based analysis to perform such a task because the whole purpose is to avoid constructing a memory-intensive distance matrix. Therefore, another measure that does not rely on the distances between the patterns, but can provide an approximate representation of them is sought. One simple solution is principal component analysis (PCA). By applying PCA on the patterns, we will retain only the four largest eigenvalues. We choose four, since we have accepted any possible approximation, and we believe that this phase of the process is not detrimental to the whole simulation algorithm. After applying PCA, a dimensionality reduction to dimension of four is obtained for each pattern. In other words, each pattern is now represented by a four-dimensional vector. Therefore, in order to find the required number of representative patterns, we will apply a simple k -means algorithm on the four-dimensional patterns. The clustering algorithm is computationally intensive when a large number of clusters (as in our case) are needed. In order to make the whole process feasible, we only allow five iterations of the k -means algorithm. Again, there are three reasons why such an approximation is acceptable. First, five iterations are enough because of the convergence speed of the k -means algorithm in the initial iterations in comparison to the rest. Second, the selected representative set still holds a great number

of the patterns, and even a random selection can provide acceptable results, let alone five iterations of the k -means algorithm. And third of all, the k -means algorithm works more accurately, with less possibility of being trapped in local minima, in situations where the dimensionality of the data has been reduced.

In summary, such an approach of applying PCA on all the patterns and selecting a smaller set out of them by a few iterations of k -means algorithm is greatly beneficial for constructing the distance matrix, and generally, a more accurate pattern-based modeling.

3.2 Comparison

Having introduced various new techniques for pattern analysis and classification for our MPS simulation framework, we will now compare them with filtersim. Filtersim method is chosen for comparison because of its similarities to DisPAT. Simpat, on the other hand, does not perform any classification on the patterns. During simulation, simpat finds the most similar pattern to the data event by an exhaustive search in the pattern database. The classification approach, used in the pattern-based MPS method of filtersim or DisPAT, provides tremendous computational improvements over simpat. In the following subsection, two different aspects of the algorithm will be illustrated. First, a comparison with filtersim in terms of pattern classification quality is conducted, where the superior classification abilities provided through distance-based modeling will be demonstrated. Next, a simple unconditional single-grid example is provided with illustrations on each step of the methodology.

3.2.1 Clustering Accuracy

In this section, the quality or accuracy of the clustering methodologies will be compared. For clarity, the detailed workflow of pattern classification is outlined in Algorithm 3.

A binary training image of size 101×101 that was also used to illustrate the previous concepts is shown in Figure 3.8. It consists of channel structures as the spatial features of the subsurface. A template of size 9×9 is used to define the essential patterns of this training image. Multi-dimensional scaling is performed on the pattern database and a lower dimensional representation of the patterns is obtained. Kernel mapping is then applied in the MDS space to linearize the data structure. The resulting MDS space and

Algorithm 3 Pattern Classification

-
- 1: $n_{Template} \leftarrow$ Pattern Template Dimension
 - 2: **for all** ($n_{Template} \times n_{Template}$) patterns **do**
 - 3: store in database
 - 4: **end for**
 - 5: Construct dissimilarity Matrix Δ
 - 6: $d \leftarrow$ Choose MDS dimensionality
 - 7: **for all** overlapping sets of patterns **do**
 - 8: **if** first set **then**
 - 9: Map the patterns in that set $\xrightarrow{\text{MDS}}$ points in \mathbb{R}^d
 - 10: **else**
 - 11: Map the (patterns in that set + overlapping points with previous set) $\xrightarrow{\text{MDS}}$ points in \mathbb{R}^d
 - 12: **end if**
 - 13: **end for**
 - 14: Map points in $\mathbb{R}^d \mapsto$ kernel space F (implicit in algorithm)
 - 15: K-means clustering in kernel space
-

kernel space are shown in Figure 3.9. The classification is then applied in kernel space and a set of 25 clusters were obtained (Figure 3.10). The cluster prototypes are shown in Figure 3.11. Different skip-sizes and multiple-grid values were tested as well as various distance functions. The classification result shown here is for the case of proximity distance transform method and a 12-dimensional feature space with 25 clusters. It is notable that by having 12 dimensions, which is a huge reduction from the original 81-dimensional pattern, a correlation coefficient of 0.99 is obtained.

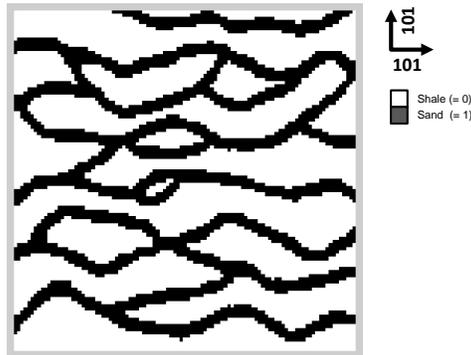


Figure 3.8: Training Image used in this study having 101×101 dimensions with 9×9 template.

The benefit of MDS space to feature space transformation is illustrated in Figure 3.9. As one can observe, the unstructured scatter of data points in the Euclidean space are now distributed in a more structured fashion in the feature space (shown using the first three principal components in that space).

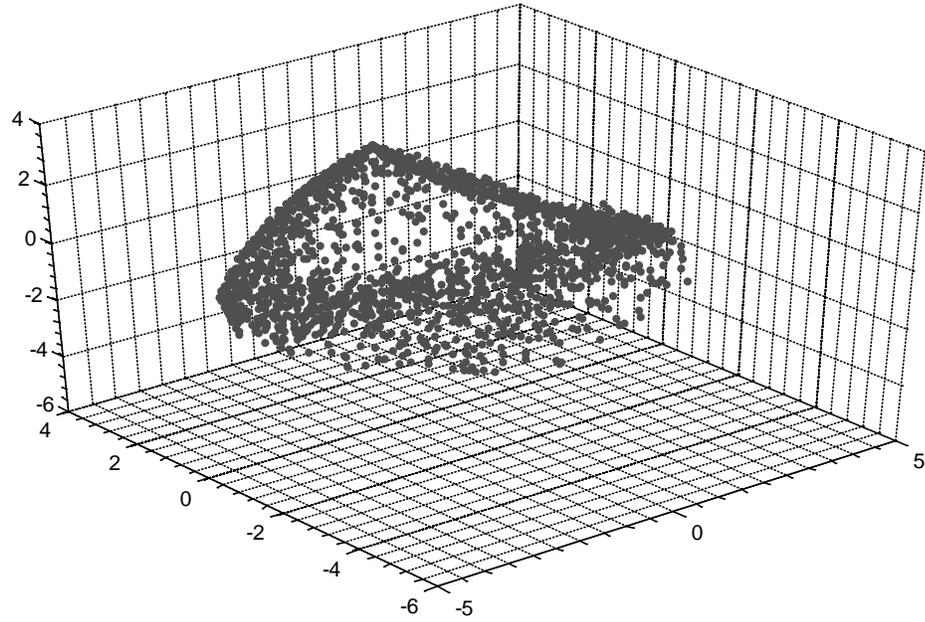
Finally, the prototypes of the resulting clusters are shown in Figure 3.11. Prototypes are pixel-wise averages of all the patterns in each cluster. Recall that if the classification was successful, then the patterns within one cluster would be similar. Visually, this translates to prototypes that are “sharp”, i.e. contain values close to zero or one. In other words, if all the patterns in one cluster are exactly the same (perfect classification) the average prototype will be exactly equal to a sample pattern from the cluster, and as such, a very sharp prototype (distinct black and white pixels) will result. On the other hand, suppose that a cluster has two completely different patterns (any pixel value of 1 in one pattern will be 0 in the other one). In this case, the average prototype will have all the pixels equal to 0.5, which results in gray prototypes (assuming a black-to-white colorbar). For instance, cluster 23 in Figure 3.11 is considered sharp, while cluster 20 is not. According to these qualitative visual concepts, a sharpness index which can aid in validating the classification accuracy is introduced. In simple words, if a pixel in a prototype is close to 0.5, which indicates poor clustering, it is rated as 0 and for the pixels close to 1 or 0, which indicates similarity of the patterns within that cluster, a rating of 1 is assigned. The sharpness index is then just an averaged sum of all these values. The formula for sharpness index of i^{th} prototype is computed as follows:

$$\text{SI}_i = \frac{1}{n_T} \sum_{j=1}^{n_T} |2 \times \mathbf{prot}_i(\mathbf{h}_j) - 1| \in [0, 1] \quad (3.10)$$

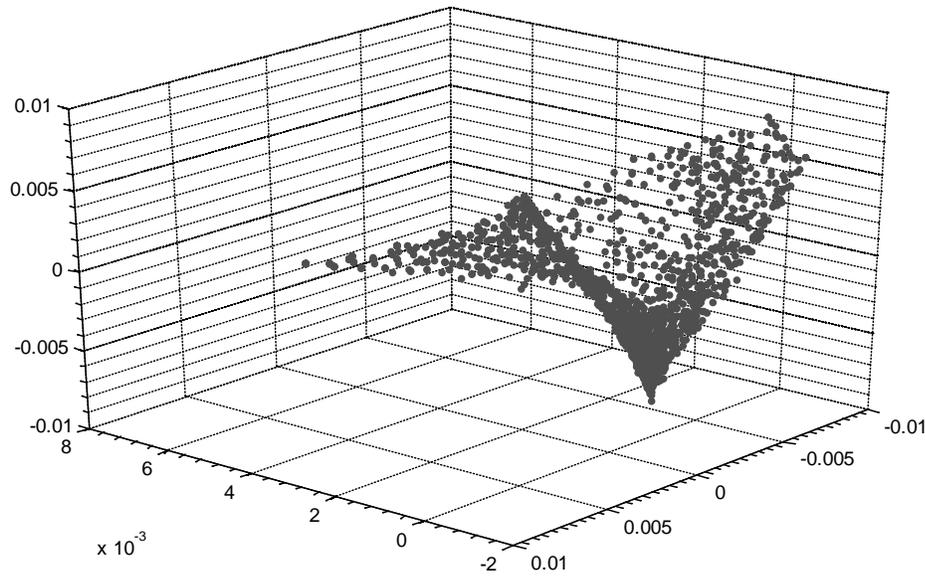
where $\mathbf{prot}_i = \frac{1}{n_{\ell_i}} \sum_{k \in \ell_i} \mathbf{pat}_{\mathbf{T}}^k$. The sharpness index, used for quality assessment of the clustering algorithm, is thus obtained as follows:

$$\text{Sharpness Index} = \text{SI} = \frac{1}{k} \sum_{i=1}^k \text{SI}_i \in [0, 1] \quad (3.11)$$

where k represents the number of clusters. However, suppose that a pattern classification, with 25 clusters, resulted in 24 clusters having only one pattern and the last cluster contains all the remaining patterns in the database. Obviously, such a clustering scenario is of poor quality. But it results in a high sharpness index, because the 24 perfect clusters, each with



(a) MDS Space



(b) Feature Space Projection

Figure 3.9: (a) MDS space of patterns, (b) 3D feature space projection using kernel PCA.

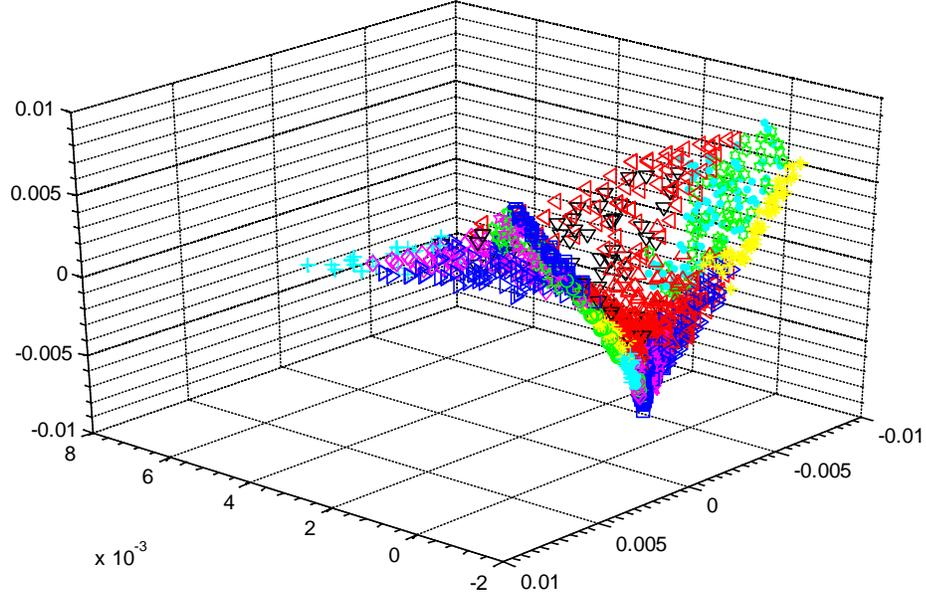


Figure 3.10: Kernel k -means clustering of 25 clusters in feature space.

$SI = 1$, are averaged with only one cluster having $SI \approx 0.5$. The average overall sharpness index of this poor clustering would be $\frac{24 \times 1 + 0.5}{25} \approx 0.98$, which adversely indicates excellent clustering quality. Therefore, to compensate for this situation, a weighted average of each pattern sharpness index is calculated, with the weight being the number of patterns in each specific cluster. As a consequence of this, the situation described above would correctly result in a small sharpness index, indicating a poor classification. The following formula is therefore proposed:

$$\text{Weighted Sharpness Index} = \text{WSI} = \frac{1}{k} \sum_{i=1}^k \left(\frac{n_{\ell_i}}{n_{\text{pat}_{\mathbf{T}}}} \right) SI_i \quad (3.12)$$

It should be noted that, although this measure of calculating the sharpness index is accurate, one shortfall happens due to the pixel-wise averaging technique used in obtaining the cluster prototypes themselves. It relies on the same Euclidean distance comparison method that was deemed unreliable (Arpat, 2005); that is to say, a “sharp” prototype results from averaging the patterns that are similar in the pixel-wise sense. This form of averaging has a misleading effect when comparing different distance functions for their final classification capability. To avoid this misinterpretation, two other sharpness indices were calculated accordingly. For both, the prototypes are obtained by averaging the proximity

transformed patterns, instead of the original patterns. In this way, each prototype would be more informative of all the patterns within that cluster.

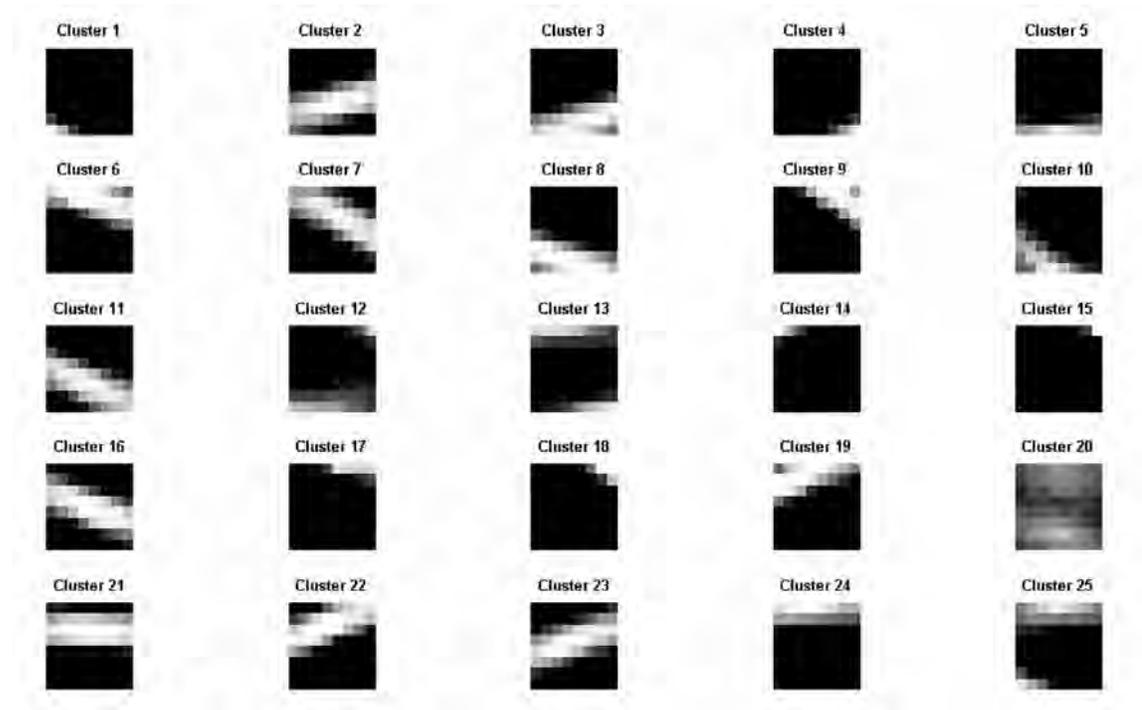


Figure 3.11: Cluster prototypes for all 25 clusters, more sharpness means better clustering

The proposed algorithm in this study presents a very general and powerful technique for pattern classification and analysis. However, being a new algorithm, a comparison with the classification technique employed in the filtersim algorithm is made. As mentioned before, filtersim uses six different filters to map the patterns into a six-dimensional space, where the patterns are classified according to their score values. On the other hand, in this analysis, only one measure - the dissimilarity between patterns - is used for pattern classification. Another advantage is that by mapping the patterns to a lower dimensional MDS space, different techniques such as k -means clustering, PCA and kernel mappings can be easily applied on the data points. In order to judge the effectiveness of the proposed algorithm a simple training image is chosen, and the pattern clustering capabilities of both methods are compared in terms of sharpness indices.

The training image used here is shown in Figure 3.12. As can be seen, this training image looks like the previous one (shown in Figure 3.8), but the dimensions of this training

image are reduced to 51×51 which is almost half the previous one. The patterns are identified using a 9×9 template. The reason of using a lower-resolution training image with the same template size as before is the introduction of more dissimilarities between the patterns in the database, and hence, a more difficult classification task. This can help us to observe the differences more clearly. In this case study, 20 clusters are chosen for the final results. In filtersim, for the sake of comparison, the better clustering algorithm of k -means (as opposed to the other option of cross partitioning) is used as the classification method.

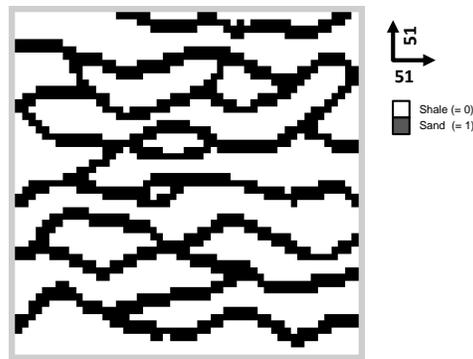
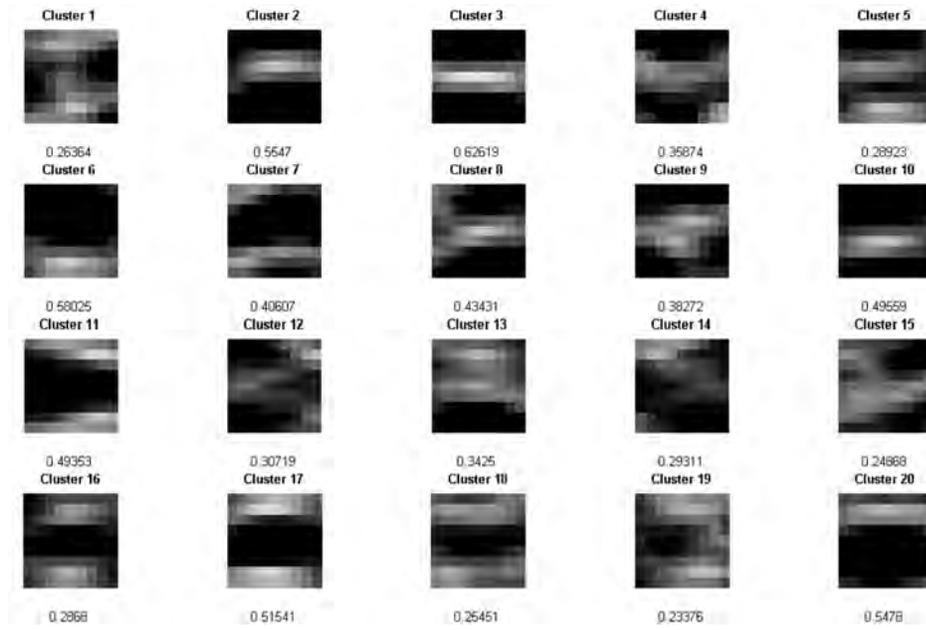


Figure 3.12: 51×51 Training Image used for comparison of DisPAT and Filtersim with a 9×9 template

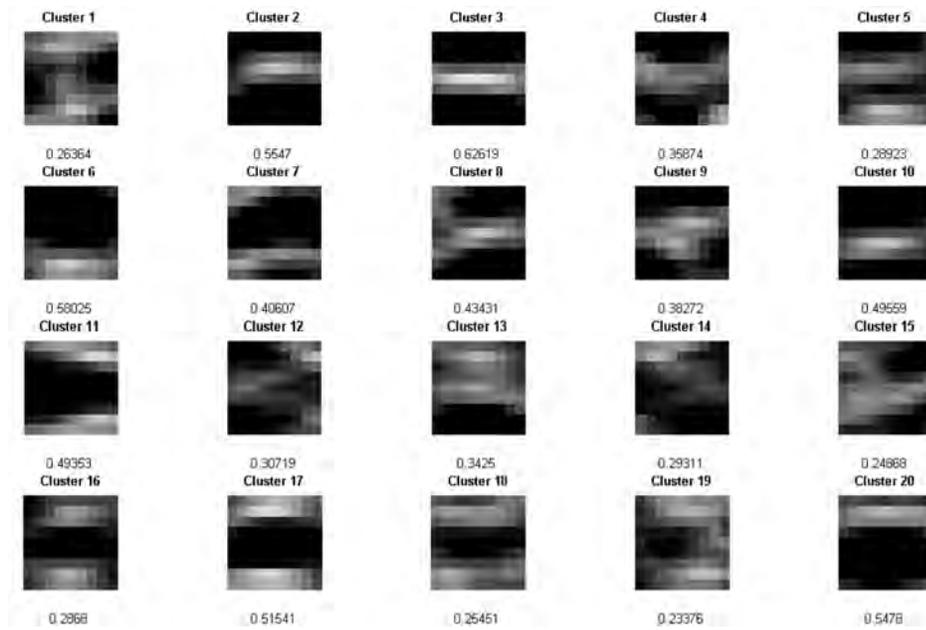
For comparison, one can look at the cluster prototypes. As was presented previously, the sharpness index of a prototype is a good measure for the accuracy of pattern classification. To this account, Figure 3.13 illustrates the prototypes of each cluster obtained by DisPAT and filtersim algorithms. The sharpness index of each prototype is written below them. A simple visual comparison demonstrates the better classification obtained by DisPAT. Nevertheless, numerically speaking, Table 3.2 summarizes the average and weighted average sharpness indices for each method. The huge difference between these values demonstrates the powerful classification capability of DisPAT in comparison with the method employed in filtersim.

3.2.2 MPS Simulation

This section shows the application of single-grid unconditional MPS simulation using DisPAT algorithm. A binary sand/shale training image of channels of size 51×51 is chosen



(a) Filtersim sharpness index



(b) DisPAT sharpness index

Figure 3.13: Comparison of clustering algorithms using (a) filter-based analysis in filter-sim, and (b) distance-based analysis in DisPAT.

	Sharpness Index	Weighted Sharpness Index
FilterSim Algorithm	0.3957	0.3996
DisPAT Algorithm	0.5237	0.5236

Table 3.2: FilterSim comparison with DisPAT in terms of sharpness index

to show the patten reproduction obtained with distance-based modeling. The same training image is also used in filtersim algorithm to show comparisons that emanate from the differences between classification qualities of the two algorithms.

The training image and two different realizations, using DisPAT, is shown in Figure 3.14. The number of clusters used in this algorithm is set to 50, and the inner patch is always set to half of the template size. As can be observed through the two sample realizations, increasing the template size from 5×5 to 9×9 will aid in better pattern reproduction and log-range continuity of channels. These realizations are constructed unconditionally. Therefore, the E-types (ensemble averages) obtained by averaging 1000 realizations should not display any features or structures. This is because the realizations are generated stochastically by using different seeds and random paths. Stochastic simulation of patterns will result in stochastic placement of the structures within the training image, and hence, no spatial certainty on the location of channels or sands can be observed through the E-type. The E-type converges to a constant image, where the constant is the mean of the input training image.

The same training image of size 51×51 is used for generating realizations using the filtersim algorithm. All the other algorithm-specific parameters are similar to DisPAT simulation. Two sample realizations with template sizes of 5×5 to 9×9 are shown in Figure 3.15. The realizations are not honoring the multiple-point statistics given by the training image. It can be attributed to the poor classification capabilities through filters. It causes uninformative prototypes, which do not have high sharpness indices, and as a result, the search for the most similar pattern is not optimal. Also, there exist some dissimilarities among the patterns within one cluster. In other words, the search for the most similar pattern to the data event does not reach a local minimum. During simulation, inappropriate patterns get pasted on the simulation grid. This can distort the realization, and bring undesirable features to the generated models; such as discontinuity between the channels.

Therefore, one can conclude that the distance-based modeling platform furnishes MPS

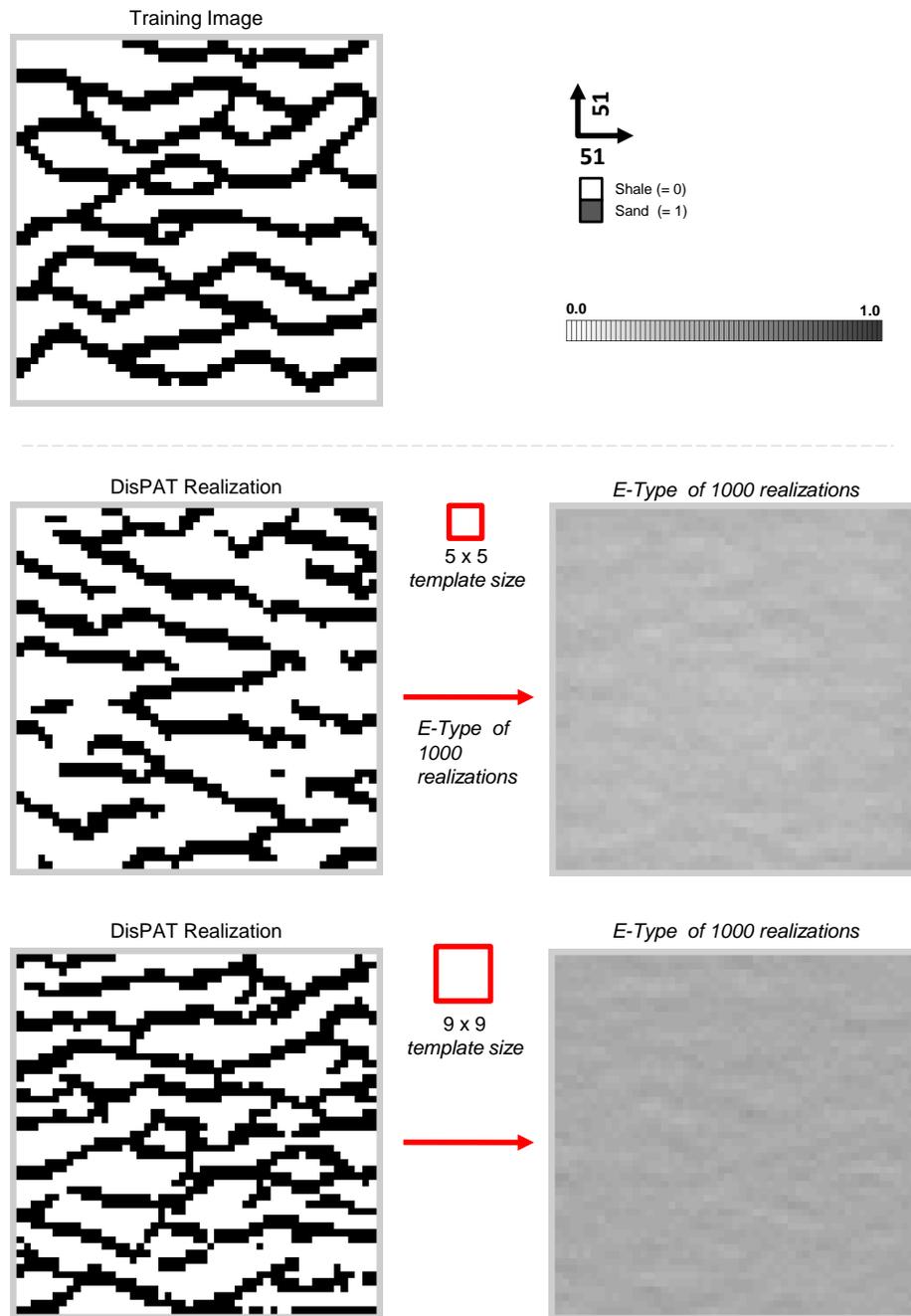


Figure 3.14: 51 × 51 Training Image used with DisPAT algorithm to generate single-grid unconditional realization with two different template sizes of 5 × 5 and 9 × 9. The E-types of obtained from each template size over 1000 realizations is also shown next to the realizations.

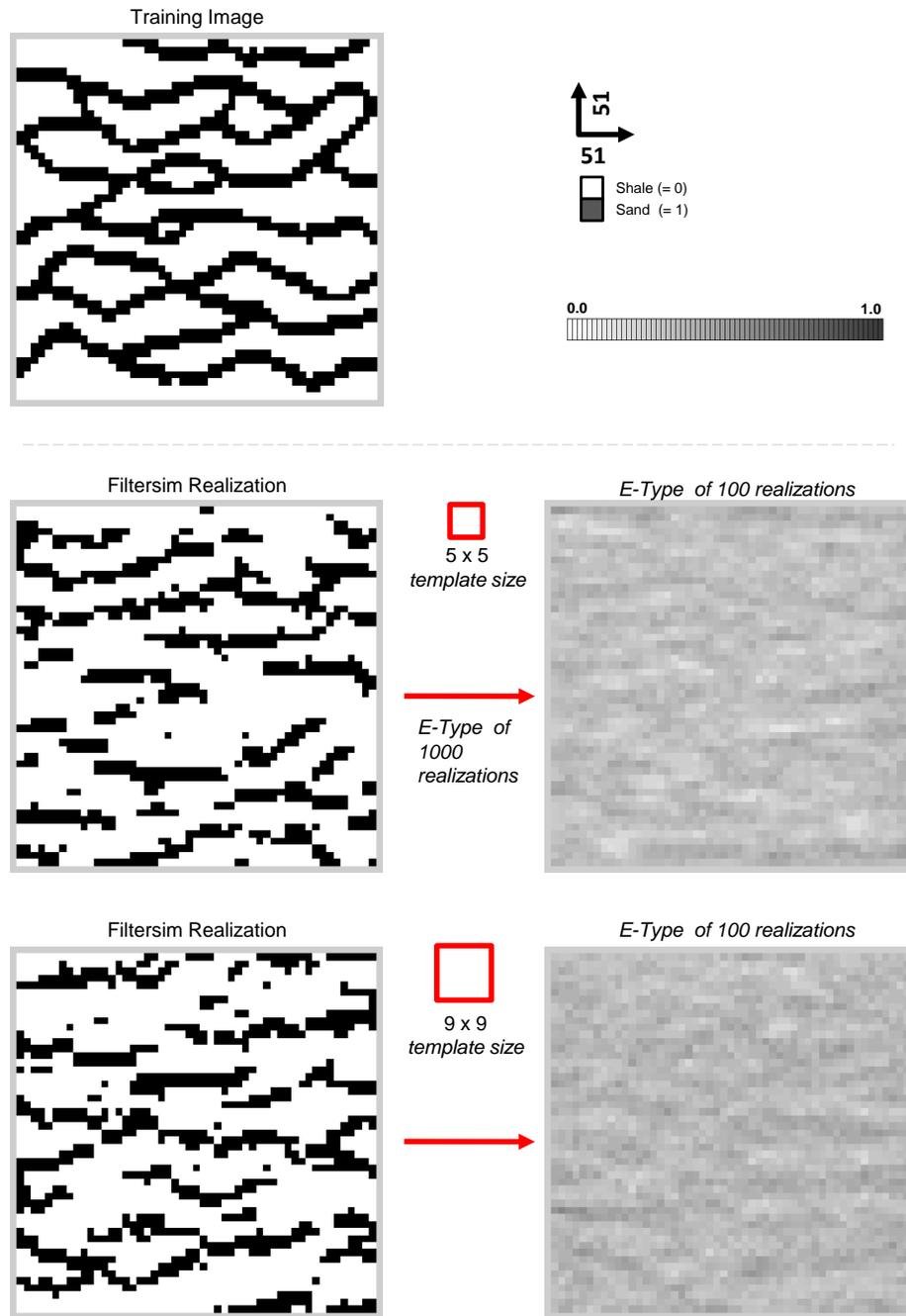


Figure 3.15: 51×51 Training Image used with filtersim algorithm to generate single-grid unconditional realization with two different template sizes of 5×5 and 9×9 . The E-types of obtained from each template size over 100 realizations is also shown next to the realizations.

methodology with a set of tools that, combined, can provide acceptable realizations. Improvements of continuity and large-scale reproductivity of patterns, which relates to long-range spatial statistics, is the topic of following chapter. In this section, a fine-scale MPS simulation of the channel training image demonstrated the superiority of the proposed stochastic pattern modeling framework. In the final section of this chapter, we will examine model variability between realization and the internal inconsistencies inherent to different MPS techniques.

3.3 Study of Model Variability

Multiple-point statistics is a study which can be pursued in two opposite directions. The more familiar direction is how visually appealing the realizations are; from pattern reproduction to connectivity, to conditioning, and so on. The other direction, which is less familiar, is about the uncertainty and model variability between generated Earth models. Instead of asking how good the models are and how well they resemble the given conceptual training image, we ask instead how well does the algorithm honor the multiple-point statistics and at the same time introduce maximum variability. For example, one of the arguments against simpat algorithm is that it reduces uncertainty in the output realizations; that, the realizations look very similar to the training image. This is rooted in neglecting the core assumptions that we have set forth in multiple-point statistics. The degrees of abstraction can be confusing. In this section, we would like to elaborate on the ideas of uncertainty, internal inconsistency, and model variability.

3.3.1 Concepts

We will find that by analyzing our mathematical notions, we could acquire fresh insights into our stochastic modeling techniques. First and foremost, we need to define the concept of stationarity. In geostatistics, we rely on random functions that define the statistical models of spatial variability such as mean, and variograms. In multiple-point statistics, this random function holds the joint distribution of the multiple attributes. In general, we need to somehow extract these statistical moments. For example, in order to calculate the porosity in a field, the engineer looks at porosities obtained through well data and infers some statistics from his sample. He will assume that the mean of the attribute porosity

is the same all over the field. This assumption about porosity means that all the data are coming from the same source, or the same cumulative distribution function. In other words, one assumes stationarity within this field. Therefore, a stationary process has the property that its mean, variance, or any other of its statistical moments does not change in space or time. For example, a first-order stationary process can be defined as a process where its first order density function does not change in space or time. If we assume the random variable to be X , then $E[X] = \text{constant}$. In two-point statistical simulations, one assumes first and second-order stationarity by defining a spatial model of variability through the mean and a variogram. Mean determines the one-point statistics, and variogram defines the two-point statistics prior model.

The same holds true in multiple-point statistics. A training image can be seen as a statistically explicit prior random function model that provides the statistics needed for simulation; single-point (histogram), two-point (variogram), or multiple-point information (Journel and Zhang, 2006). Thus, MPS algorithms rely on stationarity in order to be able to extract these statistics from the conceptual training image. Otherwise, without any additional sources of information, one cannot deduce any conclusive statistics from the training image. For example, if a Gaussian model exhibits a spatial trend, then we cannot infer the mean of that attribute, spatially. In MPS techniques, the situation is more complex. We need a rich training image that can provide enough statistics to give an empirical joint multivariate distribution of the subsurface. Therefore, stationarity is the most important assumption in multiple-point statistics, and stochastic modeling framework.

Traditional two-point statistics get the prior model in terms of means and variograms. Then, the algorithm is formulated in such a way that it honors those statistics. So given a large realization grid, one should be capable of obtaining the same statistics within the generated realizations. For instance, the realizations have the same input histogram and the same variogram as in the prior model. We expect the same in multiple-point statistics. The generated realizations should honor the same prior models.

Therefore, to compare different methods, assess the variability between the generated models, or analyze the uncertainty space of realizations, we need to favor those algorithms that behave comparably similar to an ideal algorithm. An algorithm which provides results not in accordance with an ideal algorithm is having internal inconsistency. In other words, the algorithm is not consistent with its own assumptions and characteristics in principle. In the next subsection, we will introduce the mathematical instruments used to quantify how

close an algorithm is to an ideal one.

3.3.2 JS-MPH Methodology with Example

Traditional methods of comparing two methodologies lied in the visual inspection of their resulting realizations. A practitioner would decide with his/her own subjective judgement if one method is superior to another. In order to eliminate this subjectivity, one needs to quantify the pattern reproductivity of each method. In other words, one would try to analyze the multiple-point statistics of each algorithm with the training image. The closer they are together, the better the performance of the algorithm. At the same time, the more the perceptual variability among realizations, the better the stochasticity among resulting realizations. So, we are looking for a method that provides more variability between the models and less error in multiple-point statistical reproductions.

For the first characteristic, two-point statistics cannot often fully characterize the geological arrangements containing curvilinear features and other complex shapes. Therefore, a common multiple-point statistic, called multiple-point histogram, is chosen for the analysis (Deutsch and Gringarten, 2000). For example, with a multiple-point template of the size $3 \times 3 \times 2$, every single binary configuration of this template is searched in the realization and their frequencies are obtained. This multiple-point template can produce $\sum_{i=1}^{18} (2^{i-1}) = 262143$ different configurations. These multiple-point statistics also contain all lower-order statistics as well. For computational purposes in this example, and because of the limitations in real cases, we assume that the ‘essence’ of the training image can be captured by this choice of template. By indexing each configuration, one can plot the frequency with respect to the configuration. The multiple-point histogram (MPH) can be plotted for the training image (reference) and the realizations for each method. There are always some differences with the reference multiple-point histogram due the inconsistencies in the algorithm. Be that as it may, one can calculate the absolute error between these histograms as a measure of the goodness of the realization. The error is obtained according to the norm-1 difference as follows:

$$\text{error} = \sum_{k=1}^d |C_k^{TI} - C_k^{\text{realization}}| \quad (3.13)$$

where C_K is the count of the patterns with configuration index of k , and d is the number

of configurations, $d = \sum_{i=1}^{n_x n_y n_z} (2^{i-1})$, where n_x, n_y, n_z are the multiple-point template dimensions.

One may argue that the absolute error between the histograms is not a good measure since the proportions of sand and shale can significantly change the one-point statistics of a histogram. In order to obtain a better measure, we calculate the Jensen-Shannon (JS) divergence (Endres and Schindelin, 2003). It is a popular method of measuring the similarity between two probability distributions, which is based on the Kullback-Leibler divergence. The Kullback-Leibler (KL) divergence is a measure in statistics that quantifies in bits how close a probability distribution $p = \{p_i\}$ is to a model (or reference) distribution $q = \{q_i\}$ (Cover and Thomas, 1991),

$$D_{KL}(p \parallel q) = \sum_i p_i \log \frac{p_i}{q_i} \quad (3.14)$$

D_{KL} will be zero when the distributions exactly match. Because of the non-symmetric behavior of KL divergence, a similar divergence will be used instead. Jensen-Shannon divergence (JS) is a symmetrized and smoothed version of the Kullback-Leibler divergence $D_{KL}(p \parallel q)$. It is defined by

$$D_{JS}(p \parallel q) = \frac{1}{2} D_{KL}(p \parallel m) + \frac{1}{2} D_{KL}(q \parallel m) \quad (3.15)$$

where $m = \frac{1}{2}(p + q)$. Intuitively speaking, a common technical interpretation is that the JS divergence is the coding penalty associated with selecting a distribution q to approximate the reference distribution p . In the case of the multiple-point histogram, the closer D_{JS} is to zero, the closer are the two histograms, and hence, the better pattern reproductivity.

In the previous sections, the proposed methodology was compared to filtersim using the channel training image. It was shown that the proposed method can generate more geologically realistic realizations. Also, there is a much better pattern reproduction and large-scale spatial continuity of the structures (more examples will be provided in the next chapter). One may argue that the generated realizations look more similar to the training image, and therefore, there is less variability between the generated realizations in comparison with the realization obtained by filtersim. In other words, the space of uncertainty spanned by the proposed method realizations may be seen as rather limited in comparison with filtersim. This, for example, may affect the uncertainty analysis that a reservoir engineer makes for reservoir characterization. This section will address the issue of variability observed in the

realizations and verify these assertions.

In order to roughly visualize the uncertainty space spanned by a set of realizations $\mathcal{L} : \{l_1, l_2, \dots, l_N\}$, we select a distance function that can approximately provide a measure of variability between any two realizations. This distance function should have the property that if two realizations have the same structures and patterns, then it would give zero as their dissimilarity. Euclidean distance and other variants related to $f(\|l_i - l_j\|)$ does not have this capability since they only account for collocated locations. A better distance function to quantify the variability between two realizations should instead account for multiple-point information. Therefore, it must be a function of the multiple-point histogram of the two realizations. In other words, $f(l_i, l_j) = g(\text{MPH}(l_i), \text{MPH}(l_j))$, where f represents the distance function and g represents another distance function defined over two probability distributions. In our analysis, we use Jensen-Shannon divergence (JS) as the distance function between two multiple-point histograms.

Having defined a distance function that can approximately quantify the variability between any two realizations, we can map all the realizations into the MDS space. The variability obtained using a specific technique can be explored by visual analysis of the cloud of MDS points. The larger the cloud, the higher the variability in the generated realizations. It should be mentioned that all the analysis is approximate due to the infeasibility in the calculation of the multiple-point information histogram at all scale. As mentioned earlier, in a binary case, a square template of size $t \times t$ will result in a multiple-point histogram vector of the size 2^{t^2} . This prevents us from calculating any multiple-point information beyond a 4×4 template in Matlab. However, it is sufficient for visualizing the space of variability.

The channelized 2D training image (sand/shale) is used for this analysis. 100 realizations are generated using the proposed method and `filtersim`. After applying the dissimilarity distance function between each pair of the realizations, MDS mapping is performed and the three most significant coordinates are used for illustration. The results are shown in Figure 3.16. By looking at the 3D plot and three 2D views of it, one can see the larger cloud of points for the `filtersim` case.

Showing the cloud of points in three dimensions is unsuitable for this case as the dimensionality of the data is larger than three. Therefore, each dimension (or a coordinate) will be shown on the x -axis and the coordinate value will be shown on the y -axis. This

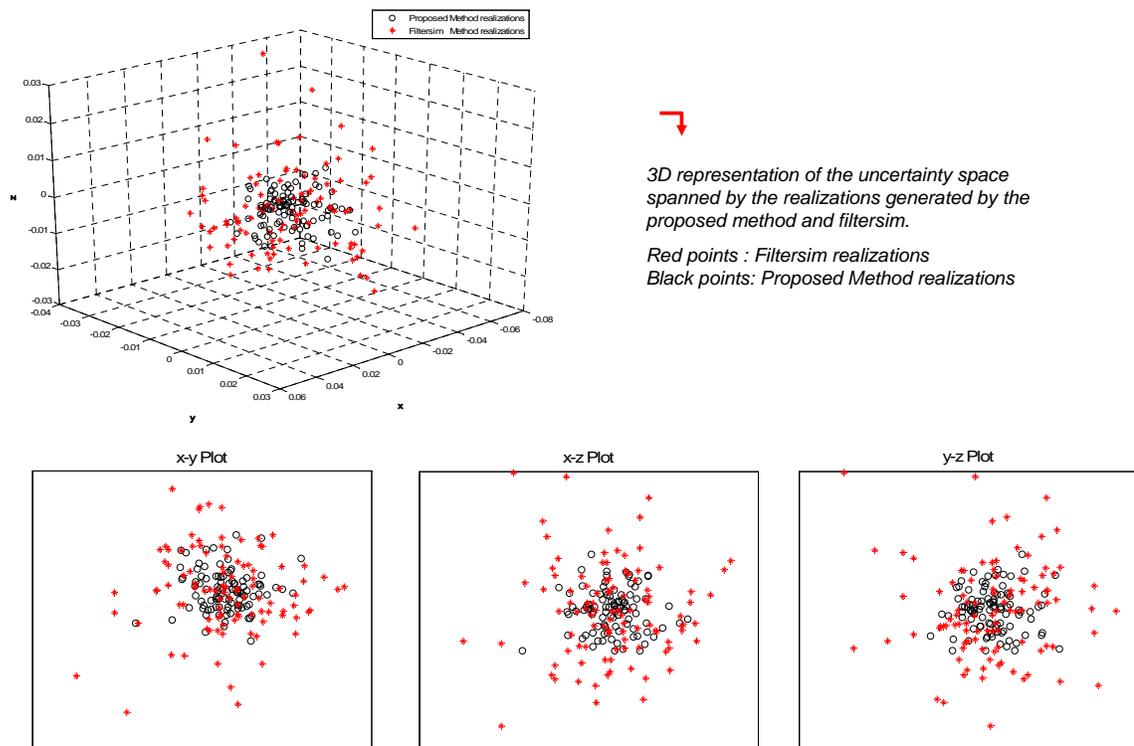
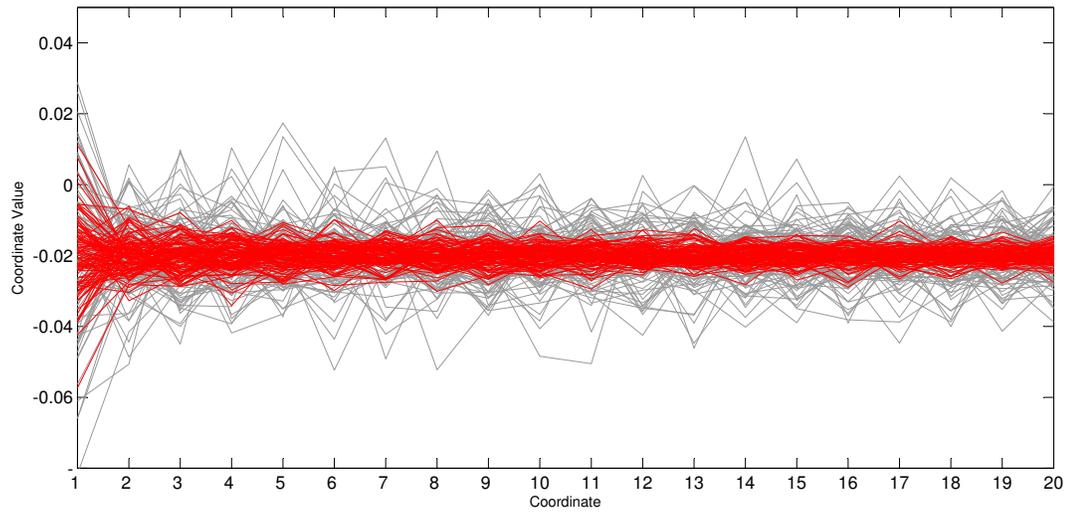


Figure 3.16: Comparison of the variability between the 100 realizations generated by filtersim (red) and the proposed method (black) by the first three principal coordinates of the space of uncertainty. Two dimensional views of the data are also plotted for clarity.

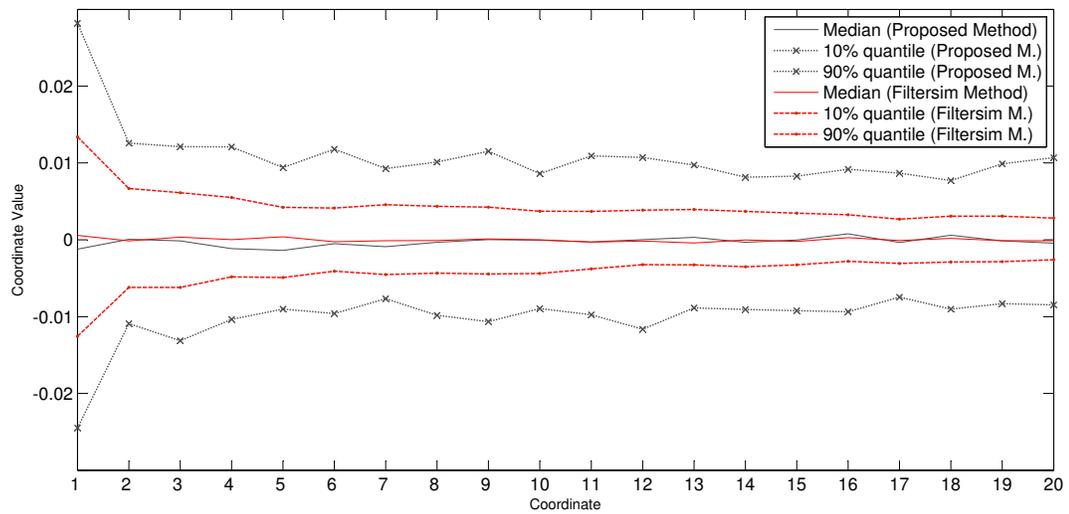
parallel-coordinate representation of the data with the first 20 dimensions is shown in Figure 3.17(a). In this figure, the gray lines represent the coordinates of all the realizations of filtersim and the red lines represent the realizations obtained by the proposed method. Larger coordinate values clearly demonstrate the larger variability in the filtersim results. Figure 3.17(b) shows the 10% and 90% quantiles of the data. The variability range is distinctly revealed on this plot.

However, one needs to realize where these variabilities are coming from, or more importantly, if they are relevant.

Inherent to any MPS simulation is a methodology for capturing the patterns/features of the training image. In snesim by a probabilistic formulation and in simpat, filtersim, or DisPAT by a pattern-based approach, one attempts to capture the geological features explicitly provided by the training image. The question on the uncertainty produced by the



(a) Full MDS variability



(b) Quantile MDS limits

Figure 3.17: Space of variability is shown in parallel coordinates with 20 first dimensions in (a), and the 90% and 10% quantiles are shown in (b). Red lines represent the proposed method's realizations variability and gray lines the filtersim.

generated realizations is of less relevance. This is due to two reasons. First, as stated before, the better the patterns are reproduced, the smaller the variability would be. This is because of the limited set of statistics existing in a stationary training image. For example, if the first-order statistics are not to be honored in MPS simulation, one can have an uncertainty space much larger than when the input proportions are to be satisfied. The second reason for the irrelevance of those questions regarding uncertainty concerns the pattern reproduction. A method that does not entirely capture the features within a training image results in some patterns in the simulated realization that are not in accordance with the provided conceptual geological scenario. Hence, such an increase in uncertainty/variability among realizations is merely an artifact that is not controlled or motivated by any geology. It can be understood, in a different perspective, that the larger uncertainty provided by filtersim is caused by a set of unknown training images that are implicitly fabricated during the course of simulation. Hence in any algorithm, by changing the parameters, one can degrade the visual appearance of the realizations (such as degradation in large-scale continuity, etc.), and artificially introduce more variability in the simulated realizations. This is a sign of internal inconsistency in the algorithm.

This postulation can also be re-examined by looking at the distribution of the errors between the realizations and the training image (as a reference). The error is defined by the Jensen-Shannon divergence of the multiple-point histograms. The results for both filtersim and the proposed method are provided in Figure 3.18. It is seen that the pattern reproduction in the proposed method is much better than in filtersim, and hence, more random patterns (corresponding to artifacts, not to actual geological scenarios) are created in the filtersim simulation. Essentially, an issue arises when one does not have control on the implicit training images used to generate these random patterns.

In conclusion, this example demonstrates that the variability obtained by using the proposed methodology is smaller than the variability obtained from filtersim simulations. In other words, one may relate the capability of any algorithm in pattern reproduction to the variability observed in the generated realizations. Better pattern reproduction induces smaller variability in the realizations, and hence, reduces uncertainty. However, not all those models with poor pattern reproduction are to be accepted. The generated realizations do not honor the prior multiple-point statistics. What we obtain as variability is fallacious and misleading. Therefore, one expects this variability, in terms of multiple-point statistics, to be as small as possible. On the other hand, the realizations themselves should have

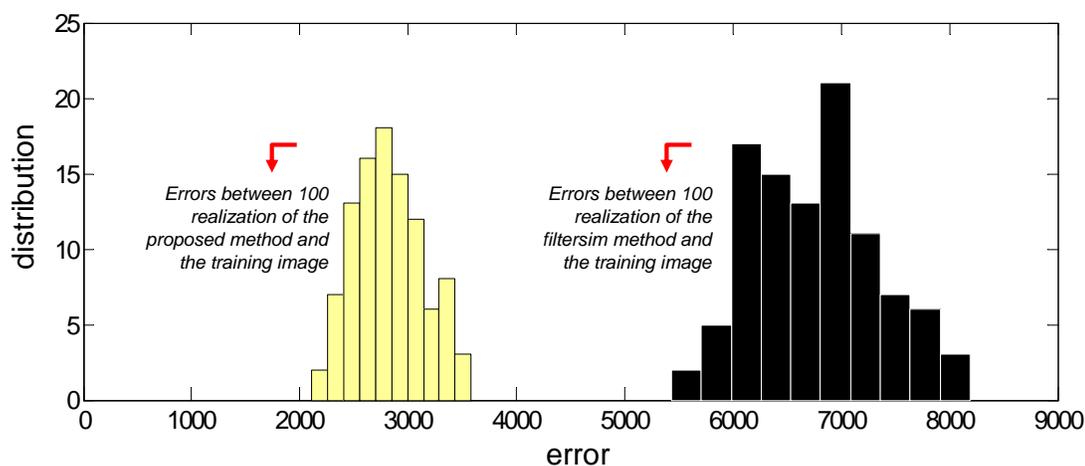


Figure 3.18: The error between the realizations generated by the proposed method (yellow) and filtersim method (black) with respect to the training image in terms of pattern reproducibility.

the highest stochasticity. Since, all these methods are stochastic in nature and different random paths can lead to the desired variabilities, we only analyzed the capability of the algorithm to get as close as possible to the explicit statistical/spatial prior model defined by the training image.

One should emphasize that, in real world, uncertainty is mostly related to the decisions that a geostatistician or a geologist makes about the training image itself. Such uncertainties originate from the knowledge or understanding of the geologist about the reservoir structure under study (or a lack thereof). In these situations, the uncertainty behind the geological scenario can be taken into account by using a set of training images instead of one.

3.4 Conclusion

In this section, we formulated a multiple-point geostatistical algorithm, called DisPAT, within a distance-based pattern modeling framework. We showed that by defining similarities, and constructing a metric space, we can better analyze the complex relationship among the patterns. It was established that when two patterns are similar, all their properties are also similar; no matter what are the actual features in their domains. It is merely their representation relative to each other that counts. This framework aids in simplifying

the patterns, such that, different mathematical algorithms can more easily capture the inherent intricacies among patterns. We used this framework for pattern classification, and demonstrated superior clustering quality and accuracy within the proposed methodology in comparison to *filtersim*. The same simulation platform as in *filtersim* was applied to generate realizations. Better pattern reproduction and continuity of the features were observed in the generated realizations. Even though more elaborate examples will be given in the next chapter, it would suffice to say that distance-based modeling proved itself to be effective.

Finally, we analyzed the variability between realizations. We stated the misbelief regarding pattern variabilities, and claimed that uncertainty produced by the generated realizations is an irrelevant point of view in multiple-point statistical algorithms. What is of value is how our realizations are honoring the statistical prior random function model provided by the conceptual training image.

In the next section, we will introduce multi-scale concepts to generate realizations that can better capture the long-range continuity of the features. In other words, we would like to have an algorithm that can better reproduce the statistical information given by the training image in both the short range and the long range. Two different notions of multi-grid and multi-resolution concepts will be explained and a wide variety of examples of the DisPAT algorithm will be provided.

Chapter 4

Multi-Scale MPS Simulation

Now we shall begin to see in detail how the rich and complex order of a town can grow from thousands of creative acts. For once we have a common pattern language in our town, we shall have the power to make our streets and buildings live, through our most ordinary acts. The language, like a seed, is the generic system which gives our millions of small acts the power to form a whole.

CHRISTOPHER ALEXANDER

In the previous chapter, we introduced the single-grid unconditional simulation using distance-based methods. We observed that a small template size prohibits us from getting all the desired statistics and continuity of large-scale structures of the training image. In this chapter, we will introduce a multi-scale approach to multiple-point geostatistical simulation, where the training image is analyzed at different scales. Two different multi-scale techniques will be described in this chapter. The first one, called multi-grid approach, is widely used in geostatistics. We will review this technique and provide some examples. The second one, called multi-resolution approach is a novel technique that can provide better long-range spatial characteristics. We will end this chapter by providing a wide variety of examples and their comparisons with filtersim method.

Before introducing the methods, let us demonstrate with an example why do we need multi-scale approaches. A channelized training image is shown, and two realizations, one with single-grid approach and one with multi-grid approach are illustrated in Figure 4.1. The training image and realizations are of size 101×101 , and the template has a small size

of 7×7 . In a single-grid approach, the channels are not connected over a long distance as opposed to the ones depicted in the training image. This discrepancy between the realization and the training image is due to inability to honor all the multiple-point statistics of the training image because of a too small template size. However, with the multi-scale approach, we are able to circumvent this limitation, and take advantage of the computational speed of a small template size, while improving upon the statistical reproductions in realizations. Although, a bigger template size could have provided acceptable results, but it has two drawbacks: (1) the variability between the realizations reduces, (2) and the computational time increases. Therefore, multi-scale modeling is a necessity for the MPS algorithm, one that, while honoring the statistical prior model, increases the uncertainty space of the generated realizations.

4.1 Multi-Grid Approach

4.1.1 Previous Work

The multi-grid approach has been widely used in previous multiple-point geostatistical methods. Tran (1994) introduced the multiple-grid concept for large-range variograms in dense simulation grids of sequential algorithms. Afterwards, MPS algorithms applied a similar concept to reproduce large-range features in the training image with a small search neighborhood or a small pattern template size. The *snesim* algorithm used it within a probabilistic simulation context (Strebelle, 2000). Next, *simpat* and *filtersim* used the same technique within their pattern-based MPS simulation (Arpat, 2005; Zhang, 2006; Wu, 2007).

In a multi-grid framework, the grid \mathbb{G} is defined by n_g cascading grids \mathbb{G}^g , where $g = 0, \dots, n_g - 1$ and n_g is the total number of multiple-grids. The corresponding templates for each grid \mathbb{T}_g is simply a scaled version of the original template. The coarse template \mathbb{T}_g is defined by the vectors $\mathbf{h}_\alpha^g = 2^{g-1} \cdot \mathbf{h}_g$. This means that the number of nodes stays the same in all the templates, but the scale doubles in every multiple-grid. In the course of simulation, at multi-grid level g , only the realization grid \mathbb{G}^g is visited in the random path. Figure 4.2 shows the realization grid nodes that are visited for each multi-grid level of 1 to 3, and their corresponding templates.

The multi-grid concept can be easily integrated into the distance-based modeling platform. The only difference is in the construction of the pattern database. Instead of using

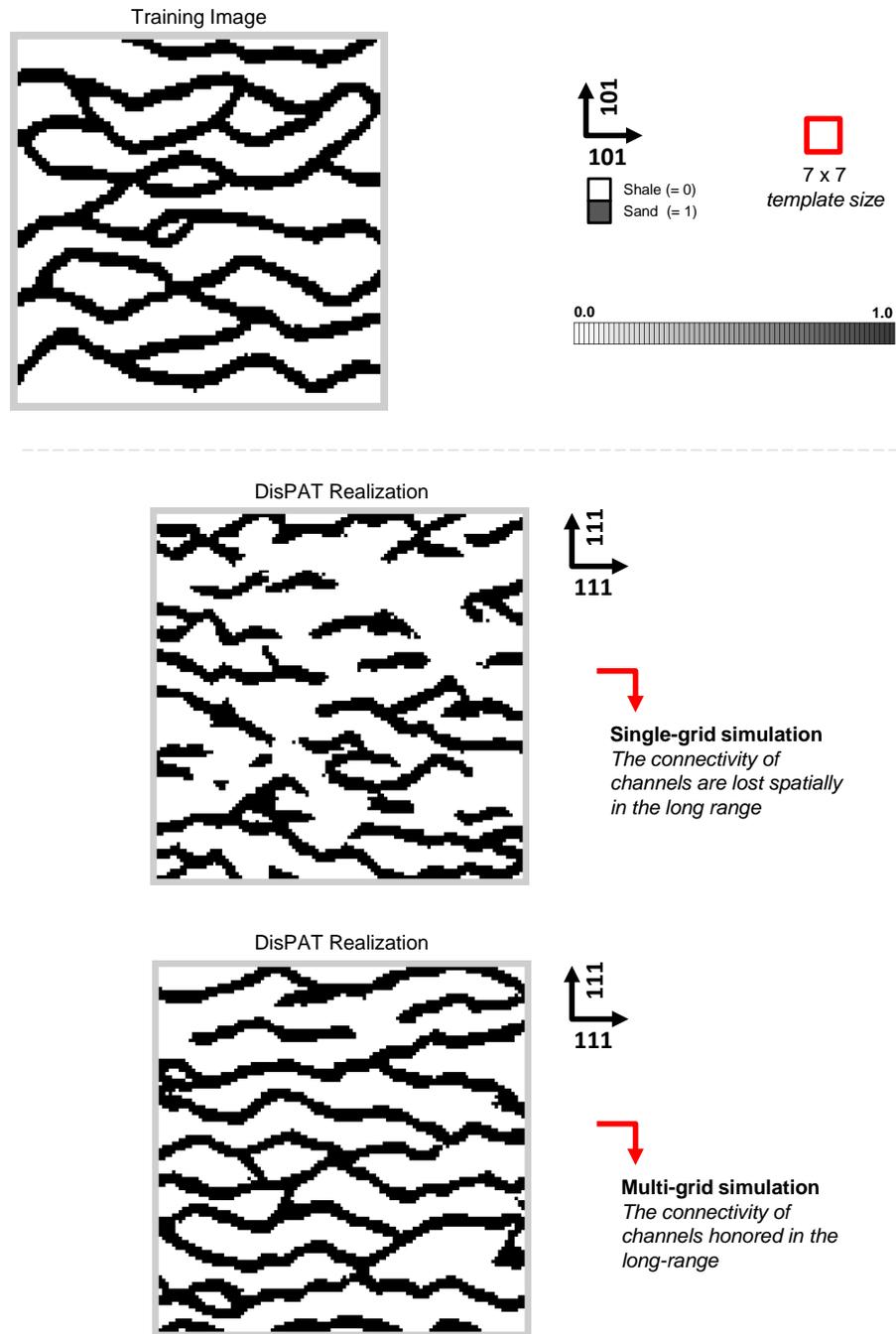


Figure 4.1: The channel training image of size 101×101 is shown. One realization using the single-grid (middle), and one using multi-grid (bottom) is illustrated, where the differences in long-range connectivity is demonstrated.

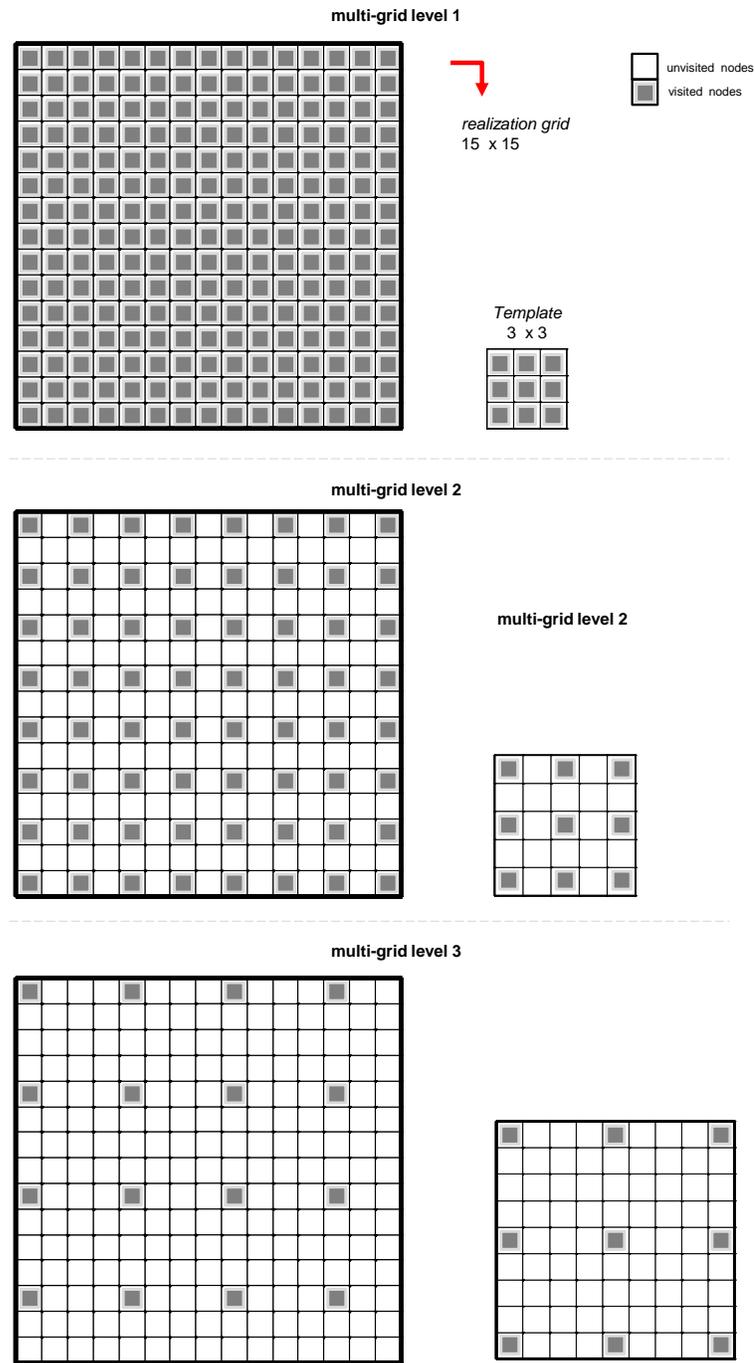


Figure 4.2: Multigrid concept is illustrated with three levels of multiple-grid. The realization is of size 15×15 and the template is 3×3 . In multi-grid 2, the visited nodes are expanded in spacings of 2, and in multi-grid 3, they are expanded in multiples of 4.

the template \mathbf{T} to scan the training image, and storing them in the database $\mathbf{patdb}_{\mathbf{T}}$, we scan the training image with template \mathbf{T}_g for each multi-grid level g . In this situation, the patterns in the training image are simply the ones obtained from the expanded template. The advantage of a distance-based modeling approach becomes more evident in the multi-grid case since the complexity between the patterns increases dramatically at higher multi-grid levels.

The simulation in the multi-grid approach proceeds with the coarsest multi-grid. In each level, only the patterns with template \mathbf{T}_g are stored for analysis. The realization grid, \mathbb{G}^g , is used, where only certain nodes are included in the random path. The simulation continues on this realization grid, and the final realization is used as the starting point for a lower multi-grid level. That is to say, \mathbb{G}_{re}^{g-1} is set equal to \mathbb{G}_{re}^g . In this lower multi-grid level, $g - 1$, all the nodes in \mathbb{G}^{g-1} are visited in the new random path. However, this time, some nodes are already informed from previous multi-grid level. These informed nodes carry the information about the larger-scale features, obtained using template \mathbf{T}_g , to the finer-scale simulation levels. The same procedure continues for all grid levels. The simulation in the finest grid level is exactly similar to the single-grid method described in previous chapter. The premise of the multiple-grid concept is that there would be more informed nodes during the similarity searches of the prototypes to the given data events, which can in turn contribute in maintaining the large-scale features of the training image.

The multi-grid approach is summarized in Algorithm 4.

Algorithm 4 DisPAT Unconditional Simulation

Require: Template size \mathbf{T}

Require: Multi-grid levels, g

- 1: **for** multi-grid $g = n_g - 1$ to 0 **do**
 - 2: Construct pattern database $\mathbf{patdb}_{\mathbf{T}}$ using template \mathbf{T}_g .
 - 3: Apply distance-based pattern modeling to obtain cluster prototypes \mathbf{prot}^g .
 - 4: Define a random path on the grid \mathbb{G}_{re}^g of realization.
 - 5: Simulate one realization with template \mathbf{T}_g and use it for next multi-grid, $g - 1$
 - 6: **end for**
 - 7: **return** realization \mathbf{re} .
-

Next, we demonstrate the application of multi-grid concept using the same channelized training image. We show the realizations at each multi-grid level to demonstrate how the information gets carried over to finer realization. The results are shown in Figure 4.3. It can be observed, specifically between multi-grid levels 2 to 1, how the large-scale structures

in the realization are refined to obtain the final result.

4.1.2 Coarse-grid Rescaling

The multi-grid concept is an instrument that allows capturing features which span a larger spatial domain than the template size can examine. However, the approach still imposes a hierarchy from coarse grids to finer ones. Therefore, Arpat (2005) introduced the concept of dual templates. It is similar to multi-grid concept, but upon pasting a pattern in the coarse grids, besides the coarse nodes within template \mathbf{T}_g , all the finer grids would also be filled by copying the entire convex volume that template \mathbf{T}_g covers (refer to Arpat (2005) for details and examples). One of the shortcomings of such an approach is the reduction in the variability among realizations. Although simpat is a stochastic modeling algorithm, but pasting large patterns on the realization grid brings larger visual similarities to the training image. In other words, there would be a decrease in the stochasticity of the algorithm, and the E-types may even converge to the training image, when big template sizes and large multi-grids are chosen.

We would like to provide another simple technique that does not decrease the stochasticity, but at the same time, aids the pattern similarity searches. Basically, from one multi-grid level to the next one, one is faced with the possibility of distortion of the features. This distortion is so evident that it can be easily spotted by a human observer. For example, the continuity of a channel gets interrupted from one multi-grid to the other. The reason behind this distortion lies in the pattern similarity search, or more specifically, finding the closest prototype, \mathbf{prot}^* , to the data event, $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$.

We have already seen that the similarity function searches only the informed nodes. Uninformed nodes are given a weight of zero. For instance, in a 2D training image, only $\frac{1}{4}$ th of the nodes in the data event are informed at the beginning of the multi-grid level 1 simulation. This lack of informed nodes would generate small errors in the similarity search $s \langle \mathbf{dev}_{\mathbf{T}}(\mathbf{u}), \mathbf{prot}_i^* \rangle$, for $i = 1, \dots, k$. In other words, the optimal pattern satisfying the given data event might be located in another cluster. This is due to the initial pattern classification that was performed with the entire pattern domain, not an incomplete/uninformed one. In general, the more informed the data event is, the more the chances of a correct classification. In this section, we would like to reduce the possibility of misclassification by filling up all the nodes of the simulation grid, only for the finest level of simulation.

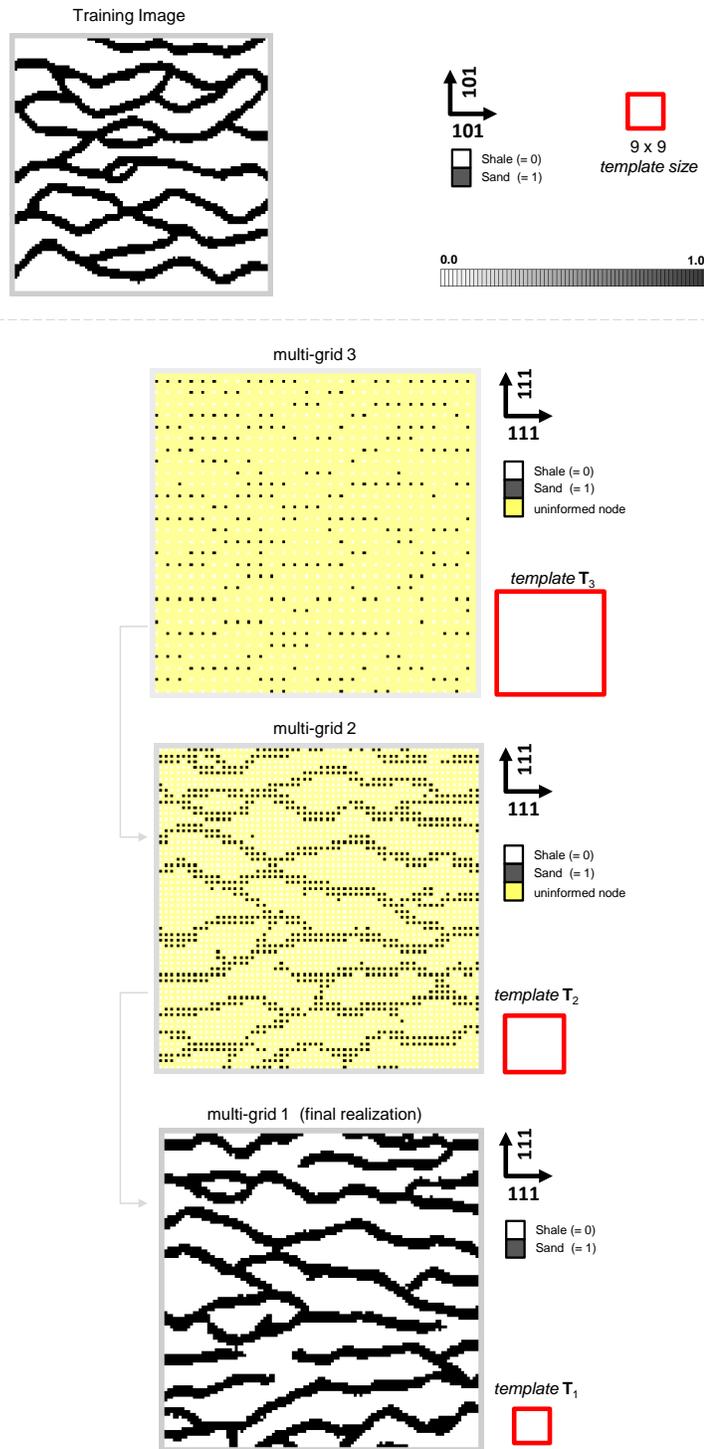


Figure 4.3: An example application of multi-grid concept with three levels.

A method, called coarse-grid rescaling, is introduced to fill up the entire uninformed nodes of the finest grid G_1 . The idea is to simply interpolate the uninformed nodes using the values at informed locations; in other words, to re-scale the multi-grid level 2, $G_2 \mapsto G_1$. Any interpolation algorithm can work; however, we employ a very simple one. We apply a smoothing filter with a template size of dimensions 3 (in 2D 3×3 , and in 3D $3 \times 3 \times 3$) on the uninformed node locations. The smoothing filter, however, would not take into account other uninformed nodes, and will only average out the informed values. Figure 4.4 shows a simple illustration of the application of smoothing filter on different locations of the uninformed nodes in a two-dimensional grid.

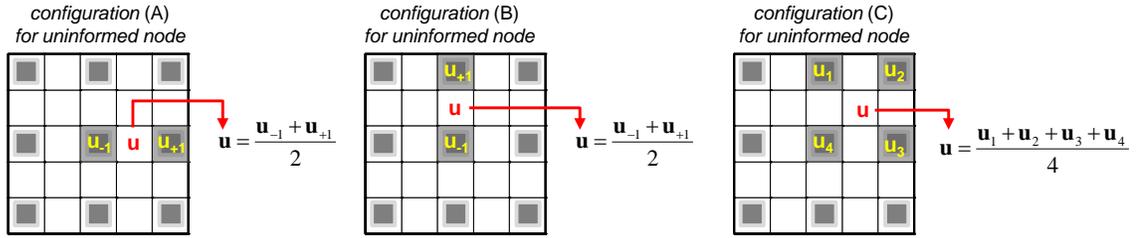


Figure 4.4: Illustration of "coarse-grid rescaling" concept to fill the uninformed nodes, performed only at the end of multi-grid level 2. It shows the simple interpolation scheme used for rescaling the grid G_2 .

We demonstrate the application of coarse-grid rescaling method on a 101×101 channelized training image. One realization is shown in Figure 4.5 for each method; with and without the coarse-grid rescaling. One can observe the slight improvement in pattern connectivity obtained with the proposed interpolation method. Yellow circles have been drawn around the locations on both realizations. These circles indicate the improvements obtained through the application of coarse-grid rescaling scheme. In addition, the filtered multi-grid realization, G_2 , is shown in this figure. Noticeably, there is now perceptually more information available within the realization grid to aid the similarity search. It should be mentioned that for a fair comparison, the random seed was set to a constant value such that the second multi-grid, G_2 , is similar for both runs.

In conclusion, we successfully applied the multi-grid concept to our distance-based modeling algorithm, DisPAT. It was demonstrated that by the use of multi-grid approach the long-range reproductivity of patterns is guaranteed. Consequently, the multiple-point statistics of the training image are better honored within the generated realizations. We also

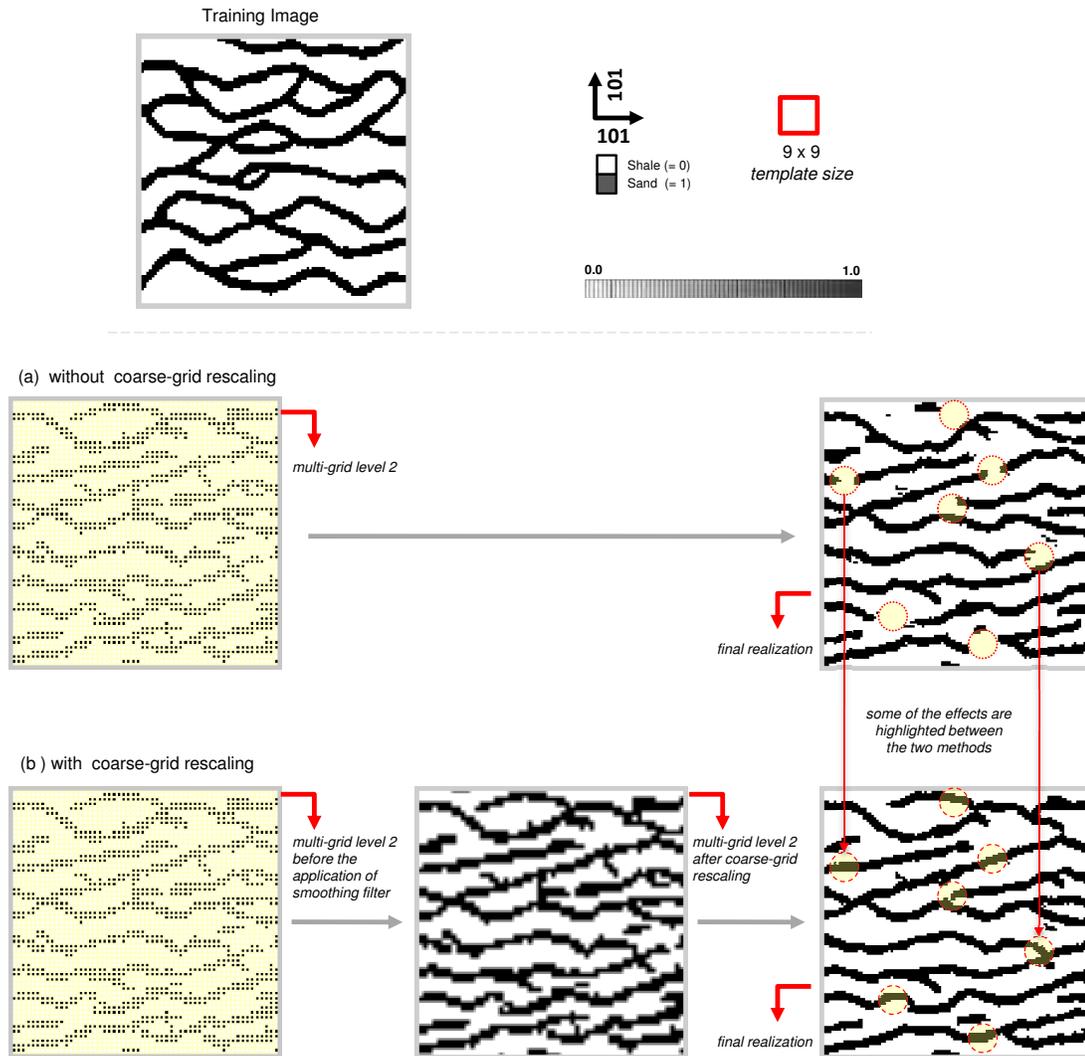


Figure 4.5: Coarse-grid rescaling method is illustrated for channelized training image. (a) depicts the MPS methodology without coarse-grid rescaling, and (b) shows the same results with coarse-grid rescaling applied. The final realizations are shown and yellow circles are used to represent the differences between the two simulations. Clearly, the pattern connectivity in the bottom realization has improved.

introduced a new coarse-grid rescaling methodology to fill the uninformed nodes of the penultimate multi-grid. A case study illustrated the advantages of such an approach due to the better and more accurate similarity searches that are obtained during simulation. In the next section, another multi-scale idea, named multi-resolution approach, that is inspired from biology will be introduced and compared with the multi-grid approach.

4.2 Multi-Resolution Approach

There are some shortcomings inherent to the multiple-grid concept. Simulation during the multiple-grid of n_g makes the simulation to continue on every $2^{n_g-1} - th$ node of the grid. Therefore, the patterns in the database are constructed from a much sparser template. This sparseness is a source of complexity/randomness in the patterns, which will in turn induce complexity/randomness in the generated realization at that multiple-grid level. More randomness suggests less structure and leads to lower pattern reproduction quality.

One solution to this problem is the concept of dual templates, where during the largest multiple-grid simulation, instead of just pasting the sparse pattern, all finest nodes of the finest grid \mathbf{G} will also be pasted. This reduces the variability that can be brought into the simulated realizations because of the explicit provision of finer information at the initial multiple grid. On the other hand, conditioning data may impose certain problems in a multiple-grid setting. For example, a particular multiple-grid \mathbf{G}^g might not include the location containing the conditioning data. Relocating the hard data to a location within the range of the multiple-grid reduces the conditioning precision of the simulated realizations. Moreover, using a dual template search strategy does not provide the true small-scale variability between the training image and the simulated realizations. In other words, pasting the most similar dual template (with respect to the hard data) on the simulation grid will also impose the small-scale features as is. Variability of these features is therefore reduced because the realization will not change significantly in the subsequent multiple-grid simulations. Besides, the search for the dual template that honors the hard data is of limited scope and a situation with no exact match can easily arise. A new method, called multiple-resolution simulation, that can reduce the existing limitations is introduced in this section.

4.2.1 Resolution Theory

Measurements, in any physical science, are obtained by integrating some property with an instrument. For example, a camera integrates the incoming light and transforms it into an electrical signal. Our visual system is no more different. We make measurements by integrating the information around us, either small objects, or large scenes. The eye does not limit itself to either small or large objects. It has the flexibility to understand its surroundings without any prior information on what to expect; whether the object is big or small, whether its square or a line, it is able to analyze it. The visual front-end of an eye can be regarded as a “multi-scale geometry engine” (Koenderink, 1984). The multi-scale capability of human visual system allows understanding perceptual information at different resolutions simultaneously. The human visual front-end should be capable of sampling images at all scales since the physical world around us consists of objects and structures with different sizes. There is no preference for any specific scale. When an observation is made in the space, all the information are transmitted to the brain, with not just one single image, but a stack of them at different scales. This is the *scale-space theory* that explains how our visual front-end performs (Lindeberg, 1994).

The same concept is adapted in our pattern-based approach. We construct an algorithm that does not rely on a specific domain for functioning. It should be a flexible system that can solve a large amount of visual information regardless of the design, and aims for generality. In principle, we are unaware of the specific tasks to be solved, and no presumptions can be made about our conceptual models. The scale-space theory suggests that, among different approaches, we are to select the one that assumes no strong a priori knowledge. In this regard, by representing conceptual training images at multiple resolutions, we can explicitly render the multi-scale aspect of the models. Moreover, this approach implicitly removes unnecessary and insignificant details, such that subsequent processings get more simplified.

We propose a *multi-resolution* approach based on the psycho-visual model of human optical system. However, pattern-based simulation algorithms are not able to handle all resolutions simultaneously due to the complexities introduced by additional sources of information; such as well logs and seismic data. Instead, we analyze patterns at different resolutions separately. In the same way as humans analyze a structure by first ‘blurring’ the features to get a big picture representation of the model, and then focusing on details and fine-scale patterns, we process the training image starting from the coarsest resolution

and follow a top-down simulation approach. Simulation proceeds with the lower-resolution training images to the original fine-scale model. The final generated realization benefits in three ways. First, the large-range structures in the training image can be easily captured at different scales. And second, there is no reduction in variability or stochasticity in the generated realizations. And finally, the patterns that may go undetected at one resolution might be easier to detect at another. Effectively, one expects more accurate multiple-point statistics and better pattern reproduction with the proposed multi-resolution approach. The details of the algorithm are described next with some examples.

4.2.2 Simulation Algorithm

In order to effectively reproduce the large-scale and fine-scale pattern structures of a training image, the simulation is performed at different resolutions. The multi-resolution view of a grid \mathbb{G} is defined by n_r decreasing resolution grids \mathbb{G}^r for $r = 1, \dots, n_r$, but with the same initial template \mathbf{T} . Figure 4.6 illustrates this concept with three multi-resolutions ($n_r = 3$). The original grid in this figure, \mathbb{G}^1 , is of size 15×15 . The coarser-resolution grids of \mathbb{G}^2 and \mathbb{G}^3 are of sizes 8×8 and 5×5 , respectively. On the other hand, the original template defined for this example, \mathbf{T} , is of size 3×3 . The template \mathbf{T} stays the same throughout all three resolutions: $\mathbf{T} = \mathbf{T}_1 = \mathbf{T}_2 = \dots = \mathbf{T}_{n_r}$. In other words, the vectors defining the template \mathbf{T} are the same n_T vectors $\{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_\alpha, \dots, \mathbf{h}_{n_T}\}$ initially defined for the algorithm. This is the fundamental difference in multi-resolution approach, that is, unlike multi-grid approach, the grids themselves change but templates not.

Applying the multi-resolution technique involves both the training image, \mathbf{ti} , and realization, \mathbf{re} , grids. We denote the training image at multi-resolution level of r by \mathbf{ti}_r , and the realization by \mathbf{re}_r . Assume that the original training image \mathbf{ti} is of size $n_x \times n_y \times n_z$. Then the training image at resolution r , \mathbf{ti}_r , is of dimensions $\lceil \frac{n_x}{r} \rceil \times \lceil \frac{n_y}{r} \rceil \times \lceil \frac{n_z}{r} \rceil$, where $\lceil x \rceil = \min\{n \in \mathbb{Z} \mid n \geq x\}$ is the mathematical notation for ceiling. For example, Figure 4.6 displays three different resolution levels for a 15×15 training image. Different resolutions of the training images are obtained by the application of this formula using $r = 1, 2, 3$.

The same process is applied to the realization grid \mathbf{re}_r . Each resolution of the realization grid has different sizes obtained by dividing the original size by a factor r . This ensures the consistency between realization and training image resolutions. Presumably, each resolution is considered as a separate modeling problem, in which, the realization \mathbf{re}_r is linked to its

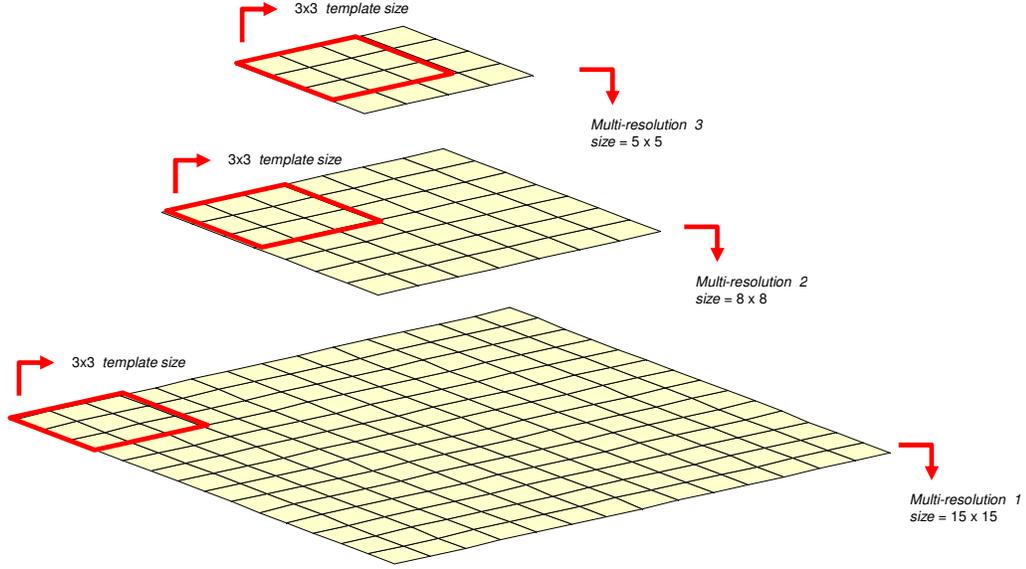


Figure 4.6: Multi-resolution concept is illustrated with $n_r = 3$ resolutions. The original grid \mathbb{G}^1 is of size 15×15 . The second and third multi-resolution grids, \mathbb{G}^2 and \mathbb{G}^3 , are of sizes 8×8 and 5×5 dimensions, respectively. The template \mathbf{T} remains the same (3×3) in all resolutions.

own training image \mathbf{ti}_r . Similar to the single-grid approach, introduced in the previous chapter, patterns extracted from the training image \mathbf{ti}_r can be denoted by $\mathbf{pat}_{\mathbf{T}}^k$, and the data events denoted by $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$, regardless of resolution r .

Application of the simple multi-resolution method starts with the construction of the lowest-resolution training image, $\mathbf{ti}_{r=n_r}$. Then, the simple single-grid simulation is applied to the lowest-resolution realization grid $\mathbb{G}_{re}^{r=n_r}$ using the initial template \mathbf{T} . Once the lowest-resolution realization is fully informed, r is set to $r - 1$ and the process is repeated until the finest-resolution grid $\mathbb{G}_{re}^{r=1}$ simulation is complete. In other words, the multi-resolution approach is similar to the multi-grid approach, where a successive single-grid algorithm is applied to increasingly higher resolutions models, and besides the initial realization grid $\mathbb{G}_{re}^{r=n_r}$, all other simulations start with a fully populated/informed realization. Having fully informed data events in the multi-resolution approach is highly beneficial for similarity searches.

In multi-grid simulation, the grids \mathbb{G}_{re}^g have the same spatial domain and extent, only the visited nodes in the random path are different. In the multi-resolution approach, on

the other hand, the entire grid domain is different in each scale. There is no one-to-one correspondence between the nodes in one scale and the nodes in the following scale. Therefore, node values need to be mapped from one resolution n_r into the next $n_r - 1$. The same is true for training images. In the multi-grid approach, the training image grid \mathbf{G}_{ti} is fixed during simulation, and only the template \mathbf{T}_g changes. But in the multi-resolution approach, the training image grid \mathbf{G}_{ti}^r changes while the template \mathbf{T} remains fixed.

Therefore, we need a method to find a one-to-one correspondence between different resolutions of both the training image and the realization. The multiple-resolution views of the training image is obtained by resizing it to $\frac{1}{r}$ th of the original size by interpolation. In other words, we resample the training image \mathbf{ti} by a factor r to get \mathbf{ti}_r . The simplest sampling procedure consists of overlaying the lower-resolution training image grid \mathbf{G}_{ti}^r with the original grid \mathbf{G}_{ti} . Then for each node location in the lowest resolution grid, we assign the value of its nearest node in the original training image,

$$ti_r(\mathbf{u}) = \min_{\|\mathbf{u}'-\mathbf{u}\|} ti_1(\mathbf{u}') \quad (4.1)$$

This is a very simple approach, but may produce undesirable artifacts in complex training images; such as distortion of straight edges. A more suitable approach is to use not just one, but 16 nearest neighbors in 2D, or respectively 64 nearest neighbors in 3D, to estimate the node value $ti_r(\mathbf{u})$ at a specific location \mathbf{u} . 16 neighbors in 2D corresponds to the 4×4 neighborhood of location \mathbf{u} , and $4 \times 4 \times 4$ neighborhood for 3D, correspondingly. The value assigned to the node $ti_r(\mathbf{u}')$ can be obtained using equation,

$$ti_r(\mathbf{u}') = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} u_x^i u_y^j \quad (4.2)$$

in 2D, and,

$$ti_r(\mathbf{u}') = \sum_{i=0}^3 \sum_{j=0}^3 \sum_{k=0}^3 a_{ijk} u_x^i u_y^j u_z^k \quad (4.3)$$

in 3D, where u_x , u_y , and u_z are defined such that,

$$\mathbf{u} = u_x \hat{\mathbf{i}} + u_y \hat{\mathbf{j}} + u_z \hat{\mathbf{k}} \quad (4.4)$$

where $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and $\hat{\mathbf{k}}$ represent the Cartesian axes of the grid \mathbf{G}_{ti} . There are 16 unknown

coefficients a_{ij} in 2D, or 64 a_{ijk} in 3D that needs to be determined. In 2D, these are obtained by the set of 16 Equations 4.2 written for 16 nearest neighbors of location \mathbf{u} , or similarly in 3D, the 64 Equations 4.3 written for 64 nearest neighbors of location $\mathbf{u} \in \mathbb{G}_{ti}$; where \mathbf{u} is the corresponding nearest neighbor of location $\mathbf{u}' \in \mathbb{G}_{ti}^r$,

$$\mathbf{u} = \arg \min_{\mathbf{u} \in \mathbb{G}_{ti}} \|\mathbf{u}' - \mathbf{u}\| \quad (4.5)$$

and the 16 nearest neighbors of location \mathbf{u} in 2D can be defined by the following set \mathcal{N}^{2D} :

$$\mathcal{N}^{2D} = \left\{ \mathbf{u} + \mathbf{h}_{i,j} \mid \forall i, j \in [-1, 2] \times [-1, 2] \right\} \quad (4.6)$$

and similarly, the 64 neighbors of location \mathbf{u} in 3D can be defined by the following set \mathcal{N}^{3D} :

$$\mathcal{N}^{3D} = \left\{ \mathbf{u} + \mathbf{h}_{i,j,k} \mid \forall i, j, k \in [-1, 2] \times [-1, 2] \times [-1, 2] \right\} \quad (4.7)$$

where \mathbf{h}_{ijk} represent the vector pointing to a specific neighboring node of location \mathbf{u} . This is illustrated in Figure 4.7 for clarity.

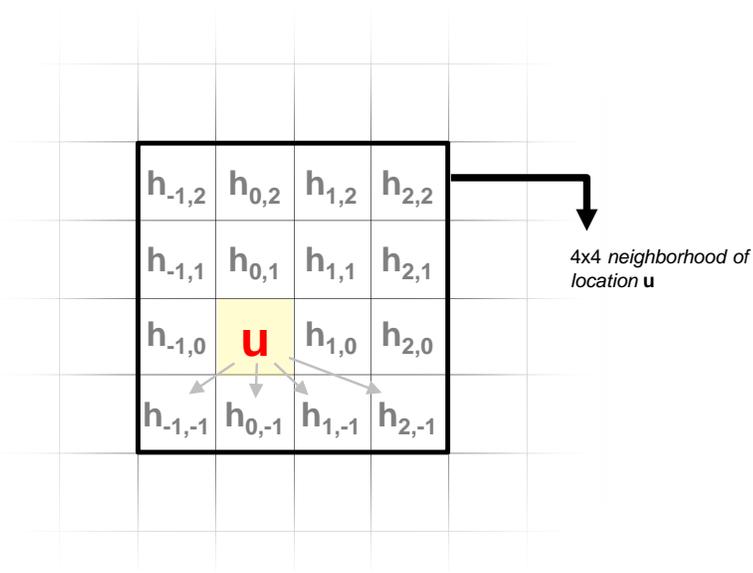


Figure 4.7: A sample illustration for 2D of 16 neighboring nodes around location \mathbf{u} , inclusive.

Equations 4.2 and 4.3 are basically the application of cubic interpolation in 2D and 3D, respectively. This method of resampling does a better job than simple nearest neighbor

interpolation in preserving fine details of the model. We apply this interpolation on the training image for each desired resolution, $r = 2, \dots, n_r$.

Having constructed a set of multi-resolution training images, \mathbf{ti}_r for $r = 1, \dots, n_r$, we can continue with the simulation algorithm. We start with the coarsest resolution of the training image $\mathbf{ti}_{r=n_r}$ and its corresponding realization $\mathbf{re}_{r=n_r}$ to simulate one model using the single-grid approach with template \mathbf{T} . Upon completion of this simulation phase, we continue with the next resolution $r - 1$. However, since there is no one-to-one correspondence between realization grids $\mathbb{G}_{re}^r \xleftrightarrow{?} \mathbb{G}_{re}^{r-1}$, we use the resampling technique of cubic interpolation as the mapping function. In other words, the realization \mathbf{re}_{r-1} is obtained by application of Equations 4.2 or 4.3 on realization \mathbf{re}_r using the resizing factor of $\frac{r}{r-1}$. The process continues until all resolutions have been simulated. The final realization would have the same size as the original realization grid.

The cubic interpolation technique, used for resizing grids to the desired resolutions, is based on an equation that performs a weighted sum of coefficients, hence, results in continuous attributes. This is problematic for categorical variables that only get discrete values. For example, a binary training image should consist of only zeros and ones in all resolutions. In such categorical cases, we threshold the interpolated training image to obtain an alternative categorical coarse-resolution. The same procedure is applied on the realization when going from one simulation scale to the next. In order to find a suitable threshold, Otsu's method is employed (Otsu, 1979). Let us assume that the training image to be thresholded contains two classes of node values (e.g. foreground and background). Then, Otsu's method calculates the optimum threshold separating those two classes such that their combined spread (intra-class variance) is minimal. No thresholding is performed for continuous variable training images.

One may argue that in training images consisting of only fine-scale features, the interpolation procedure would be unable to preserve perceptual saliency in the results. For example, the chessboard training image, shown in Figure 4.8, consists of sharp narrow parallel lines, indicating high-frequency components in the image. Figure 4.8 illustrates the application of multi-resolution method for two lower resolutions of $r = 2$ and 3. The interpolated training images, before thresholding, show an unclear representation of the input. However, thresholding will provide an unambiguous image that can reasonably characterize those salient features. This is more evident in the third multi-resolution image, where a coarse sketch of chessboard lines is displayed. Therefore, the multi-resolution approach

captures the fine-scale features without any problem. Nevertheless, the application of the multi-resolution in such sharp training images is not needed in the first place; it can simply be modeled using only one scale, its own. There is no need for large-range pattern reproduction, and a proper template can accurately capture multiple-point statistics of such a training image. On other hand, training images that consist of both fine-scale and coarse-scale features do no longer pose a challenge, since the coarse-scale view of the training image is reproduced in initial stages of simulation, and the fine details will be incorporated at the final multi-resolution stage.

In conclusion, we generate, beforehand, all n_r resolutions of the training image and realization grids. The simulation proceeds with the lowest-resolution training image, but with the same non-sparse template used for the fine-grid simulation. At the end of each stage of multi-resolution simulation, the realization (obtained so far) will be resized towards the next finer multi-resolution scale. Simulation in each resolution continues with its own corresponding training image until all multi-resolution levels are finished. Data conditioning now consists of applying the multi-resolution concept on the data themselves (explained in detail in Chapter 6). It is noteworthy that there is no need for a dual template in this case because all nodes in the simulation grid are informed. Hence, the multi-resolution concept brings consistency and robustness in MPS simulations. The proposed multi-scale approach should perform better than multi-grid method due to,

- more accurate pattern/prototype similarity search with fully informed data events
- simpler pattern database for each scale of the training image
- similarity to the theoretical scale-space model of human front-end visual system

The multi-resolution approach for the unconditional case is outlined in Algorithm 5.

4.2.3 Examples

The multi-resolution concept is examined for the 2D sand/shale binary training image. Three multi-resolution settings are chosen for the analysis. The training image, \mathbf{t}_1 , and the two other multi-resolutions, $\mathbf{t}_2, \mathbf{t}_3$, are shown in Figure 4.9. It can be observed that the training image is scaled from coarse to fine-scale. This way there is no need for a sparse template and the structural information are preserved as much as possible.

The detailed process of multi-resolution approach is illustrated with an example in Figure 4.10. The training image is the same channelized training image shown in Figure 4.9.

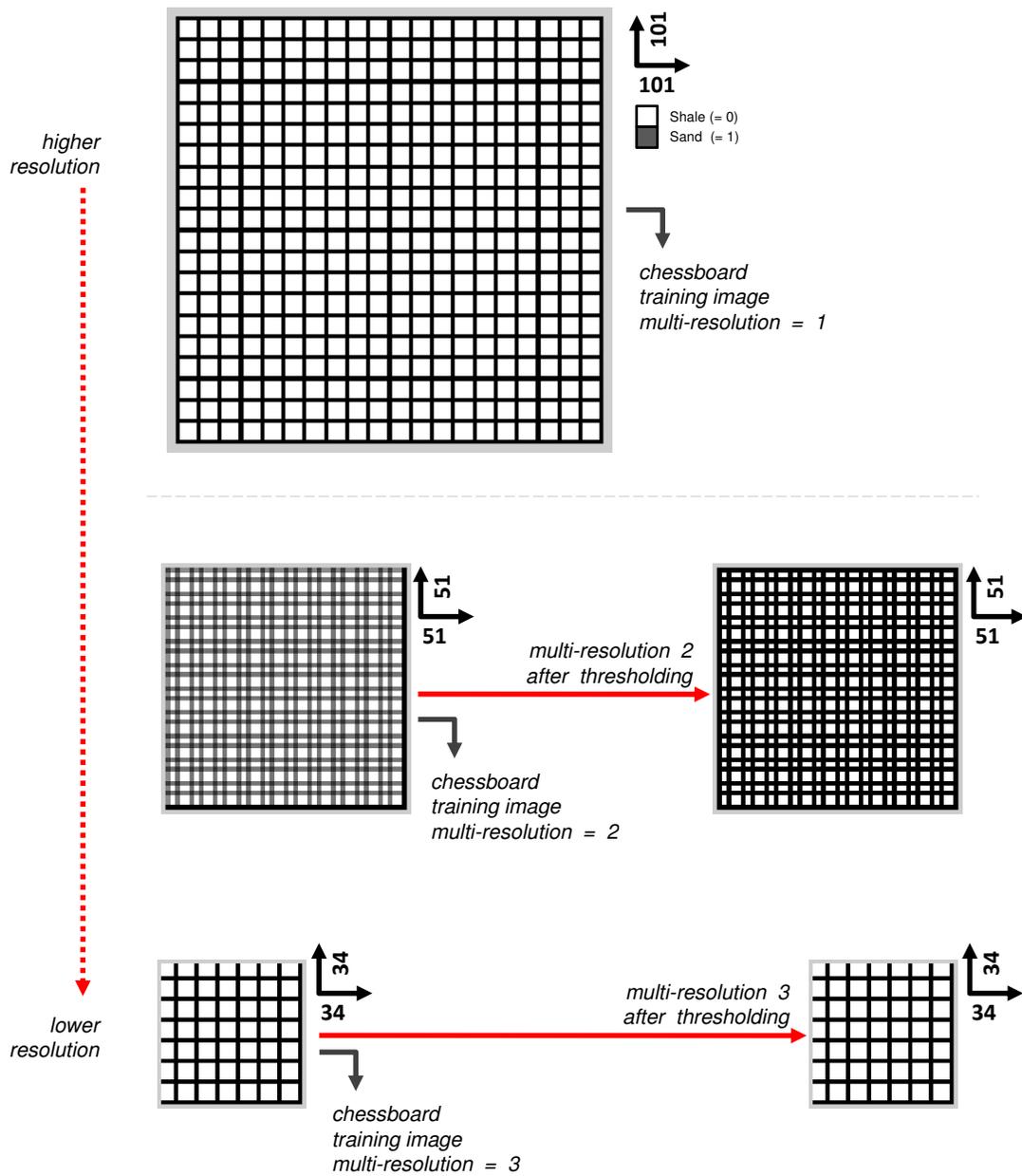


Figure 4.8: Sample experiment with multi-resolution approach where the chessboard training image consists of only fine-scale features. Three resolutions of $r = 1, 2, 3$ are shown with their thresholded results.

Algorithm 5 Unconditional Multi-Resolution Simulation**Require:** n_r = number of multi-resolutions**Require:** \mathbf{ti} = Training Image**Require:** \mathbf{T} = search Template

```

1: for  $r = n_r$  to 1 do
2:    $\mathbf{ti}_r \leftarrow$  resize  $\mathbf{ti}$  by factor  $\frac{1}{r}$  using multi-dimensional cubic interpolation
3:   if categorical training image then
4:      $level \leftarrow$  find the threshold of  $\mathbf{ti}_r$  using Otsu's method
5:     Threshold  $\mathbf{ti}_r$  by  $level$  towards a binary values of 0/1.
6:   end if
7:   if  $r == n_r$  then
8:     Initialize realization  $\mathbf{re}_r$  according to grid  $\mathbb{G}_{re}^r$ 
9:   else
10:    realization  $\mathbf{re}_r \leftarrow$  resize  $\mathbf{re}_{r+1}$  by a factor of  $\frac{r+1}{r}$ 
11:    if categorical training image then
12:       $level \leftarrow$  find the threshold of  $\mathbf{re}_r$  using Otsu's method
13:      Threshold  $\mathbf{re}_r$  by  $level$  towards a binary values of 0/1.
14:    end if
15:  end if
16:  Simulate  $\mathbf{re}_r$  using single-grid approach, with training image  $\mathbf{ti}_r$  and template  $\mathbf{T}$ .
17: end for

```

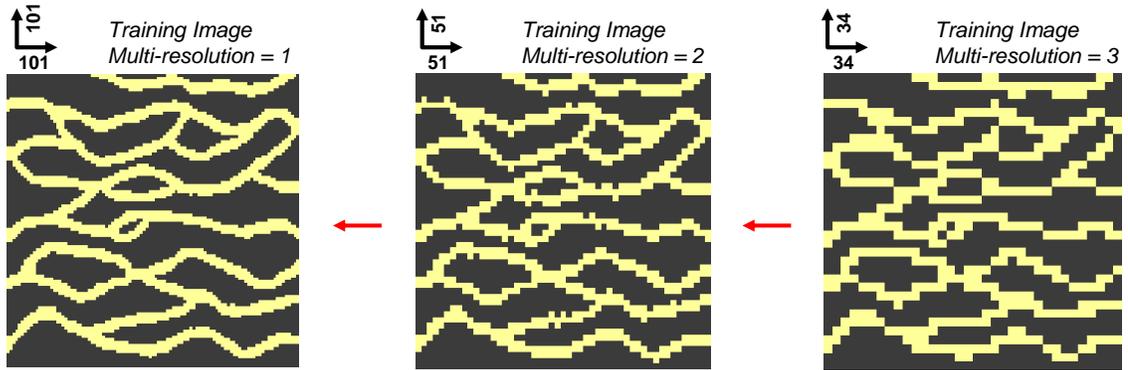


Figure 4.9: Multi-resolution training images used for each stage of the coarse to fine-scale simulations.

According to the figure, simulation starts with multi-resolution level $r = 3$, where both the training image \mathbf{ti}_3 and the realization \mathbf{re}_3 are $\frac{1}{3}$ of their original sizes, \mathbf{ti} and \mathbf{re} . Upon completion of this level, the final realization \mathbf{re}_3 is resized to the next multi-resolution level of $r = 2$, \mathbf{re}_2 . Again, a single-grid simulation occurs using the resized training image of

level 2, \mathbf{ti}_2 . In the final stage, the realization \mathbf{re}_2 is resized to the original size \mathbf{re} and used with the original training image \mathbf{ti} to generate the final realization. The entire algorithm can be conceptualized as a seamless transformation from an initial sketch of the model to a detailed realization. The approach iteratively replaces the coarse sketch with the detailed patterns referring to the initial sketch as a guide.

Some simulation results using both the multi-resolution and multiple-grid concepts are provided in Figure 4.11 for comparison. It is difficult to correctly evaluate the structure/pattern reproduction among different realizations, but visually speaking, the multi-resolution concept comparatively enhances the large-scale continuity and pattern reproducibility. Moreover, the structures with less repetitiveness (lower multiple-point probabilities) are skillfully captured and reproduced in the multi-resolution approach.

In order to quantify the differences between the multi-grid and multi-resolution approaches, 200 realizations were generated and their multiple-point histogram differences with the training image were computed by a 4×4 template. The meandering 2D training image as shown in Figure 4.12(a) is used as the training image in this experiment. The training image is complex and has high degrees of non-linearities among its patterns. Multiple-point errors between the training image and each realization are calculated, and the distributions are plotted in Figure 4.12(b) for both multiple-grid (red) and multiple-resolution (black) simulations (using the methodology proposed in Section 3.3.2 for error calculations). Moreover, the distribution of errors for filtersim simulations that use multiple-grid approach is shown in yellow in Figure 4.12(b). As expected, the multiple-resolution approach better reproduces the underlying patterns of the training image within the specified multiple-point template of size 4×4 . This demonstrates the improvements brought by the proposed methodology in terms of pattern reproduction in unconditional simulations. Effectiveness of multiple-resolution approach on data conditioning is exemplified later in Section 6.1.4.

4.3 DisPAT Simulation Examples

In this section a comparison of the proposed method will be made with the filtersim method. The filtersim results are obtained by using the public domain SGeMS software (Remy et al., 2009). Hereafter, several different training images corresponding to different subsurface structures are used for comparison.

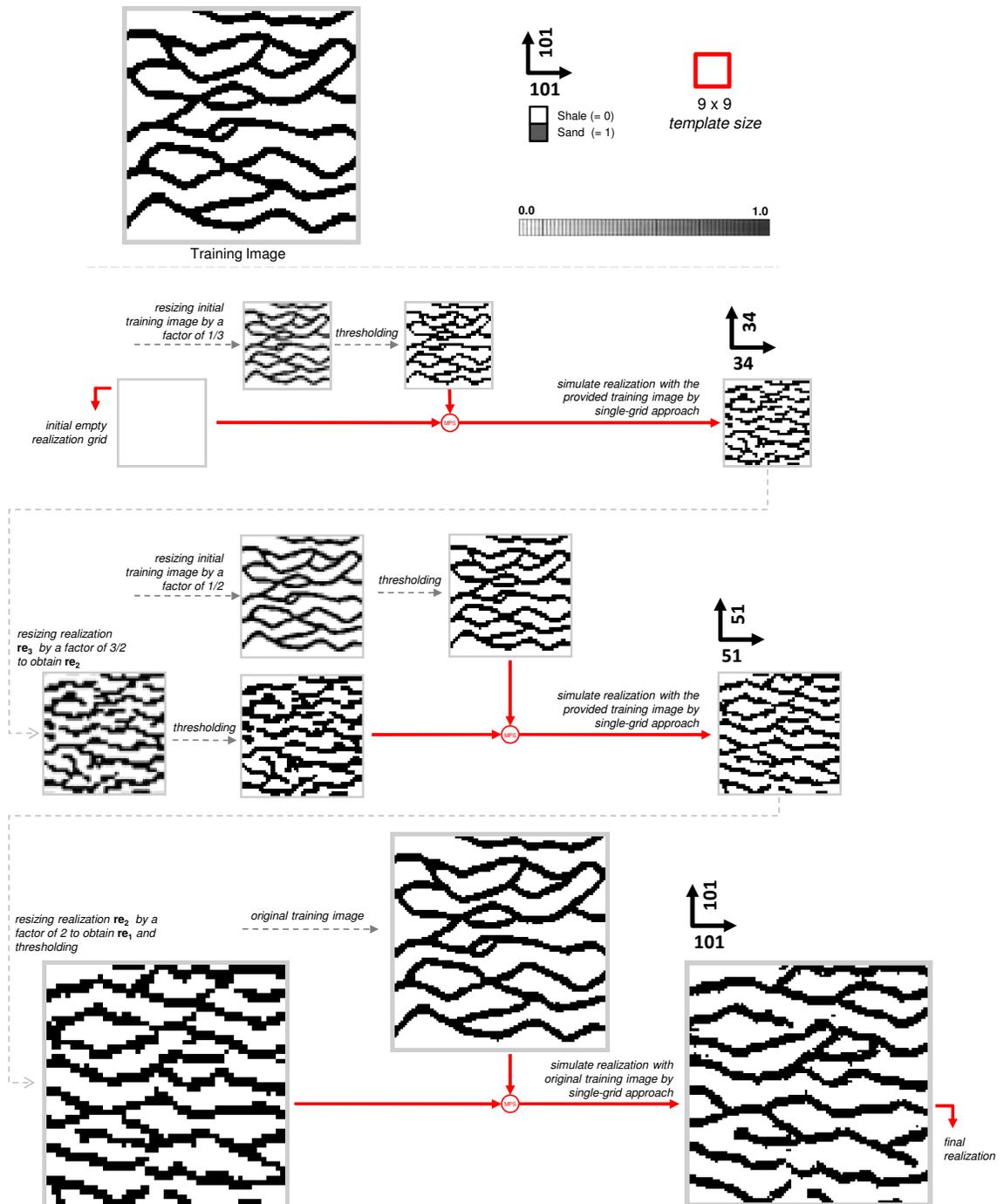


Figure 4.10: Illustration of multi-resolution approach on channelized training image in three levels.

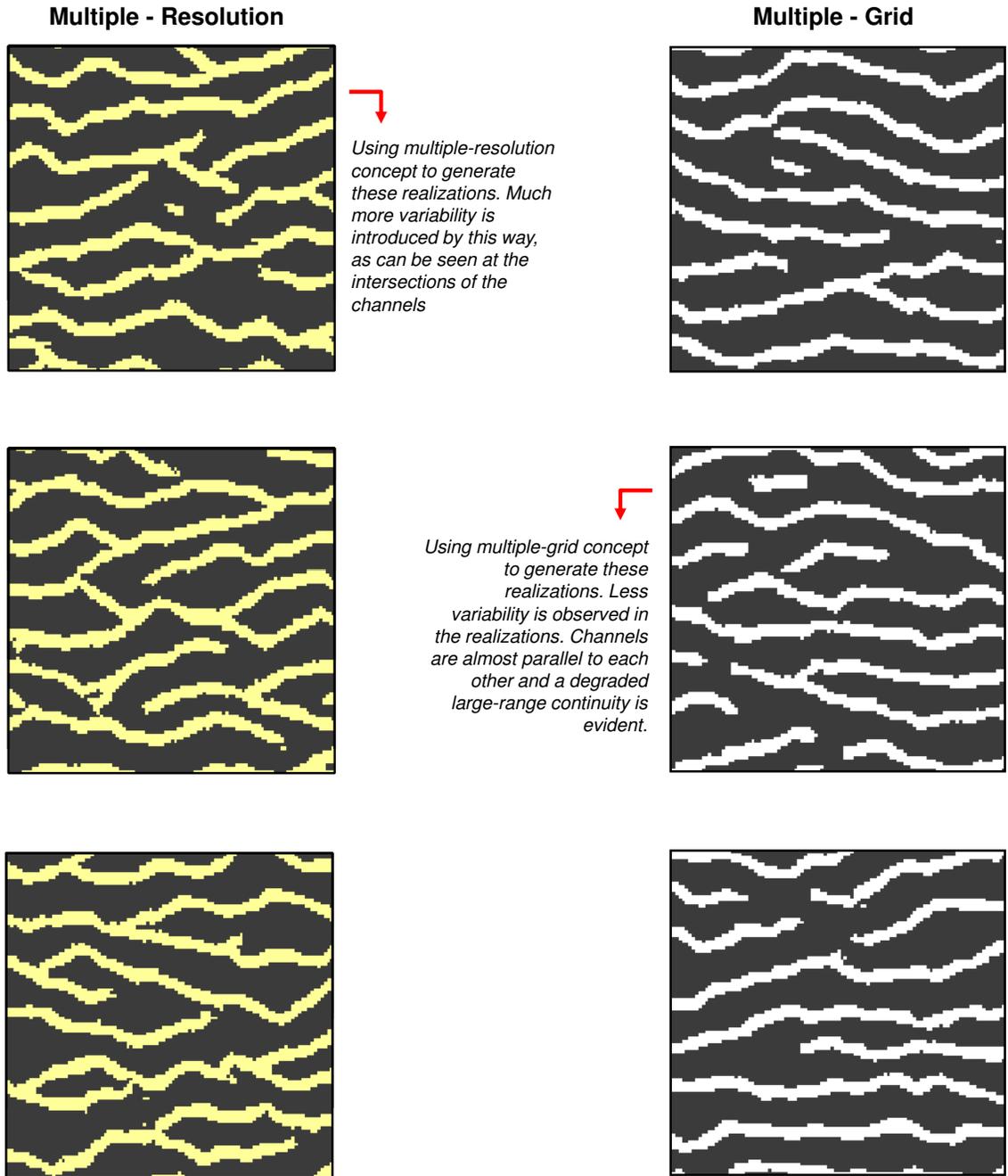
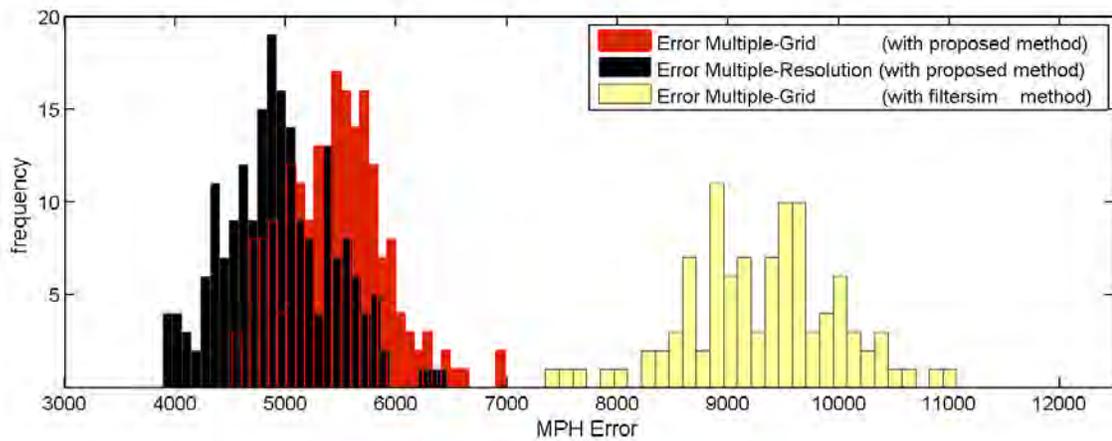


Figure 4.11: Comparison of multiple-resolution simulation with multiple-grid simulation over three generated realizations.



(a) Training Image



(b) Error distribution

Figure 4.12: Multiple-pint histogram error distribution between 200 realizations and the training image. Red distribution illustrates multiple-grid simulation errors and the black distribution for multiple-resolution case. In both distributions the proposed method is used for the simulation. The yellow distribution shows the corresponding filtersim error distributions with a multiple-grid setting.

Simple binary 2D training image

A simple training image is used for comparison. This training image is similar to a checkerboard, consisting of distinct rectangles only (Figure 4.13). The size of each rectangle is 6×6 . We conducted the simulation with inner patch = 5×5 and template size = 9×9 . The number of clusters are initially chosen at $k = 100$, due to the small number of distinct patterns in this simple training image. Some unconditional simulations were done and the most visually-appealing simulated realization among three different attempts was chosen for illustration. It is observed in Figure 4.13 that filtersim does not reproduce the true patterns inherent in the training image as well as the proposed method. There are many discontinuous lines in the filtersim results. This stems from the poor pattern classification methodology in filtersim. However, by increasing the template size to 13×13 a better pattern reproduction is observed (Figure 4.13). Besides a small region on the top left corner of the filtersim realization, the rest of the patterns follow the conceptual training image. However, in this scenario, the proposed methodology provides a perfect pattern reconstruction.

Complex binary 2D training image

Next, a more geologically realistic training image, namely the training image shown in Figure 3.8, is used for comparison. This training image represents a channelized depositional system. The parameters of the simulation in both filtersim and the new proposed methodology are provided in Table 4.1.

Parameter	Value
Pattern Template	9×9
Inner Template	5×5
Number of MultiGrids	3
Clustering Method	k -means
Number of clusters	100

Table 4.1: Simulation parameters

Two different unconditional realizations are generated using the proposed method and filtersim (Figure 4.14). It is observed that in the case of filtersim, with 100 clusters, the complex spatial patterns provided through the training image are not reproduced. Several

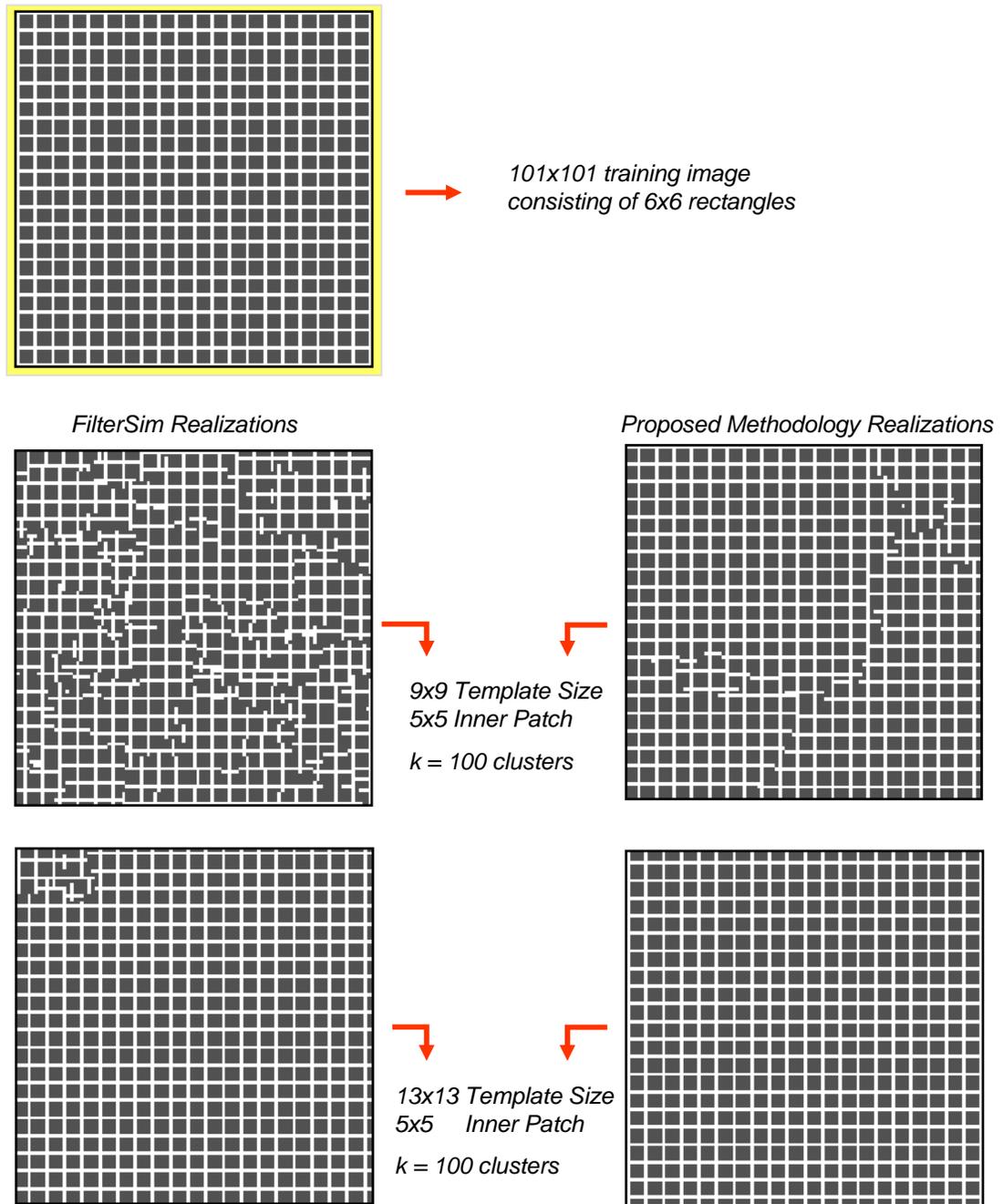


Figure 4.13: Comparison of the filtersim method with the proposed methodology for the training image shown above. Different template sizes of 9×9 and 13×13 are illustrated for comparison.

discontinuities exist. Also, the large-scale spatial features are not captured to any extent. On the other hand, the proposed method has clearly enhanced the pattern reproductivity.

The improved pattern reproduction in the DisPAT method does not render filtersim as obsolete. One of the key parameters affecting filtersim simulated realizations is the number of clusters. In principle, by choosing a much larger number for this parameter, namely 1600 in this case, the same quality of pattern reproduction and connectivity, as seen in the proposed methodology, can be observed (as will be investigated next). However, considering the total number of patterns in the pattern database, this large number of clusters results in the filtersim algorithm to perform similar to the simpat method. In simpat, each data event is compared with all the patterns in the database to find the most similar one. Choosing a large value for the number of clusters, such as 1600, will result in nearly each of the clusters to consist of only one pattern, and therefore, the search for the closest cluster-prototype is basically a search for the closest (most similar) pattern.

The effect of changing the number of clusters in filtersim is tested and a comparison is made with the proposed method. Intuitively, by increasing the number of clusters, the realization will look more like the training image, pattern-wise. However, due to the poor classification capability, filtersim can not reach the effectiveness of the proposed method which uses only 100 clusters. The results are illustrated in Figure 4.15. The improvement brought by this method is evident in the provided realizations.

Another aspect that strongly affects the simulated realizations is the search template and inner patch sizes. The choice of the template size depends on the complexity and the scale of the patterns to be reproduced. By increasing the size of the template and the inner patch, realizations will be much closer to the training image patterns and hence the pattern reproduction will improve. In order to see this effect on both filtersim and the proposed method, a larger template size of 13×13 and a larger inner patch of 7×7 is chosen for simulation. The results are shown in Figure 4.16. Noticeably, the increase in the search template in filtersim does not produce much better realizations for cluster counts of 100 and even 400. On the other hand, the proposed method is robust in the sense that the pattern reproduction is not affected by an increasing template size. This demonstrates the robustness of the algorithm and its low sensitivity to the parameters required as input to it.

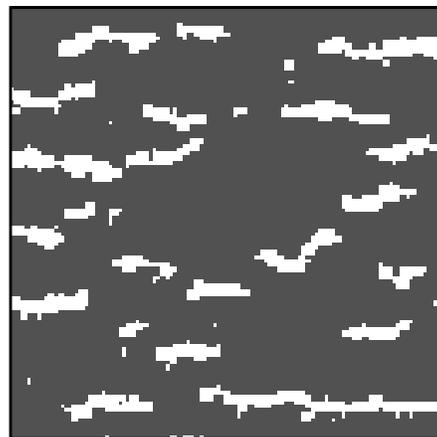
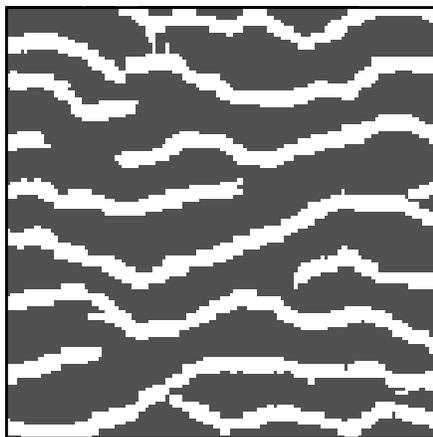
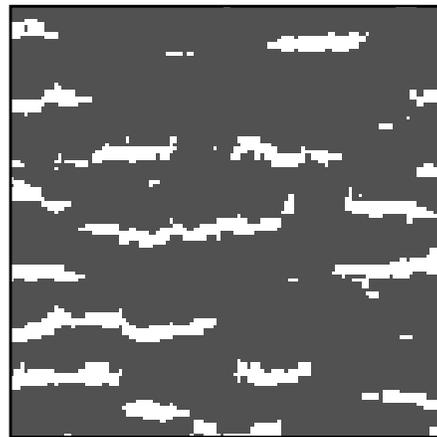
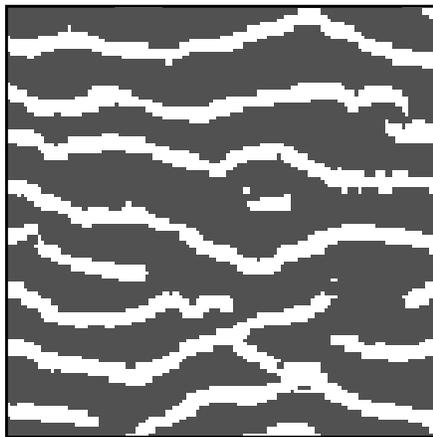
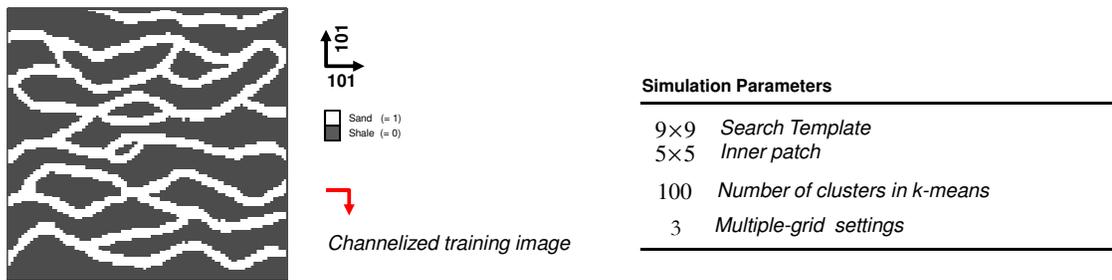


Figure 4.14: Simulated realizations using both the (a,c) new proposed methodology and (b,d) filtersim. There is a clear superiority in terms of spatial pattern reproducibility in the proposed method by using the same set of parameters. Two different realizations have been generated.

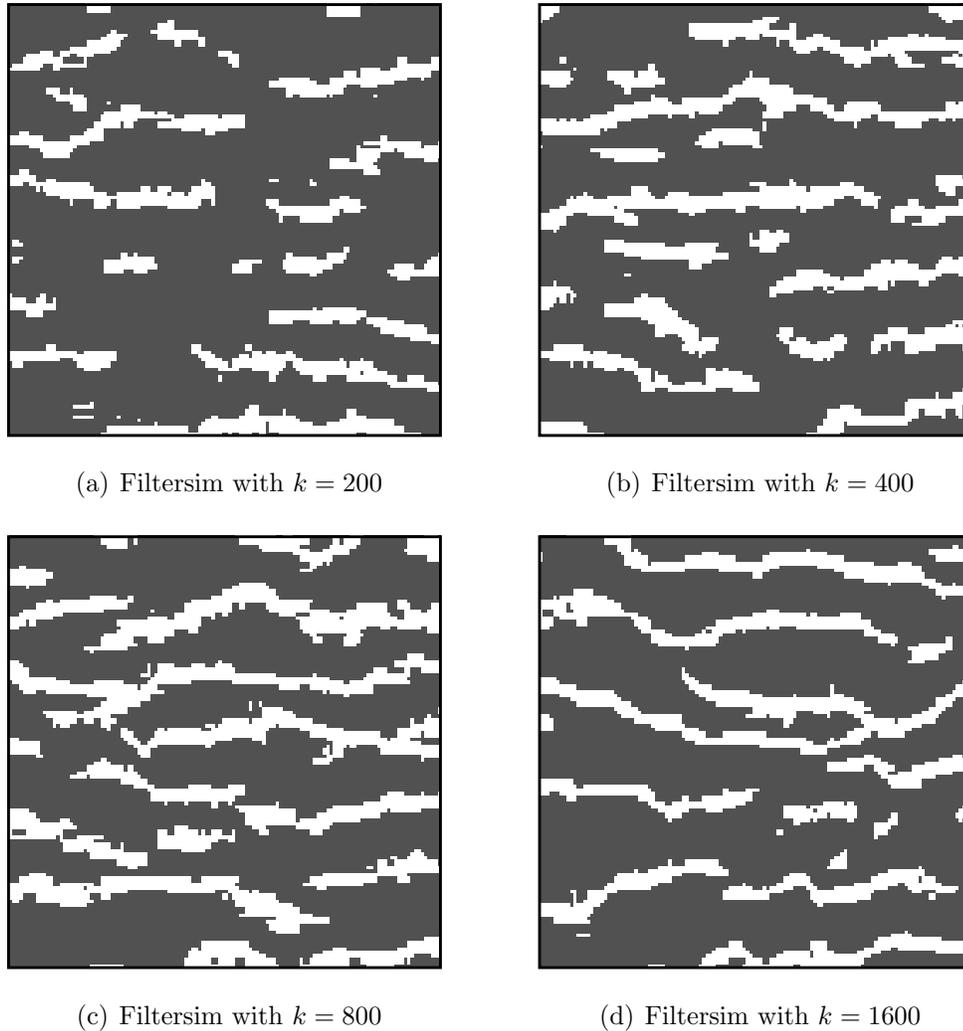


Figure 4.15: Filtersim resulting realizations with increasing number of clusters, k , in k -means clustering.

Quasi-stationary binary 2D training image

A more challenging training image that represent meandering channels is chosen in this analysis. According to Scheidegger (2004), the meandering trains are assumed to be the result of the stochastic fluctuations in the direction of flow due to the random presence of direction-changing obstacles in the river path. The training image characterizing this behavior exhibits less stationarity, hence challenging, channel structures. The application of the proposed methodology is tested on this training image and is compared with filtersim

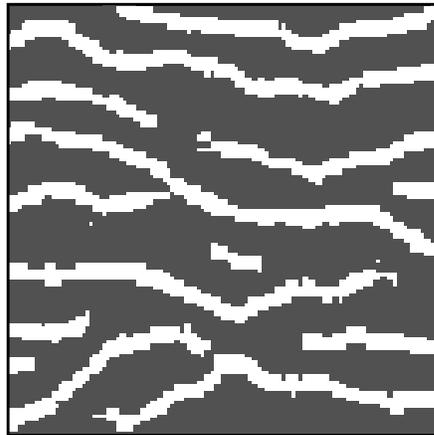
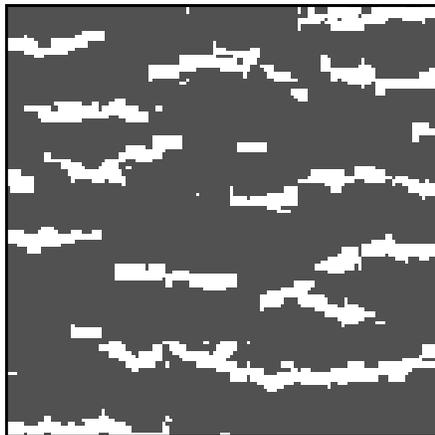
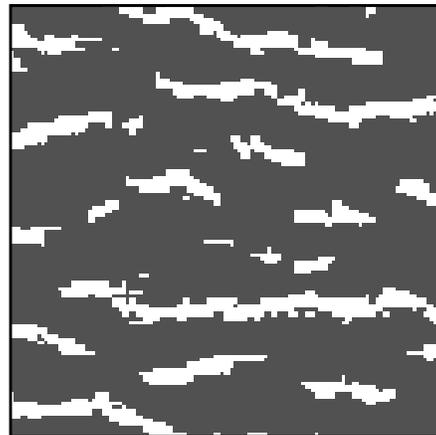
(a) proposed method, $k = 100$ (b) Filtersim, $k = 100$ (c) Filtersim, $k = 400$

Figure 4.16: Simulated realizations with search template of 13×13 and inner patch of 7×7 . (a) For the proposed method with 100 clusters, (b,c) for Filtersim method with 100 and 400 clusters respectively.

results.

Figure 4.17 shows the 111×111 training image and the parameters used for the multiple-point simulation. A search template of size 15×15 and an inner patch of 9×9 is selected. The number of clusters is set to 1000 for the proposed methodology, and 2000 for filtersim method. An advantage is provided for filtersim by doubling the number of clusters to reduce the approximation effects of random pattern selection from each cluster, inasmuch as the

number of patterns inside a cluster is significantly reduced. Three different realizations, using both methods, are shown in Figure 4.17. Clearly, filtersim results are less structured as compared to the proposed method. It should be mentioned that by changing the parameters of the algorithm, one can obtain better realizations that honor large-scale features. However, the general approach of selecting the closest cluster and randomly selecting a pattern from that cluster causes poor pattern reproduction especially in the final multiple-grid simulation.

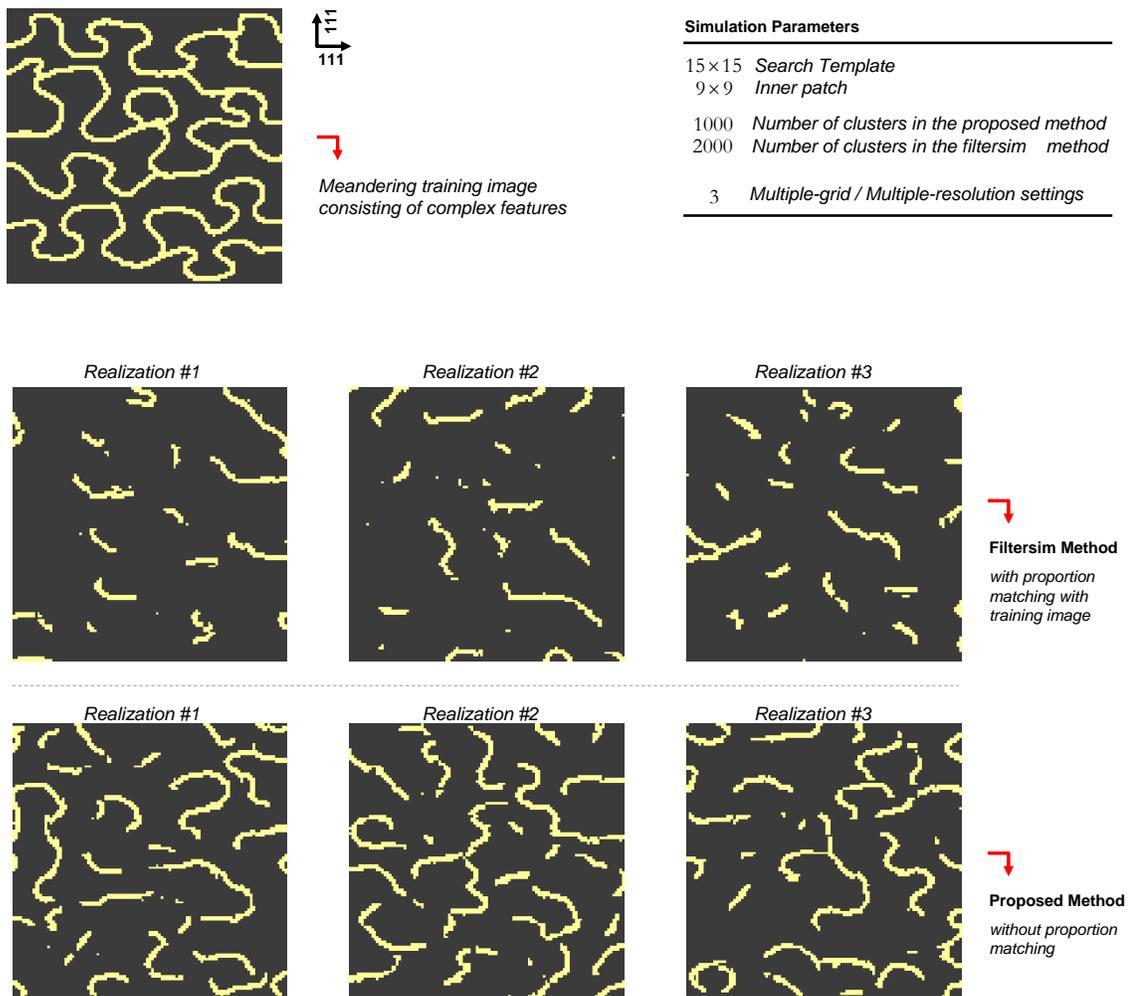


Figure 4.17: Comparison of the filtersim method with the proposed methodology for a quasi-stationary meandering training image. Three realizations are shown for filtersim (top) and the proposed method (down).

4-facies categorical 2D training image

Next, some unconditional simulations are performed on a categorical training image with four facies using filtersim and the proposed method. The training image used is of size 101×101 , which is a rescaled version of the training image of Wu (2007). A template size of 11×11 and inner patch of 7×7 were chosen according to the pattern homogeneity scale. Different realizations for each method are shown in Figure 4.18. Similar to previous results, the proposed method better honors the provided conceptual geological model. More distortion exists in the filtersim realizations. This is due to the incapability of the filtersim algorithm to provide accurate classifications; especially at the coarsest scale (first stage of simulation), where a large scale template is used. On the other hand, in the proposed method, a better coarse-scale simulation can considerably improve the final pattern reproductivity.

It should be mentioned that the filtersim realizations have been forced to match the training image proportions, whereas no histogram matching has been performed in the proposed method. This proportion matching attempts to produce realizations that look similar to the training image in one-point statistical sense. In order to see how filtersim behaves without this additional constraint on the simulated realization, three different unconditional simulations are generated using filtersim. The result are shown in Figure 4.19. A poor pattern and histogram reproduction in the realizations is evident. This observation emphasizes the considerable improvement of the proposed method over filtersim, since both sets of realizations are generated with the exact same template size.

Continuous-valued 2D training image

A continuous training image of size 159×159 , consisting of extreme features with thick and narrow cracks, is used for analysis (obtained from Zhang (2006)). For this continuous multiple-point simulation, a larger number of 200 clusters is chosen for the k -means algorithm to capture the increased variability in the pattern database. The dissimilarity function used in the continuous case is the Manhattan distance function, since the distance proximity transform cannot be used for a continuous pattern. Three different realizations are generated using filtersim and the proposed method (shown in Figure 4.20). In this specific continuous case, it is more difficult to distinguish between a good or a bad pattern reproduction, because high values that are close to one or the edges of thick cracks alter ones perception about the goodness of the reproduced patterns. However by visual inspection of

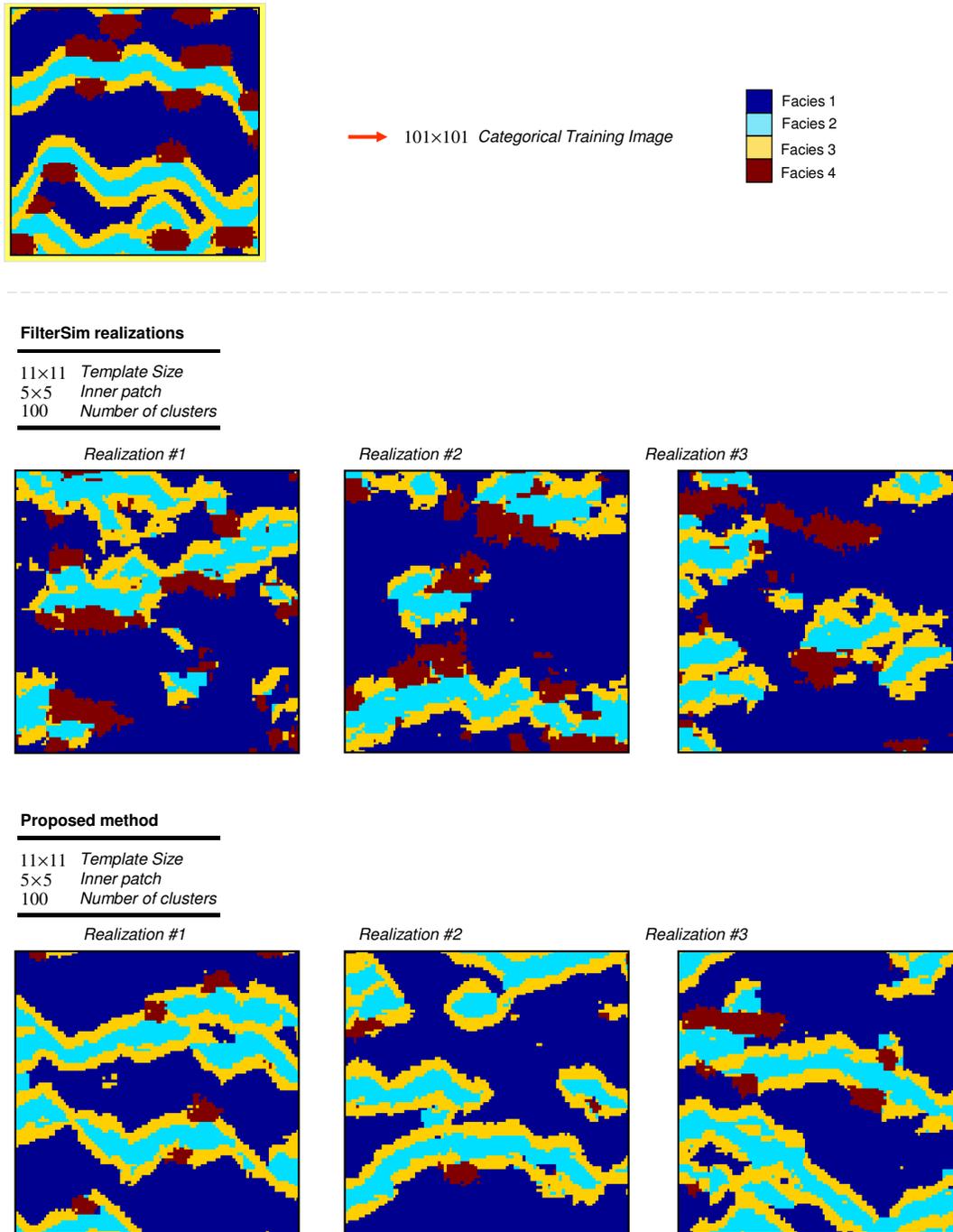


Figure 4.18: Comparison of filtersim with the proposed methodology for a categorical training image. Template size of 11×11 and an inner patch of 7×7 are used for simulation. Three realizations are shown for filtersim (top) and the proposed method (down).

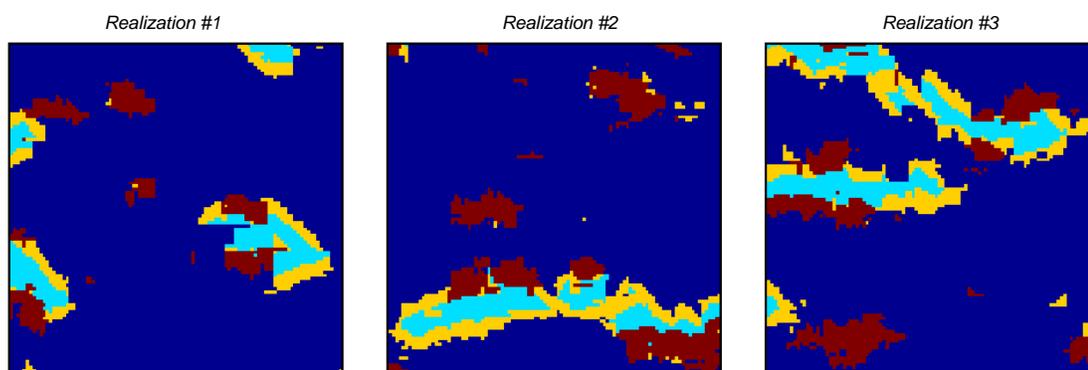


Figure 4.19: Unconditional filtersim results using the previous categorical training image but without any histogram matching. Template size of 11×11 and an inner patch of 7×7 are used for simulation. These are the realizations that should be compared with the ones generated using the proposed method.

Figure 4.20, it can be inferred that the proposed method produces slightly better realizations than filtersim. One of the reasons for reasonable results in filtersim is the application of filters in continuous images. In other words, the specific filters defined in filtersim can correctly characterize the patterns of this specific training image. The difference in quality of the filtersim realizations, seen through a variety of training images, confirms that filters cannot provide a universal measure for pattern similarity.

Binary 3D training image

The application of a 3D training image on both methods is tested next. The training image is a binary sand/shale conceptual model with the dimensions of $69 \times 69 \times 39$. A template size of $15 \times 15 \times 9$ and an inner patch of $9 \times 9 \times 5$ with a three multiple-grids levels is chosen for the simulations. The resulting realizations are shown in Figure 4.21. It demonstrates that the proposed methodology can adequately model complex, 3D geological scenarios; on the other hand, filtersim realization displays poor pattern reproduction. Figure 4.22 shows three random horizontal slices of the training image and the realizations for better visual analysis. Hence, the proposed methodology can also handle 3D training images as well as it does 2D training images. Moreover, due to the dimensionality reduction obtained by MDS mapping, the pattern dimensions, which are $15 \times 15 \times 9 = 2025$, are reduced to 118 dimensions with the automatic dimensionality selection algorithm. This has a great impact

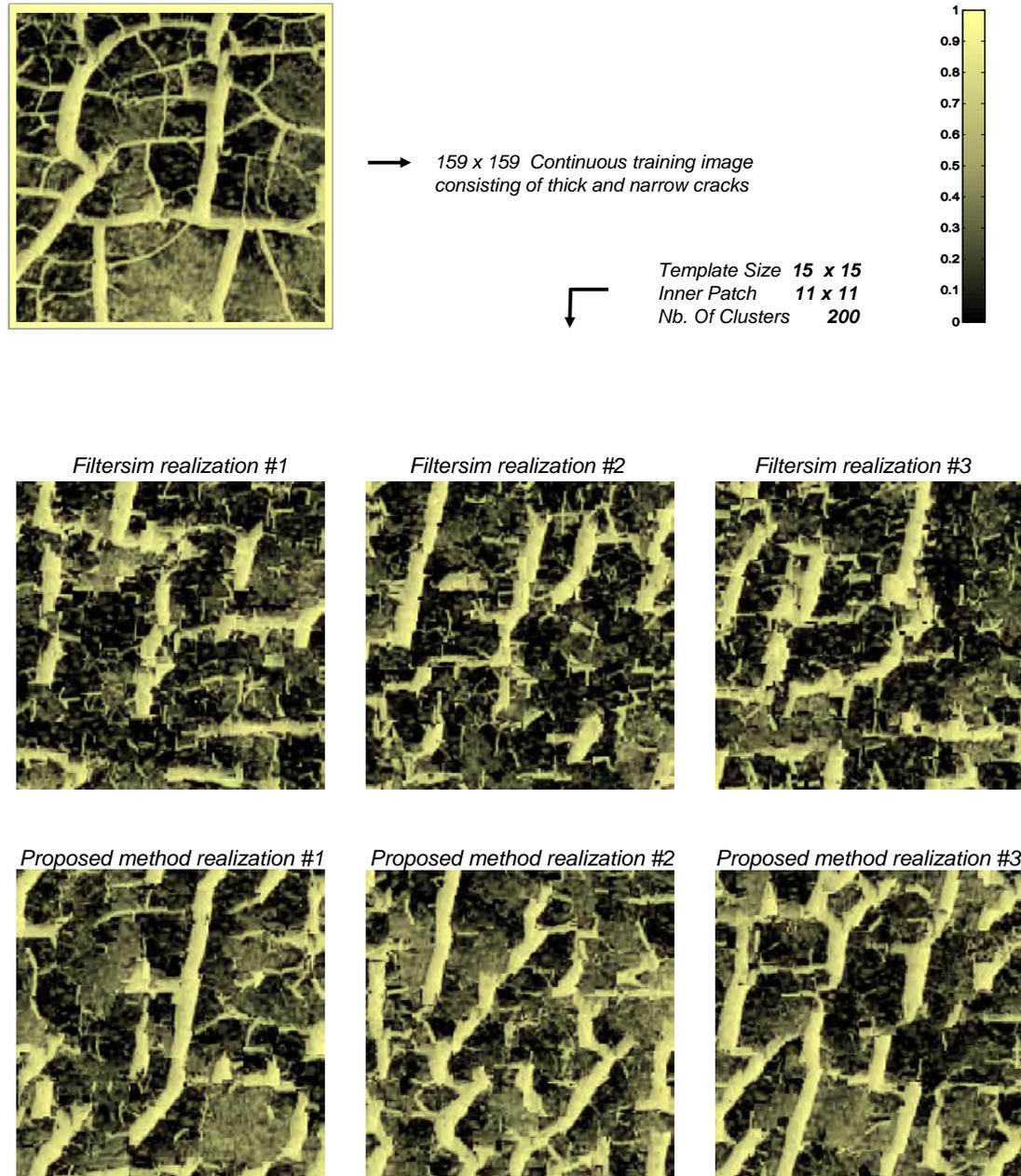


Figure 4.20: Continuous training image (159×159) shown on top is chosen for comparison. Three different filtersim realizations (middle) and three different realizations using the proposed method (bottom) are shown.

on the speed of the simulation (i.e., pattern similarity search during the simulation).

The improvement of the proposed method was established by visual inspection of the training image. In order to eliminate this subjectivity, one needs to quantify the pattern reproductivity of each method. We use the method introduced in Section 3.3.2 for this matter. The multiple-point histogram (MPH) plots for the 3D training image (reference), one filtersim realization and one realization of the proposed methodology are calculated using a multiple-point template size of $3 \times 3 \times 2$, and shown in Figure 4.23. There are some differences between each realization MPH and the training image MPH. This can be due to (1) bad template size selection for simulation, and (2) the variation between the original template size used for the simulation and the one used for the histogram evaluation. The absolute error between these histograms is used as a measure of the goodness of the realizations. It is calculated according to Equation 3.13, repeated here:

$$\text{error} = \sum_{k=1}^d |C_k^{TI} - C_k^{\text{realization}}| \quad (4.8)$$

The resulting errors are shown in Table 4.2. It can be seen that the proposed method has a better multiple-point histogram reproduction than filtersim.

Method	Multiple-point histogram error
Filtersim Method	60912
The proposed Method	37496

Table 4.2: MPH errors

In addition, we can compute the errors, not by absolute differences of histograms, but by dissimilarities between the probability distribution functions using the Jensen-Shannon (JS) divergence (described in Section 3.3.2). The resulting divergences are shown in Table 4.3. It is observed that the realization generated with the proposed method provides a smaller D_{JS} than the one from filtersim. This demonstrates better multiple-point statistical outcome of the proposed methodology.

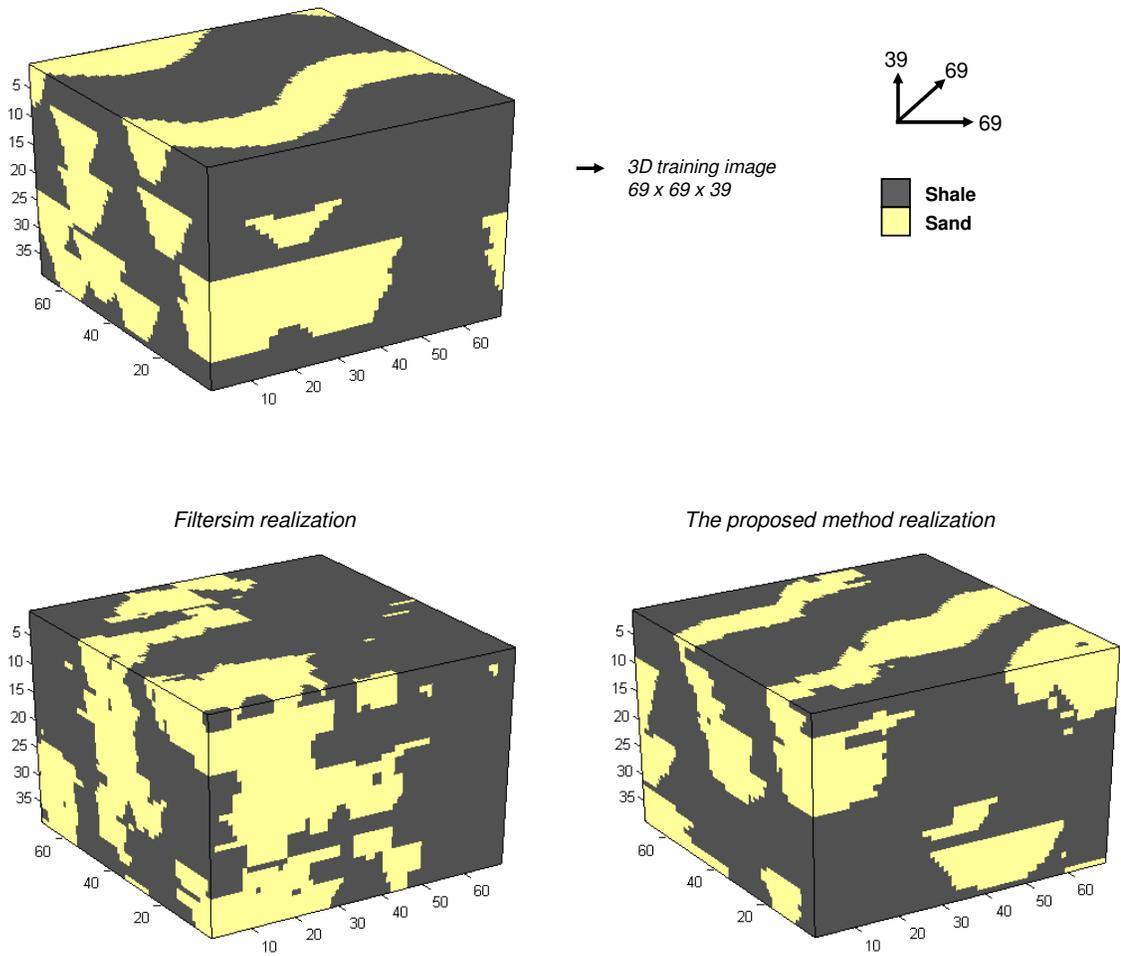


Figure 4.21: 3D unconditional simulation on a $69 \times 69 \times 39$ training image shown in top. Filtersim realization (on left) and the proposed method's realization (on right) are illustrated for comparison.

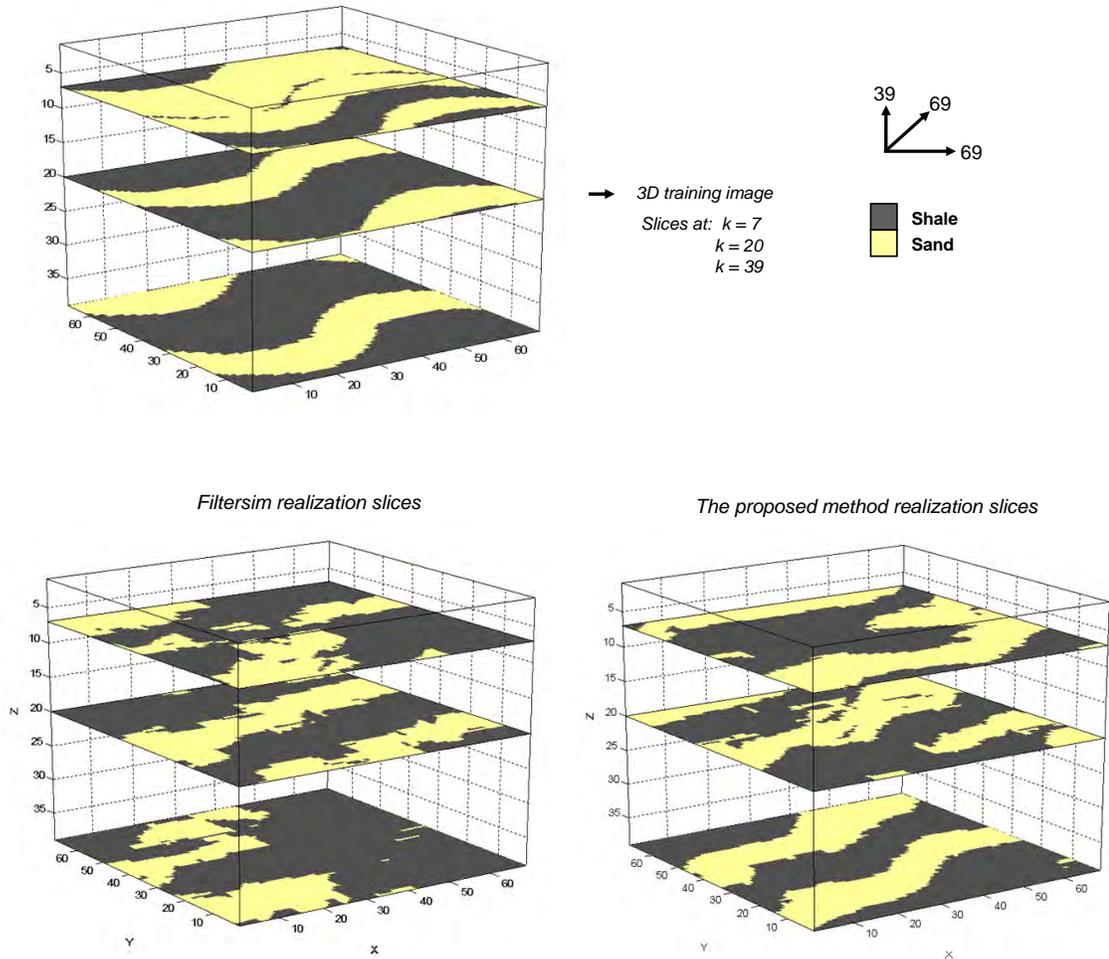


Figure 4.22: 3 slices of the training image (on top) and two realizations, one from filtersim (on left) and the other from the proposed method (on right), are shown on horizontal planes at locations 7, 20 and 39.

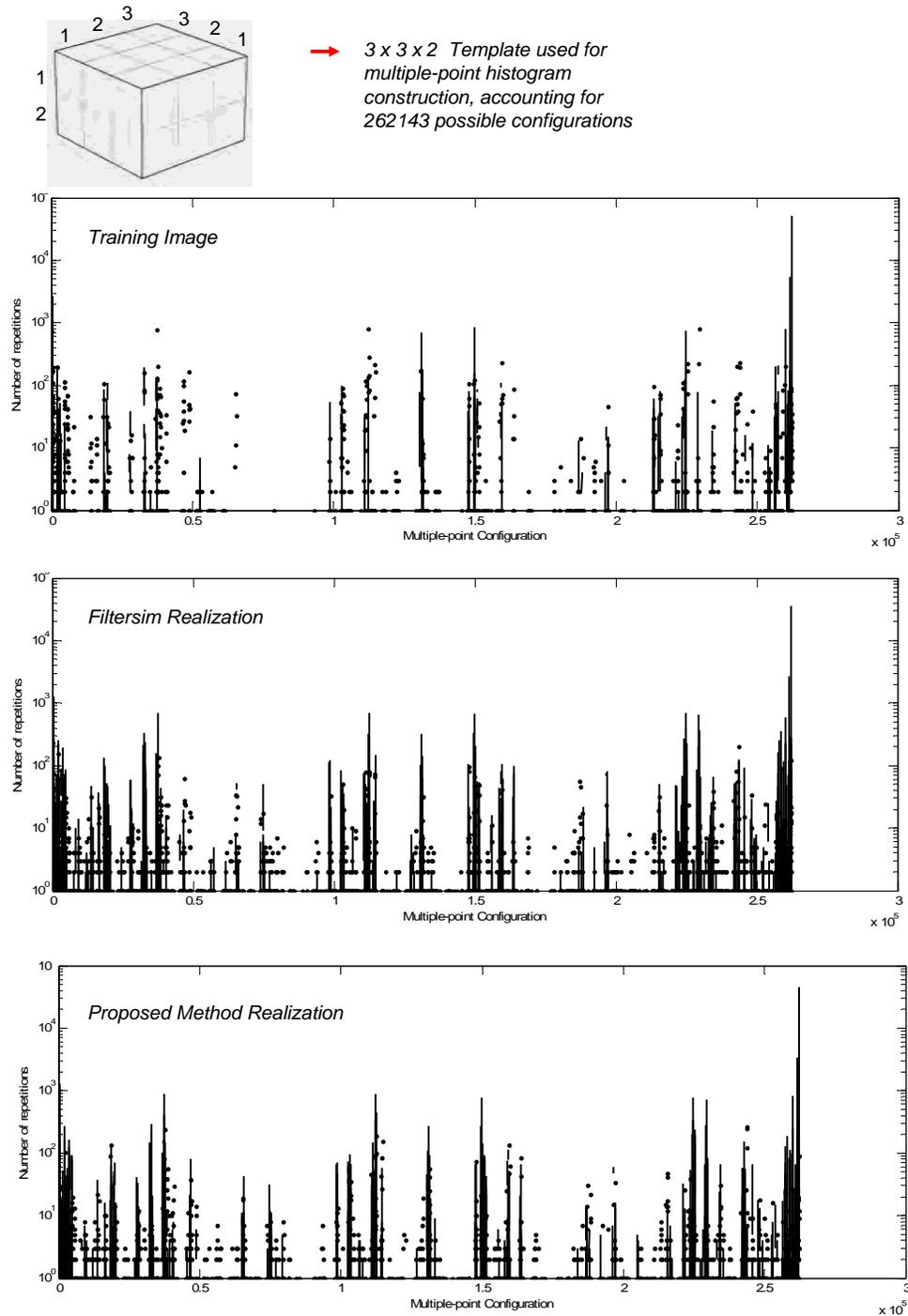


Figure 4.23: Multiple-point histogram (MPH) of the models, considering the multiple-point template of size $3 \times 3 \times 2$. x -axis is labeled according to the index related to the multiple-point configurations, and y -axis represents the number of observed instances of each configuration.

Method	$D_{JS}(\text{hist}_{\mathbf{t}_i} \parallel \text{hist}_{\mathbf{r}_e})$
Filtersim Method	0.0705
The proposed Method	0.0513

Table 4.3: Jensen-Shannon divergence between the multiple-point histogram of the training image with respect to each realization.

4.4 Simulation Time Comparison

In this section, we will analyze the computational efficiency of the proposed approach. DisPAT has been implemented in SGeMS (Stanford Geostatistical Modeling Software). Time comparisons are made with other MPS techniques of: snesim, simpat, filtersim, and the direct-sampling method. For the snesim method, instead of using the SGeMS software, we use the highly optimized implementation of Petrel 2009.

It should be mentioned that each MPS approach has its own set of input parameters. The parameter selection usually requires a compromise between the quality of the results and the computational time. For example, in the snesim algorithm, increasing the number of nodes in the search template leads to more visually-appealing realizations, but at a higher computational cost. To make a fair time comparison analysis, we attempted to remove any bias in parameter selection by generating reasonably similar realizations in all methods. The experiments were run on a 32-bit single-core Intel 2.66 GHz computer with 2GB system memory.

One 2D and one 3D training image are considered (Figure 4.24). The 2D training image is of size 101×101 , and the 3D training image is of size $69 \times 69 \times 39$. For each of these two models, the realization grid size is set equal to the corresponding training image dimensions. We replicate a real geomodelling task by generating a set of 100 unconditional realizations. The computation times are recorded for many realizations because of the goal of geostatistics in capturing the spatial uncertainty by generating not just one, but several Earth models.

The simulation times for the 2D training image are shown in Table 4.4. This table shows the great computational advantage offered by the DisPAT method.

For the 3D training image (shown in Figure 4.24(b)), the simulation times are provided in Table 4.5. This 3D example demonstrates the feasibility of the algorithm in large-scale problems. The computational speed improvements in DisPAT can be attributed to both

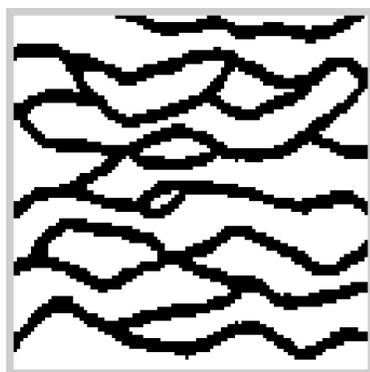
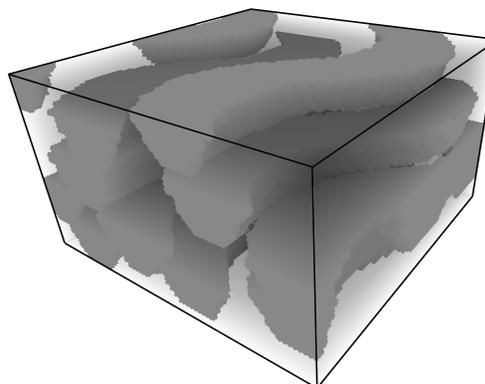
(a) 2D, 101×101 training image(b) 3D, $69 \times 69 \times 39$ training image

Figure 4.24: The two training images used for computation time comparison of different MPS techniques.

Algorithm	Simulation Time
Simpat	54 ^{min} 45 ^{sec}
Filtersim	11 ^{min} 33 ^{sec}
Direct Sampling	44 ^{sec}
Snesim (Petrel 2009.1)	40 ^{sec}
DisPAT	2 ^{sec}

Table 4.4: Time comparison for generating 100 unconditional simulation with different MPS methodologies using a two-dimensional training image (Figure 4.24(a)).

the algorithmic enhancements and the programmatic design optimizations.

Algorithm	Simulation Time
Simpat	> 300 ^{days}
Filtersim	2 ^{days} 13 ^{hrs} 57 ^{min}
Snesim (Petrel 2009.1)	22 ^{hrs} 33 ^{min}
DisPAT	15 ^{min}

Table 4.5: Time comparison for generating 100 unconditional simulation with different MPS methodologies using a three dimensional training image (Figure 4.24(b)).

4.5 Conclusion

In conclusion, we have introduced the techniques of multi-grid and multi-resolution for modeling the long-range spatial continuity of training image features. These methods facilitate MPS modeling of complex geological phenomena with the advantages of computational speed and adequate pattern variabilities. The novel technique of multi-resolution simulation is based on human front-end visual system, or simply the scale-space theory. It has been demonstrated that such an approach can improve on the multiple-point statistical reproduction of training image features.

Finally, many examples of such multi-scale modeling were illustrated with a variety of training images. The realizations were compared with *filtersim*. In all examples, an increase in pattern reproductivity and long-range continuity in the DisPAT method was observed. These examples demonstrate the versatility of the distance-based modeling framework with a variety of pattern structures.

Chapter 5

Parameter-free Learning

Like snowflakes, the human pattern is never cast twice. We are uncommonly and marvelously intricate in thought and action, our problems are most complex and, too often, silently borne.

ALICE CHILDRESS (1887 - 1948)

5.1 Its Perils

The ability to “learn how to learn” is a key component for producing robust geostatistical algorithms that can adapt to various geological models and complexities. Developing a geostatistical model concerns two presumptions:

- model form
- model complexity

where model form is motivated by its representation ability (i.e. two-point or multiple-point information) and our prior knowledge of the subsurface. For example, sequential Gaussian simulation is a direct consequence of the interpretations that the data provide. The explicit function class of Gaussian kernel used in the proposed methodology is also one example of a choice in model form. This section is concerned with the second presumption concerning the statistical methods; those that define model complexity.

One of the strengths of a statistical method is its ability to mathematically quantify its own model complexity. For example, in fitting a model to the observed data, one may decide on a model that can perform extremely well on the training data, but comes with inaccuracies in predicting new data; or otherwise, select a model that will have errors even in the training data, but has a good generalization capability. These are two examples of overestimating and underestimating the model complexity. It can also be attributed to the concept of ‘bias-variance tradeoff’ in statistical learning theory where a small variance is introduced by having a smooth function for modeling the data, which on the other hand introduces bias, as details of the function (sharp peaks and valleys) will be lost (Geman et al., 1992).

Traditionally in multiple-point geostatistics, one is required to choose the model complexity, subjectively. For example, choosing the correct search template in snesim algorithm is crucial to the outcome of the simulation and the quality of realizations. However, such knowledge is not easy to come by, and is difficult to estimate a priori. The user would spend a significant amount of time to perform trial-and-error experiments with this parameter to obtain acceptable performances. Multiple-point geostatistics, as a powerful geological modeling framework, has not yet gained the widespread acceptance in industry. This is a direct impact of the complexities inherent to application of these techniques. Only a limited set of expert geostatisticians have enough knowledge to fully take advantage of MPS modeling capabilities. One needs a technique that can self-adapt itself to the input data and automatically choose the correct model complexity for the task at hand.

The assumption in this chapter is that prior knowledge about the data and the geological model can help us make an educated guess about model parameters and the complexity. Two key parameters crucial to the performance of DisPAT are the choice on size of the template \mathbf{T} , and number of clusters used for pattern classification, k . The ultimate goal is to reach a parameter-free statistical algorithm that can perform optimally in its own domain. In the first section, we emphasize the importance on template size selection and describe an algorithm that can, statistically, provide the optimal size. Afterwards, we explain how to pick a correct number of clusters as a compromise between speed and model complexity. We demonstrate these two techniques with a variety of examples.

5.2 Template Size

In order to generate realizations in any geostatistical simulation, some form of a statistical prior model is required. For example, in sequential Gaussian simulation, a mean and a variogram alone are sufficient to generate realizations. Similarly, in multiple-point geostatistics, one acquires statistics from a conceptual training image. In two-point statistics, the variogram range is implicitly taken into account by the formulation. But in multiple-point modeling, the relevant statistics are implicitly defined through the choice of the size of the template. For example, in the probabilistic approach of *snesim*, this corresponds to the search neighborhood for calculation of probabilities, and in pattern-based approaches, such as *simpat* and *filtersim*, it corresponds to the size of the patterns that are used to generate realizations.

Therefore, in pattern-based MPS approaches, the scanning template can be seen as an interface to retrieve relevant multiple-point statistics of the training image. The training image is analyzed using the selected template \mathbf{T} to store the corresponding patterns vectors, $\mathbf{pat}_{\mathbf{T}}^k = \mathbf{ti}_{\mathbf{T}}(\mathbf{u})$, in pattern database $\mathbf{patdb}_{\mathbf{T}}$. Patterns in the database are thus location-independent, and the training image is assumed to be fully characterized by the pattern database. The simulation proceeds knowing only the patterns in the database. The initial choice on the template has considerable impact on the reproduced patterns. The reliability of the final generated Earth model is highly affected by the choice and shape of the scanning template. The size and shape of the template are equivalent to the search neighborhood in the traditional two-point statistical algorithms. Too big of a size will result in sampling data from far away locations on the training image that are unrelated to the unsampled location, and retaining small template size restricts long-range pattern reproductions. If these patterns are deemed necessary according to the conceptual geological model, then the training image needs to be scanned with a larger template size (Tran, 1994).

So far, selection of an optimal template size, associated with the training image, has been made with cumbersome trial-and-errors by analyzing the reproduction of patterns and large-scale structures in the final simulated realization. For example, Dujardin et al. (2006) conducted extensive sensitivity analysis on parameters used in *filtersim*, and concluded that the parameters should be adapted to the dimensions of the training image and the simulation grid, and a search template of half the size of the realization grid at the coarser grid scale is reasonable. Evidently, their conclusions are valid for the specific training images and

model dimensions used. In this section, an algorithm that automatically selects the optimal template size will be presented with example applications on a variety of training images.

5.2.1 Concept

In order to select a template out of all possible sizes, some criterion of optimality needs to be defined. The template size should depend on the small-scale structures as well as on the training image resolution and the actual features themselves. If the template size is chosen very small with respect to the actual features (i.e. the actual features are insufficiently represented within the template), the statistics will differ strongly for each template window. Therefore, the template size should be chosen as small as possible to improve small-scale structures, but large enough to represent the actual features occurring in a training image.

Previously, the application of two-point entropy for analyzing the training image and obtaining a 2D entropy map of points in an arbitrary template has been studied in Mackay (2003). The 2D entropy map provided an insight into the features of the training image and was used to reveal the template size. However, a clear methodology on the calculation of template size was not provided, and it relied on a subjective visual analysis of the map for this selection. In this section, the same general concept of entropy is used to find an optimal template size. However, instead of two-point entropy, a much simpler single-point entropy is considered.

Entropy is a statistical measure of randomness that can be used to characterize the texture of the training image. Similar to Shannon's treatment of the English language (Shannon, 1948), we can analyze patterns as realizations of random variables. A simple model would assume that each node is an i.i.d. realization. The normalized histogram of a pattern can be an estimate of the underlying probability of node values. The entropy of a pattern with the dimensions of $n_x \times n_y \times n_z$ can be computed as follows:

$$H = \sum_{i=1}^K p_i \log(p_i) \quad (5.1)$$

where K = number of possible outcomes of the random variable, and p_i represent the probability mass function (i.e. histogram). High entropy relates to more randomness. Generally speaking and despite some exceptions, as the template size grows, the entropy of the patterns of a training image should increase. The reason behind that lies in the Shannon's

source coding theorem. In information theory, Shannon's entropy measures the information contained in a message as opposed to the portion of the message that is determined (or predictable). This concept, when applied to patterns, can determine the minimum information required in order to reliably represent the pattern as encoded by binary digits. Therefore, by increasing the template size in a stationary training image, two different behaviors will be observed. In the first stage, the entropy will sharply increase since the average number of bits of information needed to encode or compress the patterns is increasing. At a later stage, where the template size has increased above the optimal window that represents the stationary features of the training image, Shannon's entropy would increase at a much slower pace; since the information needed for encoding a large pattern that has some stationarity features is approximately the stationary feature of the training image itself.

To provide another view on the role of entropy on template size selection, assume that a specific pattern, such as a star-shaped object, is being repeated abundantly over the training image grid. In this situation, an optimal template size would be the one that captures the star-shaped object, namely the size of the object itself. As the template size increases below the optimal value, the amount of information contained in the extracted patterns will increase. Upon reaching the optimal template size, the features of the training image, being a star-shaped object, is perfectly encoded within the patterns. However, as we increase the size beyond this template, patterns would consist of a set of repeated objects, and hence, the amount of carried information ceases to increase. In other words, there are no additional statistics that have not already been captured by the smaller templates. The choice of the template size becomes clear as the increase in information, characterized by the entropy, diminishes.

According to these concepts, the algorithm for finding the optimal template size is straightforward. Note that the choice of this optimality criterion (Equation 5.1) is still subjective. However, as demonstrated with examples, it performs well on different training images.

5.2.2 Methodology

The algorithm for optimal template selection starts by scanning the training image with different template sizes. A bound is provided for the sizes that need to be tested. Generally in MPS algorithms, since the template sizes are odd values, the bound for the 2D square

templates is the following: $\mathcal{T} = \{(3 \times 3), (5 \times 5), \dots, (n' \times n')\}$, where n' is chosen arbitrarily as $n' = 0.4 \times \max(N_x, N_y)$, where N_x and N_y are the dimensions of the training image. 3D training images follow the same procedure, but unlike the 2D case, a template size should also be calculated for the vertical dimension. Without loss of generality, we first focus on the 2D case, and afterwards, the differences for a three dimensional case are discussed.

The process in 2D starts by calculating the mean entropy (ME) of all the patterns having a specific template size $w \times w$. We denote the template with a specific square-size of $w \times w$ by \mathbf{T}^w . The formula for calculating the mean entropy (ME) for template \mathbf{T}^w is:

$$\text{ME}(w) = \frac{1}{n_{\mathbf{T}^w}} \sum_{k=1}^{n_{\mathbf{T}^w}} \text{entropy} \left(\text{pat}_{\mathbf{T}^w}^k \right) \quad (5.2)$$

where $n_{\mathbf{T}^w}$ represents the number of patterns that can be extracted with the specific template size of $w \times w$. The resulting mean values ($\text{ME}(w)$) are then plotted with respect to the template size, w . The mean entropy should increase dramatically in the beginning, and start to flatten out as the optimal template size is reached. Hence, the elbow of that plot would correspond to the optimal template size. Similar to the algorithm explained in Section 2.3.3, the maximum in the profile log-likelihood of the mean entropy can be chosen as the elbow. However, in order to obtain a more robust value, which would not depend on the maximum number of template sizes analyzed within the bound, we apply the maximum profile likelihood on the forward second difference of the mean entropy curve ($f''_i = f'_{i+1} - f'_i = f_{i+2} - 2f_{i+1} + f_i$). The reason is that the second derivative of a curve represents its curvature; hence, its curvature increases from a negative value and significantly flattens out near zero in the later stage. The approach is outlined in Algorithm 6. It should be noted that this approach is not CPU demanding in comparison with the effort of manually finding the ideal template size. Entropy calculations are fast due to the single-point entropy measure used throughout the algorithm.

The same algorithm can be applied in three dimensions. However, the dimensions are analyzed separately. First, template sizes for the horizontal plane are obtained by the application of the proposed method over all 2D slices of the training image simultaneously. That is, for each specific template size $w_{xy} \times w_{xy} \times 1$, we compute the mean entropy, $\text{EW}(w_{xy})$, by averaging the entropies of all 2D patterns within all the 2D slices of training image. Next, the vertical direction is analyzed separately. However, in the vertical case,

Algorithm 6 Automatic Template Selection

Require: Stationary training image $\{\mathbf{ti} (N_x \times N_y)\}$

- 1: $n' \leftarrow 0.4 \times \max(N_x, N_y)$
 - 2: Set the bound for template sizes: $\mathcal{T} = \{(3 \times 3), (5 \times 5), \dots, (n' \times n')\}$
 - 3: **for** Template size $(w_i \times w_i) = \mathcal{T}(1)$ to $\mathcal{T}(\text{end})$ **do**
 - 4: Window $W \leftarrow W(w_i \times w_i)$
 - 5: Construct pattern database $\{Patdb_W : \text{patterns obtained with the window } W\}$
 - 6: $H \leftarrow 0$
 - 7: **for all** Patterns $\in Patdb_W$ **do**
 - 8: $H \leftarrow H + Entropy(\text{Patterns})$
 - 9: **end for**
 - 10: $E_i \leftarrow H / (\text{number of patterns} \in Patdb_W)$
 - 11: **end for**
 - 12: **for** $i = 1$ to $(n' - 2)$ **do**
 - 13: $E_i = E_{i+2} - 2E_{i+1} + E_i$
 - 14: **end for**
 - 15: $P \leftarrow$ Maximum profile log-likelihood of $E_{1:(n'-2)}$
 - 16: **return** Optimal template size: $P \times P$
-

the template size is $1 \times 1 \times w_z$, which represent a vertical column of height w_z . Finally, we would combine these two results to provide the 3D template of $w_{xy} \times w_{xy} \times w_z$. In the next section, some examples will be provided on the effectiveness of the proposed methodology. The sequence of first determining the horizontal template size then the vertical is in line with the typical sedimentary nature of most training images.

5.2.3 Empirical Evaluations

The proposed template size selection is a theoretically sound algorithm. The concept of entropy as a measure of randomness in the patterns can provide the tools for understanding the essence of a stationary training image. In this part, we analyze different training images to evaluate the described method. A set of stationary training images with increasing complexities, and one non-stationary training image are used.

Square TI

In order to effectively assess the applicability of this algorithm, we start with a very simple training image; one with an apparent optimal template size. The training image is constructed by filling the grid with two perpendicular lines, each filling 5 grid nodes. Therefore,

the true optimal template size should be 5×5 , as is the stationary feature of the training image (Figure 5.1). Stationarity can be obtained by tiling of two perpendicular lines of size 5 next to one another over the entire grid. The mean entropy curve is also plotted in Figure 5.1. It can be seen that there is a large jump at template size of 5×5 , and for larger templates, becomes almost flat. The profile log-likelihood used for automatic selection of the elbow demonstrates the perfect match with our visual inspection.

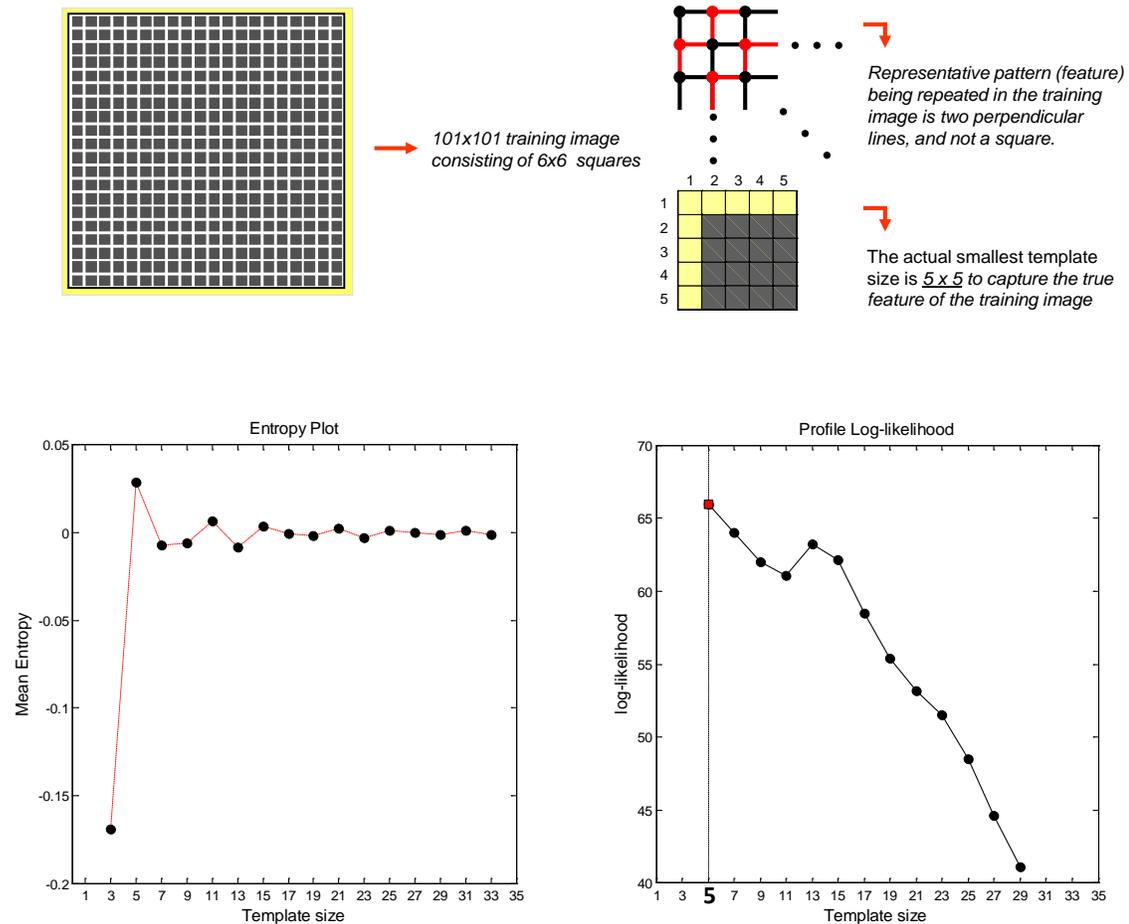


Figure 5.1: The application of automatic template selection on a square-filled training image, showing mean entropy curve (lower left) and profile log-likelihood of that curve (lower right), indicating the best template to be of size 5×5 .

Circle TI

A more complex binary training image, generated using a boolean technique, is analyzed next. The only feature chosen for constructing the training image is a circle with the diameter of 11. However, there is more complexity in this training image than in the previous one due to the interaction of the circles with each other. No single feature could be extracted as the optimal one. For that matter, the template size that leads to the most visually-appealing realization is used as a reference. Six different template sizes of (11×11) , (13×13) , \dots , (21×21) are chosen for simulation. Because of our prior knowledge on the smallest feature (circle) in the training image, the minimum template size is set to 11×11 .

However, the selection of the best realization out of these six cases is a subjective task. To simplify this selection, the realizations that have reproduced the original feature (a clean circle) will be assumed acceptable. The training image and the resulting realizations are shown in Figure 5.2. The realizations obtained by template sizes of (17×17) , (19×19) , (21×21) contain a clean circle similar to the original feature. Therefore, one expects the algorithm to produce the same results. The application of the algorithm on this training image is shown in Figure 5.3. The best template size, as expected, is obtained to be 17×17 (although 19×19 is equally good, at least visually).

Channel TI

A more complex 2D training image consisting of channels (sand/shale) is also tested with this methodology. A sensitivity analysis of filtersim parameters has previously been made on the exact same training image (Dujardin et al., 2006). The templates that were tested are (7×7) , (11×11) , (15×15) . The best realization, which resulted in good channel connectivity and large-scale pattern reproduction, has been generated by a template size of 11×11 . Applying the proposed algorithm on this training image yields 13×13 for the optimal template size. The results are shown in Figure 5.4. Although this template dimension was not tested in the mentioned paper, a closer look at the profile log-likelihood of the entropy curve suggests that 11×11 is also a good estimate for the template dimension as it is very close to the global maximum.

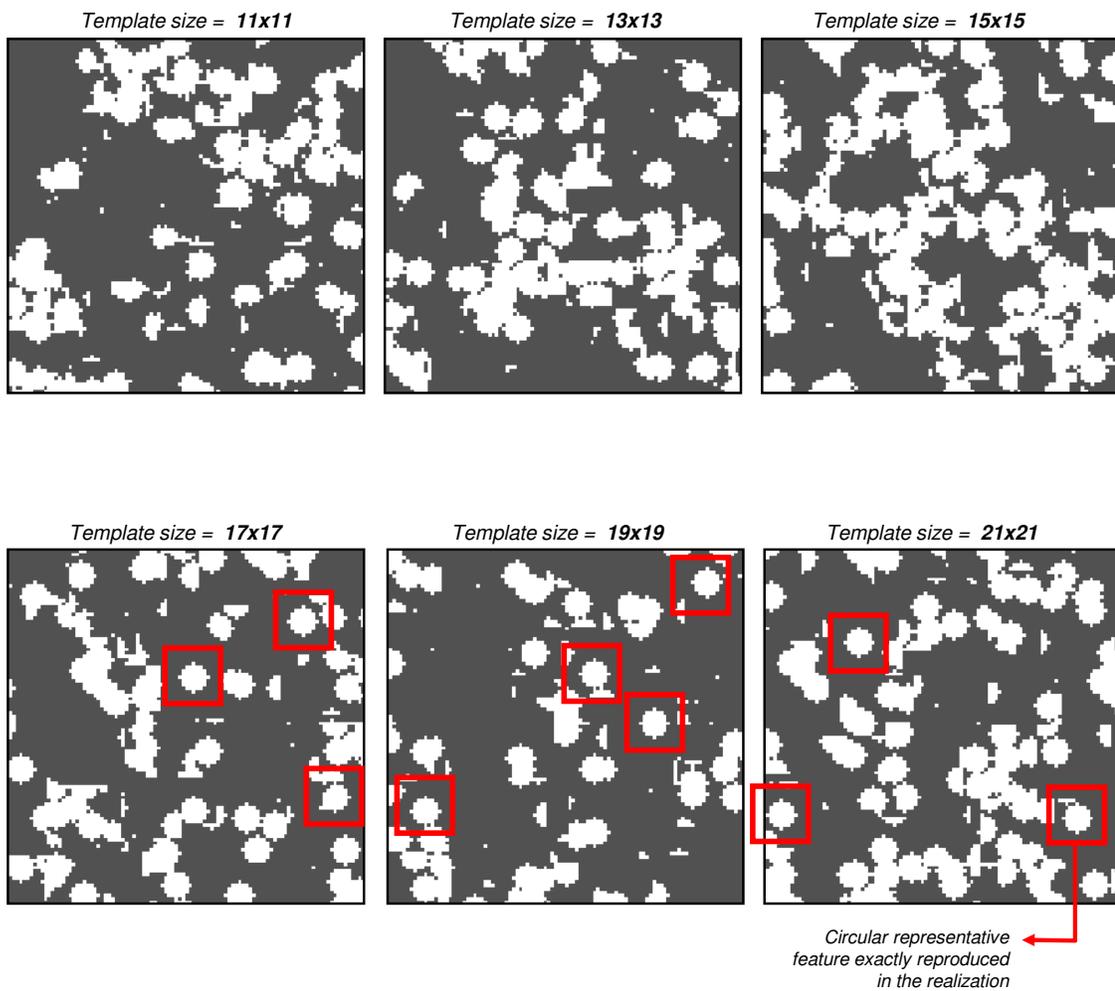
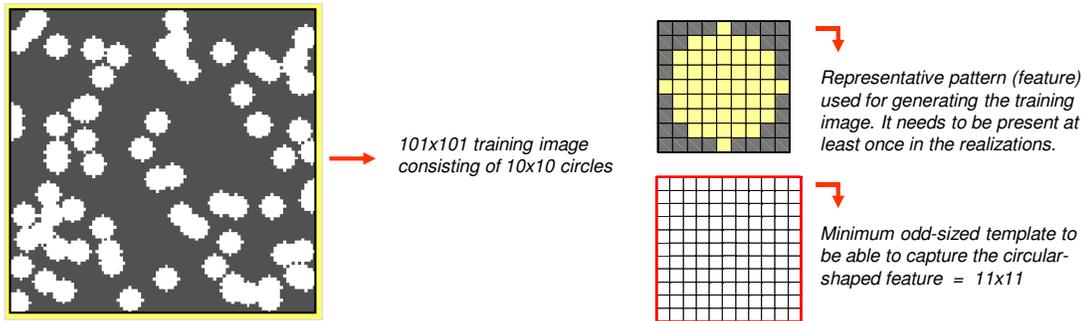
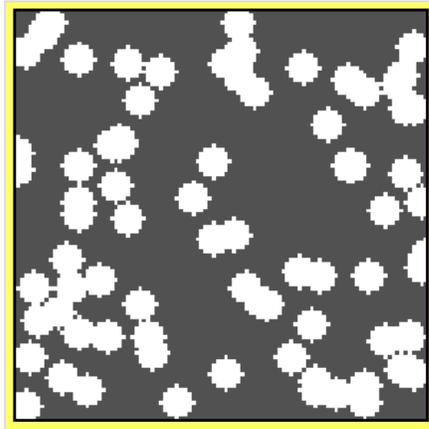
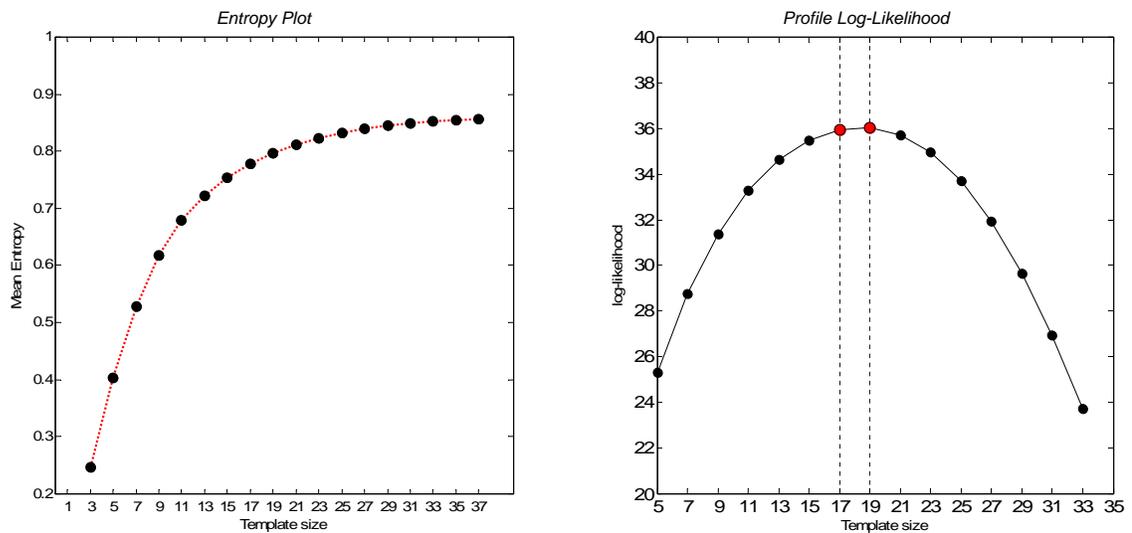


Figure 5.2: The training image and the original feature used in its construction are shown on top. Six different realizations using different template sizes are shown in the middle and lower parts of the figure.



(a) Training Image



(b) Mean Entropy Curves

Figure 5.3: The application of automatic template selection on a 2D training image consisting of circles (a). Mean entropy curve (b-left) and profile log-likelihood of that curve (b-right) indicate best template size of 17×17 .

Non-stationary TI

A worthwhile study would be the application of the proposed approach for template selection on a non-stationary training image. A non-stationary training image does not have a

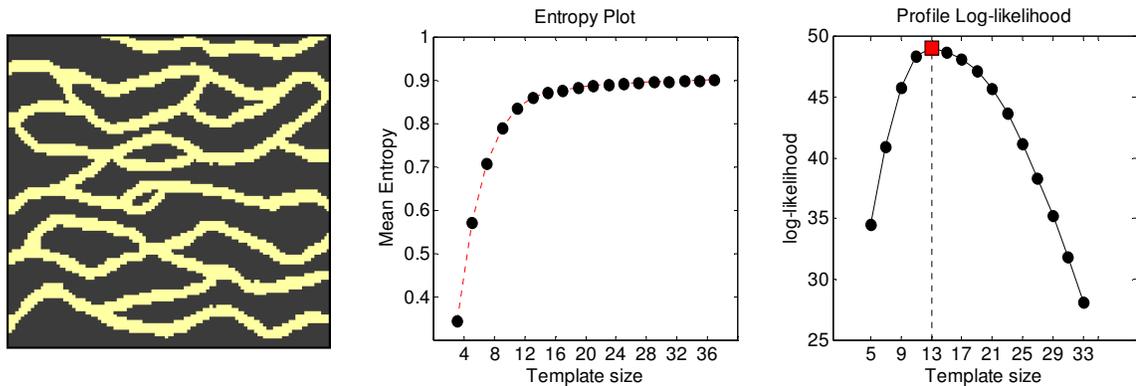


Figure 5.4: The application of automatic template selection on a 2D channelized training image. The figure shows the training image (left), the mean entropy curve (middle) and the profile log-likelihood (right). The best template dimension of 13×13 has been obtained.

clear optimal template size because no template, besides the one as big as the training image itself, can adequately establish a specific scale for the features. Therefore, one expects the proposed method to provide an entropy curve that increases non-linearly without any apparent elbow. In other words, the second derivative, f'' , of the entropy curve is almost constant in the non-stationarity case. The non-stationary training image and the comparison with a stationary one are shown in Figure 5.5. The entropy of both training images are shown in the lower left part. Evidently, a clear elbow is spotted in the stationary case, but no elbow can be identified in the non-stationary case. If we calculate the second derivative of these two curves, as shown in the lower middle plot, the non-stationary case leads to a constant curvature in contrast to the stationary one. In spite of all these differences, one expects the variance of the entropies observed within each specific template dimension to significantly decrease to zero when the template size grows larger than the optimal size. However, in contrary to the stationary case, the variance of the non-stationary training image does not drop to zero. This indicates the existence of non-stationary regions at all sizes of the template.

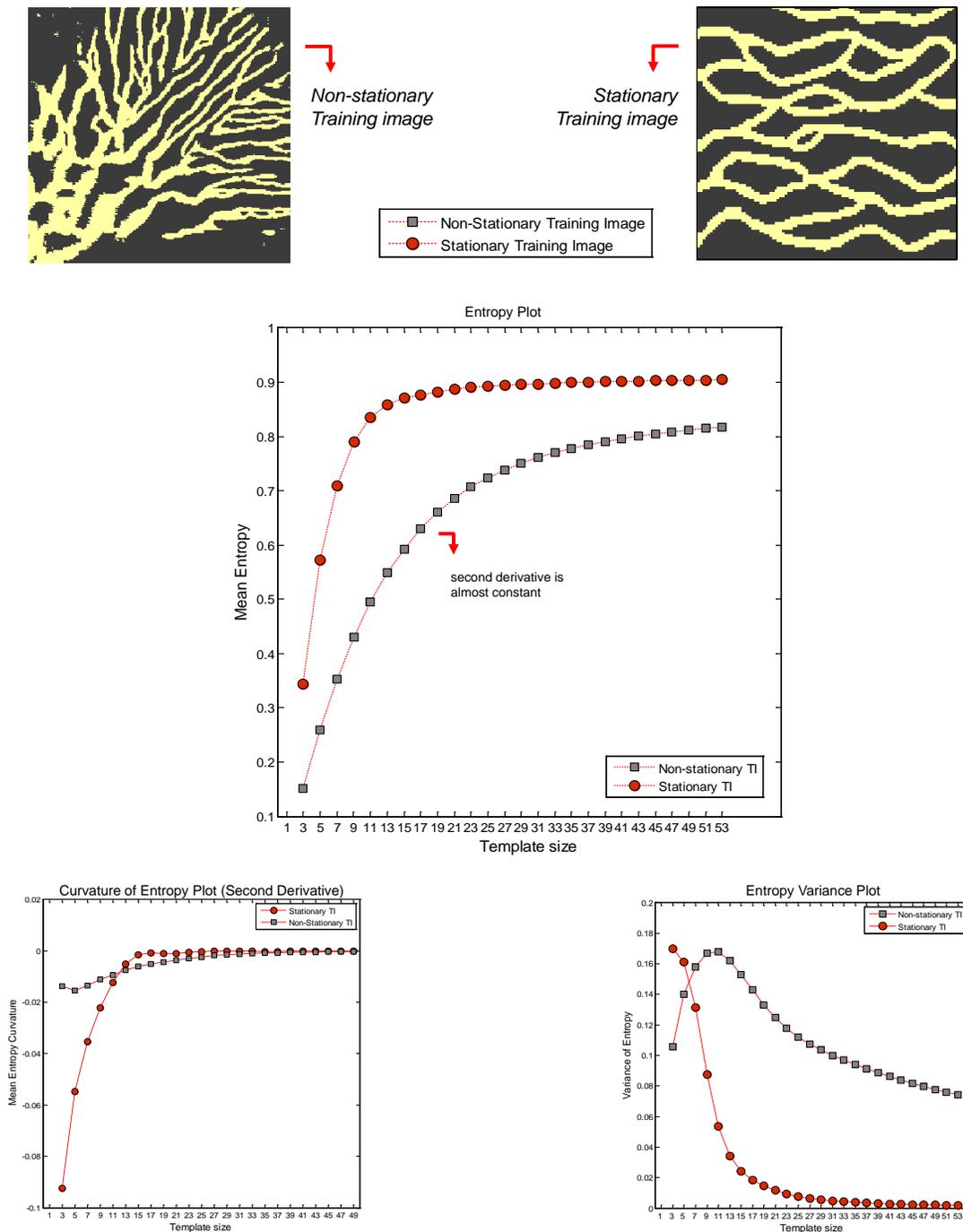


Figure 5.5: The application of automatic template selection on a non-stationary training image and the comparison with a stationary training image. The plots represent the mean entropy curve (left) and the second derivative of entropy curve (middle) and the variance of the entropy values (right) for both stationary and non-stationary cases. The plots demonstrate the non-existence of an appropriate template size for non-stationary training image.

5.3 Clustering

5.3.1 Concept

One of the main issues in any classification is the decision on the number of classes (clusters) prior to the algorithm. An adequate choice is often ambiguous. It depends on the shape and the distribution of patterns in the MDS space. Increasing the number of clusters (k) in the algorithm will reduce the errors in classification. In the extreme case, if we choose k to be equal to the number of data points, then the error would become zero and each point would belong to its own class. On the other hand, by choosing one cluster for the entire data set ($k = 1$), we are capable of reaching maximum compression. The main purpose in clustering is to compress the data into similar groups. Intuitively, the choice of the number of clusters is thus a compromise between low clustering error and high compression of the data.

In previous pattern-based method of filtersim, k needed to be selected a priori to simulation. The complexity of patterns and the internal details of each algorithm restrict the user from selecting an acceptable number beforehand. In filtersim, one needed to try different values for the number of clusters in order to obtain reasonable realizations. For example, Dujardin et al. (2006) conducted a sensitivity analysis on the number of clusters, or even the clustering method, for filtersim. There is a clear compromise in MPS methodologies caused by this classification. A higher value for the number of clusters results in accurate pattern classification, where the prototypes get sharper and similarity searches between a data event and the prototypes becomes more precise. However, the computational burden increases dramatically with more clusters because more similarity comparisons are required at each node during the simulation. On the other hand, choosing a small number of clusters will make the algorithm run much faster, but with a drawback of inaccurate pattern classification. Low number of clusters does not provide optimal compression, and the prototypes will not have the desired sharpness. As a result, the similarity searches between a data event and the limited prototypes become extremely error-prone. As mentioned earlier in this chapter, model complexity plays an important factor in parameter selection, as a compromise between speed and accuracy. In this section, we will introduce the technique for automatic selection of the number of clusters.

5.3.2 Methodology

In Section 2.4.2, two methods for choosing an optimal number of clusters were introduced for the k -means algorithm. Both of those methods work on the original data. Clustering in the proposed MPS methodology is performed in kernel-induced feature space. The data are now infinite-dimensional and only the inner products between the data are available. The assumptions of hyper-spherical clusters are not necessarily required in feature space as was the case for the k -means algorithm. Therefore, previous techniques are incapable of correctly analyzing the data structures in feature space.

In Honarkhah and Caers (2008), the optimal number of clusters was selected based on the *Minimum Description Length* (MDL) principle, in a trade-off between the likelihood of the model to the given data and the complexity of the model itself. However, the method is computationally expensive and all candidate number of clusters needed to be tested explicitly. In other words, to test for the number of clusters in the range of 1 to k , one needs to perform k independent classifications. Thus, a faster method for computing the optimal number of clusters is proposed. The simplicity of such an algorithm comes from the fact that kernel analysis of patterns has provided a database of all the pairwise distances between the points. This, in fact, can be exploited to find the optimal number of clusters. By analyzing the inherent structures within the distance matrix, which is already available from kernel transformation, one is capable of finding the best number of clusters.

In MDS space, a $N \times d$ dimensional data matrix needed to be manipulated for the optimization of the sum-of-squares criterion in k -means. The feature space counterpart now requires the manipulation of a $N \times N$ dimensional kernel matrix K . Each element of the kernel matrix represents a distance in the feature space; therefore, if the matrix elements belonging to the same cluster are re-ordered to lie next to each other, the kernel matrix will have a block diagonal structure. Let us provide an example to better understand the advantages of a kernel matrix. Assume we have eight data points grouped into two clusters in a two dimensional space. The data are provided in matrix X below:

$$X = \begin{bmatrix} 2.6625 & 1.1069 \\ 2.2941 & 1.5472 \\ 0.7372 & 0.7908 \\ 2.7260 & 1.6995 \\ 1.1467 & 0.8607 \\ 2.4963 & 1.6729 \\ 0.8335 & 0.5753 \\ 0.7268 & 1.1422 \end{bmatrix} \quad (5.3)$$

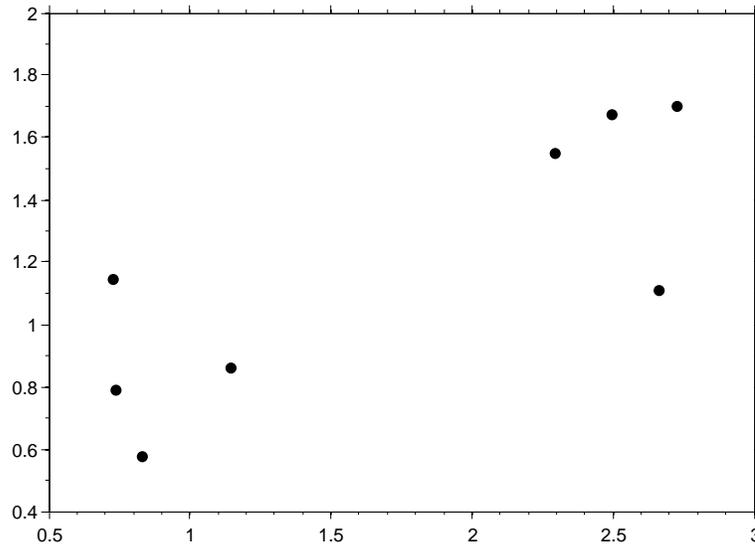


Figure 5.6: Eight sample data points shown in \mathbb{R}^2 , where two clear cluster of data is noticeable.

where each row represents one data point in \mathbb{R}^2 . The data points are shown in Figure 5.6 in a two dimensional Cartesian space. If we compute the kernel transformation of this data using a Gaussian kernel with bandwidth of 0.45, we would obtain the kernel matrix shown below:

$$K = \begin{bmatrix} 1.0000 & 0.2423 & 0.0081 & 0.2295 & 0.0226 & 0.2330 & 0.0091 & 0.0084 \\ 0.2423 & 1.0000 & 0.0139 & 0.3228 & 0.0368 & 0.5556 & 0.0131 & 0.0184 \\ 0.0081 & 0.0139 & 1.0000 & 0.0045 & 0.3585 & 0.0078 & 0.5583 & 0.4197 \\ 0.2295 & 0.3228 & 0.0045 & 1.0000 & 0.0121 & 0.5650 & 0.0044 & 0.0059 \\ 0.0226 & 0.0368 & 0.3585 & 0.0121 & 1.0000 & 0.0205 & 0.3512 & 0.2870 \\ 0.2330 & 0.5556 & 0.0078 & 0.5650 & 0.0205 & 1.0000 & 0.0073 & 0.0104 \\ 0.0091 & 0.0131 & 0.5583 & 0.0044 & 0.3512 & 0.0073 & 1.0000 & 0.2406 \\ 0.0084 & 0.0184 & 0.4197 & 0.0059 & 0.2870 & 0.0104 & 0.2406 & 1.0000 \end{bmatrix} \quad (5.4)$$

The kernel matrix now holds all the information necessary to characterize the data points. Without loss of generality and in order to observe structure in the kernel matrix, we permute the row/columns of the kernel matrix. The re-ordered matrix is shown in Figure 5.7, where colors represent the values of the matrix.

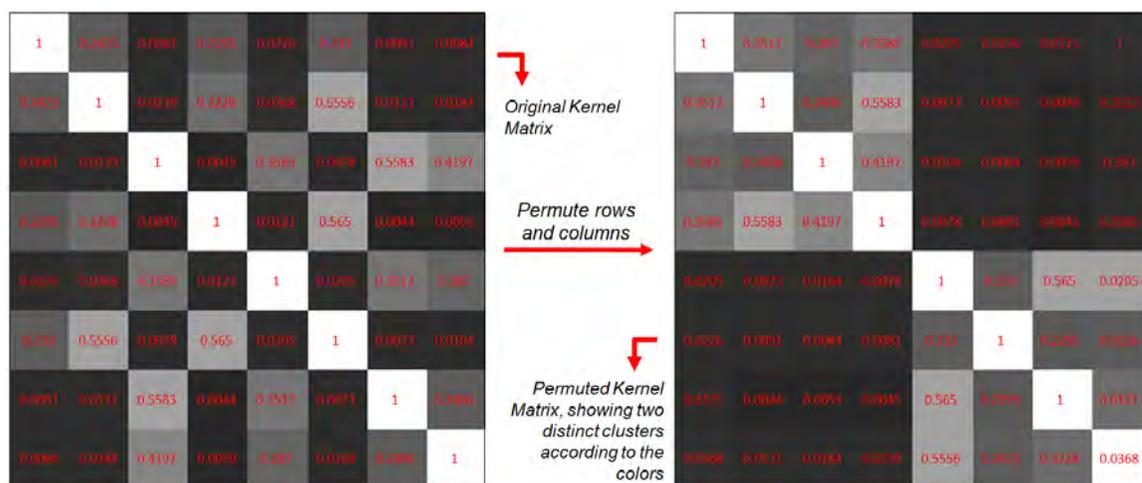


Figure 5.7: Kernel matrix is shown where each element is colored according to its value. Left figure shows the original kernel matrix, and right figure show the one where rows and columns are permuted. A clear structure between the data, indicating two clusters, is evident in the right figure.

This example illustrates how the structure of the kernel matrix can be exploited to find the desired clustering. In k -means algorithm one tries to minimize the sum-of-squares criterion. In other words, the sum of within-cluster distances, $\text{Tr}(S_w)$, is the objective function of k -means algorithm; where, Tr represents the trace of the matrix, and S_w is the

within-cluster distance matrix as follows:

$$S_w = \frac{1}{N} \sum_{k=1}^K \sum_{i=1}^N a_{ki} (\mathbf{x}_i - \mathbf{m}_k) (\mathbf{x}_i - \mathbf{m}_k)^T \quad (5.5)$$

where \mathbf{m}_k represent the cluster center as was calculated in Section 2.5.3, and a_{ki} represents the membership of datum \mathbf{x}_i to cluster k such that:

$$a_{ki} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in \mathbb{C}_k; \\ 0, & \text{if } \mathbf{x}_i \notin \mathbb{C}_k. \end{cases} \quad (5.6)$$

Similarly, in kernel k -means, the objective function for Gaussian kernel can be obtained as follows:

$$\text{Tr}(S_w^\Phi) = \frac{1}{N} \sum_{k=1}^K \sum_{i=1}^N a_{ki} (\Phi(\mathbf{x}_i) - \mathbf{m}_k^\Phi)^T (\Phi(\mathbf{x}_i) - \mathbf{m}_k^\Phi) \quad (5.7)$$

where $\mathbf{m}_k^\Phi = \frac{1}{N_k} \sum_{i=1}^N a_{ki} \Phi(\mathbf{x}_i)$ is the centroid in feature space. The properties of a Gaussian kernel help in simplifying the objective function formula as below:

$$\text{Tr}(S_w^\Phi) = 1 - \sum_{k=1}^K \left(\frac{N_k}{N} \right) S_{w|\mathbb{C}_k}^\Phi \quad (5.8)$$

where $S_{w|\mathbb{C}_k}^\Phi = \frac{1}{N_k^2} \sum_{i=1}^N \sum_{j=1}^N a_{ki} a_{kj} K_{ij}$, represents the average sum of all within-cluster distances of cluster k . On the other hand, according to the convolution theorem of Gaussians (Friedman and Tukey, 1974):

$$\int_{\mathbf{x}} p(\mathbf{x})^2 d\mathbf{x} \approx \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N K_{ij} \quad (5.9)$$

Accordingly, one can concur with the following relationship: $S_{w|\mathbb{C}_k}^\Phi \approx \int_{\mathbf{x} \in \mathbb{C}_k} p(\mathbf{x} | \mathbb{C}_k)^2 d\mathbf{x}$, (Girolami, 2002). This represents a measure of distribution compactness of cluster k . It is based on a non-parametric estimate of the probability density function of the data, $\int_{\mathbf{x} \in \mathbb{C}_k} p(\mathbf{x} | \mathbb{C}_k)^2 d\mathbf{x}$. Therefore, the objective function in the kernel k -means algorithm can be seen as maximizing the sum of compactness of all clusters 1 to k . In this case, the kernel matrix can be re-ordered according to the desired cluster k such that the compactness of each cluster is maximized. However, without any prior estimate on the number of clusters, one

can analyze the compactness of the entire dataset by observing the behaviors of $\int_{\mathbf{x}} p(\mathbf{x})^2 d\mathbf{x}$ instead. Since $\int_{\mathbf{x}} p(\mathbf{x})^2 d\mathbf{x} \approx \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N K_{ij} = \mathbf{1}_N^T \mathbf{K} \mathbf{1}_N$, we can analyze the kernel matrix itself. Here, $\mathbf{1}_N$ is an $N \times 1$ dimensional vector with values $\frac{1}{N}$.

However, the previous example (Figure 5.7) showed that the original kernel matrix is unstructured. We need a technique that does not rely on the correct re-ordering or permutations of the kernel matrix. Since the eigenvectors of a permuted matrix are the permutations of the original matrix, an indication of the number of clusters within the data may be given from the eigenvalue decomposition of the kernel matrix (Girolami, 2002). If \mathbf{u}_i denote the eigenvectors of matrix \mathbf{K} , and λ_i the eigenvalues, we can write:

$$\int_{\mathbf{x}} p(\mathbf{x})^2 d\mathbf{x} \approx \mathbf{1}_N^T \mathbf{K} \mathbf{1}_N = \sum_{i=1}^N \lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2 \quad (5.10)$$

Therefore, the number of clusters that are obtained by analyzing the distribution compactness of clusters can be obtained by analyzing the dominant terms of $\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2$. In other words, by plotting $\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2$ against i , a curve with an apparent elbow should be obtained. By finding the elbow of this plot (using the formulations of Section 2.3.3), we can obtain an optimal number of clusters prior to the classification.

The procedure is explained in Algorithm 7. It should be mentioned that two extra steps are required for a robust performance of this algorithm. First, we find the elbow of not the $\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2$ value, but their logarithm, $\log(\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2)$. This is needed to account for the huge shifts at the beginning of the plot (i.e. $i = 1$ stands for only one cluster). The next step is to smooth the data. The profile log-likelihood methodology of Section 2.3.3 is very sensitive to the variance in the data. Therefore, we provide a smoothed variant instead of the original function to find the elbow.

Algorithm 7 Number of Clusters in Kernel Space

- 1: find the eigenvalue decomposition of the kernel matrix, $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$.
 - 2: Sort $\mathbf{\Lambda}$ and \mathbf{U} from the highest to the lowest eigenvalue.
 - 3: Let $\mathbf{1}_N = [\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N}]_{(1 \times N)}$
 - 4: **for all** eigenvalues (λ_i) and eigenvectors (\mathbf{u}_i) **do**
 - 5: Plot $\log(\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2)$ with respect to i
 - 6: **end for**
 - 7: Smooth the plot using an averaging filter of size 3.
 - 8: number of clusters \leftarrow find the *Elbow* of this plot.
-

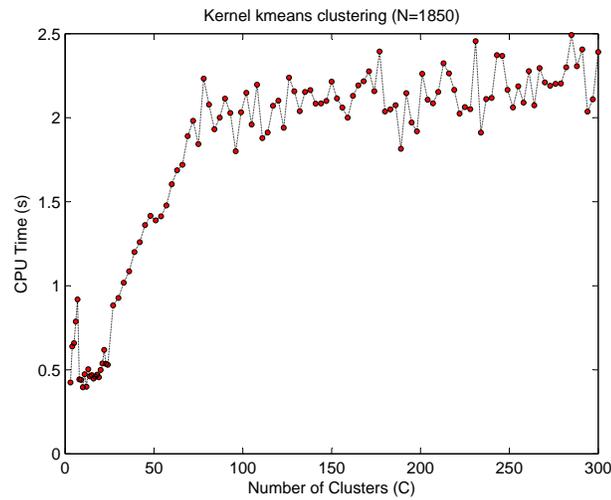
5.4 Empirical Evaluation

Evaluating the methodology on the data set obtained from the patterns of a training image is a subjective task. This is because of the complexities among the patterns, and most importantly, lack of distinct and clear clusters. Data are all spread out over space, and a visual estimation of the number of clusters is somewhat speculative.

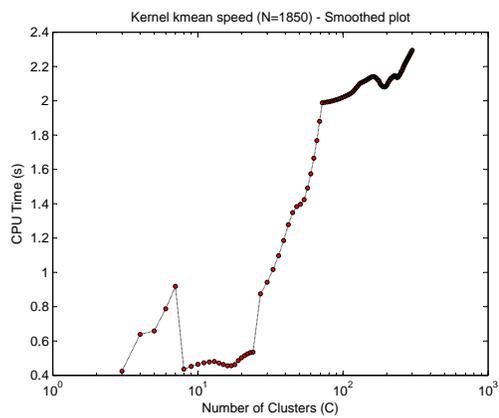
In order to test this methodology, a comparison in computational complexity of the kernel k -means algorithm is made with respect to the number of clusters. It is believed that the time complexity of the clustering algorithm is related to the true nature of data structures, and hence, to the actual number of clusters. For instance, it would be easier for a clustering algorithm to converge to a solution if the number of clusters is accurately provided. Therefore, the behavior in the complexity of the k -means algorithm will get affected dramatically according to the number of clusters.

A set of 1850 patterns is obtained from the channel training image shown previously in Figure 3.8, and were mapped to a 15 dimensional MDS space. The kernel k -means algorithm was applied on the dataset. The computational time of the algorithm was obtained by an average of 30 attempts over the cluster count of 3 to 300. The result is shown in Figure 5.8(a). As observed, the behavior changes at some values related to particular number of clusters. This can be clearly observed in the semi-log smoothed version of the same plot in Figure 5.8(b). Four different behaviors are noticeable. Figure 5.8(c) shows the log-log behavior and different computational complexities associated to each region of the cluster counts. Presumably, the true number of clusters is related to the observed turning points. Since a visual inspection was not possible, this computational test has been carried out instead as it implicitly provides us with the true number of clusters. The Algorithm 7 on the methodology for finding the number of clusters will now be tested on this dataset, and the behavior will be analyzed according to the turning points observed in the computational test.

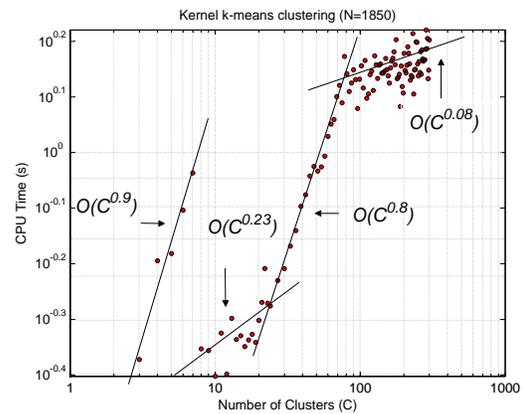
According to Algorithm 7, $\lambda_i \{ \mathbf{1}_N^T \mathbf{u}_i \}^2$ with respect to i is plotted and the elbow of the plot is calculated. However, due to the complexity of data and the elbow selection methodology, we have provided three different interpretations on the optimal number of clusters. Evidently, the higher the number of the clusters is, the better the clustering becomes. Here, we observe three possible candidates for the number of clusters. They are all in exact agreement with the previous analysis on the computational complexity of kernel



(a) Computational Speed



(b) Semi-Log smoothed



(c) Log-Log

Figure 5.8: Computational speed of the kernel k -means algorithm with respect to the number of clusters. (a) showing the normal plot, (b) showing the smoothed version of the semi-log plot with four distinct regions, (c) the Log-Log plot showing different speed behaviors in each region.

k -means with respect to the number of clusters. The first choice is to have eight clusters such that the kernel matrix can have a satisfactory block diagonal structure (which manifests a good clustering) . It is shown in the semi-log plot in Figure 5.9(a). Next possible candidate is 25 clusters (Figure 5.9(b)), which corresponds to a better clustering and is occurring at the same point where the behavior in Figure 5.8(b) changes. Finally, the last one with

80 clusters is also another optimal choice for MPS simulations (Figure 5.9(c)). In here, only the first 80 values of $\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2$ contribute to the overall compactness of data; thus indicating 80 dominant clusters within the data sample.

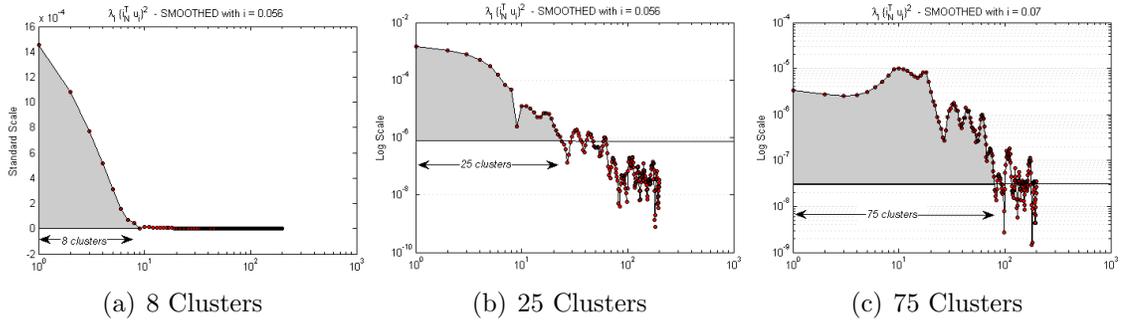


Figure 5.9: Plot of $\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2$, and observing the contribution of each of these terms to the overall value. The results are interpreted at different scales of normal or logarithmic (considering the importance of the accuracy obtained with clustering).

The provided interpretations are subjective. In order to exactly evaluate how the algorithm would work on the chosen dataset, we will show it step-by-step. Figure 5.10 displays all necessary steps in the algorithm. On top of this figure, a plot of $\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2$ versus i is shown. The curve has a steep behavior that flattens out after the first three values. Therefore, as mentioned before, we analyze the logarithm of this function, that is $\log(\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2)$, instead. The middle plot in this figure shows a clear elbow. Finally, we apply the profile log-likelihood method on the smoothed variant of the middle plot to obtain the elbow; indicating the optimal number of clusters. As can be seen in the smoothed version (bottom-left plot), the elbow becomes more evident. Upon the application of profile log-likelihood, a reasonable range of around $k = 65$ to $k = 85$ is obtained. This range is in good agreement with our previous analysis using the computational time of the kernel k -means algorithm ($k = 75$). This example demonstrates the effectiveness of the proposed algorithm without any subjective decisions. Although the eigenvalue decomposition of a large kernel matrix can become computationally expensive, the computation time would be much less than the time needed for a practitioner to find a reasonable choice by trial-and-error, and also, much less than performing the kernel k -means clustering for validating each of the number of clusters. In addition, not all eigenvalues need to be calculated in the proposed approach. We limit the search to $3\sqrt{N}$ numbers, where N denotes the number

of patterns. It would then be of great computational advantage not to do a full decomposition of the kernel matrix, but use the *Arnoldi iteration* method to obtain only the most dominant eigenvalues with their corresponding eigenvectors.

In conclusion, once the kernel matrix has been defined, then, only one eigenvalue decomposition is required to establish an optimal partitioning. This is contrary to the method proposed in Honarkhah and Caers (2008) where for each candidate partitioning, the outcome of a nonlinear optimization routine is used in validating the partition based on the data. Therefore, at least as many nonlinear optimization routines as there are possible clusters would be required. The proposed approach, on the other hand, provides us with a more robust selection on the number of clusters. The method of profile log-likelihood is then used for an automatic selection of the number of clusters.

5.5 Conclusion

In this chapter, we attempted to eliminate the subjectivity in choosing the parameters of the algorithm. It could be seen as a technique that finds the correct parameters of the spatial model of continuity. The ultimate goal is to have a modeling framework that can objectively characterize and learn the geological features of the subsurface.

Two main parameters in the DisPAT methodology were the choice on the size of the template and the number of cluster. The former directly defines the spatial model of continuity and the multiple-point statistical features of the training image. A correct choice of this parameter will not only capture the inherent stationary features of the conceptual geological model, but will also lead to visually-appealing realizations. The second parameter of the number of clusters has a direct impact in both the quality and complexity of the results as well. A method that can take advantage of the information carried in the kernel matrix was introduced. It was demonstrated that such a technique is capable of revealing an optimal representation of the patterns with a set of clusters.

The objective analysis for these two main parameters aids us in reaching a parameter-free learning approach that facilitates the application of MPS methodologies in real data sets.

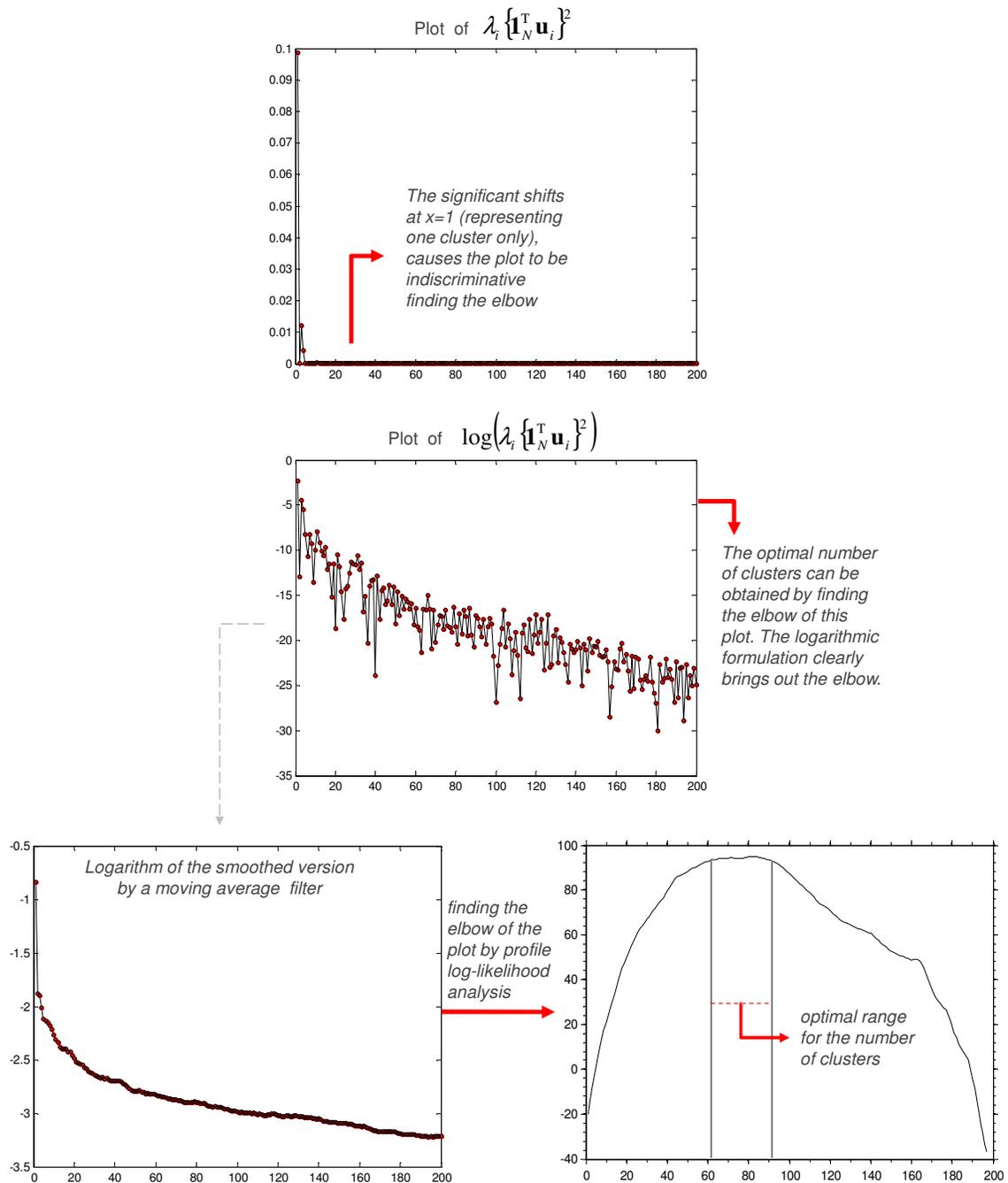


Figure 5.10: Example application of finding the optimal number of clusters. Top figure is the plot of $\lambda_i \{\mathbf{1}_N^T \mathbf{u}_i\}^2$, middle figure is the logarithm of the top figure. The bottom-left plot is the smooth version, and the bottom-right is the application of profile log-likelihood on finding the elbow.

Chapter 6

Data Integration

Science is facts; just as houses are made of stones, so is science made of facts; but a pile of stones is not a house and a collection of facts is not necessarily science.

HENRI POINCARÉ (1854 - 1912)

Geomodeling is the process of generating models that can capture the subsurface structures of a field. The ultimate goal of these geological models is to predict the flow performances. For instance, the models are used in flow simulators to obtain the uncertainty range in the cumulative oil production of the field and the business strategy. Constructing an accurate model is thus vital in the decision making process. However, accurate models are never attainable. There are many sources of uncertainty in the subsurface. The permeability of the field, the porosity, water-oil contact level, fault networks, geology of the field are all simple examples of the unknown subsurface properties or characteristics. However, not everything is unknown. Some prior knowledge is available. In order to model the field under study, one needs to take them into account. For this matter, geostatistics provide tools for generating models that can capture the inherent spatial uncertainties. More specifically, multiple-point geostatistics can generate Earth models with a prior geological concept in mind. It can reproduce the structures and features deemed relevant by the geologist.

To further reduce spatial uncertainty, all available sources of information need to be integrated within the models. Models should be able to reflect our knowledge to the full extent. There are two main sources of information, local and global. They provide auxiliary

information that, if integrated within the model, will reduce the uncertainty in the generated realizations. The local information are the ones obtained locally from the well data. They are accurate measurements on specific locations in the field, and hence, are called hard data. The other source of information is the one that globally measures some sort of characteristic about the field. For example, seismic data provide a low resolution characterization of the field. These sources of data are less accurate, and less detailed than the hard data, and hence, are called soft data. They provide soft information about the field. While the expertise from different disciplines is gathered and conveyed through a training image, various sources of additional data, with different scales, should also be integrated into the reservoir modeling practice. One of the main advantages of multiple-point statistical methods over object-based simulations is their capability in conditioning models to dense hard and soft data. Data integration enables us to obtain better prediction capabilities in our modeling processes.

The concepts of data integration can be similarly explained within a Bayesian perspective. If we denote our models by \mathcal{D} , then without any additional information, our models are generated according to their prior distributions, $p(\mathcal{D})$. However, additional sources of information can alter this probability. Let's assume two sources of information are available; hard data and soft data denoted by \mathcal{B} , and \mathcal{C} , respectively. Then, the posterior distribution of our models obtained through stochastic simulation can be calculated by $p(\mathcal{D} | \mathcal{B}, \mathcal{C})$. This probability distribution provides more accurate representations of our models than just the priors. In this section, we will describe the techniques for integrating the two sources of information; namely hard and soft data. It is a necessary step towards stochastic modeling and uncertainty quantification. First, we will describe the techniques for hard data conditioning, and then, soft data integration is explained. Many examples will be provided for comparison, and the capabilities of these approaches will be demonstrated.

6.1 Hard Data Conditioning

6.1.1 Method

Hard data are provided as local information on node values within a grid. Denote the hard data grid by \mathbf{hd} that is discretized by the same cartesian grid \mathbb{G}_{re} used for realization \mathbf{re} . All the nodes of \mathbf{hd} grid are unknown, and only the hard data locations are filled with values.

An example grid is shown in Figure 6.1, where two hard data are available; indicating sand and shale at different locations.

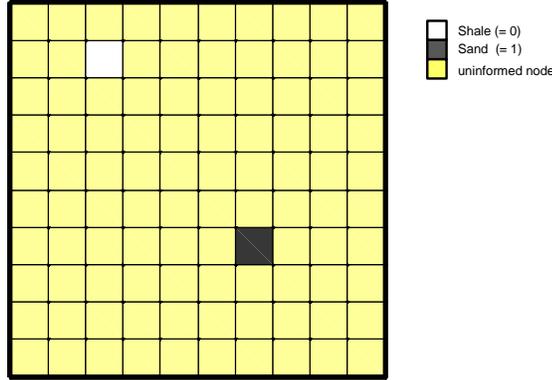


Figure 6.1: Example of a hard data grid \mathbf{hd} , with two hard data one with a value of zero (indicating shale), and the other with one (indicating sand).

In pattern-based methodology of *simpat*, hard data conditioning is performed by a two stage process (Arpat, 2005). In the first stage, upon obtaining the hard data event $\mathbf{hdev}_T(\mathbf{u})$ from the hard data grid \mathbf{hd} , it calculates the similarities between $\mathbf{hdev}_T(\mathbf{u})$ and the patterns in the database. It then stores all patterns that have a similarity greater than a specific threshold, $s\langle \mathbf{hdev}_T(\mathbf{u}), \mathbf{pat}_T^k \rangle$, in a new database. In the second stage, it searches for the most similar pattern \mathbf{pat}_T^* to the data event $\mathbf{dev}_T(\mathbf{u})$ obtained from realization \mathbf{re} , from the newly constructed pattern database that has already been matched with hard data. Finally, it pastes the pattern \mathbf{pat}_T^* on the realization grid \mathbf{re} . This process works well for *simpat*. However, for the proposed framework where the data events are only compared with cluster prototypes, it becomes impractical.

In our proposed pattern-based multiple-point statistical method, the same approach taken by *filtersim* is employed (Wu, 2007). The idea behind the process is to incorporate the hard data grid \mathbf{hd} into similarity search during the simulation in only one stage. Hard data are assumed known and certain, and therefore, their values are to be copied exactly on the realization grid \mathbf{re} . In other words, the simulation starts by copying \mathbf{hd} into \mathbf{re} . During the simulation, the hard data locations \mathbf{u}_{hd} are not included in the random path. In addition, their values are fixed on the realization grid; to the extent that the patterns pasted on their neighboring nodes are prohibited to change the value at \mathbf{u}_{hd} .

In *filtersim*, hard data conditioning is carried out by giving different weights to different

node locations in the pattern similarity calculation. For example, hard data nodes are given the highest weight of 0.5, and frozen nodes 0.3, and the rest of the nodes 0.2. This is an arbitrary choice of weights that the practitioner can adjust according to the problem at hand. Then the closest cluster prototype is found by minimizing the weighted similarity function. In order to have a reliable comparison in this section, the same approach as `filtersim` is implemented in the proposed algorithm. Therefore, the distance from a data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ to a prototype \mathbf{prot}_i can be calculated according to different weights for each node as follows:

$$d\langle \mathbf{dev}_{\mathbf{T}}(\mathbf{u}), \mathbf{prot}_i \rangle = \sum_{j=1}^{n_{\mathbf{T}}} \omega_j \cdot |dev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_j) - \mathbf{prot}_i(j)| \quad (6.1)$$

where ω_j represents the weight associated with the node at location $\mathbf{u} + \mathbf{h}_j$. If that node is a hard data, a large weight of 0.5 will be assigned to it. Other kinds of nodes will have smaller weights of 0.3 or 0.2. This scheme of weighting ensures the transfer of spatial information inherent in the hard data to the simulation grid. For example, if the training image is composed of elongated objects in E-W direction, the presence of a hard data value of 1 at \mathbf{u} will not only enforce $re(\mathbf{u}) = 1$, but will also increase the possibility of occurrence of that object in neighboring nodes. This is because an elongated object exists in more than one grid node, and when a location \mathbf{u} consists of that object, the neighboring spatial locations along the E-W direction will also consist of the same object.

To perform a full comparison of the `filtersim` method with the proposed methodology, hard data conditioning is examined in the next section. One expects higher data conditioning capability for the proposed methodology than for `filtersim` due to the same reasons that resulted in better pattern reproduction in generated models; namely better pattern modeling and classification.

6.1.2 Examples

The binary training image of Figure 3.8 is chosen for analysis. In the first test, only five hard data are selected for simulation. The ensemble average of a given set of realizations (EA) is used to check the quality of conditioning. If the algorithm conditions properly, the ensemble average should provide less uncertainty near hard data locations. 100 realizations are generated using the proposed method, `filtersim`, and `sgsim` (with a proper variogram),

and their ensemble averages (EA) are calculated. The results are shown in Figure 6.2. The EA of the proposed method shows the channel structure around the hard data locations and also the expected spatial structures as one moves away from the hard data. The continuity of the channel passing through the hard data at the center of the grid is evident. On the other hand, in filtersim, the proportions are not satisfied and a stronger tendency towards shale facies is observed. Moreover, the structural patterns are not fully captured in the filtersim ensemble average. In order to verify this, the ensemble average of 100 sgsim realizations is also shown. The EA from sgsim accounts only for two-point statistics, yet, provides similar results as filtersim. This demonstrates the better data conditioning obtained with the proposed method.

Next, 100 hard conditioning data are used for the comparison. The results are shown in Figure 6.3. Recall that filtersim was used without proportion control, resulting in more shale facies present in the realization and making the comparison difficult. However, by observing the structures, one can verify that the proposed method (in an average sense) has converged to the reference case.

Next, a conditional simulation is performed with two neighboring hard data. In this analysis, a sand and a shale facies are assigned to two vertical neighboring nodes respectively. This configuration indicates the limit of a channel passing through the sand location. The same binary channel training image is also used for data conditioning. A search template of 9×9 and an inner patch of 5×5 is chosen. In this case, a large number of clusters, 1000, is chosen in order to improve the filtersim results. 100 realizations are generated and an ensemble average is calculated for each of the three MPS methods: filtersim, snesim and DisPAT. The parameters for snesim are given in such a way as to closely emulate the pattern template used in pattern-based methods. The search template geometry in snesim is $(9, 9)$, and the number of nodes in the search neighborhood is 25. It is worthwhile mentioning that every algorithm has its own set of parameters that needs tuning for better results, but for the sake of comparison, we have assumed similar situations in all MPS simulations. The results are shown in Figure 6.4.

All MPS methods exhibit the expected sand and shale structures that were provided through the neighboring hard data. However, the results obtained by filtersim and snesim demonstrate worse pattern reproductivity as one moves away from the hard data locations. According to the provided training image, laterally, there should be a channel passing

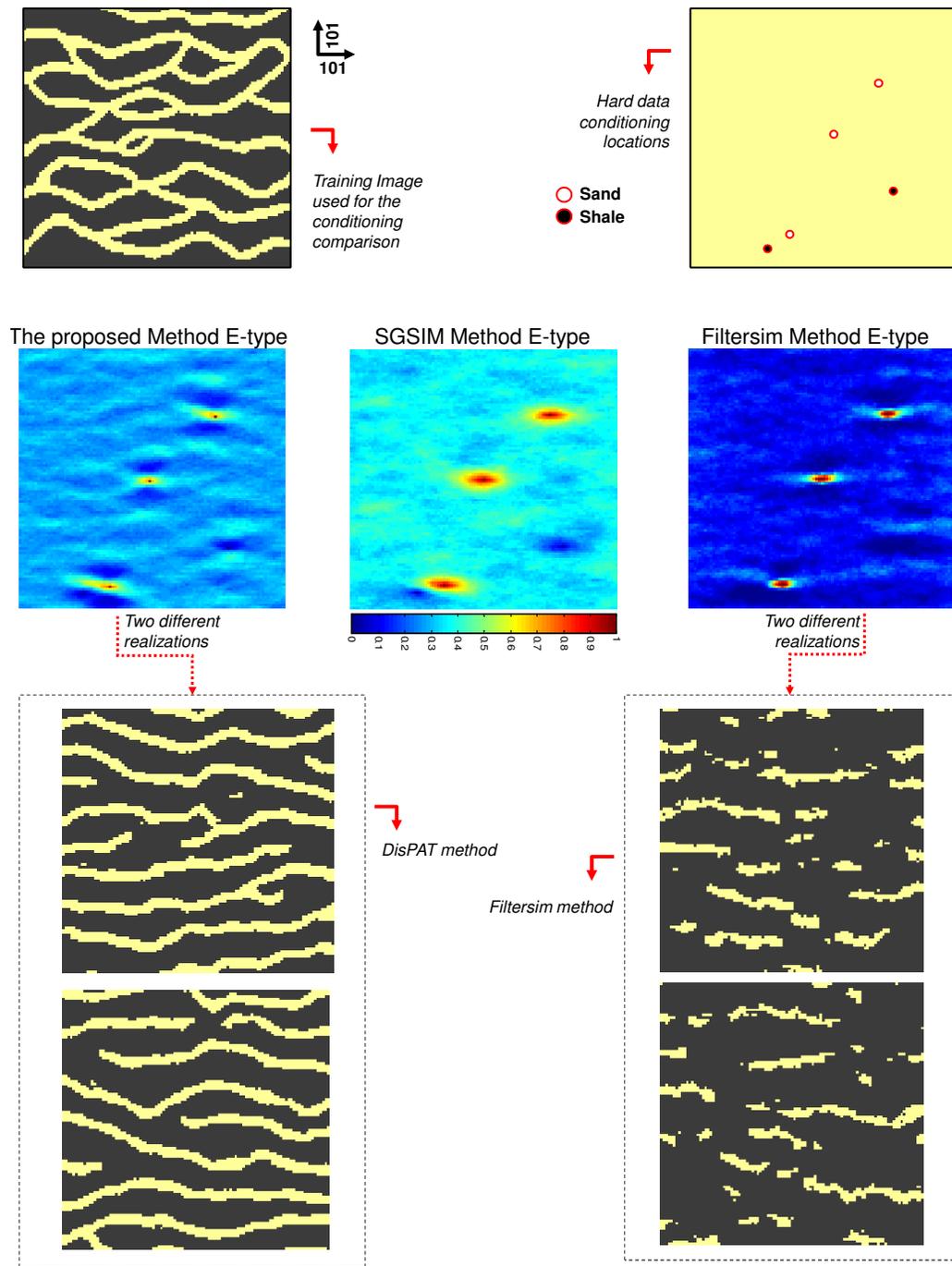


Figure 6.2: The hard data conditioning locations are shown in top. The ensemble average (E-Type) of 100 realization generated using the proposed method (left), the sgsim method (middle) and filtersim method (right) are shown. The multiple-point structures are better captured in the proposed methodology as seen in the ensemble averages. Two different MPS realizations of each method is also shown on the bottom.

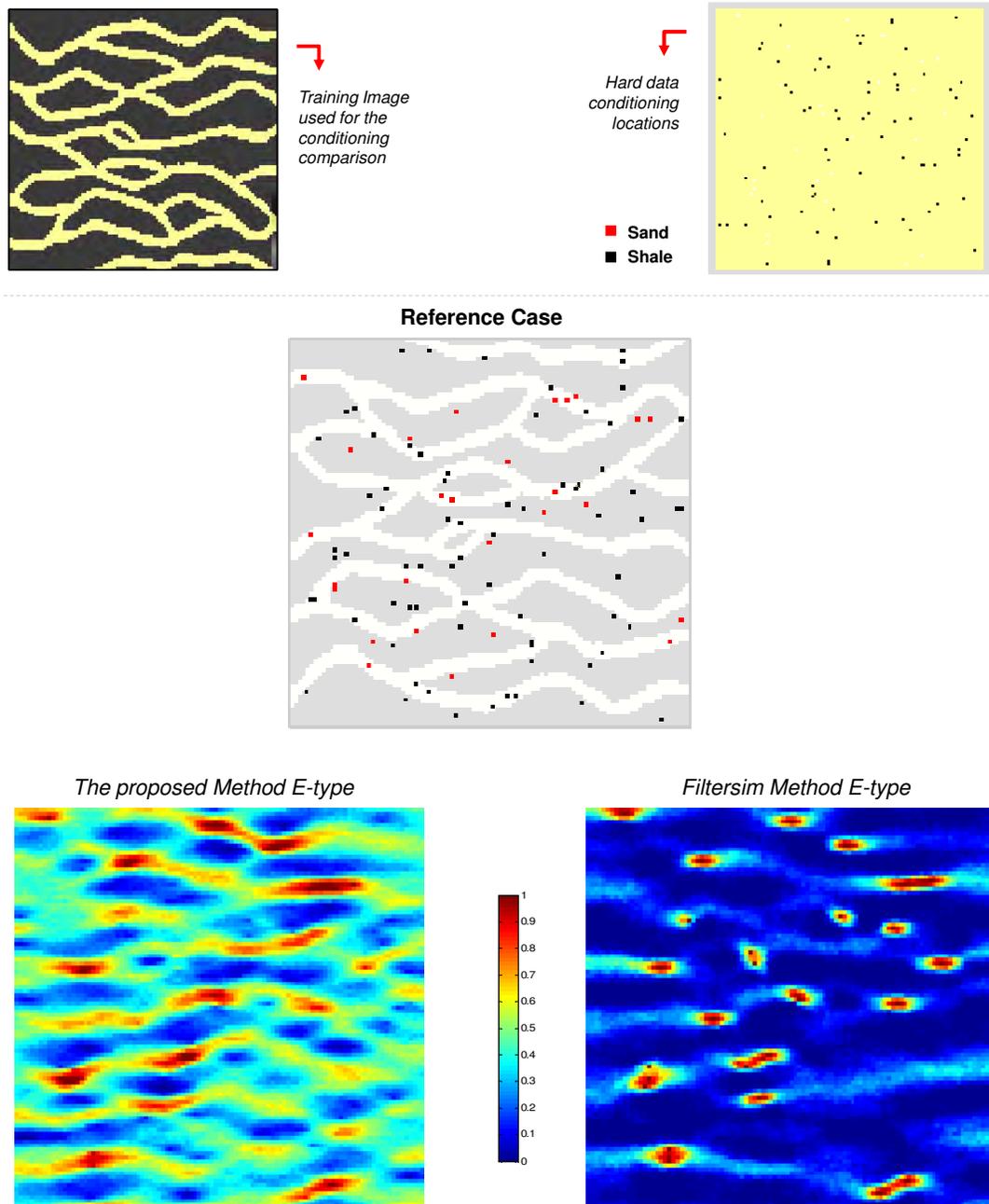


Figure 6.3: The hard data conditioning locations are shown in top. The ensemble average (E-Type) of 100 realizations generated using the proposed method (left-bottom) and filtersim method (right-bottom) are shown. The reference case is also shown on left.

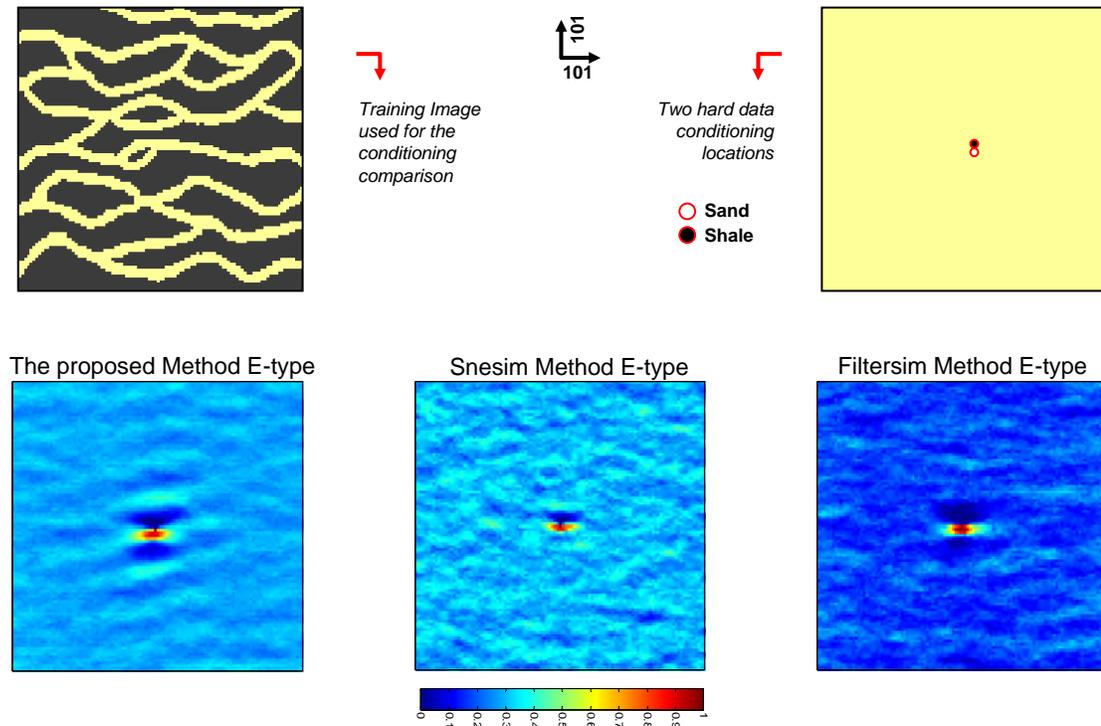


Figure 6.4: The neighboring hard data conditioning locations are shown in top. The ensemble average (E-Type) of 100 realizations generated using the proposed method (left), snesim method (middle) and filtersim method (right) are shown in the bottom.

through the sand hard data in the realization and a shale structure on top of it. Furthermore, knowing the average width of a channel, a human expert can infer a shale structure underneath the channel structure with reasonable certainty. Neither of the algorithms, snesim or filtersim, accurately captures the multiple-point information that was provided through the training image. On the other hand, the ensemble average realization obtained by the proposed methodology demonstrates the same channel structures as would be determined by a human expert. In order to confirm this, a window of size 35×35 is used to scan over the training image and store all the patterns that honor the two hard data. Figure 6.5 shows the ensemble average of this 35×35 pattern, which is the reference target to be achieved. It is perceived that the training image structures correctly resemble the ones seen in the central part of the ensemble average obtained by the proposed methodology, as opposed to snesim and filtersim.

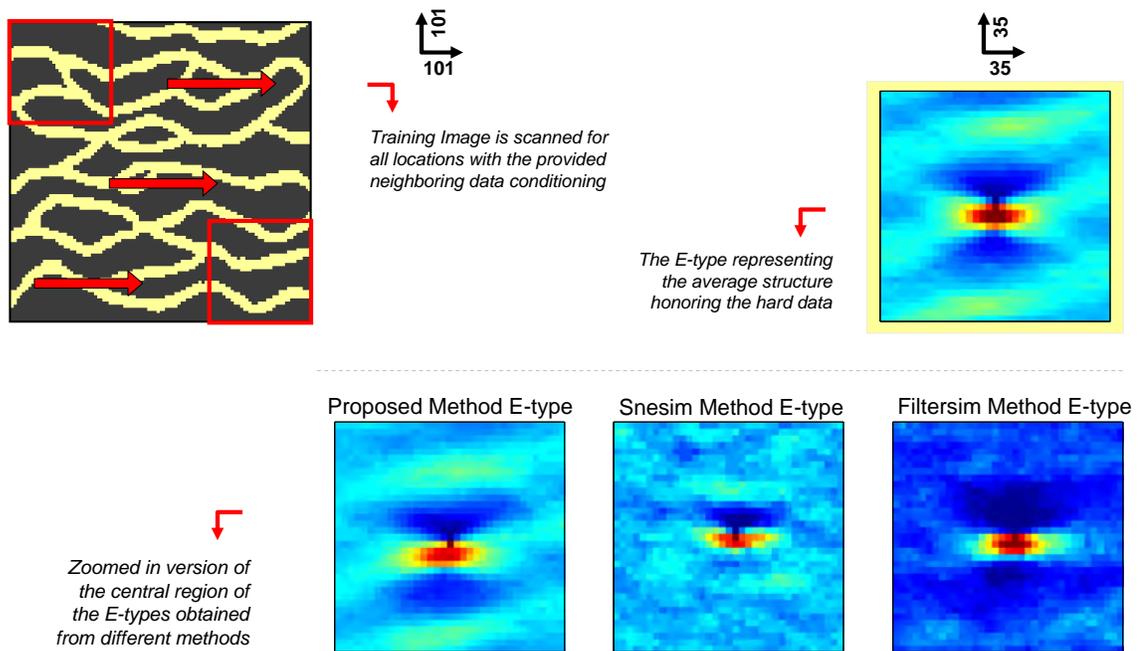


Figure 6.5: The average of all 35×35 patterns in the training image that honor the neighboring hard data at their central nodes is shown on top-right corner. The zoomed-in version of the ensemble average (E-Type) of different methods is shown for comparison on the bottom.

The provided examples demonstrate that the conditioning in the proposed method is better than other algorithms, and at the same time, produces the essential features of the training image. This has the advantage of generating better and more realistic geological models.

In this section, we introduced a hard data conditioning approach and analyzed its capabilities. This method dealt only with single-grid simulations. In the next section, we describe the additional steps required for multi-scale hard data conditioning. All previous MPS techniques follow the same conditioning methodology for a multi-grid setting. We review the multi-grid data conditioning method. We will also show some of its inherent shortcomings, and provide solutions accordingly. Afterwards, we introduce the conditioning algorithm for the new multi-resolution approach, and demonstrate it with some examples.

6.1.3 Multi-Grid Approach

Data Relocation

The fundamental basis in hard data integration is for them to be exactly reproduced on the realization grid. However, this is not a sufficient basis, and one needs to ensure that large-scale spatial information corresponding to hard data to be honored as well. It was mentioned that multi-grid approach can model the long-range continuity of features and multiple-point statistics of the training image. The same procedure can be applied on hard data to take advantage of their joint spatial information.

One solution proposed by Strebelle (2000) for snesim is to assign hard data to all multi-grid realizations \mathbf{re}_g , for $g = n_g - 1, \dots, 0$. Assigning or copying the hard data grid \mathbf{hd} to the finest realization grid \mathbf{re}_0 is trivial, but the problem arises when copying hard data to another grid $\mathbf{re}_{g>0}$. In those situations, the nodes belonging to the multi-grid realization may not be the same nodes where hard data information exist, $\mathbf{u}_{hd} \notin \mathbf{G}_{re}^g$. Strebelle (2000) suggested a data relocation approach, where the hard data are relocated to their closest node in the multi-grid realizations. This ensures that hard data are taken into account from initial stages of simulation, and as a result, the finer grids are somewhat informed about the larger scale statistical dependencies of hard data. A simple data relocation example is illustrated in Figure 6.6.

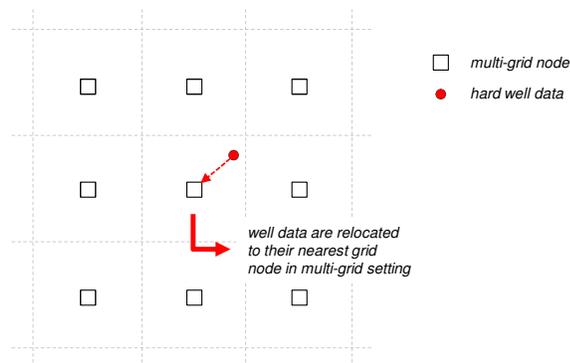


Figure 6.6: An example on how data relocation approach will move the hard data \mathbf{u}_{hd} from the hard data grid \mathbf{hd} to the nodes on the multi-grid realization \mathbf{re}_g .

However, Arpat (2005) argued that this method of data relocation does not fully honor the data; either the data location or the data values are not honored. As a solution,

Arpat (2005) introduced a dual-template approach for multi-scale simulation in *simpat*, and performed a data conditioning approach accordingly. In the dual-template approach, the hard data are not relocated in the multi-grid setting. But the dual-template, which is a template with same spatial domain of multi-grid template, but including all finer nodes, is used to search for all dual patterns that match the hard data in the fine grid, and consequently, only those matched patterns will be used for similarity search with the data event. Although the methodology does not rely on data relocation and can fully honor the hard well data, but it has a drawback regarding uncertainty. By restricting the search to multi-grid patterns, which have their dual-templates honoring the well data, the number of possible patterns reduces significantly. In dense well data, it may become even more difficult to match all data, and the most similar dual-pattern may be incapable of fully honoring the hard data, let alone the data events.

We are introducing a new approach much similar to the one in *snessim* on data relocation. In the next part, we will first analyze all the limitations inherent to the original data relocation approach of *snessim*. Afterwards, a solution that eliminates those problems will be described.

Limitations

Data relocation is a crucial step in hard data conditioning, since unreasonable shifts in the well data locations may cause dramatic differences in the final flow performances. For example, in a large grid and within the biggest multi-grid setting, data relocation can cause the wells to be moved hundreds of feet away from their original locations. Moreover, dense hard data in the multi-grid setting can be problematic due to the inability of the algorithm to handle such cases. In general, there are two inherent limitations to the current data relocation algorithm.

The first limitation involves the situation with spatially close well information. It can be due to dense well data in a field or simply two wells located in close proximity. The situation arises more than often due to the preferential drilling in profitable locations, where two or more wells are explored within a small area of the field. The current algorithm cannot adequately relocate the data to the correct multi-grid node. In order to understand this limitation we first need to understand how the algorithm behaves. The data relocation algorithm proceeds by sequentially going over all the informed nodes on the hard data grid \mathbf{hd} , that is locations \mathbf{u}_{hd}^i for $i = 1, \dots, n_{hd}$, where n_{hd} denotes the number of hard well

data. At each node location \mathbf{u}_{hd}^i it will find the closest multi-grid node $\mathbf{u}^* \in \mathbb{G}_{re}^g$. Before relocating it to node \mathbf{u}^* , it will check to see if any hard data has already been relocated to that location. It will then decide which one, the new hard data or the previously visited one, is closer to \mathbf{u}^* , and will only re-assign $re(\mathbf{u}^*)$ value if the new hard data is closer. In case no hard data has been assigned to node \mathbf{u}^* , it will automatically copy it to that grid node ($re_g(\mathbf{u}^*) = hd(\mathbf{u}_{hd}^i)$). The process continues until all n_{hd} hard data are analyzed. According to this methodology, it becomes clear that some hard data may need to be dropped in this process, that is, they never get assigned to any node location in the realization grid \mathbf{re}_g . We illustrate this situation with a simple example. Note that in all these figures, the hard data are represented by a circle indicating their locations, and their values are not relevant in this analysis. Figure 6.7 shows the dropped hard data in the relocation process. This has an impact on large-scale reproduction of features conveyed through those hard data. The inaccuracy caused by eliminating one hard datum from the simulation in that multi-grid setting can propagate to the final realization. This is one source of inaccuracy and inconsistency inherent to the current data relocation algorithm.

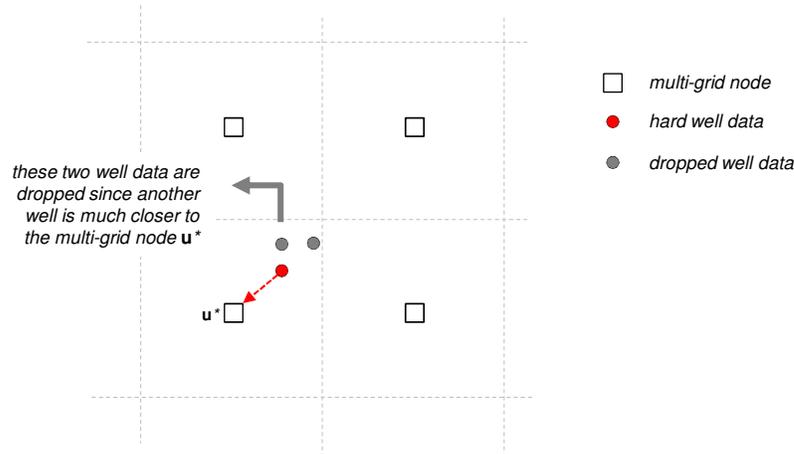


Figure 6.7: An illustration of the first limitation in current data relocation practice, where well data in proximity to each other can be dropped and eliminated in the process.

The second limitation originates from the specific multi-grid definition. The grid defined for multi-grid level g is denoted by \mathbb{G}_{re}^g and is fixed in all simulations. The problem arises from data relocation of a hard data to a new location on grid \mathbf{u}_{re}^* for all realizations. This produces an artifact or a spatial inconsistency within all realizations (Caers, 2011). The

ensemble average of all generated realizations does not necessarily reproduce the multiple-point structures of the training image. In order to illustrate this concept, we will provide two approaches for hard data conditioning. A simple example with only one hard datum is analyzed. In the first approach, the original hard data integration algorithm is used during simulation. For the second approach, we will apply rejection sampling. Rejection sampling provides the most accurate methodology for data integration where unconditional realizations are constructed by the algorithm, and only those that match the hard data are accepted, while others are rejected. The rejection sampling process does not rely on any conditioning algorithm. Thus, one expects a correct data conditioning algorithm to reproduce similar results, or the ensemble averages of the method to converge to the solution obtained by rejection sampling. Figure 6.8 illustrates the training image used and one hard datum with some realizations generated by both *snesim* and the rejection sampling. The ensemble averages (E-types) of the two methods are also depicted in the bottom figures. Evidently, there is a spatial mismatch between the current algorithm and the one obtained through rejection sampling. This demonstrates the internal multiple-point inconsistencies produced by the current data relocation algorithm.

In the next section, we will introduce two solutions for each of these limitations. These solutions will resolve the mentioned issues and reproduce the same expected outcomes of the computationally infeasible, but accurate, rejection sampling.

Smart-Vibrating Multi-Grid Approach

In this section, a ‘smart-vibrating multi-grid approach’ is provided that can resolve the two previously mentioned limitations. In the first part, the limitation concerning the inability of data relocation to accommodate dense well data or wells within close proximity is resolved. Afterwards, the second limitation regarding the distortions or internal inconsistencies caused by data relocations themselves will be addressed.

For the first limitation, the algorithm should be smart enough to understand which hard well data are to be copied to which nodes on the multi-grid realization \mathbf{re}_g . It should also know which hard data are to be dropped within a very complex dense configuration of wells. In order to better understand the final goal of the proposed data relocation approach we will provide three different examples with increasing complexities, and analyze the expected optimal behavior for hard data assignments. It should be mentioned that all examples are provided for a two dimensional grid, but the same arguments apply to three dimensions.

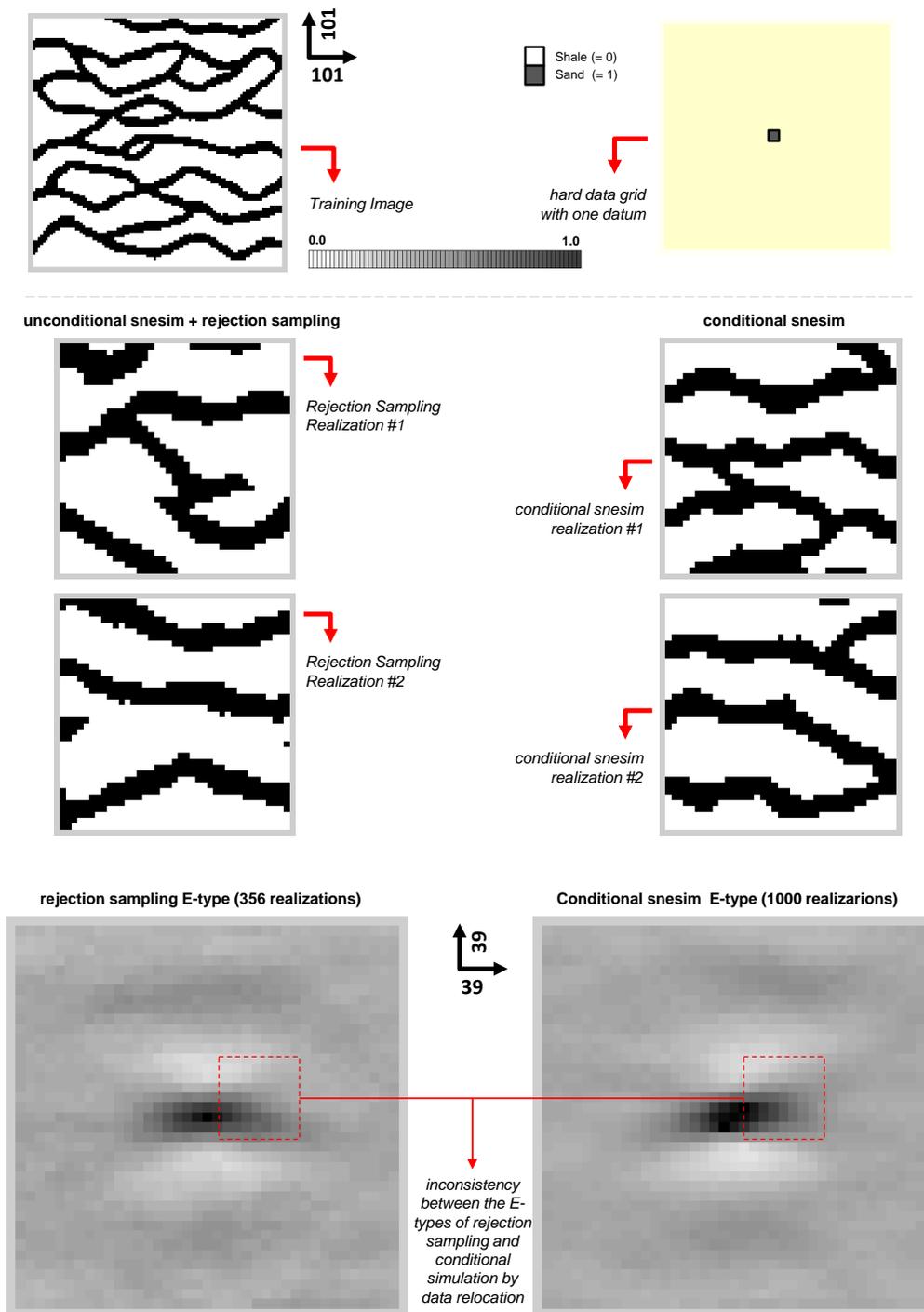


Figure 6.8: An illustration of the second limitation in current data relocation practice, where a clear spatial deformity is observed in the E-type obtained through conditional snesim simulation. Rejection sampling on left shows the correct E-type that should be obtained with the algorithm.

The first example is shown in Figure 6.9. The situation in this example is similar to the one introduced in the previous section (Figure 6.7). There are two well data in close proximity of each other, w_1 and w_2 . The previous method drops the well that is farther away from its closest node on the grid, i.e., drop w_2 from being assigned to node \mathbf{u}_1 . However, a better approach in multi-scale simulation is for that well data w_2 to be assigned to the next closest node, that is, to node \mathbf{u}_2 . This guarantees transfer of information even at the coarsest scale (i.e., \mathbf{r}_{e_g}), such that the finer scales can take advantage of previous coarser-scale simulations (i.e., $\mathbf{r}_{e_{g-1}}$). For instance, assume two neighboring hard well data, one indicating sand and the other shale, exist on the hard data grid \mathbf{hd} . From a visual perspective, one would expect the coarsest scale to be capable of providing models that have a perception of a sand and a shale next to each other. For example, in a channel training image, this corresponds to the edge defined between a channel and a shale structure, and as a consequence, the coarsest scale should be aware of such an edge.

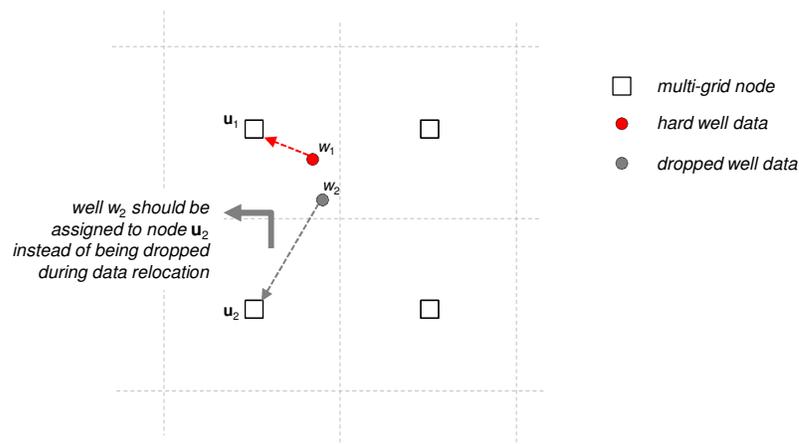


Figure 6.9: Example indicating that well w_2 is better to be assigned to \mathbf{u}_2 rather than being dropped. Previous implementations would drop w_2 since w_1 was closer to \mathbf{u}_1 .

The second example involves a situation where another well w_3 also exists close to node \mathbf{u}_2 . This is shown in Figure 6.10. The question is that which node should the well w_2 be assigned to. In this situation, the initially dropped well w_2 is expected to be assigned to the next closest node of \mathbf{u}_2 , however, the data relocation algorithm must have already assigned well w_3 to this node. Now, since w_3 is closer to \mathbf{u}_2 than w_2 , the data relocation algorithm needs to look for the next (third) closest node, that is, node \mathbf{u}_4 . The proposed relocation

will keep the overall configuration of points intact while minimizing the loss of hard well information.

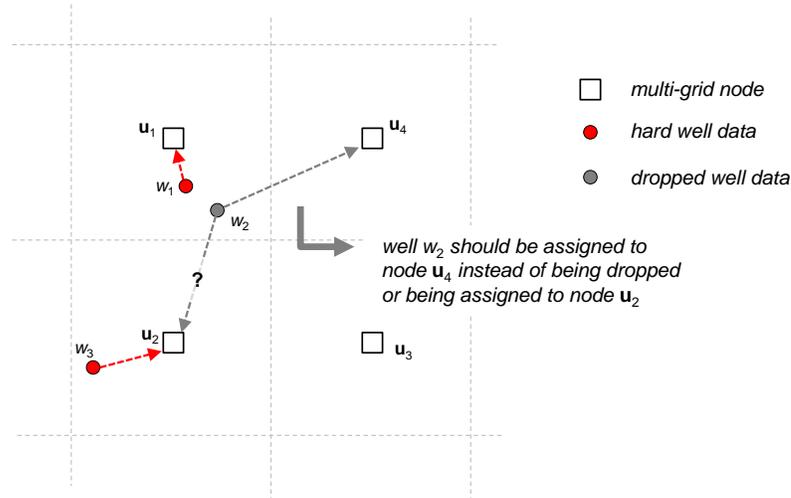


Figure 6.10: Example indicating that well w_2 should not anymore be copied on node u_2 , and instead be copied over u_4 due to a third well w_3 being a better candidate for relocation on node u_2 .

The third example is slightly more complicated than the previous ones, but is necessary for establishing correct data relocation rules. The example is shown in Figure 6.11. There are five wells with the specified configuration. Data relocation for three of them, w_1 , w_3 , and w_5 , are apparent, since they undoubtedly belong to their closest nodes of u_1 , u_2 , and u_5 respectively. These initial relocations lead to two dropped hard data, namely w_2 and w_4 . As seen in the previous example (Figure 6.10), well w_2 cannot be assigned to node u_2 , therefore the third best choice is now node u_4 . But even though no data has been assigned to node u_4 , well w_1 is actually closer to u_4 than well w_2 , and if such a relocation of w_2 to u_4 occurs, a distortion in the original configuration of wells will result. One can visualize such data relocation constraint by a geometrical rule, in the following way: wells can only be assigned towards nodes in a radial direction, where the origin of the radial direction corresponds to the center of mass of all hard data located between the four neighboring nodes. This geometrical rendition provides an abstract rule that prohibits any change in the overall configuration of hard data upon relocation.

Now that well w_1 is closer to node u_4 , we would assign w_2 to its fourth closest node, i.e., u_3 . However, there is another dropped well w_4 on the other hand. That well would

also follow the same rules and gets assigned to its second closest node of \mathbf{u}_3 . One can observe that w_4 is actually closer to \mathbf{u}_3 than well w_2 . Therefore, node \mathbf{u}_3 gets a copy of the value of well w_4 instead of w_2 , and well w_2 must be dropped for this multi-grid simulation level. Therefore, one can see that there are some situations where the hard data cannot be relocated to any grid node. It may either be because of the disruption on the hard data configuration, or the fact that none of the neighboring nodes are uninformed. As will be described later, the data relocation algorithm shall realize when to stop looking for the next closest node of a well, and entirely drop the hard well data.

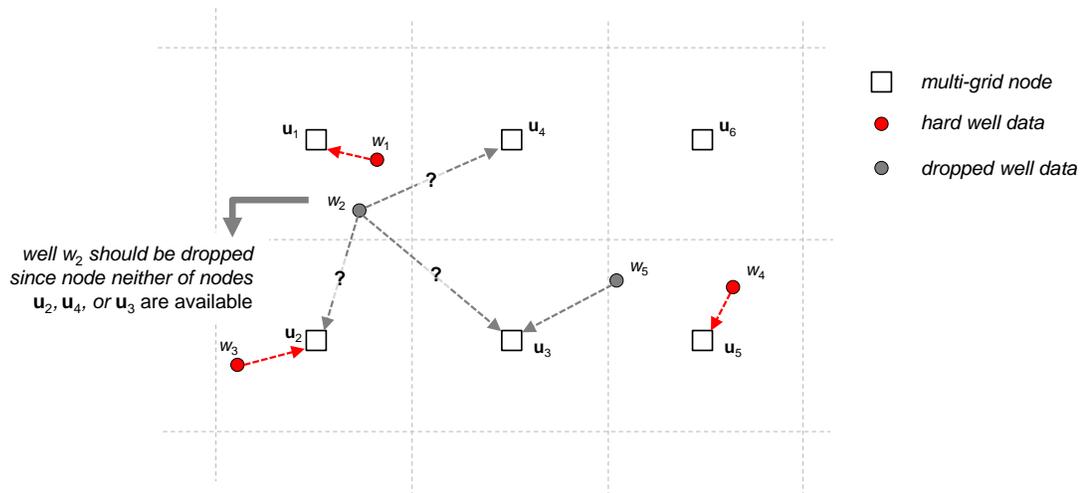


Figure 6.11: A complicated example configuration of wells and multi-grid nodes is shown. Different rules can be elaborated by such an example.

The provided examples lay out the foundations for a smart data relocation algorithm. Although, at first thought, it may require a complex set of rules, but can be simply formalized with the following three axioms:

1. Each data node is assigned the value of its closest hard well data
2. Each hard well data is assigned to its closest node
3. Each hard well data can only be assigned to the nodes within its enclosing volume

The first two axioms must be valid at the same time in order to ensure a correct data relocation process. In other words, by making sure that a two-way relationship exists between the well and the node, i.e., well w is the closest to node \mathbf{u} , and node \mathbf{u} is the closest to the well w , one will comply with all the rules set out in previous examples. However,

these axioms only apply to the dropped nodes. In the first stage of the proposed algorithm, each well is assigned to its closest node similar to the traditional approach. Some dropped wells may exist upon completion of this initial stage. In that situation, the second stage of the algorithm proceeds by re-assigning wells to grid nodes following the three axioms mentioned before. The desired relocation is achieved at the end of this stage.

Now that the theoretical foundations of the algorithm are presented, we describe the process itself. As mentioned before, there are two stages to data relocation. In the first stage, each of the n_{hd} wells are analyzed sequentially from $i = 1, \dots, n_{hd}$. Each well w_i will be assigned to its closest node \mathbf{u}^* . If node \mathbf{u}^* has already been informed by a previous well data, i.e., $\text{re}(\mathbf{u}^*) = \text{hd}(\mathbf{u}^{w_{i' < i}})$, then we will re-evaluate this assignment and relocate the hard data that is actually closer to the node \mathbf{u}^* , that is:

$$\text{re}(\mathbf{u}^*) = \begin{cases} \text{hd}(\mathbf{u}^{w_{i'}}), & \text{if } \|\mathbf{u}^{w_{i'}} - \mathbf{u}^*\| < \|\mathbf{u}^{w_i} - \mathbf{u}^*\|; \\ \text{hd}(\mathbf{u}^{w_i}), & \text{otherwise.} \end{cases} \quad (6.2)$$

If the above situation happens, one of the wells $w_{i'}$ or w_i will be dropped. We store all the dropped hard data in a vector denoted by \mathbf{dhd} , and continue with this process until all n_{hd} wells have been analyzed. Therefore, at the end of first stage, we have a vector \mathbf{dhd} of n_{dhd} dropped data. The second stage proceeds by analyzing all these dropped data. For each well $w_i^{\text{dhd}} \in \mathbf{dhd}$, all the neighboring nodes, which provide an enclosing volume to the well, are obtained, $\Omega_{w_i^{\text{dhd}}} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{2d}\}$ where $d = 2$ for 2D and $d = 3$ for 3D grids. Afterwards, the nodes in $\Omega_{w_i^{\text{dhd}}}$ are evaluated one by one from the one closest to w_i^{dhd} well to the farthest. With this ordering, for each node $\mathbf{u}^* \in \Omega_{w_i^{\text{dhd}}}$, we check all original n_{hd} wells to see whether well w_i^{dhd} is the closest. If there is such a match and the well is actually the closest one to node \mathbf{u}^* , we will assign it to that node; that is $\text{re}(\mathbf{u}^*) = \text{hd}(\mathbf{u}^{w_i^{\text{dhd}}})$. Otherwise, we will continue with the next node $\mathbf{u}^* \in \Omega_{w_i^{\text{dhd}}}$ until either a match occurs, or we have run through all the nodes in $\Omega_{w_i^{\text{dhd}}}$, and then that well w_i^{dhd} will be dropped entirely. This second stage continues in a similar fashion until all nodes in \mathbf{dhd} have been considered. This smart approach for hard data relocation will resolve all the issue shown by previous examples. An example application of this methodology will be shown later in this section (for the ‘Smart-Vibrating Multi-Grid’ approach). For simplicity, the process is described in Algorithm 8.

In order to test this new data relocation approach, we will condition the MPS simulations to a complex configuration of hard data, and compare the results with rejection sampling.

Algorithm 8 Smart Data Relocation Approach

Require: n_{hd} wells defined on grid \mathbf{hd} .

Require: realization grid \mathbf{re}_g , with multi-grid setting g .

```

1: for well  $w_i = w_1$  to  $w_{n_{hd}}$  do
2:   find the closest node  $\mathbf{u}^*$  to  $w_i$ 
3:   if  $\mathbf{u}^*$  is already assigned to well  $w_{i'}$  then
4:     if  $\|\mathbf{u}^{w_{i'}} - \mathbf{u}^*\| > \|\mathbf{u}^{w_i} - \mathbf{u}^*\|$  then
5:        $\mathbf{dhd} \leftarrow$  assign  $w_{i'}$  to dropped hard data vector
6:     else
7:        $\mathbf{dhd} \leftarrow$  assign  $w_i$  to dropped hard data vector
8:     end if
9:   else
10:    assign  $w_i$  to node  $\mathbf{u}^*$ 
11:   end if
12: end for
13: for well  $w_i^{\mathbf{dhd}} \in \mathbf{dhd}$  do
14:   obtain enclosing node set  $\Omega_{w_i^{\mathbf{dhd}}} \leftarrow \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{2d}\}$ 
15:   Sort  $\mathbf{u}_{\mathbf{dhd}}^* \in \Omega_{w_i^{\mathbf{dhd}}}$  in ascending order w.r.t. the distances  $d = \|\mathbf{u}_{w_i^{\mathbf{dhd}}} - \mathbf{u}_{\mathbf{dhd}}^*\|$ 
16:   for each  $\mathbf{u}_{\mathbf{dhd}}^i \in \Omega_{w_i^{\mathbf{dhd}}}$  do
17:     find closest well  $w^* \in \mathbf{hd}$  to node  $\mathbf{u}_{\mathbf{dhd}}^i$ 
18:     if  $w^* == w_i^{\mathbf{dhd}}$  then
19:       assign  $w_i^{\mathbf{dhd}}$  to node  $\mathbf{u}_{\mathbf{dhd}}^i$ 
20:     end if
21:   end for
22: end for

```

The same binary channel training image, shown in Figure 6.12, is considered. A set of seven hard data, within close spatial distance from each other, is specified. It should be noted that these hard data are not sampled from any specific reference model. Therefore, conditioning to such a hard data configuration is difficult because they may not be representative of the input spatial model. In other words, they may not conform to the training image patterns. The configuration of the seven hard data and their values are shown in Figure 6.12.

Rejection sampling provides the true conditioning results for our comparison. For rejection sampling in this experiment, 52000 unconditional realizations of size 39×39 with three multi-grid settings are generated. Then, only those realizations that match all the seven hard data are accepted. Since there is no conditioning algorithm in this sampling methodology, the E-type obtained from those realizations represent the true expected result. It should be mentioned that due to the complexity of such a data configuration, only 22 of the realizations could match all the hard data.

For the comparison of the conditioning capability of the proposed data relocation approach with the traditional approach, 150 conditional simulations using both the DisPAT method and the snesim method are generated. All realizations are now conditioned to these seven hard data. The E-types of both methods are obtained. A correct data conditioning capability of an MPS algorithm should result in an E-type much similar to the one obtained by rejection sampling. Figure 6.12 shows the E-type obtained from the correct rejection sampling method, and the ones from both the DisPAT and snesim methods. One can clearly observe that the new data relocation approach considerably improves the conditioning capability in the DisPAT method. On the other hand, the snesim E-type shows lesser conditioning capability, resulting from the traditional data relocation methodology. As mentioned before, the reason behind such a distinction in the E-type of the snesim method with rejection sampling lies behind the inaccuracy in data relocation and the excessive data dropping in higher multi-grid levels, and therefore, less information is revealed after data relocation. The poor conditioning in higher multi-grids prohibits a correct final conditioning in the finest multi-grid level. On the contrary, in the proposed data relocation approach, the higher-multi-grid levels are more informed about the expected fine-scale details for hard data conditioning.

The second limitation to be resolved concerns the data relocation itself. As shown through Figure 6.8, data conditioning may not statistically honor the multiple-point statistics of the training image. In other words, rejection sampling, as a perfect conditioning

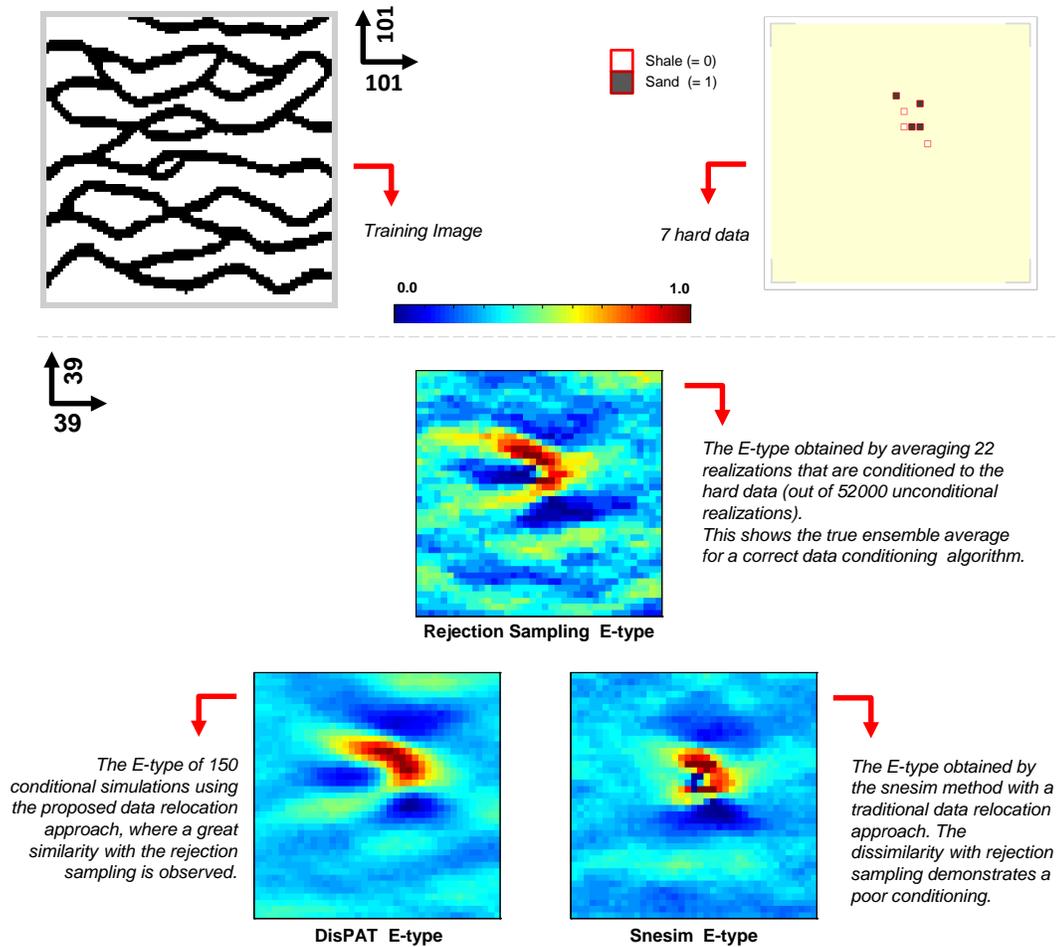


Figure 6.12: Investigation of the new data relocation approach. The E-type of rejection sampling method is shown in the middle as the truth. The two lower E-types from the DisPAT, and the snesim algorithms demonstrate the improved conditioning capability obtained by the proposed data relocation approach. Simulations are done in three multi-grid levels.

algorithm, will statistically converge to a different solution according to the E-type. Relocating hard data to specific nodes on the simulation grid for each Earth model may lead to irregularities. The situation arises due to the fixed locations of the active grid nodes on each simulation multi-grid \mathbf{re}_g . For example, the nodes on the random path always start from the origin (i.e., lower left corner of the grid in 2D), and include every 2^{g-1} -th nodes. Therefore, a hard data not originally on the realization grid of \mathbf{re}_g is always moved to a specific location $\mathbf{u}_{hd} \rightarrow \mathbf{u}^*$. The repetition of such a relocation in all Earth models will

induce false certainty towards the hard data value at location \mathbf{u}^* . Statistically speaking, if the node at location \mathbf{u}^* in all realizations is always assigned with the same hard data value, it would decrease the spatial variance in the neighborhood of \mathbf{u}^* due to less stochasticity. To solve that, stochasticity will be introduced in the proposed hard data relocation approach such that the randomness removes any undesirable artifacts in the E-type.

To resolve this problem one needs to disrupt the grid node locations in the multi-grid settings of each simulation. In other words, the grid nodes for realization \mathbf{re}_g should not necessarily start at the same origin. For each realization, a shift in the origin of the first active grid node of realization \mathbf{re}_g is made. With this approach, different realizations differ in the locations of multi-grid nodes, as if the coarse grids are vibrating on the finer ones.

Mathematically speaking, assume that the origin in a 3D realization grid is located at $(0, 0, 0)$, then each realization will have its origin at (s_x, s_y, s_z) , where s_x , s_y , and s_z represent the shifts in x , y , and z directions. The values of s_x , s_y , and s_z are randomly selected from the range $[0, 2^{n_g-1})$. This can be computed by the formula below:

$$\text{origin} = (s_x, s_y, s_z), \quad \text{where} \quad \begin{cases} s_x = [2^{n_g-1} \times U[0, 1)] \\ s_y = [2^{n_g-1} \times U[0, 1)] \\ s_z = [2^{n_g-1} \times U[0, 1)] \end{cases} \quad (6.3)$$

where $U[0, 1)$ represents a random number from the uniform distribution in the range $[0, 1)$, and $[\cdot]$ represent the floor function. A 2D illustration of the vibrating process of shifting multi-grids is shown in Figure 6.13.

Finally, the smart-vibrating approach introduced in this section will not only provide a better data relocation algorithm, but it will also reduce any artifacts caused by the idea of relocation. In order to compare the proposed technique with previous approaches, we have conducted the same experiment shown in Figure 6.8. In that experiment, we performed many unconditional simulations, and obtained an ensemble average of models that have been correctly conditioned to the hard data with rejection sampling. On the other hand, to compare the conditioning capability of the proposed method, we simulated some conditional realizations as well. The results are shown in Figure 6.14. As can be observed, the E-types of both approaches have similar structures, and there is no deformity or distortion evident in the E-type of the conditional DisPAT. This demonstrates the improved conditioning algorithm and an internally consistent algorithm in DisPAT.

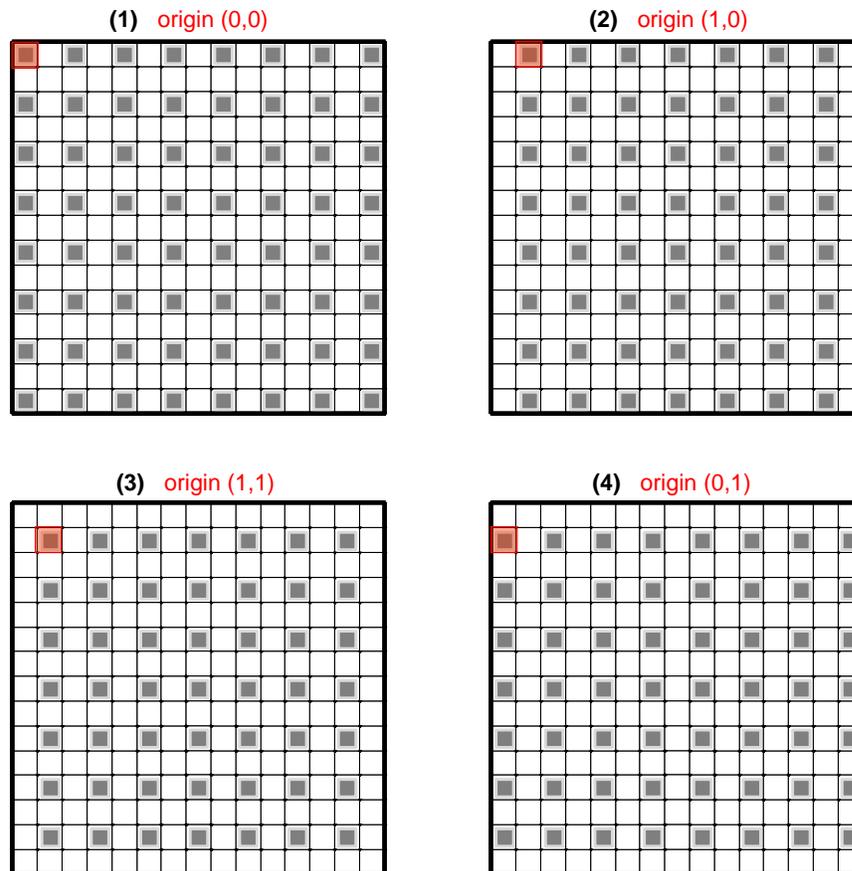


Figure 6.13: The vibrating multi-grid shifting approach is illustrated for $n_g = 2$ multi-grid setting. There are four possible shifts in the origin, thus introducing stochasticity within hard data relocation.

6.1.4 Multi-Resolution Approach

Method

The multi-resolution approach was introduced in Section 4.2 for multi-scale simulation much like the multi-grid approach. The difference lied in the way it handled scale using the scale-space theory of the human visual system. Instead of having similar grid sizes for all multi-grid levels, the grids in multi-resolution approach are resized according to each scale. It was demonstrated that upon generating realizations at different coarse to fine scales, one can obtain a better long-range pattern continuity and multiple-point reproduction of the training image features.

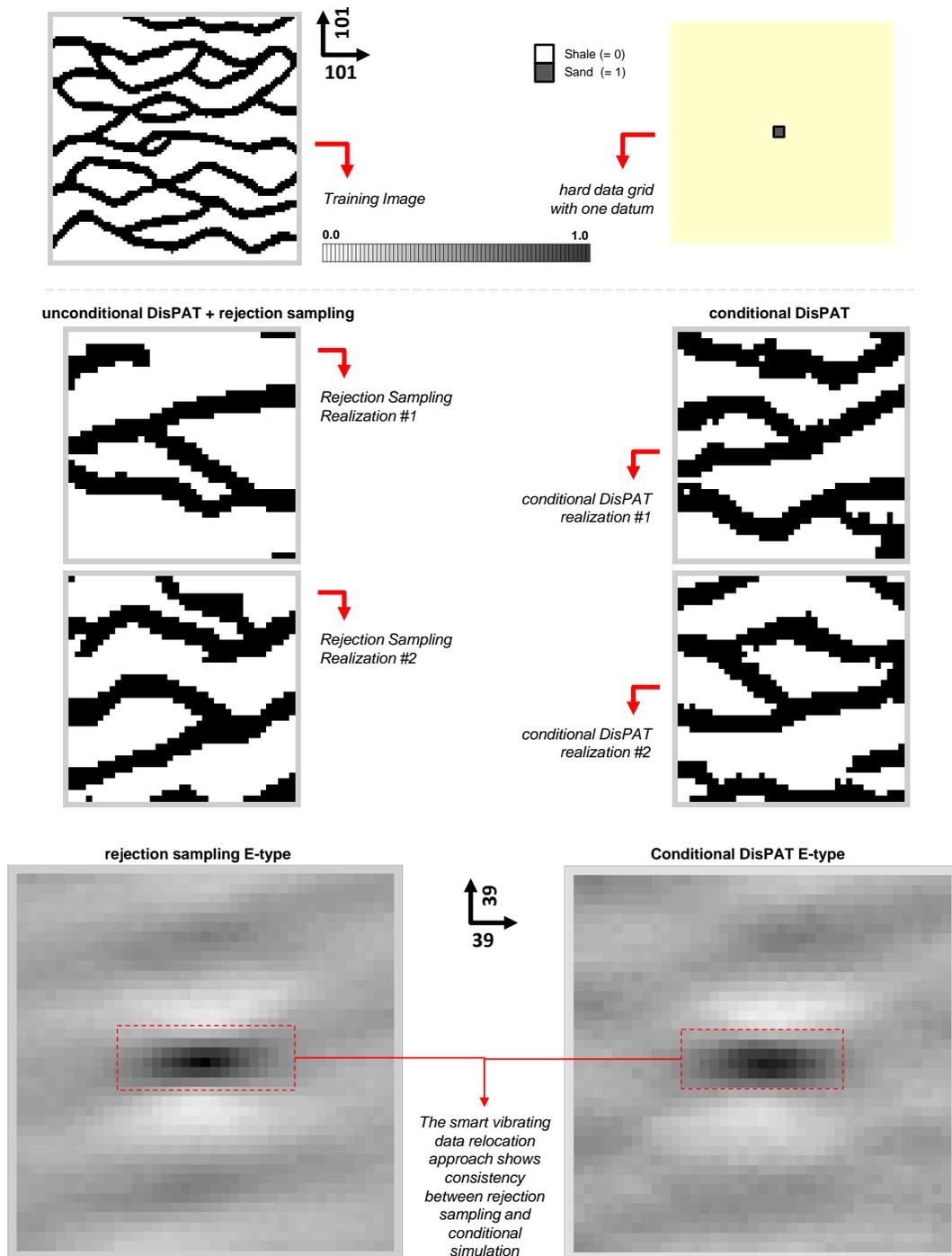


Figure 6.14: An illustration of the smart-vibrating approach, where no spatial differences is observed in the E-type obtained through conditional DisPAT simulation and rejection sampling.

However, data conditioning would not be as trivial. In the multi-grid setting, as mentioned in previous section, all the nodes on each grid \mathbf{re}_g have the same spatial location within all grids \mathbf{re}_g , for $g = 0, \dots, n_g - 1$. On the other hand, in a multi-resolution approach, the nodes are different in terms of both their sizes, and their spatial locations. Progressing from one scale \mathbf{re}_r to the next scale \mathbf{re}_{r-1} is not a one-to-one relationship. Each scale has to be resized with cubic interpolation into the next finer scale during the simulation. This non-linearity seems to make data conditioning intractable.

However, we can still relocate data to their nearest neighbor grid in the multi-resolution approach, much similar to the multi-grid method. According to the scale-space theory, in a multi-scale setting, one will perceive a blurred (or a big picture) depiction of the hard data grid \mathbf{hd} . This means that hard data are no more relocated individually to their corresponding nodes in multi-resolution realization \mathbf{re}_r . But instead, they are interpolated to their closest nodes. In other words, each grid node $\mathbf{u} \in \mathbf{G}_{re}^r$, will be assigned a value obtained by spatially interpolating all hard data in its proximity.

There are two simple interpolation techniques: kriging and inverse-distance weighting. The selection of the interpolation technique will impact the resulting hard data conditioning. In kriging the interpolated value depends on the values at neighboring locations plus knowledge about the underlying spatial configuration of data. It resolves the redundancy that exists between two close data. On the other hand, in the inverse-distance weighting approach, the interpolated value is based solely on the neighboring data and their distance from the interpolated location. In general, kriging is a more appropriate approach in spatial simulation, but for hard data relocation, we choose the simpler inverse-distance weighting scheme due to its computational advantage. In addition, since hard data conditioning will be improved/refined in successive scales, the small errors caused by ignoring the spatial configuration of data is resolved in the final Earth model. Despite this tendency for self-correction (the successive reduction in the interpolation error with finer multi-resolution levels), the configuration of hard data is not lost. This is due to the large spatial area covered for each inverse-distance weighting, and as a result, the configuration of the hard data can be relatively captured by the configuration of the realization grid nodes themselves. This is similar to the blurring effect on an image. If a human face is blurred by a smoothing filter, we are still able to recognize the face even though the spatial configuration of points was not taken into account.

The process of inverse-distance weighting for hard data relocation proceeds by visiting

all the nodes on simulation grid \mathbf{re}_r that are within ‘one coarse grid-cell’ proximity of a hard data. This is shown in Figure 6.15. For each node \mathbf{u}^* , it will store all the hard data in its proximity, i.e., $\mathcal{W} : \{w_1, \dots, w_n\}$, where n denoted the number of hard data in its surroundings. Afterwards, it will calculate the inverse-distance weighted average of all the hard data values in \mathcal{W} . The interpolated value can be computed with the formula below:

$$\mathbf{re}_r(\mathbf{u}^*) = \frac{\sum_{i=1}^n \omega_i(\mathbf{u}_{w_i}) \text{hd}(\mathbf{u}_{w_i})}{\sum_{i=1}^n \omega_i(\mathbf{u}_{w_i})} \quad (6.4)$$

where,

$$\omega_i(\mathbf{u}_{w_i}) = \frac{1}{\|\mathbf{u}^* - \mathbf{u}_{w_i}\|} \quad (6.5)$$

This process continues until all nodes are analyzed.

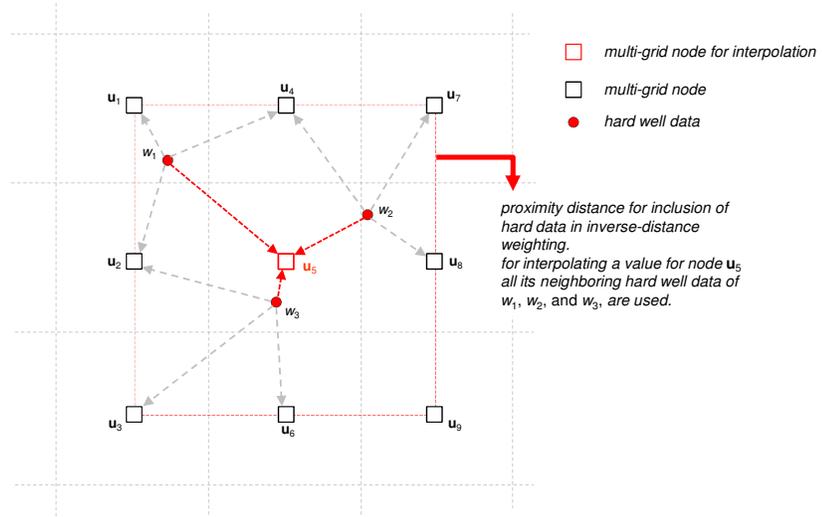


Figure 6.15: Data relocation by inverse-distance weighting approach for multi-resolution simulation. For node \mathbf{u}_5 , all the wells within the distance less than the size of a coarse grid-cell are included in analysis, that is wells w_1 , w_2 , and w_3 . The red lines shows the distances used for interpolation of node \mathbf{u}_5 . The gray lines for each well w_i show all nodes that w_i will be used for their interpolation.

The resulting realization \mathbf{re}_r is now partially informed with the interpolated hard data values. The described data relocation can be interpreted as a soft data conditioning algorithm, since hard data are not exactly copied on the coarse grid nodes, but interpolated. However, the resulting data events extracted from realization \mathbf{re}_r are more similar to the prototypes during the similarity search, and thus, more accurate pattern modeling will emerge.

The process is summarized in Algorithm 9. In next section, we will provide some examples and demonstrate the data conditioning capability of the multi-resolution technique.

Algorithm 9 Multi-Resolution Hard Data Conditioning

Require: n_{hd} wells defined on grid **hd**.

Require: realization grid \mathbf{re}_r , with multi-resolution scale r .

- 1: **for** each node $\mathbf{u} \in \mathbb{G}_{re}^r$ **do**
 - 2: find all wells within one coarse grid-cell distance from \mathbf{u} ; $\mathcal{W} \leftarrow \{w_1, \dots, w_n\}$
 - 3: **if** number of wells ($n > 0$) **then**
 - 4: find the distances between each well $w_i \in \mathcal{W}$ and node \mathbf{u} ; $d_i = \|\mathbf{u} - \mathbf{u}_{w_i}\|$
 - 5: interpolate the value of node \mathbf{u} using inverse-distance weighting of wells in \mathcal{W} .
 - 6: assign interpolated value to node \mathbf{u}
 - 7: **end if**
 - 8: **end for**
-

Examples

The application of hard data conditioning within a multi-resolution approach is investigated in this section. In order to demonstrate the capabilities of the multi-resolution method in data conditioning, the results are compared with the multi-grid approach.

It was demonstrated that pattern reproduction is improved in a multi-resolution setting for unconditional simulations. For conditional simulations we adapted the inverse-distance weighting of hard data for coarse resolution grids. In the first example, we investigate the effectiveness of the proposed approach on the same example shown in Figure 6.3. There are 100 hard data on a 101×101 realization grid. Figure 6.16 displays the application of data conditioning in both multi-resolution and multi-grid approaches. Evidently, multiple-point statistics and hard data are better honored within the new multi-resolution technique. One can clearly observe the training image structures and features that are revealed through hard data. The reason for such improvements is twofold. First, the multi-resolution approach can better capture statistics of the training image. And second, the inverse-distance weighting method for data relocation provides a more accurate representation of hard data configuration in coarse resolutions.

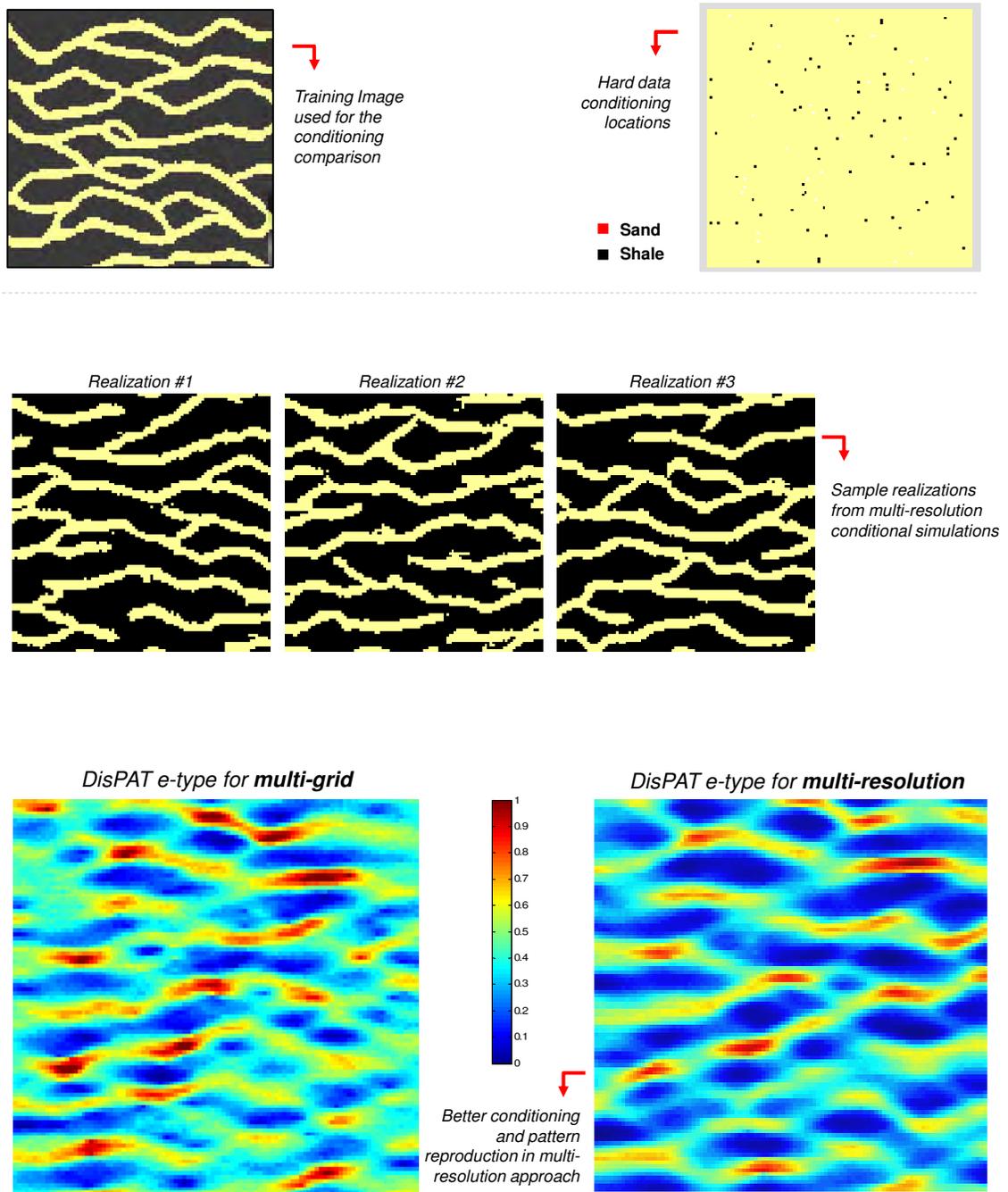


Figure 6.16: Example application of data conditioning in multi-resolution simulation using inverse-distance weighting. There are 100 hard well data of sand/shale in the 2D grid size of 101×101 . Both multi-grid (bottom-left) and multi-resolution (bottom-right) E-types of DisPAT algorithm are shown. Better conditioning is obtained in multi-resolution approach.

6.2 Soft Data Conditioning

6.2.1 Review of Methods

In geostatistics, seismic data are typically viewed as a soft, indirect or secondary information (Caers and Ma, 2006). Soft data information, contrary to hard well data, are not sparse and provide more insight into lateral variations of attributes. For instance, seismic data provide a rich source of low resolution information covering the entire field; such as depositional facies.

In most multiple-point geostatistical methods, a soft probability cube is used as the input for soft data. For example, in a binary training image consisting of sand and shale, the probability cube is simply the sand probability derived from seismic data. One method for integrating all additional sources of information, such as hard and soft data, is the probabilistic approach. Consider computing the probability of occurrence of an attribute, $P(\mathbf{A})$, where \mathbf{A} represents the variable under study. Furthermore, assume that there are two sources of additional information available, \mathbf{B} and \mathbf{C} , which denote hard and soft data respectively. $P(\mathbf{A})$ is the prior probability of occurrence of variable \mathbf{A} without any other information. But with the existence of \mathbf{B} and \mathbf{C} , the posterior probability of \mathbf{A} can be obtained by integrating the two additional sources of information according to the Bayesian rule, as follows:

$$P(\mathbf{A} \mid \mathbf{B}, \mathbf{C}) = \frac{P(\mathbf{A}, \mathbf{B}, \mathbf{C})}{P(\mathbf{B}, \mathbf{C})} \quad (6.6)$$

where $P(\mathbf{A}, \mathbf{B}, \mathbf{C})$ can be calculated as follows:

$$\begin{aligned} P(\mathbf{A}, \mathbf{B}, \mathbf{C}) &= P(\mathbf{A}) \cdot P(\mathbf{B} \mid \mathbf{A}) \cdot P(\mathbf{C} \mid \mathbf{A}, \mathbf{B}) \\ &= P(\mathbf{A}) \cdot P(\mathbf{C} \mid \mathbf{A}) \cdot P(\mathbf{B} \mid \mathbf{A}, \mathbf{C}) \end{aligned} \quad (6.7)$$

The main challenge for computing the posterior probability of \mathbf{A} is to calculate joint probability $P(\mathbf{B}, \mathbf{C})$, and conditional probability of $P(\mathbf{C} \mid \mathbf{A}, \mathbf{B})$ or $P(\mathbf{B} \mid \mathbf{A}, \mathbf{C})$. In real field situations, these probabilities are hard to obtain due to few available hard data or large uncertainties on the soft probabilities. The simplest assumption that allows a simplified calculation is the one of the independence between the two additional sources of information, \mathbf{B} and \mathbf{C} . According to the independence assumption:

$$P(\mathbf{B}, \mathbf{C}) = P(\mathbf{B}) \cdot P(\mathbf{C}) \quad (6.8)$$

However, this is a strong assumption that does not hold true in reality. Moreover, the posterior probability may go out of the permissible range of $[0, 1]$. These two limitations motivated a less constraining approach of “permanence of ratios” by Journel (2002) for probabilistic data integration. Data are only assumed to be conditionally independent of variable \mathbf{A} . There is no more an assumption of independence between \mathbf{B} and \mathbf{C} . This conditional independence assumption leads to the following simplification in calculating the posterior probability of \mathbf{A} given \mathbf{B} and \mathbf{C} .

$$P(\mathbf{A} | \mathbf{B}, \mathbf{C}) = \frac{P(\mathbf{A}) \cdot P(\mathbf{B} | \mathbf{A}) \cdot P(\mathbf{C} | \mathbf{A})}{P(\mathbf{B}, \mathbf{C})} \quad (6.9)$$

It can be noted that the joint probability of \mathbf{B} and \mathbf{C} still exists. In order to eliminate this problem, the posterior probability of the complement of \mathbf{A} is used. Complement of \mathbf{A} , denoted by $\bar{\mathbf{A}}$ represents the event where \mathbf{A} does not occur. In other words, $P(\bar{\mathbf{A}}) = 1 - P(\mathbf{A})$. Writing the posterior probability with respect to $\bar{\mathbf{A}}$ is shown as follows:

$$P(\bar{\mathbf{A}} | \mathbf{B}, \mathbf{C}) = \frac{P(\bar{\mathbf{A}}) \cdot P(\mathbf{B} | \bar{\mathbf{A}}) \cdot P(\mathbf{C} | \bar{\mathbf{A}})}{P(\mathbf{B}, \mathbf{C})} \quad (6.10)$$

Therefore, upon dividing Equation 6.10 by 6.9, a new relationship is obtained as follows:

$$\frac{x}{b} = \frac{c}{a} \quad (6.11)$$

where,

$$\begin{aligned} x &= \frac{P(\bar{\mathbf{A}} | \mathbf{B}, \mathbf{C})}{P(\mathbf{A} | \mathbf{B}, \mathbf{C})} \\ a &= \frac{P(\bar{\mathbf{A}})}{P(\mathbf{A})} \\ b &= \frac{P(\bar{\mathbf{A}} | \mathbf{B})}{P(\mathbf{A} | \mathbf{B})} \\ c &= \frac{P(\bar{\mathbf{A}} | \mathbf{C})}{P(\mathbf{A} | \mathbf{C})} \end{aligned} \quad (6.12)$$

The resulting equation states that the incremental information provided through \mathbf{C} is the same irrespective of knowledge of \mathbf{B} , and vice versa. In other words, the additional information of the seismic data for estimating the probability of facies \mathbf{A} is independent

from the information obtained through hard data. As a result, the “permanence of ratios” assumption leads to the following calculation of the posterior probability of \mathbf{A} given \mathbf{B} and \mathbf{C} :

$$P(\mathbf{A} \mid \mathbf{B}, \mathbf{C}) = \frac{a}{a + bc} \in [0, 1] \quad (6.13)$$

However, in geology, there are rarely satisfying justifications for such a conditional independency between the data. In order to tune down this strong assumption of conditional independency Journel (2002) introduced the Tau model with an addition of a new parameter τ , as follows:

$$\frac{x}{b} = \left(\frac{c}{a}\right)^\tau \quad (6.14)$$

where τ determines the redundancy between well and seismic data. The Tau model has been widely used and extensively studied in the past (Lopez et al., 2004; Krishnan et al., 2005; Caers et al., 2006; Castro et al., 2006; Tang et al., 2007; Krishnan, 2008). One of the advantageous was the non-convexity of the solution, that is, the posterior probability $P(\mathbf{A} \mid \mathbf{B}, \mathbf{C})$ can go beyond the intervals posed by the individual probabilities of $P(\mathbf{A} \mid \mathbf{B})$ and $P(\mathbf{A} \mid \mathbf{C})$. Some variant of the Tau model has also been proposed. For example Polyakova and Journel (2007) introduced the Nu model for probabilistic integration of multiple sources of information.

Snesim multiple-point geostatistical algorithm adapted the Tau model formulation for integrating hard and soft data with the training image to probabilistically generate Earth model. However, in pattern-based statistics, the probabilistic assumptions cannot be directly used during simulation.

Simpat, as the first pattern-based method, integrated soft data by introducing a soft training image \mathbf{sti} besides the original training image \mathbf{ti} (Arpat, 2005). The two training images now are used to establish a relation between the hard geological data and soft variables. For example, a simple averaging filter applied on the training image can provide the desired lower resolution soft training image. In simpat, preprocessing the training image involves storing both the patterns of the training image \mathbf{pat}_T^k and the patterns of the soft training image \mathbf{spat}_T^k in the pattern database \mathbf{patdb}_T . The simulation then proceeds by obtaining not just the data event $\mathbf{dev}_T(\mathbf{u})$, but also the soft data event $\mathbf{sdev}_T(\mathbf{u})$. The similarity search would jointly incorporate both data events with their corresponding

patterns in the database. In other words, it will try to maximize the following function for obtaining the most similar pattern \mathbf{pat}_T^* :

$$s = s \langle \mathbf{dev}_T(\mathbf{u}), \mathbf{pat}_T^k \rangle + \omega_{sd}(\mathbf{u}) \times s \langle \mathbf{sdev}_T(\mathbf{u}), \mathbf{spat}_T^k \rangle \quad (6.15)$$

where ω_{sd} is a location specific weight factor that reflects the confidence on soft data.

The approach in *simpat* could not be applied to the more practical filter-based MPS technique of *filtersim*. The reason is that in *filtersim* no direct access to the pattern database is allowed. Instead, the simulation can only interact with cluster prototypes. Therefore, Zhang (2006) and Wu (2007) introduced a new way of soft data integration. Two different methodologies were proposed, one for continuous, and one for categorical variables.

For continuous variables, the approach was simple. During the simulation, both the data event $\mathbf{dev}_T(\mathbf{u})$ and the soft data event $\mathbf{sdev}_T(\mathbf{u})$ are retrieved. Then, all the uninformed nodes in the data event $\mathbf{dev}_T(\mathbf{u} + \mathbf{h}_\alpha)$ are replaced by their corresponding nodes on soft data event $\mathbf{sdev}_T(\mathbf{u} + \mathbf{h}_\alpha)$. The similarity search between the data event and the cluster prototypes are performed similar to the unconditional case, and the closest prototype is found, \mathbf{prot}_T^* . So the data event will have some soft information carried into it. For example, if the data event is fully uninformed, it will be simply replaced by the soft data event for the similarity search.

However, for categorical variables, the methodology combines both the pattern-based approach of *simpat* with the probabilistic Tau model of *snesim*. It combines the two sources of information to obtain a new data event $\mathbf{dev}_T^*(\mathbf{u})$. For the fully uninformed data event $\mathbf{dev}_T(\mathbf{u})$, the process is similar to the continuous variable where the data event is simply replaced by the soft data event, $\mathbf{dev}_T^*(\mathbf{u}) = \mathbf{sdev}_T(\mathbf{u})$. However, for the partially informed data event, it will first look for the most similar prototype \mathbf{prot}_T^* to the data event $\mathbf{dev}_T(\mathbf{u})$. Then, the Tau model is used to combine soft data event $\mathbf{sdev}_T(\mathbf{u})$ with the prototype \mathbf{prot}_T^* to obtain a new data event $\mathbf{dev}_T^*(\mathbf{u})$. Finally, another search is performed to find the closest prototype to this newly created data event $\mathbf{dev}_T^*(\mathbf{u})$.

Although the *filtersim* approach is applicable in our proposed methodology, we propose another simpler method. The methodology for fusing two different sources of information is described next.

6.2.2 Soft Distance-based Fusion

DisPAT is based on the concept of distances for pattern modeling. Integration of soft data with both hard data and the training image follows the same concept. The soft data event being a probability cube will be directly used during the data event similarity search with cluster prototypes.

The hard data integration aspect will stay the same. In other words, using different weights for the nodes of the data event, the distance between each prototype \mathbf{prot}_T^k to the data event $\mathbf{dev}_T(\mathbf{u})$ is obtained. We denote these distances by the vector \mathbf{d}_{hd} , as follows:

$$\mathbf{d}_{hd} = \{d_{hd}^1, d_{hd}^2, \dots, d_{hd}^k\} \quad (6.16)$$

where k represents the number of clusters, and d is the distance between a data event and each prototype,

$$d_{hd}^i = d \langle \mathbf{dev}_T(\mathbf{u}), \mathbf{prot}_T^i \rangle \quad (6.17)$$

This is the same procedure used for integrating hard data. Now in order to include soft information in the similarity search, we construct another distance vector for soft data. The soft data event $\mathbf{sdev}_T(\mathbf{u})$ is used in the similarity search with the prototypes. Another vector denoted by \mathbf{d}_{sd} is obtained using the soft data event as follows:

$$\mathbf{d}_{sd} = \{d_{sd}^1, d_{sd}^2, \dots, d_{sd}^k\} \quad (6.18)$$

where the distances are:

$$d_{sd}^i = d \langle \mathbf{sdev}_T(\mathbf{u}), \mathbf{prot}_T^i \rangle \quad (6.19)$$

The distances between the soft data events and the cluster prototypes can provide satisfactory classifications. The combination of variables in the similarity search is justified because the soft data, seen as a filtered version of model variables, have similar characteristics to cluster prototypes. For instance, similar to the soft training image in the simpat algorithm, soft data can be interpreted as a blurred version of the random variables. This is comparable to what the cluster prototypes represent: a blurred portrayal of patterns within a cluster.

Having found both distances, \mathbf{d}_{hd} between the data event and prototypes, and \mathbf{d}_{sd}

between the soft data event and prototypes, one can linearly fuse them together to obtain a new distance vector. More specifically, the final distance used for finding the most similar cluster prototype \mathbf{prot}_T^* is computed according to the formula below:

$$\mathbf{d} = \mathbf{d}_{hd} + f(\mathbf{dev}_T(\mathbf{u})) \times \omega_{sd} \times \mathbf{d}_{sd} \quad (6.20)$$

where $f(\mathbf{dev}_T(\mathbf{u}))$ provides the percentage of informed nodes in data event $\mathbf{dev}_T(\mathbf{u})$, and ω_{sd} represent the confidence on the soft information. For example, $\omega_{sd} = 0$ amounts to ignoring the soft data; and vice versa, a large value for ω_{sd} increases our confidence on the soft information.

It should be noted that, the distance vectors of \mathbf{d}_{hd} and \mathbf{d}_{sd} are to be normalized to the range $[0, 1]$ beforehand. Even though a binary training image \mathbf{ti} , and the soft data \mathbf{sd} have similar ranges of zero to one, the distances must still be normalized. The reason is due to the differences in the range of the distances themselves. The soft data and the prototypes are both continuous variables in $[0, 1]$ range. But in a binary training image, the data events are binary, and therefore, they are at the extreme ends of the distribution; namely, either zero or one. As a result, the range of distances of the data events to the prototypes will be different than the ones from the soft data events.

In addition to normalization of both distance vectors, there is a factor $f(\mathbf{dev}_T(\mathbf{u}))$ that is being multiplied with the soft distance vector, \mathbf{d}_{sd} . The soft data event is always fully informed through the soft data grid. However, simulation proceeds sequentially over the realization grid \mathbf{re} , and as a result, data event extracted from the realization may have uninformed nodes. The distances obtained to the prototypes when all the nodes of the data event are full are different in magnitude than the ones that are partially informed. However, we have already normalized these distance vector to $[0, 1]$ range. Not accounting for these magnitude differences (resulted from the number of informed nodes of the data event) would cause an algorithmic bias towards soft data. For example, assuming that data event has only one informed node, vector \mathbf{d}_{sd} will dominate over the vector \mathbf{d}_{hd} (with smaller values) during the fusion process of Equation 6.20. This factor f thus re-adjusts the previous compensation introduced by normalizing the distance vectors. Without this factor, the distance vector \mathbf{d}_{sd} would have the same range of $[0, 1]$ regardless of the number of informed nodes of the data event $\mathbf{dev}_T(\mathbf{u})$. Thus, factor f compensates for the differences between the number of informed nodes in the data event and the fully informed soft data

events. The factor is computed as follows:

$$f(\mathbf{dev}_T(\mathbf{u})) = \frac{\text{number of informed nodes in } \mathbf{dev}_T(\mathbf{u})}{\text{total number of nodes in } \mathbf{dev}_T(\mathbf{u})} \quad (6.21)$$

According to the proposed distance-fusion technique, a cluster prototype \mathbf{prot}_T^* is selected according to the distance vector \mathbf{d} (shown in Equations 6.22 and 6.23). Similar to the unconditional case, a pattern will be randomly selected from the cluster and pasted on the simulation grid.

$$\mathbf{prot}_T^* = \mathbf{prot}_T^{i^*} \quad (6.22)$$

where,

$$i^* = \arg \min_i \{d^i\} = \arg \min_i \{d_{hd}^i + f(\mathbf{dev}_T(\mathbf{u})) \times \omega_{sd} \times d_{sd}^i\} \quad (6.23)$$

The proposed soft data integration approach works in both multi-grid and multi-resolution simulations. In the multi-grid case the soft data events can be extracted from the soft data in a similar fashion as the extraction of data events from a realization. The situation is different for the multi-resolution approach since the scale of the grids differs at each resolution. However, the solution to this inconsistency between the scale of soft data and the coarse grid is to apply the same scale transformation to the soft grid \mathbf{sd} as well. For resolution r with realization \mathbf{re}_r that uses the resized training image \mathbf{ti}_r , one requires resizing the soft data \mathbf{sd} to the corresponding scale \mathbf{sd}_r . In other words, in the multi-resolution approach, all sources of information should be somehow transformed into their respective scales, i.e., hard data by using the inverse-distance weighting interpolation and soft data by resizing using cubic interpolation.

Finally, the general distance-fusion algorithm for soft data integration is outlined in Algorithm 10. This algorithm describes the method for either the multi-grid or multi-resolution approach. The flowchart of the algorithm is also illustrated in Figure 6.17.

6.2.3 Note on Distance-Fusion

The aim of data integration in the distance-based pattern modeling framework is to choose the appropriate class of patterns. Without any additional information, one simply picks the cluster that is closest in distance to the current data event. Additional sources of

Algorithm 10 Distance-Fusion Soft Data Integration

Require: cluster prototypes \mathbf{prot}_T^i , for $i = 1, \dots, k$ **Require:** data event $\mathbf{dev}_T(\mathbf{u})$ from realization \mathbf{re} **Require:** soft data event $\mathbf{sdev}_T(\mathbf{u})$ from realization \mathbf{sd} **Require:** ω_{sd} as a parameter describing the confidence in soft data

```

1:  $ni \leftarrow$  number of informed nodes in data event  $\mathbf{dev}_T(\mathbf{u})$ 
2: for  $i = 1$  to  $k$  do
3:   if  $ni \neq 0$  then
4:      $d_{hd}^i \leftarrow d\langle \mathbf{dev}_T(\mathbf{u}), \mathbf{prot}_T^i \rangle$ 
5:   else
6:      $d_{hd}^i \leftarrow 0$ 
7:   end if
8:    $d_{sd}^i \leftarrow d\langle \mathbf{sdev}_T(\mathbf{u}), \mathbf{prot}_T^i \rangle$ 
9: end for
10:  $f \leftarrow \frac{ni}{n_T}$ 
11:  $\mathbf{d} = \mathbf{d}_{hd} + f \times \omega_{sd} \times \mathbf{d}_{sd}$ 
12:  $i^* \leftarrow \arg \min_i \{d^i\}$ 
13: return  $\mathbf{prot}_T^{i^*}$  as the most similar prototype for pattern selection

```

information, such as hard data and soft data, will only alter this decision on the closest cluster. The process can be perceived as a naive Bayesian classifier, where different sources of information are fused together, and a class label is assigned to them (Anderson, 1974; Zhang, 2004). Some rules are thus required for the Bayesian classifier to assign a class label to the data event according to available information. A Bayesian framework can provide the probabilistic rules to determine the class probability given the data events. Assignment is then simply the task of picking the class with the highest probability. Figure 6.18 depicts an example of a naive Bayesian network.

As can be seen in Figure 6.18, the posterior probability of event \mathbf{A} is conditionally dependent upon the additional information, \mathbf{B} , \mathbf{C} , and \mathbf{D} . In our method for integrating data events with soft data events, the independence assumption is revealed by assuming a linear dependency between the data. The method is similar to the simple ‘weighted linear averaging’ of prior probabilities for calculating the posterior probability of \mathbf{A} given \mathbf{B} and \mathbf{C} . This is shown in formula below:

$$P(\mathbf{A} \mid \mathbf{B}, \mathbf{C}) = \omega_1 \times P(\mathbf{A} \mid \mathbf{B}) + \omega_2 \times P(\mathbf{A} \mid \mathbf{C}) \quad (6.24)$$

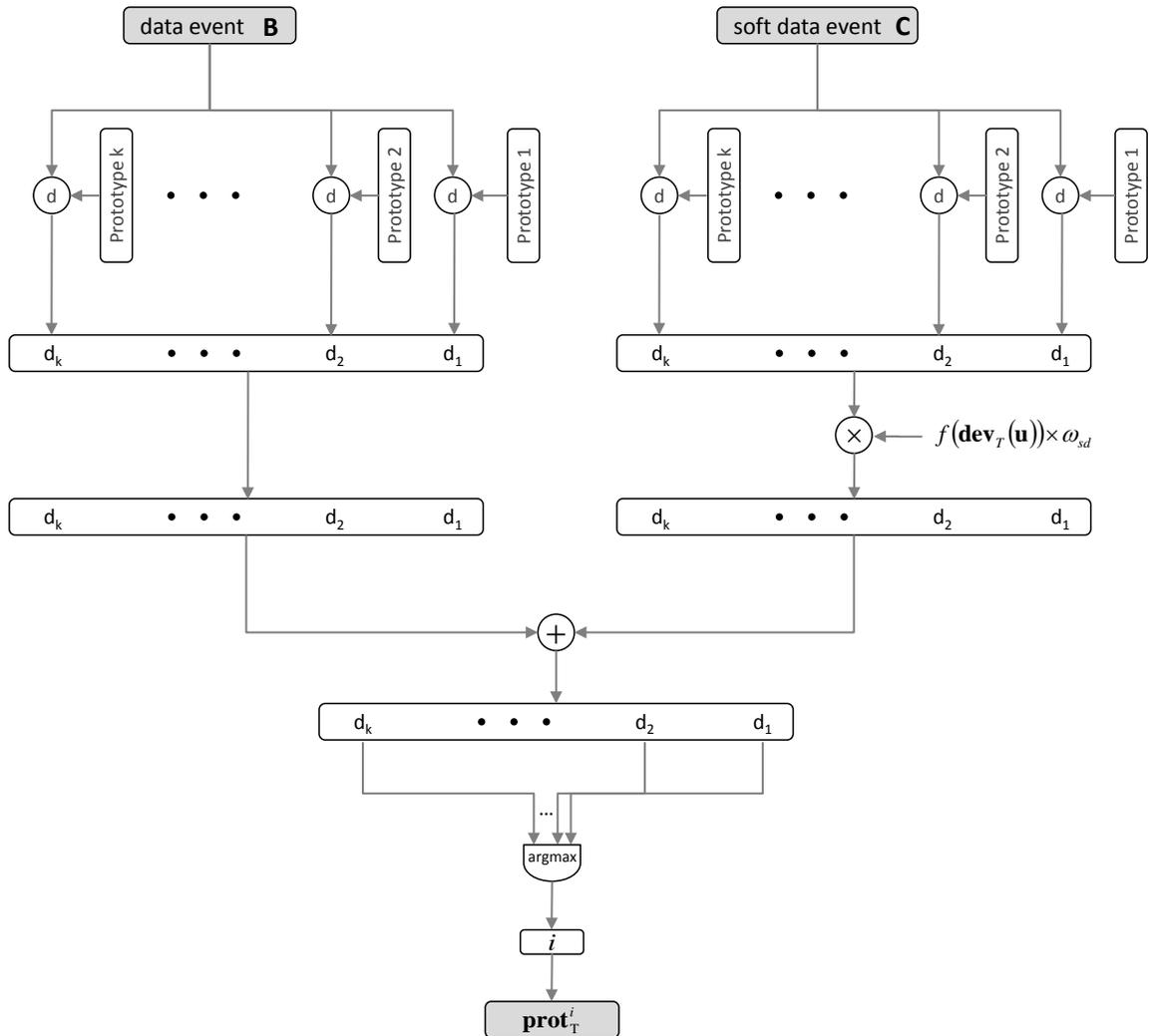


Figure 6.17: Flowchart of the distance-fusion technique for integration of soft information. Data event represents the training image and the hard data. Soft data event represents the soft data only. The result is a analogous to the naive Bayesian classifier, where the optimal cluster prototype is selected.

where, in the probabilistic framework, the weights are positive and follow: $\omega_1 + \omega_2 = 1$. However, as mentioned before, this technique imposes convexity on the posterior probability. Thus, true data integration is not possible with this method. However, the proposed method of distance-fusion is slightly different than linear averaging technique, and convexity implications are no longer valid. In order to prove this, we need to emphasize that the weight ω_{sd} in Equation 6.20 is not bound by any range. It can take values greater

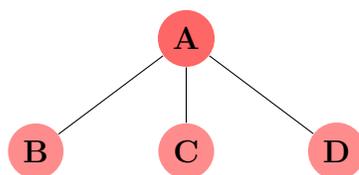


Figure 6.18: An example Bayesian network showing the Bayesian classifier, where the assumption of conditional independence is enforced. The root node, **A**, is connected to all other nodes (additional information).

than one, or lower than zero; in principle, $\omega_{sd} \in (-\infty, +\infty)$. In a probabilistic framework, this amounts to posterior probabilities $P(\mathbf{A} \mid \mathbf{B}, \mathbf{C})$ being outside the interval defined by $P(\mathbf{A} \mid \mathbf{B})$ and $P(\mathbf{A} \mid \mathbf{C})$. The non-convexity will be later proved by introducing an analogy with the “permanence of ratios” method of Journel (2002).

Before proving the non-convexity of our technique, we investigate data integration and naive classifier performance under the strong assumption of conditional independence. In reality, the assumption of conditional independence is rarely true, and one would expect poor performances. However, despite this strong assumption, it provides a surprisingly good performance on a variety of applications. In other words, the classification does not depend on the independency assumption, and according to Domingos and Pazzani (1996), naive classifiers may have high accuracy even on data sets that have strong dependency among their variables. They explained that one of the reasons of such a high accuracy is their zero-one loss function, which does not penalize the inaccurate posterior estimations as long as the correct class gets the highest probability (Friedman and Fayyad, 1997). In other words, the probabilities do not need to be precise as long as they can correctly discriminate between the classes, and it is the ranking of posterior probabilities that identifies the correct class. Even if the probability estimates are inaccurate, good classification solely depends on whether the correct class has higher probability than other classes. For example, assume that there are only two classes/clusters \mathbf{a}_1 and \mathbf{a}_2 for event **A**, and the correct posterior probabilities for each class is $P(\mathbf{a}_1 \mid \mathbf{B}, \mathbf{C}) = 0.8$, and $P(\mathbf{a}_2 \mid \mathbf{B}, \mathbf{C}) = 0.2$. But the posterior probabilities obtained with naive Bayes classifier are $P^*(\mathbf{a}_1 \mid \mathbf{B}, \mathbf{C}) = 0.63$, and $P^*(\mathbf{a}_2 \mid \mathbf{B}, \mathbf{C}) = 0.37$. It is evident that, while the posterior estimations are incorrect, the correct class \mathbf{a}_1 is identified according to the ranking of probabilities. The reason can also be attributed to the fact that the strong dependencies between variables may cancel each other out, and do not affect the classification. Friedman and Fayyad (1997) proved, for

a binary case, that even with a considerable bias in the estimation of probabilities, naive Bayes produces a low inter-class error, which results in good classification performance.

This provides a great simplification in our method for estimating the posterior probabilities. The distance-fusion technique is in accordance with the probabilistic interpretation of naive Bayesian classifiers. We can interpret distances in the context of probabilities by using a Gaussian kernel. Let us assume that the probability of a cluster given a data event is related to the distance between the cluster prototype and the data event. This is a valid assumption and can be formulated with a Gaussian function as follows:

$$\begin{aligned} P(\mathbf{prot}_T^i \mid \mathbf{dev}_T(\mathbf{u})) &= \exp(-d \langle \mathbf{prot}_T^i, \mathbf{dev}_T(\mathbf{u}) \rangle) \\ &= \exp\left(-\sum_{\alpha=1}^{n_T} |\mathbf{prot}_T^i(\mathbf{h}_\alpha) - \mathbf{dev}_T(\mathbf{u} + \mathbf{h}_\alpha)|\right) \end{aligned} \quad (6.25)$$

The equation can be equivalently written for soft data event $\mathbf{sdev}_T(\mathbf{u})$. This probabilistic representation of the distance-fusion technique demonstrates the analogy with naive Bayesian classifiers. As a result, the probability of the cluster prototype \mathbf{prot}_T^i given the data event $\mathbf{dev}_T(\mathbf{u})$ and soft data event $\mathbf{sdev}_T(\mathbf{u})$ is computed by re-writing Equation 6.20 as follows:

$$\begin{aligned} P(\mathbf{prot}_T^i \mid \mathbf{dev}_T(\mathbf{u}), \mathbf{sdev}_T(\mathbf{u})) &= P(\mathbf{prot}_T^i \mid \mathbf{dev}_T(\mathbf{u})) \\ &\quad + f(\mathbf{dev}_T(\mathbf{u})) \times \omega_{sd} \times P(\mathbf{prot}_T^i \mid \mathbf{sdev}_T(\mathbf{u})) \end{aligned} \quad (6.26)$$

Equation 6.26 describes a naive Bayesian classifier. However, unlike the linear averaging of probabilities, it has a free weight parameter of ω_{sd} that can take any real value. As a result, the solution is non-convex and can take values outside the interval defined by the minimum and maximum of $P(\mathbf{prot}_T^i \mid \mathbf{dev}_T(\mathbf{u}))$ and $P(\mathbf{prot}_T^i \mid \mathbf{sdev}_T(\mathbf{u}))$.

In addition to a good classification performance of the distance-fusion technique, it is comparable to “permanence of ratios” method and Tau model. To prove this, we start with the Tau model and develop the distance-fusion equation accordingly. Tau model is shown by introducing a τ parameter that controls the redundancy between the data:

$$\frac{x}{b} = \left(\frac{c}{a}\right)^\tau \quad (6.27)$$

or equivalently,

$$\frac{\frac{P(\bar{\mathbf{A}} | \mathbf{B}, \mathbf{C})}{P(\mathbf{A} | \mathbf{B}, \mathbf{C})}}{\frac{P(\bar{\mathbf{A}} | \mathbf{B})}{P(\mathbf{A} | \mathbf{B})}} = \left(\frac{\frac{P(\bar{\mathbf{A}} | \mathbf{C})}{P(\mathbf{A} | \mathbf{C})}}{\frac{P(\bar{\mathbf{A}})}{P(\mathbf{A})}} \right)^\tau \quad (6.28)$$

After some manipulations, one can obtain the Tau model as follows:

$$P(\mathbf{A} | \mathbf{B}, \mathbf{C}) = \frac{P(\mathbf{A}) \cdot P(\mathbf{B} | \mathbf{A}) \cdot P(\mathbf{C} | \mathbf{A})^\tau}{P(\mathbf{B}, \mathbf{C})} \quad (6.29)$$

This can be written in the following form:

$$P(\mathbf{A} | \mathbf{B}, \mathbf{C}) = \frac{P(\mathbf{A})}{P(\mathbf{B}, \mathbf{C})} \cdot P(\mathbf{B} | \mathbf{A}) \cdot P(\mathbf{C} | \mathbf{A})^\tau \quad (6.30)$$

Now if we take the natural logarithm of both sides, we get:

$$\ln(P(\mathbf{A} | \mathbf{B}, \mathbf{C})) = \ln\left(\frac{P(\mathbf{A})}{P(\mathbf{B}, \mathbf{C})}\right) + \ln(P(\mathbf{B} | \mathbf{A})) + \ln(P(\mathbf{C} | \mathbf{A})^\tau) \quad (6.31)$$

The event \mathbf{A} , \mathbf{B} , and \mathbf{C} respectively represent the cluster prototype, data event, and soft data event. In other words:

$$\begin{aligned} A &: \{\text{cluster} = \mathbf{prot}_T^i\} \\ B &: \{\mathbf{dev}_T(\mathbf{u})\} \\ C &: \{\mathbf{sdev}_T(\mathbf{u})\} \end{aligned} \quad (6.32)$$

In Equation 6.31, the prior probability $P(\mathbf{A})$ is assumed to be constant for all clusters, since there is no information indicating which cluster is more appropriate for location \mathbf{u} . For all k clusters, one can assume $P(\mathbf{A}) = \frac{1}{k}$. Consequently, at node location \mathbf{u} , the data event $\mathbf{dev}_T(\mathbf{u})$ and soft data event $\mathbf{sdev}_T(\mathbf{u})$ are known, that is, $P(\mathbf{B}, \mathbf{C})$ does not depend on the cluster prototypes. Therefore, since both $P(\mathbf{A})$ and $P(\mathbf{B}, \mathbf{C})$ are constant for each node location \mathbf{u} , the probability $\frac{P(\mathbf{A})}{P(\mathbf{B}, \mathbf{C})}$ in Equation 6.31 is constant irrespective of the clusters. We denote the logarithm of this term in Equation 6.31 by a constant c , as follows:

$$\ln(P(\mathbf{A} | \mathbf{B}, \mathbf{C})) = c + \ln(P(\mathbf{B} | \mathbf{A})) + \ln(P(\mathbf{C} | \mathbf{A})^\tau) \quad (6.33)$$

In addition, according to our definitions relating the probability to distances, one can simply assume that the probability of a prototype given the data event is equal to the probability of the data event given the prototype; and as such, for the soft data event. The reason is

that we defined the probability according to the symmetric definition of distances between the two data, that is, $d\langle \mathbf{A}, \mathbf{B} \rangle = d\langle \mathbf{B}, \mathbf{A} \rangle$, and similarly for event \mathbf{C} , $d\langle \mathbf{A}, \mathbf{C} \rangle = d\langle \mathbf{C}, \mathbf{A} \rangle$. As a result Equation 6.33 can be written as follows:

$$\ln(P(\mathbf{A} | \mathbf{B}, \mathbf{C})) = c + \ln(P(\mathbf{A} | \mathbf{B})) + \ln(P(\mathbf{A} | \mathbf{C})^\tau) \quad (6.34)$$

and if we input the probabilities into these equations, we get:

$$\begin{aligned} \ln(P(\mathbf{prot}_T^i | \mathbf{dev}_T(\mathbf{u}), \mathbf{sdev}_T(\mathbf{u}))) &= c + \ln(P(\mathbf{prot}_T^i | \mathbf{dev}_T(\mathbf{u}))) \\ &\quad + \ln(P(\mathbf{prot}_T^i | \mathbf{sdev}_T(\mathbf{u}))^\tau) \\ &= c + \ln(P(\mathbf{prot}_T^i | \mathbf{dev}_T(\mathbf{u}))) \\ &\quad + \tau \ln(P(\mathbf{prot}_T^i | \mathbf{sdev}_T(\mathbf{u}))) \end{aligned} \quad (6.35)$$

upon substitution of Equation 6.25 for the probabilities, one will obtain:

$$\begin{aligned} \ln(P(\mathbf{prot}_T^i | \mathbf{dev}_T(\mathbf{u}), \mathbf{sdev}_T(\mathbf{u}))) &= c + \ln(\exp(-d\langle \mathbf{prot}_T^i, \mathbf{dev}_T(\mathbf{u}) \rangle)) \\ &\quad + \tau \ln(\exp(-d\langle \mathbf{prot}_T^i, \mathbf{sdev}_T(\mathbf{u}) \rangle)) \end{aligned} \quad (6.36)$$

which can be simplified by the properties of Natural logarithms as follows:

$$\begin{aligned} \ln(P(\mathbf{prot}_T^i | \mathbf{dev}_T(\mathbf{u}), \mathbf{sdev}_T(\mathbf{u}))) &= c - d\langle \mathbf{prot}_T^i, \mathbf{dev}_T(\mathbf{u}) \rangle \\ &\quad - \tau d\langle \mathbf{prot}_T^i, \mathbf{sdev}_T(\mathbf{u}) \rangle \end{aligned} \quad (6.37)$$

and after re-arranging some terms one will get:

$$\begin{aligned} c - \ln(P(\mathbf{prot}_T^i | \mathbf{dev}_T(\mathbf{u}), \mathbf{sdev}_T(\mathbf{u}))) &= d\langle \mathbf{prot}_T^i, \mathbf{dev}_T(\mathbf{u}) \rangle \\ &\quad + \tau \times d\langle \mathbf{prot}_T^i, \mathbf{sdev}_T(\mathbf{u}) \rangle \end{aligned} \quad (6.38)$$

Now, if we denote the term on the left side by d_i , we will get:

$$d_i = d\langle \mathbf{prot}_T^i, \mathbf{dev}_T(\mathbf{u}) \rangle + \tau \times d\langle \mathbf{prot}_T^i, \mathbf{sdev}_T(\mathbf{u}) \rangle \quad (6.39)$$

Finally, the above equation can be written for all $i : \{1, 2, \dots, k\}$, where k is the number of clusters. The results for all k equations can be gathered into the corresponding vector, as

follows:

$$\mathbf{d} = \mathbf{d}_{hd} + \tau \times \mathbf{d}_{sd} \quad (6.40)$$

This equation is equivalent to Equation 6.20; where $\tau = f(\mathbf{dev}_T(\mathbf{u})) \times \omega_{sd}$. Therefore, we were able to derive the distance-fusion approach of Equation 6.20 from the “permanence of ratios” method (Tau model) of Equation 6.28. This derivation demonstrates that the two approaches are analogous to each other, and as a result, the distance fusion technique is a robust and accurate method for data integration of different information sources.

In the next section, we will show an example application of this technique using soft data for both the multi-grid and multi-resolution approaches.

6.2.4 Examples and Sensitivity

Consider the reference case shown in Figure 6.19. It is a 59×59 Earth model consisting of channels. Four different soft data are used for investigating soft data integration methodology. Different soft data have been obtained by applying averaging filters of sizes 2×2 , 3×3 , 4×4 , and 5×5 . It can be observed that the higher the averaging filter, the lower the resolution of the soft data (or seismic probability cubes). There are fewer distinctive features observed in the soft data as the resolution drops. Hence, one expects the resulting Earth models to have more spatial variability within the lower resolution seismic data.

In the first example, we investigate the sensitivity of the soft data integration approach on the template size used for pattern modeling. Two template sizes of 7×7 , and 9×9 are considered. Figure 6.20 shows sensitivity results to the effectiveness of data integration. Two sample realizations are shown for each template by using ‘medium’ confidence on the soft data. The confidence on soft data is characterized by the factor ω_{sd} of Equation 6.20. Medium confidence amounts to $\omega_{sd} = 0.35$. It can be observed in Figure 6.20 that the smaller template size of 7×7 can better reproduce both the patterns of the training image, and most importantly, the spatial structures of the reference Earth model (shown in Figure 6.19). This is due to higher capabilities of image reconstruction using the smaller templates than the larger ones, where freezing the nodes within a bigger inner patch prohibits the introduction of new patterns. The E-types obtained from 150 realizations are also shown in Figure 6.20. Again, the E-type from 7×7 template provides more information in terms of channel locations and their connectivities.

Next, we investigate the application of data integration using different resolutions of

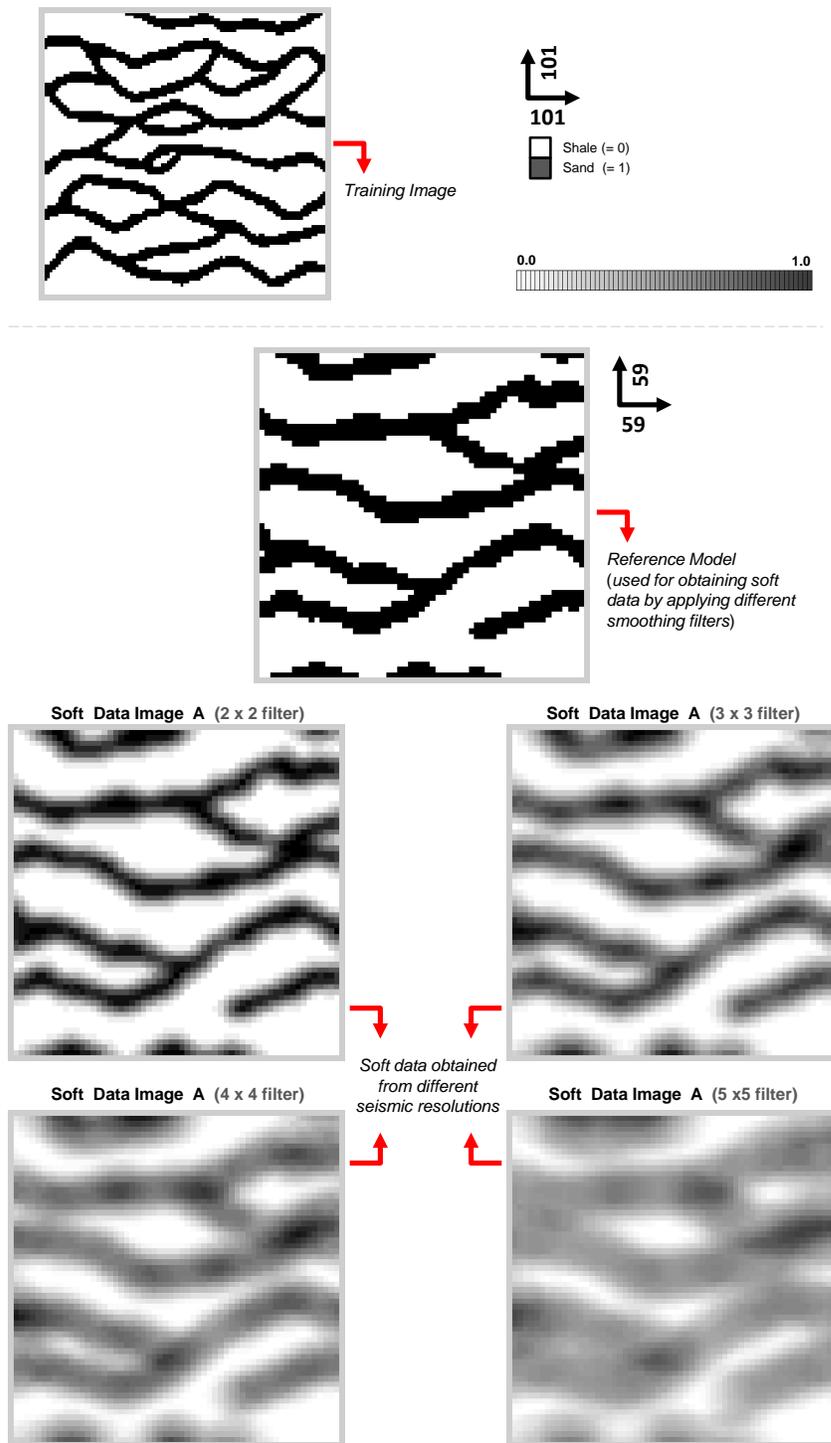


Figure 6.19: The channel training image, used for application of soft data integration, and the reference case are shown. Four soft data are generated by applying an averaging filter with windows sizes of 2×2 , 3×3 , 4×4 , and 5×5 , indicating the resolution of seismic data.

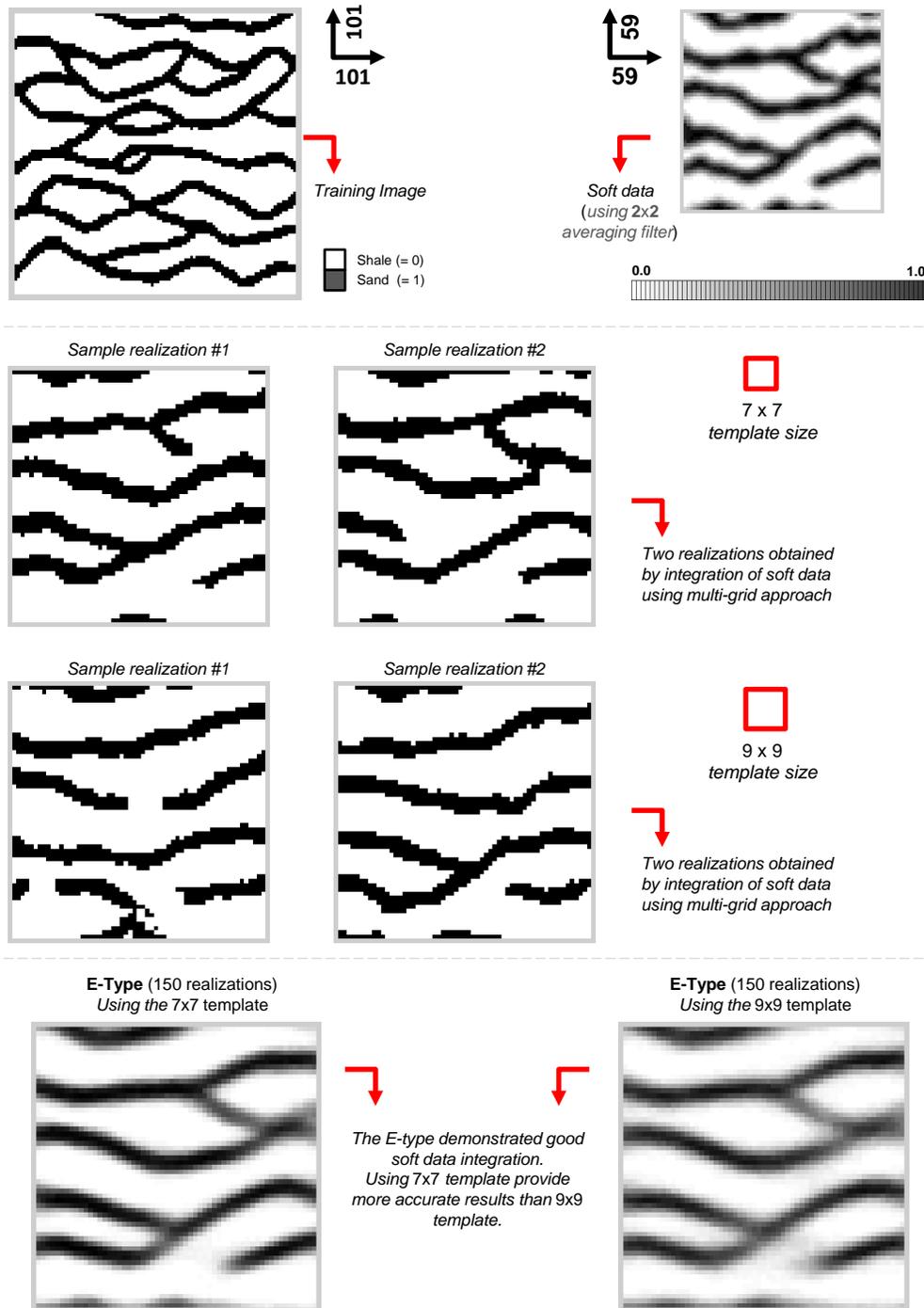


Figure 6.20: Sensitivity of soft data integration approach on template sizes T . Two sample realizations are shown for each template size of 7×7 and 9×9 . The E-types obtained from 150 realizations are also shown for each template. Smaller template of 7×7 can provide more accurate data integration.

soft data. In general, one expects the obtained E-types to be less informative in terms of the probabilities of occurrence of channels. Figure 6.21 displays two sample realizations obtained using different resolutions of soft data. 150 realizations were generated for each of the four soft inputs. The generated E-types are shown in Figure 6.22. One can observe that the E-Types show similar behavior to their corresponding soft data. More informative soft information lead to a decrease in the uncertainty within E-types. However, even in the lowest resolution case, where an averaging filter of 5×5 were used to generate the soft data, the E-type reflects a reasonable certainty in spatial location of training image features. It demonstrates the capability of the proposed data integration approach, where the similarity of soft data events to prototypes is included in the analysis.

All previous examples assumed ‘medium’ confidence over the soft data by assigning 0.35 to ω_{sd} . Due to practical uncertainties inherent in seismic imaging, one needs to investigate the behavior of data integration procedure by imposing different confidences on soft data. The soft data, obtained by an averaging filter of size 3×3 over the reference model, is used in this experiment. Different confidence levels of weak, medium, and strong are checked with $\omega_{sd} = 0.15, 0.35, \text{ and } 0.6$, respectively. The results are illustrated in Figure 6.23. It is evident that upon increasing the confidence on soft data, the resulting E-type will exhibit less spatial uncertainty. This is due to more weight (ω_{sd}) put on similarity comparisons of soft data events with prototypes than on the data events extracted from the realization. The distance-fusion technique will thus favor soft information more than other sources of information as the confidence level increases.

So far, we have explored the capabilities of the proposed distance-fusion technique in soft data integration using a multi-grid approach. Another multi-scale geostatistical modeling approach of multi-resolution simulation, introduced in Section 4.2, is investigated next. The application of data integration in a multi-resolution setting is shown in Figure 6.24 for two confidence levels of ‘weak’ and ‘medium’ over soft information. Clearly, the multi-resolution approach can properly combine different sources of information. The higher confidence level reduces the spatial uncertainty within the E-type (Figure 6.24).

6.3 Conclusion

Integrating additional sources of information is of paramount importance in geostatistical modeling. In this chapter, we introduced the techniques for integrating two different sources

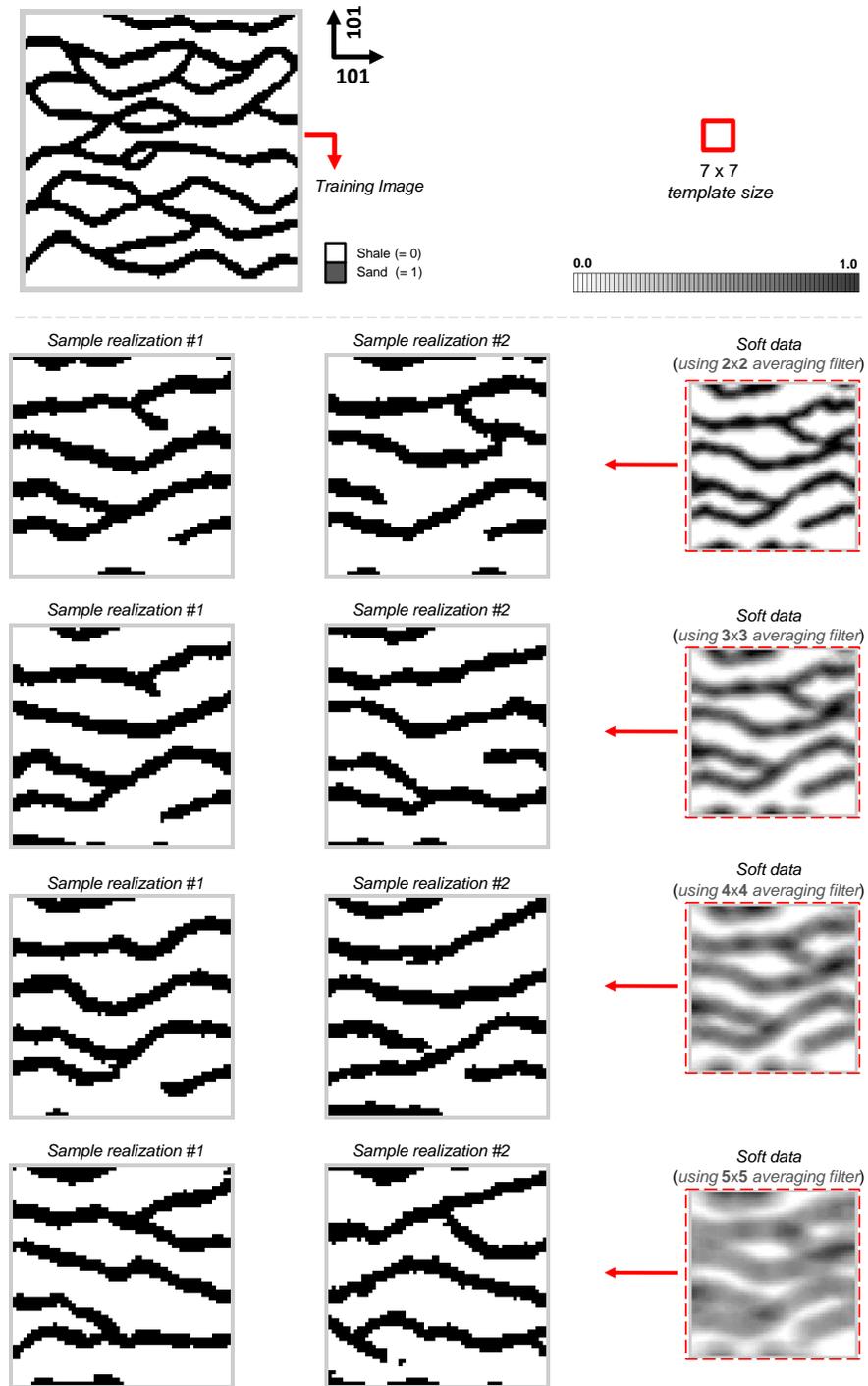


Figure 6.21: Two realizations are shown for each of the four soft data used used in a multi-grid simulation. The illustrated soft data differ with respect to only their resolutions.

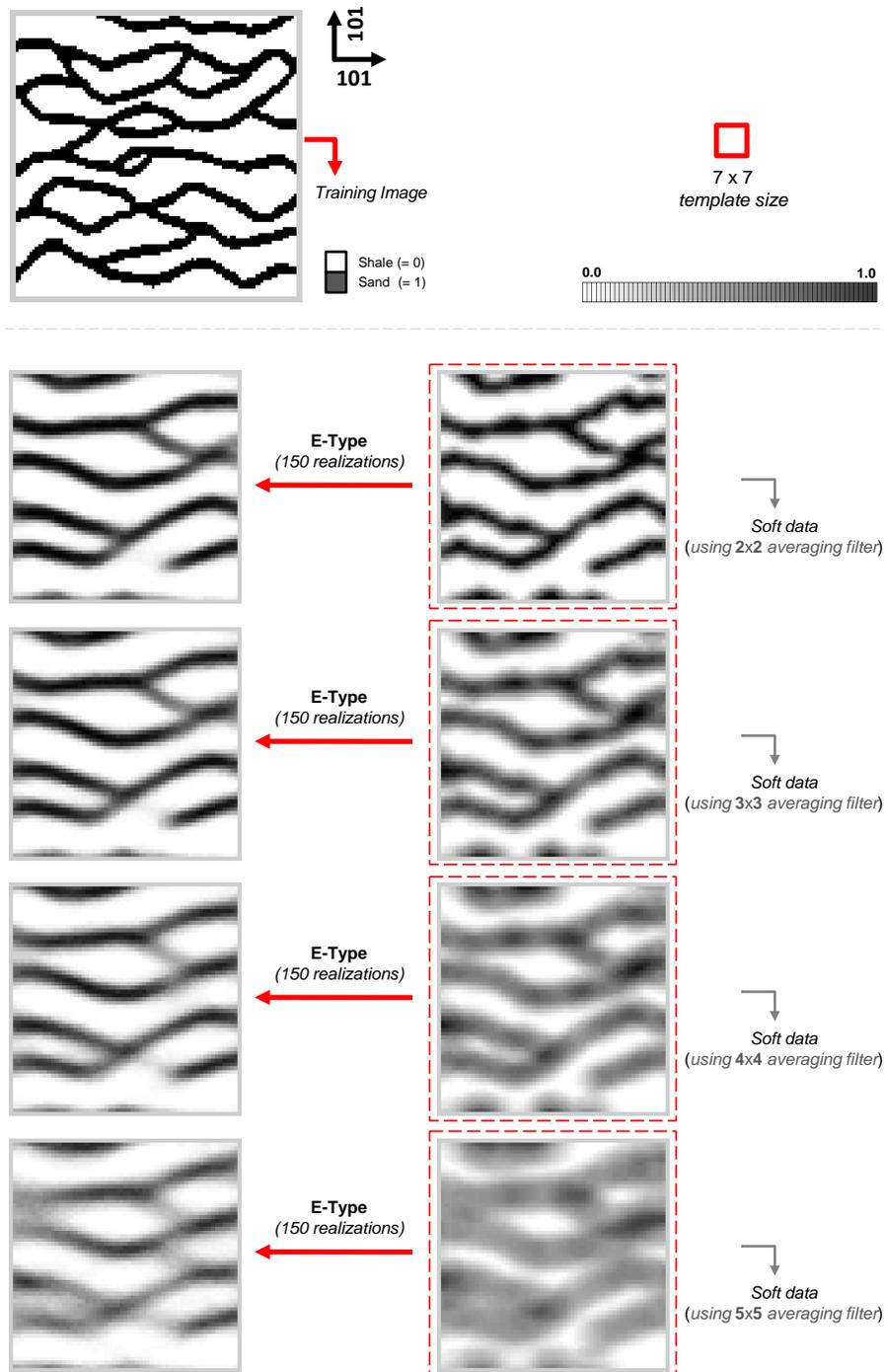


Figure 6.22: The E-types obtained from 150 Earth models generated using different soft data resolutions; indicating the increase in uncertainty by a decrease in resolution. Medium confidence interval is chosen in this analysis.

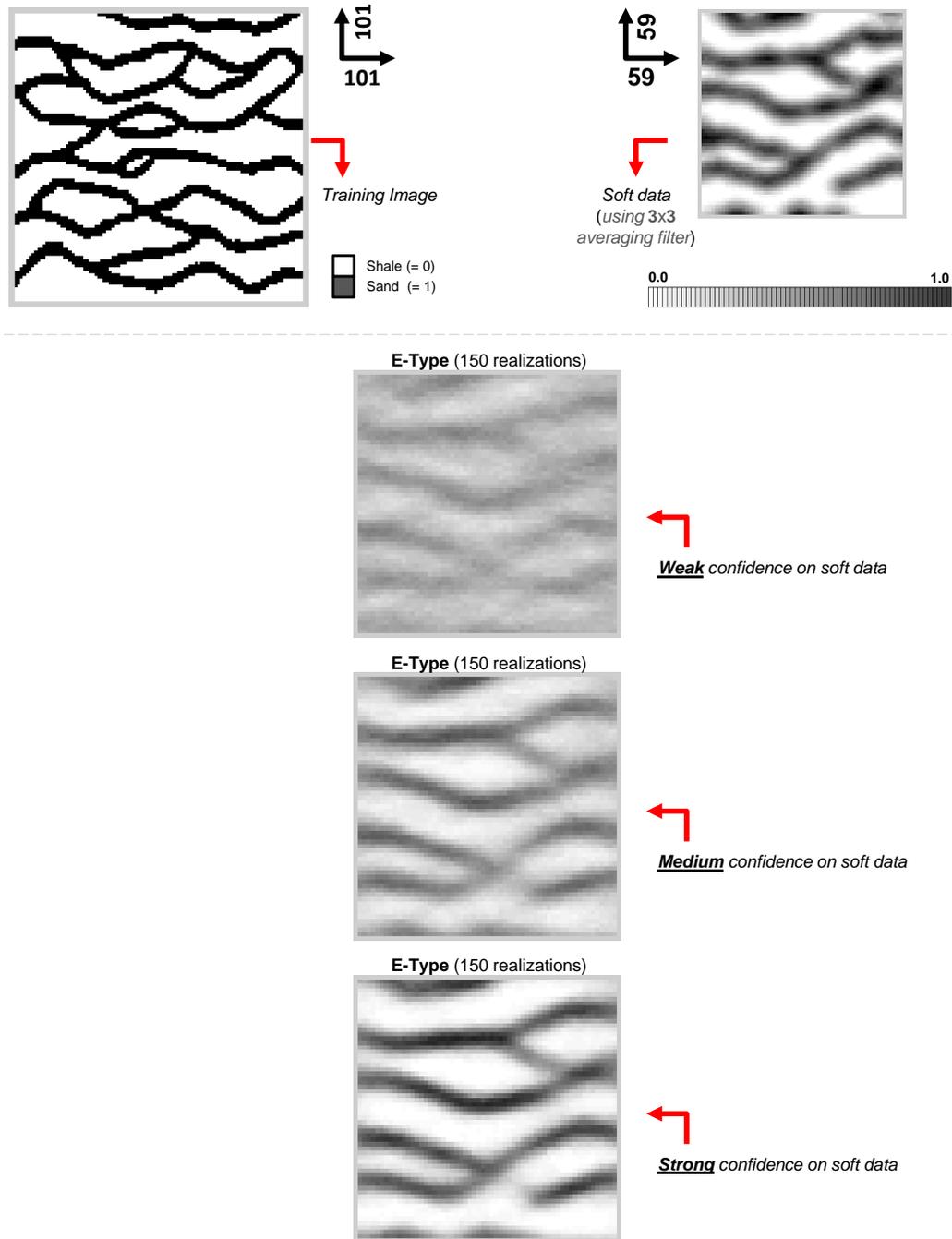


Figure 6.23: The resulting E-types with different confidence levels of weak, medium, and strong over soft data are illustrated. Confidence levels are characterized by setting $\omega_{sd} = 0.15, 0.35,$ and 0.60 respectively. It is shown that higher confidence can lead to less uncertainty in the E-type.

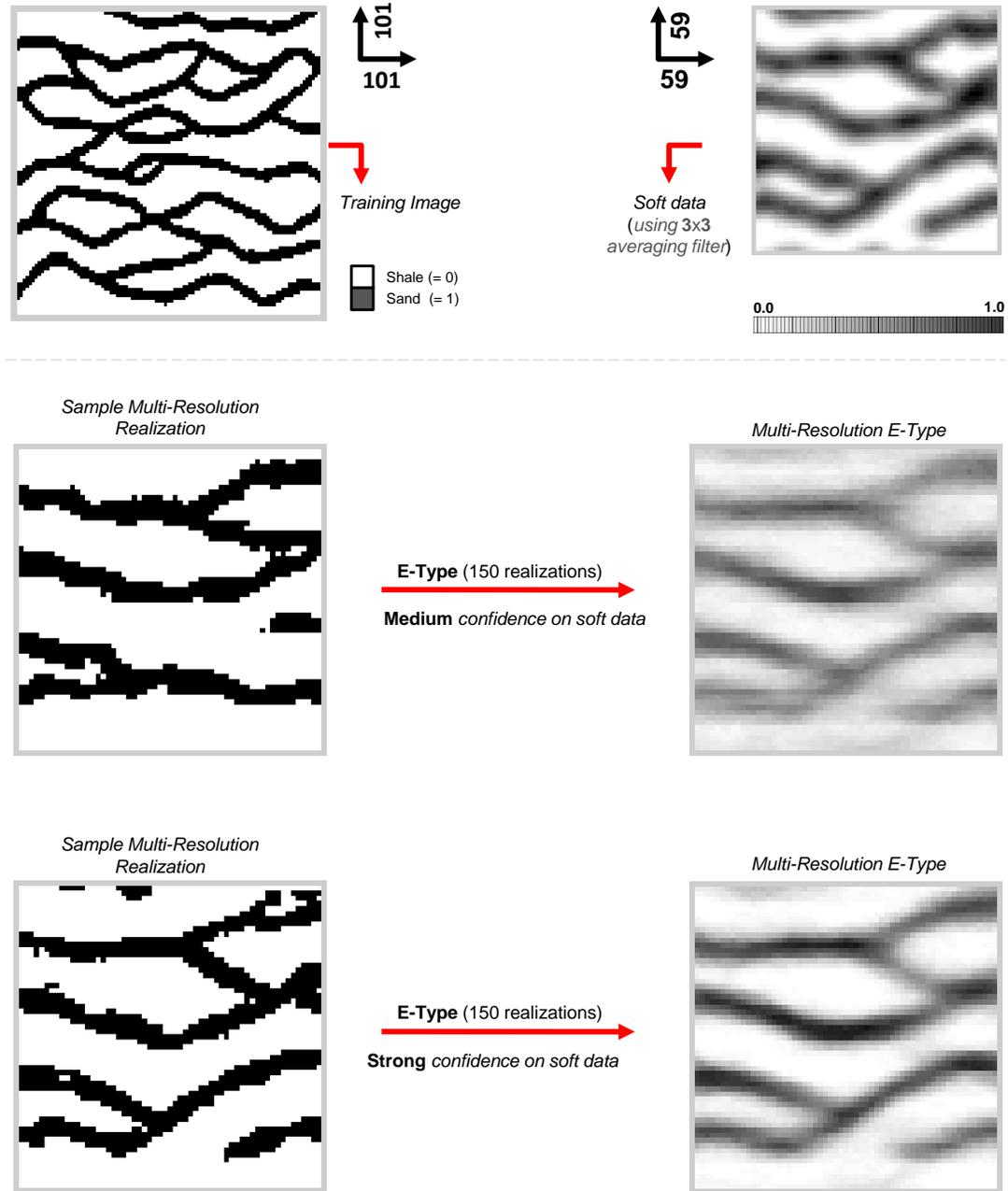


Figure 6.24: The application of soft data integration in a multi-resolution setting is depicted. The effectiveness of methodology is observed by observing the generated E-types for each confidence level of medium and strong ($\omega_{sd} = 0.35$, and 0.60 respectively).

of information; namely, hard data and soft data.

The data conditioning algorithm in DisPAT for hard data was demonstrated to be superior to previous MPS methodologies. Better pattern reproduction was obtained at hard data locations. For example, the binary channel training image showed that the connectivity and continuity of channels are much better produced in DisPAT method. Hard data integration was also investigated for multi-scale simulations. A new ‘smart-vibrating multi-scale’ approach was introduced that would not only improve upon the data relocation methodology, but also eliminate internal inconsistencies produced by traditional approaches. The example application of such a technique showed the similarities with the rejection sampling algorithm for data conditioning.

Soft data conditioning was implemented by the introduction of distance-fusion technique. In this technique, the same paradigm of distance-based modeling was adapted for fusing the distances obtained by soft data event and normal data event with each prototype. Several examples with different confidence levels on soft information were provided, and extensive sensitivity analyses to the parameters were conducted. Finally, an analogy between the distance-fusion technique and the ‘permanence of ratios’ model was provided. It was stated why the proposed distance-based classifier is capable of correctly finding the optimal prototype, even when the conditional independence assumption may not hold.

At the end, the conditioning capabilities of DisPAT algorithm to either hard data or soft data showed promising results with great improvements.

Chapter 7

Non-Stationarity

Natural science does not simply describe and explain nature, it is part of the interplay between nature and scientists.

WERNER HEISENBERG (1901 - 1976)

Consider the training image shown in Figure 7.1 conceptualizing a fluvial deltaic fan reservoir. Geostatistics is based upon the assumption of stationarity. Such a training image does not possess the statistical properties of a stationary process. Channels on the top left corner are thick and gradually become thinner towards the lower right corner, while changing direction by spreading out from the top left corner source. Not accounting for the spatial variability in this training image will result in undesired and unrealistic Earth models. One such model example is shown in Figure 7.1. It can be observed that the spatial features and local statistics of the model are different than the conceptual training image. Without the assumption of stationarity, one cannot accurately capture the multiple-point statistics existing at different spatial locations.

In geostatistics, one infers the statistics of the attribute under study, from the model of spatial variability. Multiple-point geostatistics relies on inferring multiple-point facies joint correlation moments from the provided training image. In order to infer the statistics, there is a need for repetitiveness in the samples. Therefore, the statistics are inferred by observing the samples at all locations over the training image. The training image should be stationary in a sense that statistical properties of facies distributions do not vary in space

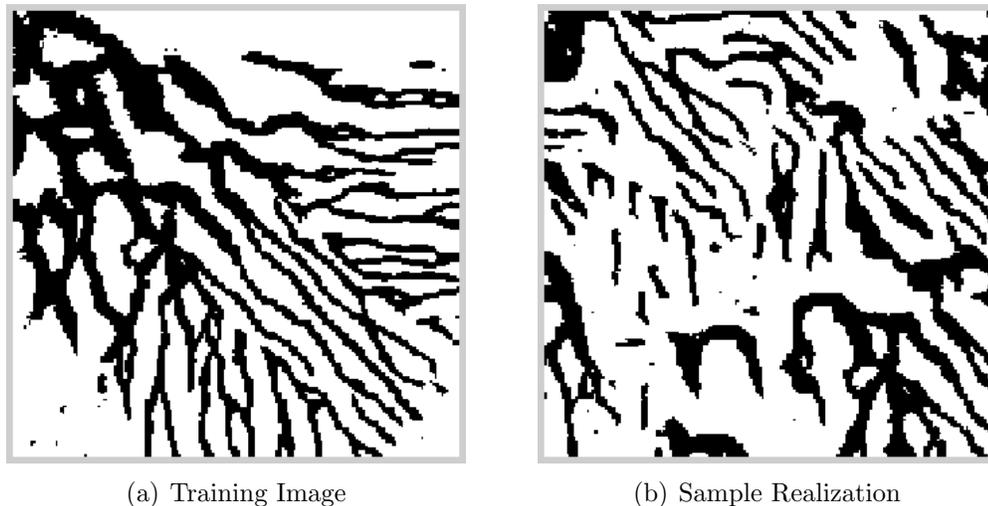


Figure 7.1: (a) Training image representing a fluvial deltaic fan reservoir, (b) one sample realization generated with MPS technique of DisPAT without accounting for non-stationarity of the training image.

(Journal, 1985). This situation rarely happens in real geological phenomena. However, the physical processes and the facies characteristics usually vary in space in a known fashion. Geological processes and topographic forces such as the presence of a salt dome, seal level degradation/progradation, or changes of sedimentation source cause spatial variations in the dimensions of facies geobodies and their directions (Strebelle and Zhang, 2005). For instance, in a channel fan model shown in Figure 7.1, the patterns representing multiple-point statistical information are not similar in all regions. The statistics from one region cannot be assumed relevant for another region. The orientation and the thicknesses of channels vary over the training image. Pooling together all the regions to obtain statistics is not meaningful in such training images. One region might have a higher proportion of sand facies than another region. Assuming stationarity on such a training image falsely implies that the patterns and structures are location-independent and can be reproduced at any location. This assumption thus results in unrealistic realizations. In this chapter, we relax this assumption of stationarity, and provide algorithms that can automatically incorporate the spatial component of pattern variability within the geostatistical modeling.

In the next section, a review of the previous attempts on modeling non-stationary geological phenomena is presented. The limitations in previous approaches will be discussed. Afterwards, three different algorithms for modeling non-stationarity will be provided. The

first two algorithms, called ‘Spatial-Similarity Method’ and ‘Neighborhood-Radius Method’, rely on similar concepts of incorporating spatial information within the multiple-point statistical modeling. The applicability of these methods will be demonstrated with examples. The third algorithm called ‘Automatic-Segmentation Method’ will eventually combine the two preceding ideas to provide a more flexible framework towards non-stationarity geostatistical modeling.

7.1 Overview

The most widely used concept for generating non-stationary models is based on the concepts of rotation and affinity. In a two-point statistical realm, an isotropic variogram model can be made location-specific using various rotation and affinity transformations of the grid coordinates to bring a locally differing anisotropy into the model (Xu, 1996).

The same concept of rotation and affinity was adapted by Zhang (2002). The assumption of stationarity is still valid, but statistics are obtained from a transformed training image according to the node location that is being simulated. It is assumed that non-stationarity is only formed by variations in: (1) direction of the features and structures, and (2) the scale of the patterns. Therefore, there is a need for an additional source of information on rotation and affinity, which could be obtained from well logs and geologic and seismic interpretations (Mondt, 1993). The conceptual training image is thus rotation and scale invariant; and additional information sources, dictating how the statistics should be obtained, are required to construct the models.

In Zhang (2002), the probabilistic approach of snesim was used to construct one search tree for the rotation and scale invariant training image. At each node location \mathbf{u} during the simulation, the data event is transformed (rotated, or scaled) with respect to the additional information regarding the non-stationarity. The transformed data event is analyzed by the search tree to obtain the multiple-point conditional probabilities. Two simple examples of the concept of rotation and scale are shown individually in Figure 7.2. The example on top only applies constant rotation angles for the entire realization grid, and the second example shows the desired scale variations on the entire domain. One can observe that one training image can lead to many different realizations with disparate multiple-point statistics. Choosing different spatial rotations or scales will force the realizations to honor this location-dependent information, and as such, a non-stationary Earth model will be

produced.

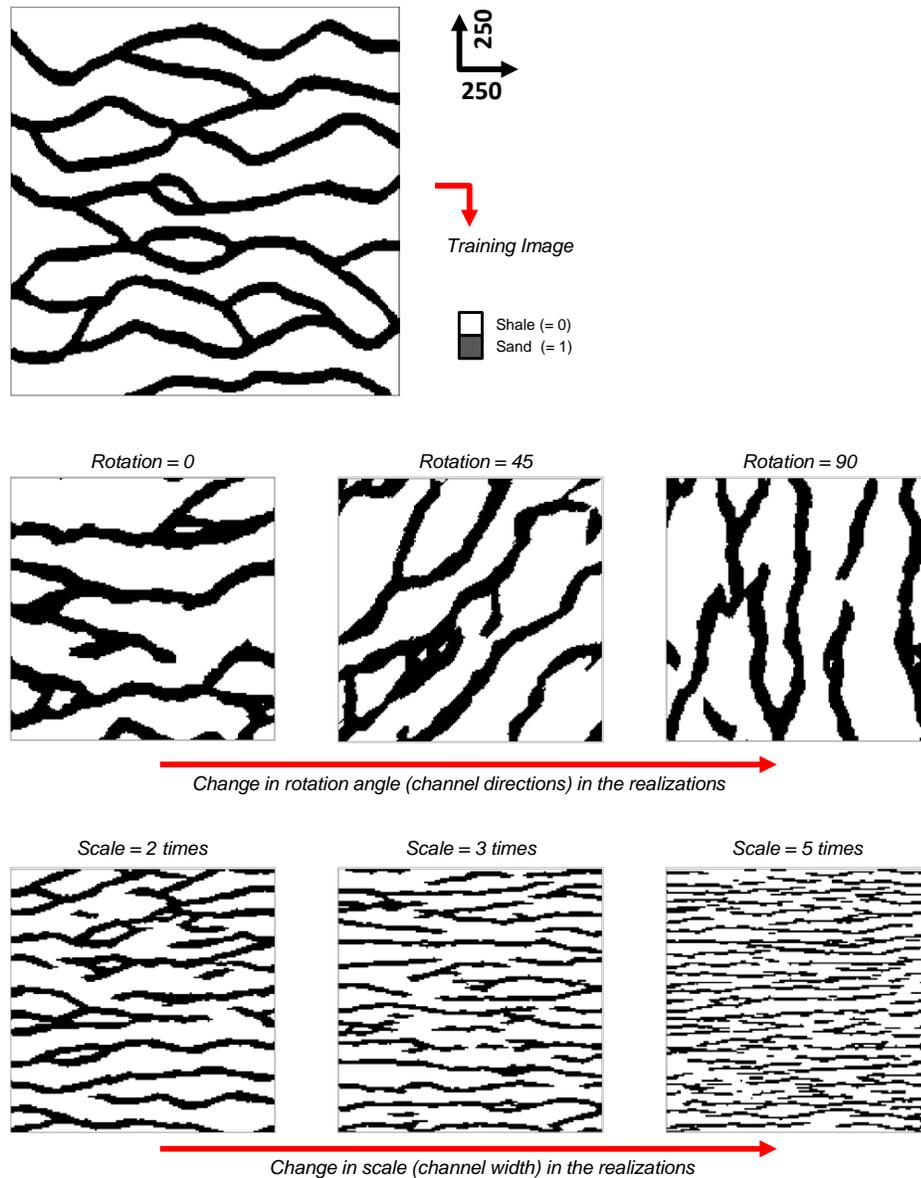


Figure 7.2: Example application of rotation and affinity concept for non-stationary analysis. The channel training image is used to construct a search tree in snesim. Different location-specific rotation constraints or scale constraints can be applied on data event to generate non-stationary realizations (models from Zhang (2002)).

The method of Zhang (2002) had some limitations in multi-grid simulations. In coarser

grids, the rotations or affinity transformations of the data template will result in a template that does not fall into the active nodes of the coarse grid. Therefore, the components of the template are relocated to the closest node of the coarse grid. This relocation can cause approximations regarding the actual locations of the conditioning data event. In addition, the method demands a CPU-intensive transformation of the data event at each simulated node location. In order to remove this limitation, Caers (2004); Strebelle and Zhang (2005); Wu (2007) proposed a similar approach. Instead of constructing one search tree for the training image, many search trees will be generated for different rotations or scales. In other words, if two rotations of 0 and 90 degrees are required, two search trees corresponding to the two rotated training images (one by 0 degree and another by 90 degree) are constructed. Construction of many search trees can be perceived as generating many training images; contrary to the previous approach that relied on only one training image. During the simulation of node \mathbf{u} , the data template will no longer be transformed, but only according to the rotation angle (0 or 90 degree), the corresponding search tree will be searched to obtain the conditional statistics. This approach of constructing different search trees according to rotation and affinity constraints reduces the approximations caused by data relocation in the previous approach of Zhang (2002). However, one of the limitations of this new approach is the number of required search trees. To alleviate the memory needed for storing all trees, the angles or scales were categorized to some pre-specified number of bins, and only a limited set of trees were constructed. Another approach is to simulate locations not in a purely random fashion, but simulating all the nodes falling within a specific bin one after another. This approach of successive region simulations, however, reduces the stochasticity in multiple-point statistical modeling.

Previous approaches were implemented for the probabilistic MPS algorithm of *snesim* using the search tree. *Simpat* and *filtersim* took a similar methodology for their pattern-based multiple-point geostatistical algorithms. However, the previous approach was only applicable to training images that can be made stationary by rotation and affinity transformation. In order to circumvent this limitation, *simpat* introduced the concept of regions. In this methodology, the reservoir is first divided into several subregions. These subregions are user-defined, according to the morphological parameters associated with them. For example, the variation in the rotation or the facies proportion defines different regions in the model. One can thus model the petrophysical properties of each subregion individually. For that, each region is associated with a different training image corresponding to

patterns deemed relevant to that region. The difference in non-stationary modeling of this approach to the previous ones is the introduction of completely distinct training images for each subregion. Unlike previous methods, there is no need for scaling or rotating one master training image for each location of the model. Simpat allowed completely distinct training images to be combined in such a way that it produces a smooth and gradual transition from one region to the other. An illustration of this approach is provided in Figure 7.3. As can be seen, the model is divided into a maximum of 10 regions related to geomorphological characteristics of scaling and rotation.

This second approach, based on regions, has a broader application than the one based on rotation and affinity. One can simply define subregions for locations with similar characteristics.

The other pattern-based approach of *filtersim*, which uses filters for pattern classification, followed the same procedure of region-based simulation for non-stationary modeling (Wu, 2007). Although in principle, one can use different training images for each region, but the implemented algorithm only accounts for rotation and affinity regions (*filtersim* in *SGeMS*). The user provides two separate grids defining the values for each transformation. Then, the algorithm automatically transforms the training image for each of the user-input stationary regions.

One of the issues in region-based approaches is the appearance of discontinuities in the border along regions when their corresponding training images are not compatible with each other. Another approach that does not suffer from this issue uses probability fields to control the occurrence of attributes (Strebelle, 2002; Harding et al., 2005). In this approach, one complex training image is used instead of many different training images; however, non-stationarity in their approach is controlled by pre-defining azimuths and local probabilities of different facies in the spatial domain. The provided probability field is integrated using the Tau model (Journel, 2002). However, this method of data integration using the Tau model can lead to unknown behaviors in obtaining conditional probabilities from the search tree.

Therefore, two issues can be identified in previous approaches. The first one concerns the analytical approximation of the conditional probabilities obtained by the Tau model. The second one is the arbitrary classification of the models into different regions, according to some auxiliary information (such as rotations, scales, proportions, etc.). Two alternative methods were proposed for non-stationary simulations to eliminate these issues.

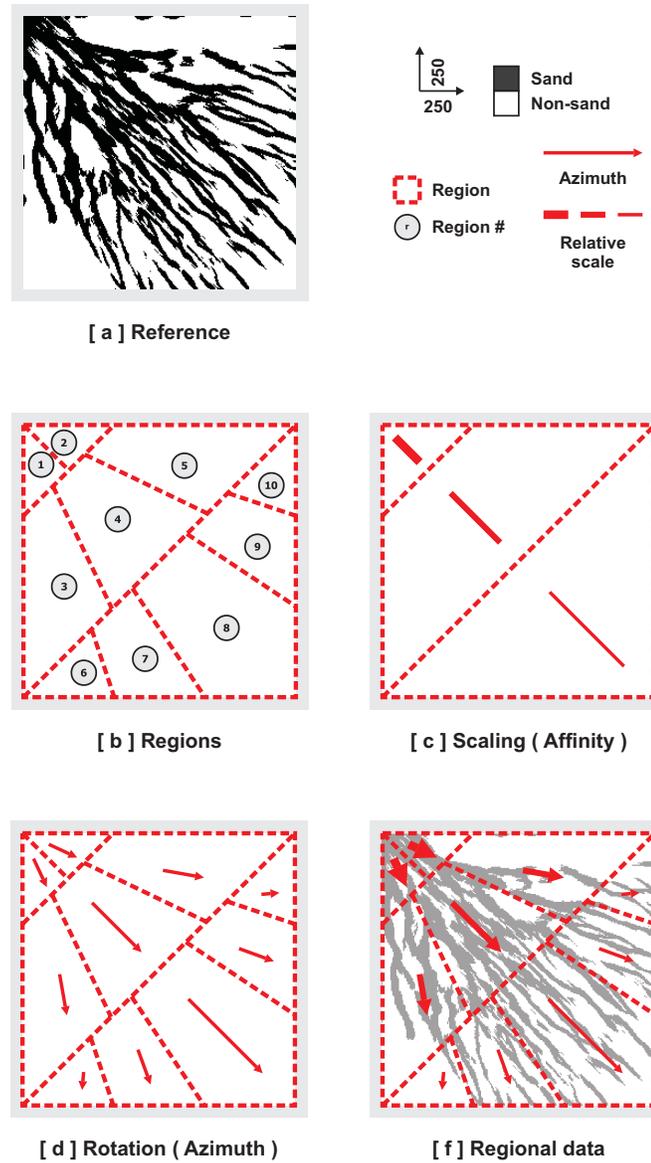


Figure 7.3: Example application of non-stationary modeling in simpat MPS methodology by defining 10 regions according to the morphological characteristics of rotation and scale (models from Arpat (2005)).

The first method introduces a continuous auxiliary variable and uses it to constrain the simulation (Chugunova and Hu, 2008). It is assumed in this method that both the principal and auxiliary training images are available. The auxiliary variable is often a smooth function

in space. For example, it could be the domain representing the orientation of the features. The simulation proceeds by constructing search trees for both the principal and the auxiliary training images. The retrieval of the conditional distributions has two steps. In the first step, the principal tree is searched for neighborhood nodes and the patterns characterized by them. A set of tree nodes would then satisfy these conditions. These nodes are then searched for in the tree corresponding to the auxiliary training image. Finally, only those nodes that are also compatible with the auxiliary data will be retained. The statistics are thus retrieved from the retained nodes and used for drawing a value at the unsampled location. The method of Chugunova and Hu (2008) thereby removed the dependency on the Tau model for auxiliary data integration. Moreover, it eliminated the need for reducing the auxiliary data into a certain number of classes/regions. However, that method is appealing when the auxiliary data that controls the non-stationarity of the training image can be easily integrated into the methodology. Each auxiliary variable requires its own set of formulations to define the implied characteristics. For example, auxiliary data for proportion control is handled differently than the ones that control rotations. The algorithm must be able to calculate the rotational attributes of the data event (an algorithm to find the orientation of features). Therefore, a database of formulations for a variety of auxiliary data needs to be available. Besides this limitation on handling auxiliary data, it can only accept one simple training image. If the features or structures under study vary significantly over the area, one would need to use different training images. For example, if the east side of the model has only channels and the west side consists of only lobes, then no single stationary training image exists that, combined with an auxiliary data, can reproduce the desired non-stationary phenomena.

The second method to overcome the previous issues (pre-defining regions, and incorporating probability fields) was introduced by Boucher (2009). This method, unlike previous ones, uses only one non-stationary training image for simulation. It is based on the concept of search tree partitioning, where the training image is subdivided into homogeneous smaller images (Boucher, 2007). These subdivisions are named ‘partition classes’. For each partition class, it will generate a corresponding search tree with the *snesim* algorithm. The methodology is similar to the region-based approaches. Here, regions are called partition classes. The difference lies in their definition. Regions are obtained by a user decision on how the non-stationarity is to be modeled. On the other hand, partition classes are obtained by processing the training image by spatial filters; the same ones used in *filtersim*

for pattern classification. Similar to filtersim, the filter scores are grouped into clusters, and the resulting clusters define the partition classes. The simulation proceeds in a fashion similar to the region-based approach, but instead of successive simulation of each region, it simulates all the partition classes in a random fashion. The search trees for each of the partition classes are smaller in comparison with the search trees generated from one big training image for each region in region-based approaches. Boucher (2009) demonstrated that having only one training image eliminates the issues regarding boundary artifacts that are inherent to region-based approaches.



Figure 7.4: Example application of constructing five partition classes for the fracture training image that consists of three different fracture networks with three orientations. The partition classes show spatially scattered regions (models from Boucher (2007)).

One example application of the method proposed by Boucher (2009) for constructing the partition classes is shown in Figure 7.4. It is noticeable that the partitions are vastly scattered in the spatial domain. There is no more a perception of distinct and confined regions for the partition classes. However, the principal training image clearly shows three different fracture networks according to the orientation of the features. Therefore, this training image needs to be partitioned in a manner that can capture the spatial non-stationary variabilities. The key to success, according to Boucher (2009) is finding the right transformation or filters that can correctly partition the training image. For example, for the fracture training image shown in Figure 7.4, an edge detection filter followed by an averaging filter can produce reasonably well partitions. Another training image might need an upscaling of the training image before the application of filter-based partitioning to provide the desired performance. The inability to generalize the technique for different training images is one of the limitations of this method. Each training image needs its own algorithm for a correct partitioning. On the other hand, the method uses a pattern-based approach of filtersim for

partitioning, while the probabilistic method of *snesim* is subsequently used for simulation. The combination of the two overall approaches on multiple-point geostatistical modeling, namely pattern-based and probabilistic-based, is not yet theoretically justified. For example, what is the relation between the size of the pattern used for finding partition classes and the data template used for constructing the search tree? or what are the prevailing effects of constructing search trees for a partition class that is spattered all over the training image (as was shown in Figure 7.4)?

In conclusion, there have been many attempts on modeling non-stationarity in multiple-point geostatistics. All the approaches relied, to some extent, on the concepts of regions, or some additional data that implicitly define regions. In the following sections, a general framework that incorporates the common concept inherent to all previous methods is provided. The framework conceptually embeds the spatial component of modeling into geostatistical simulations. First, in order to understand the fundamental objectives of the proposed methodologies, some questions need to be answered:

- What is expected to be depicted in a training image?
- How should non-stationarity be modeled?
- What are the responsibilities of the modeler and that of the geostatistical algorithm?

These questions will be answered in the next section. Afterwards, three different methods for modeling non-stationarity will be provided that can automatically, and without any additional data, generate the desired Earth models.

7.2 Note on Non-stationary Modeling

Traditional methods for non-stationary modeling can be entirely categorized in the realm of geostatistical modeling. They are merely tools that force the algorithm to provide exactly what the modeler has in mind. For instance, the general concept of providing regions for rotation, affinity, or facies proportion is seen as an external instrument that guides the multiple-point statistical simulation to generate the desired phenomena.

A fluvial deltaic fan reservoir is formed by the deposition of sandstone facies from a source, and an outward radial flow to lower elevations. Previous non-stationary approaches, for example, consider one stationary training image, consisting of only channels, and make an attempt on reproducing the same phenomenon. Providing the exact rotation angle of

the channels, or the width of the channels at each spatial location, is nothing but a clever trick for producing a model that pleases the practitioner.

Multiple-point geostatistics relies on the principle of a training image. The training image is a random function model that governs the features and structures of the subsurface. The modeler's decision on stationarity facilitates the statistical calculations. However, geological processes are rarely stationary. There are elevation changes, sea level fluctuations, sediment transports, barriers, and many more complex phenomena that invalidate the assumption of stationarity. Processes happen sequentially in time, and they depend on the previous physical state of the system. Spatial variation of the properties and heterogeneities are an inevitable characteristic in nature.

Therefore, it is hard to define the stationary components of a model, as required by traditional approaches, separately from the non-stationary ones. Providing a specific training image that can conceptualize each subregion of the reservoir is a subjective task. The interaction and boundaries between the subregions cannot be easily modeled. One should escape from these assumptions and provide a realistic approach that can handle the non-stationarity without any prior assumption. MPS is designed to generate stochastic models, not to *force* a particular behavior. It is awkward to try rotating the features or scaling them, or even changing their proportions gradually from east to west, just to generate a model that looks similar to the non-stationary phenomenon in mind.

It is expected from the geologist, geophysicist, or any expert's knowledge to provide a conceptual model that prescribes the heterogeneities of the subsurface. This is the purpose of a training image; to train or guide. The modeler should provide just the road map for the algorithm; not drive it to the destination. Previous techniques have been stripped from the ability to drive. The modeler takes the responsibility of driving them to their desired destination – a non-stationary realization. We believe that an MPS algorithm should be more relaxed by eliminating the need for auxiliary variables and information that eventually force the algorithm to its destination.

The region-based approaches or the introduction of auxiliary variables, such as probability fields, for modeling non-stationarity can also be interpreted as a tool for generating a training image, and not a model. In other words, all traditional tools try to generate a model that captures the geological concepts and spatial information that the modeler has in mind, hence, it is called a 'training image generator'. The advanced methods for generating

training images consist of defining the shapes, the objects, their interactions, their proportions, their trend, and so on for generating a conceptual model that can later be used as a training image. The situation is similar in traditional non-stationary modeling approaches. The modeler tries to set some pre-defined goals (i.e., proportion trends, orientation grids, etc.) to generate a model. On the other hand, a training image should be seen as the input to the algorithm, without any additional information. One of the main limitations prohibiting such an approach for non-stationary images is the dependency of MPS algorithms on the statistical richness of the images. However, this dependency should be eliminated such that a non-stationary training image can solely result in a non-stationary realization, in the same way that a specific variogram can solely generate a Gaussian model.

Another example that can shed light into improper practices of current non-stationary modelings is to consider the science of modeling faces. Imagine a database of human faces. A proper algorithm will proceed by learning a model of the faces for generation of new faces. For instance, one can apply a kernel PCA or a neural networks approach on the database, and then, use that model to generate new faces. The algorithm is inherently analyzing the spatial relationship between the components of a face to infer new ones. The approach taken in current non-stationary practices can be envisaged by another unorthodox course of action. Instead of automatic generation of faces, they provide additional information. First, they would provide some images of eyes, noses, mouths, and other body parts (equivalent to stationary images), and then, they provide some additional data as to, i.e., the regions that eyes should be placed, the locations admissible for placing a mouth, and so on (equivalent to auxiliary data or probability fields). Finally, their algorithm will try to put the eyes and the nose and the mouth at their pre-defined locations to generate a face. One can clearly see that, even though the method can provide a face, but the approach taken to reach this goal is unsystematic and too indirect.

In spite of tedious modeling paradigms of current practices, there are some intrinsic issues that can make them an extremely challenging task. First, a real geological process has more complexity than can be captured by a stationary process. One may not be able to provide a stationary training image that relates to specific subregions of the field, nor can it model the complex interweaving of stationary processes. For example, imagine a simple model that consists of two sets of channels, one with a north-south direction only in the middle of the field, and the other with an east-west direction passing similarly through the middle of the field (illustrated in Figure 7.5). This non-stationary sand/shale model consists

of two stationary processes. However, traditional approaches need to define four stationary regions: (1) shale region, (2) N-S channels, (3) E-W channels, and (4) interweaving of N-S and E-W channels. Obviously, more complex examples of such geological phenomena, where the modeler is unable to characterize the non-stationarity by the means of defining stationary regions, or probability fields, or rotation fields, exist.

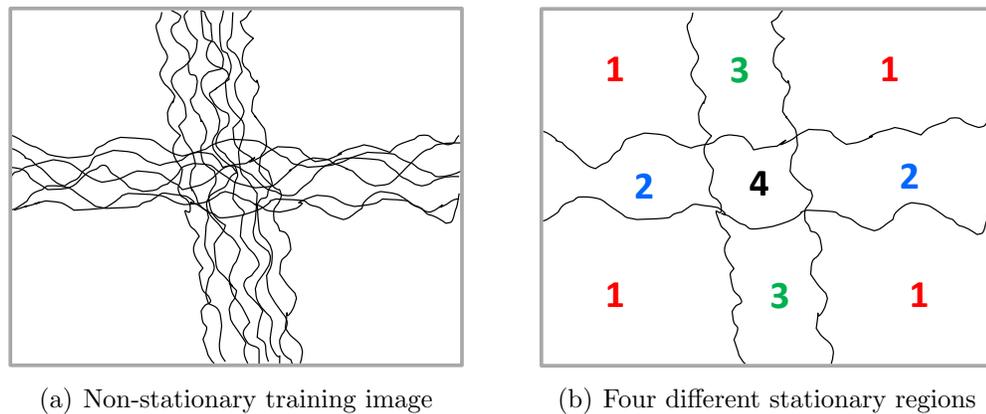


Figure 7.5: A non-stationary training image is shown that consists of two stationary processes of E-W and N-S channels. The subregions on the other hand should define four regions in traditional approaches.

In addition, for situations where non-stationarity can be easily defined through some auxiliary data, no appropriate tool for generating those auxiliary data exists. Imagine partitioning a non-stationary image into stationary parts. One would need a tool to can define those partitions either by computer drawings, hand drawings, or in rare situations analyzing the geophysical or other properties of the field. This is a challenging task that goes beyond the responsibilities of a modeler.

Another critical aspect of current practices is the question of uncertainty. It is both the quality/quantity of available information and the subjective interpretations of an expert about a system that shapes our views on uncertainty. There are many sources of uncertainty in geological modeling of the subsurface. Geostatistics provides us with the tools to assess the inherent spatial uncertainties. Generating different models of subsurface allows us to describe, prescribe, or predict the uncertain behaviors of the geological system. Therefore, it is of utmost importance to account for all causes of uncertainty; being the lack of information or abundance, the ambiguities on the data, or even different subjective beliefs. One should try not to unwillingly limit nor reduce the uncertainty space. This may have a tremendous

effect in predicting the behavior of the geological system. Current practices on modeling non-stationarity impose a certain belief on the system that causes undesirable reduction of uncertainty. More specifically, defining deterministic regions or fixed probability fields forces the algorithm to exactly follow the modeler's decision. A modeler, by defining a set of regions each with its own stationary feature, is basically specifying a set of *certain* boundaries for each region. One then limits the capabilities of the geostatistical algorithm to explore the entire uncertainty space by changing the region boundaries. The subjective decision of the modeler, introduced by the region concept or by the auxiliary information, should be minimized as much as possible. Therefore, one seeks an unconstrained paradigm for non-stationary modeling; one that does not need any auxiliary information.

Therefore, the modeling paradigm should be formulated such that a conceptual non-stationary training image (one that accurately depicts the subsurface geological phenomena) would be the sole input to the MPS methodology, and adequate for generating non-stationary Earth models. The following sections will introduce an approach that satisfies this modeling paradigm by incorporating the spatial components of the training image into pattern modeling. The method removes any dependency on modeler's subjective decisions on how to model the non-stationarity. It simply provides a road map and lets the algorithm drive to the destination.

It is noteworthy to mention that a non-stationary training image *must* depict the modeler's interpretation of the subsurface. It is not the task of the algorithm to understand what the modeler has in mind. The training image is the sole prerequisite for the MPS technique. For example, the modeler cannot provide a non-stationary training image for a small area of the field and expect the algorithm to expand it to fill a much larger spatial domain (of the simulation grid). The stationary principle in traditional geostatistical modeling allows such a difference between the conceptual model and the simulation grid domains. However, in a non-stationary framework, there are no more location-independent moments for the attribute, and one cannot infer conditional probabilities at unsampled locations. If a fluvial fan-deposit is provided as a training image with a square grid, the algorithm cannot figure out how to fill a much larger simulation grid that is no more square-sized. This is shown in Figure 7.6. As can be seen, the realization is twice the size of the training image. Three possible non-stationary realizations are depicted. The main question is which one of these realizations is the desired one? The algorithm cannot make this decision. It is the task of the modeler to provide this guidance by providing a training image that has the same size

as the realization grid. In other words, since the geological features are non-stationary, the modeler needs to define the features beyond the initial domain. The algorithm has no way of knowing how to populate the grid (similar to the situation shown in Figure 7.6). Therefore, the training image should capture the modeler’s perspective about the subsurface. One may argue that the algorithm should be able to accept a set of rules on how to populate the simulation grid at locations that are beyond the training image domain. Although this can be managed, to some extent, by defining rules regarding stretching, repeating, extending, and so on, but the possibilities are infinite. Every modeler may come up with his own understanding on how the algorithm should behave. As a result, the entire spatial domain needs to be conceptually provided to the MPS algorithm for non-stationary modeling.

In conclusion, this section clarified the modeling paradigm for geostatistical simulations of non-stationary processes. The three following sections will provide algorithms that reduce the previous issues, and bring us closer to the desired modeling paradigm. In all three approaches, a notion of “spatial component” of the models will be integrated into the proposed distance-based modeling approach of DisPAT. The introduction of this spatial component, i.e. the coordinates of patterns, relieves the non-stationary algorithm from any dependence on auxiliary variables. The reason for integration of the spatial component is related to the physical behavior of geological systems. If a specific location has experienced a certain geological phenomenon, its surrounding or neighborhood has also undergone the same process. For example, in a fluvial reservoir, if streams of channels are flowing down the hill, at one specific location, one will expect the channels to have similar orientations than to farther away locations. Training image shown in Figure 7.6 is an example where the channels have similar width or direction to the ones in their spatial neighborhoods. This is the main geological concept that will be used throughout this chapter for modeling non-stationarity.

7.3 Spatial-Similarity Method (SSM)

Consider the non-stationary training image \mathbf{ti} . Instead of just storing all the pattern of the training image in the pattern database $\mathbf{patdb}_{\mathbf{T}}$, we will also store their spatial (i, j, k) locations in another database, called location database and denoted by $\mathbf{locdb}_{\mathbf{T}}$. Each row of this database refers to the location of the corresponding pattern in the pattern database. In other words, the pattern database is no more a location-independent matrix. Processing

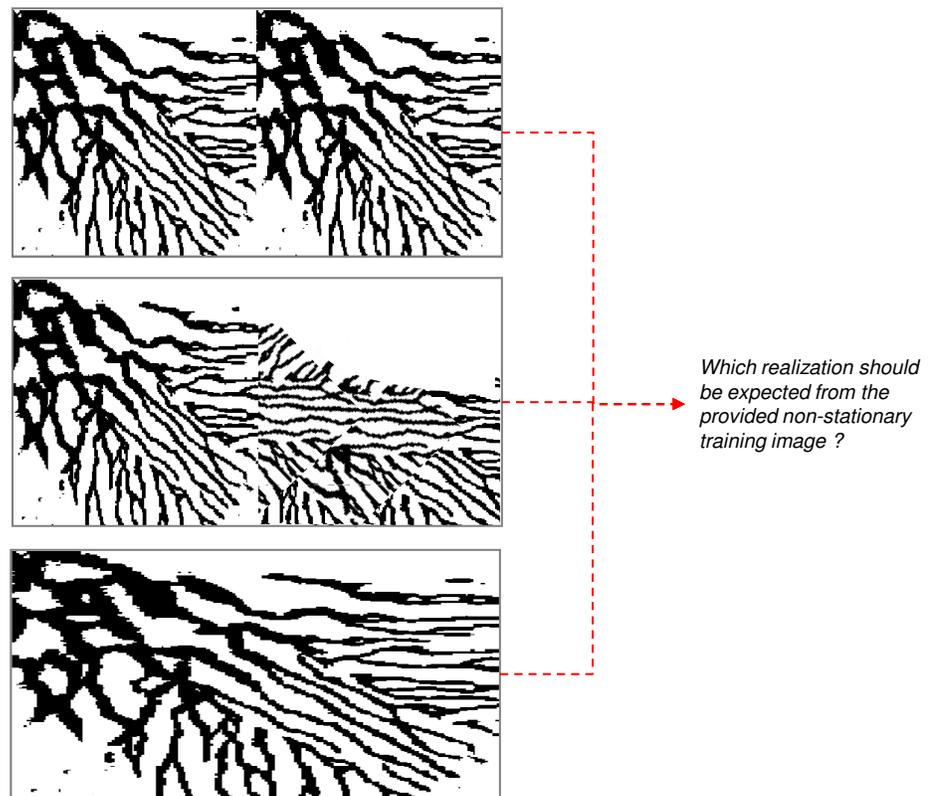
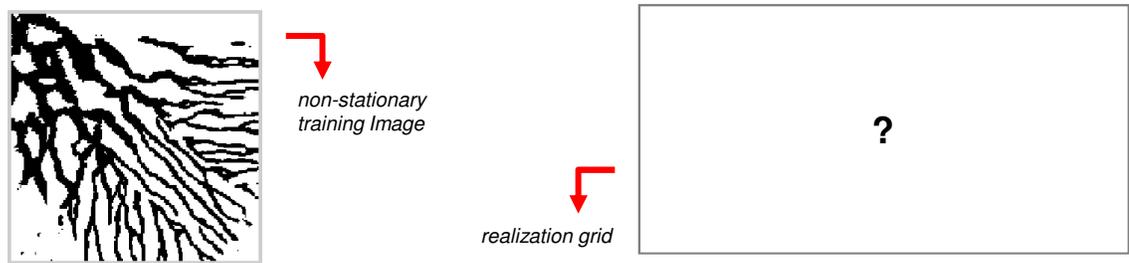


Figure 7.6: An example non-stationary fluvial fan training image with a realization grid with different dimensions is shown. Three different non-stationary simulation results are depicted emphasizing that the algorithm is unable to understand the modeler’s perspective.

the training image thus concerns the extraction of patterns from the training image and storing them with their corresponding locations. Consider the pattern at location \mathbf{u} . The corresponding pattern that is stored in the k -th row of the pattern database, $\mathbf{patdb}_{\mathbf{T}}$, is $\mathbf{pat}_{\mathbf{T}}^k = \mathbf{ti}_{\mathbf{T}}(\mathbf{u})$. In addition to storing the pattern, the k -th row of location database,

$\mathbf{locdb}_{\mathbf{T}}$, is $\mathbf{loc}_{\mathbf{T}}^k = \mathbf{u}$.

Therefore, each pattern is also equipped with a location. The simulation proceeds similar to the unconditional pattern-based technique. A random path is chosen and the nodes of the simulation grid \mathbf{re} are visited sequentially. At each node location \mathbf{u} , the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ is extracted from the realization \mathbf{re} . In order to find the most appropriate pattern to be pasted on the simulation grid, $\mathbf{pat}_{\mathbf{T}}^*$, one will search for similarities between the data event and the patterns within the pattern database $\mathbf{patdb}_{\mathbf{T}}$, and also, the similarities between the location \mathbf{u} and all the locations in the location database $\mathbf{locdb}_{\mathbf{T}}$. The similarity search can be formulated as follows:

$$d_{pat}\langle \cdot, \cdot \rangle = d\langle \mathbf{dev}_{\mathbf{T}}(\mathbf{u}), \mathbf{pat}_{\mathbf{T}}^k \rangle \quad (7.1)$$

for the pattern similarity searches with data events, and

$$d_{loc}\langle \cdot, \cdot \rangle = d\langle \mathbf{u}, \mathbf{loc}_{\mathbf{T}}^k \rangle \quad (7.2)$$

for the similarity searches of the spatial component. $d\langle \cdot, \cdot \rangle$ represents the Euclidean distance between the two corresponding vectors. If we assume the location $\mathbf{u} \in \mathbb{G}_{re}$ to be represented by a three dimensional vector $\mathbf{u} = (i, j, k)$, and the k -th row of location database be represented by $\mathbf{loc}_{\mathbf{T}}^k = (i', j', k')$, then the distances are

$$d\langle \mathbf{u}, \mathbf{loc}_{\mathbf{T}}^k \rangle = \sqrt{(i - i')^2 + (j - j')^2 + (k - k')^2} \quad (7.3)$$

The distances of $d_{pat}\langle \cdot, \cdot \rangle$ and $d_{loc}\langle \cdot, \cdot \rangle$ are calculated for all the rows, $k = 1, \dots, N_{\mathbf{T}}$, of pattern database $\mathbf{patdb}_{\mathbf{T}}$, and location database $\mathbf{locdb}_{\mathbf{T}}$. The corresponding vectors are denoted by \mathbf{d}_{pat} and \mathbf{d}_{loc} respectively. The integration of these two distances, or in other words, the integration of the spatial components into MPS simulation is performed as follows:

$$\mathbf{d}_{ns} = (1 - \omega_{SSM}) \cdot \frac{\mathbf{d}_{pat}}{\|\mathbf{d}_{pat}\|} + \omega_{SSM} \cdot \frac{\mathbf{d}_{loc}}{\|\mathbf{d}_{loc}\|} \quad (7.4)$$

where ω_{SSM} defines the non-stationarity level of the simulation for the proposed *spatial-similarity method* (as will be explained later). Therefore, for each visited node location \mathbf{u} on the realization grid during the simulation, one will calculate the vector of distances \mathbf{d}_{ns} , called non-stationary distance vector (by computing the distances for all rows of $\mathbf{patdb}_{\mathbf{T}}$ and $\mathbf{locdb}_{\mathbf{T}}$ according to Equations 7.1 to 7.4). The optimal pattern $\mathbf{pat}_{\mathbf{T}}^*$ to be pasted

on the realization grid at node location \mathbf{u} is the one with the minimum value in the non-stationary distance vector, \mathbf{d}_{ns} ,

$$\mathbf{pat}_{\mathbf{T}}^* = \mathbf{pat}_{\mathbf{T}}^i = \arg \min_i d_{ns}(i) \quad (7.5)$$

where $\mathbf{pat}_{\mathbf{T}}^i \in \mathbf{patdb}_{\mathbf{T}}$. The simulation then continues until all the nodes of the grid \mathbf{G}_{re} are visited. The application of the multi-grid approach or the multi-resolution approach can be similarly extended by constructing the location database $\mathbf{locdb}_{\mathbf{T}}$ in addition to the pattern database $\mathbf{patdb}_{\mathbf{T}}$ for each multi-scale training image grid.

The idea of including the spatial locations of the patterns into the similarity search causes the patterns to not only be ranked according to their similarity with the data event, but also be ranked according to their spatial closeness to the simulated node \mathbf{u} . In other words, the geological patterns are, most probably, similar to their local surroundings even in a non-stationary training image. The important factor is the way the spatial components are integrated into the modeling framework. The method proposed here, called spatial-similarity method or SSM, provides a smooth and convex distance weighting of patterns of the entire grid area. In other words, all the patterns of the database are included in the similarity search, but they are weighted according to their geometric distances (on the original training image grid) to the location of the visited node (on the realization grid). For a 2D grid, the smooth and convex function of distances that is used to influence the pattern similarities is shown in Figure 7.7.

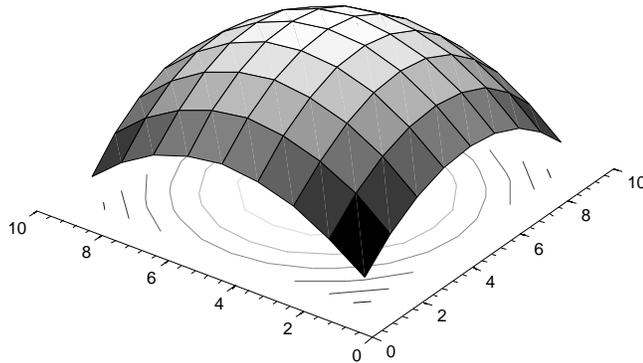


Figure 7.7: The convexity or smoothness of the spatial weighting function used to integrate spatial components into non-stationarity modeling. Patterns similarities change with respect to their spatial proximity with the node being simulated.

An example procedure for a simple training image and one grid node location \mathbf{u} is shown in Figure 7.8. The training image is assumed to be non-stationary. A node location \mathbf{u} on the realization grid \mathbf{re} is being simulated. The data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ is extracted and shown below the realization grid. Two distance vectors are visually depicted in a gray color bar; one for distances between patterns of the training image and the data event, and the other for the distance between location \mathbf{u} and all locations in the location database $\mathbf{locdb}_{\mathbf{T}}$ of the training image. These two distance vectors are combined using $\omega_{\text{SSM}} = 0.55$ according to Equation 7.4. As can be seen in the 2D depiction of non-stationary distance \mathbf{d}_{ns} , the most appropriate pattern is obtained at the position with the minimum distance. This pattern is then pasted on the realization grid, and the simulation continues with the next unvisited node.

The spatial-similarity method does not require any pre-defined regions as in previous approaches. Inherently, the concept of regions exists with a more gradual boundary than a strict one. The region is the entire grid domain without any clear boundary. The stationarity is still an assumption in this method. This assumption is greater in the local neighborhood of the location to be simulated, and slowly fades away as the spatial distance increases. No deterministic regions are selected. In addition, only one training image is required; unlike the previous approaches that needed either auxiliary information or different training images for each region.

The non-stationarity level of the proposed algorithm can be controlled by ω_{SSM} . It is a weight that can take values between zero and one, $\omega_{\text{SSM}} \in [0, 1]$. A value of zero for ω_{SSM} indicates a stationary simulation, since the spatial component is completely excluded from the simulation and only the patterns similarities to the data event are included. On the other hand, a value of one for ω_{SSM} forces the strictest non-stationarity such that all the generated models resemble exactly the training image. In other words, since only the spatial components, and not the patterns, are included in the similarity search, the structures in the training image are copied precisely at same locations on the realization grid. In principle, ω_{SSM} can steer the modeling from pure stationarity to strict non-stationarity. Therefore, a value of ω_{SSM} between zero and one can provide the desired non-stationary models, while at the same time, introduce stochasticity. The spatial randomness and uncertainty can be increased by decreasing the value of ω_{SSM} . In general a value of 0.5 is a reasonable starting point for analysis. Note that the modeler, now, has control over the desired non-stationarity of the models, and can decide on the weight parameter according to the task at hand.

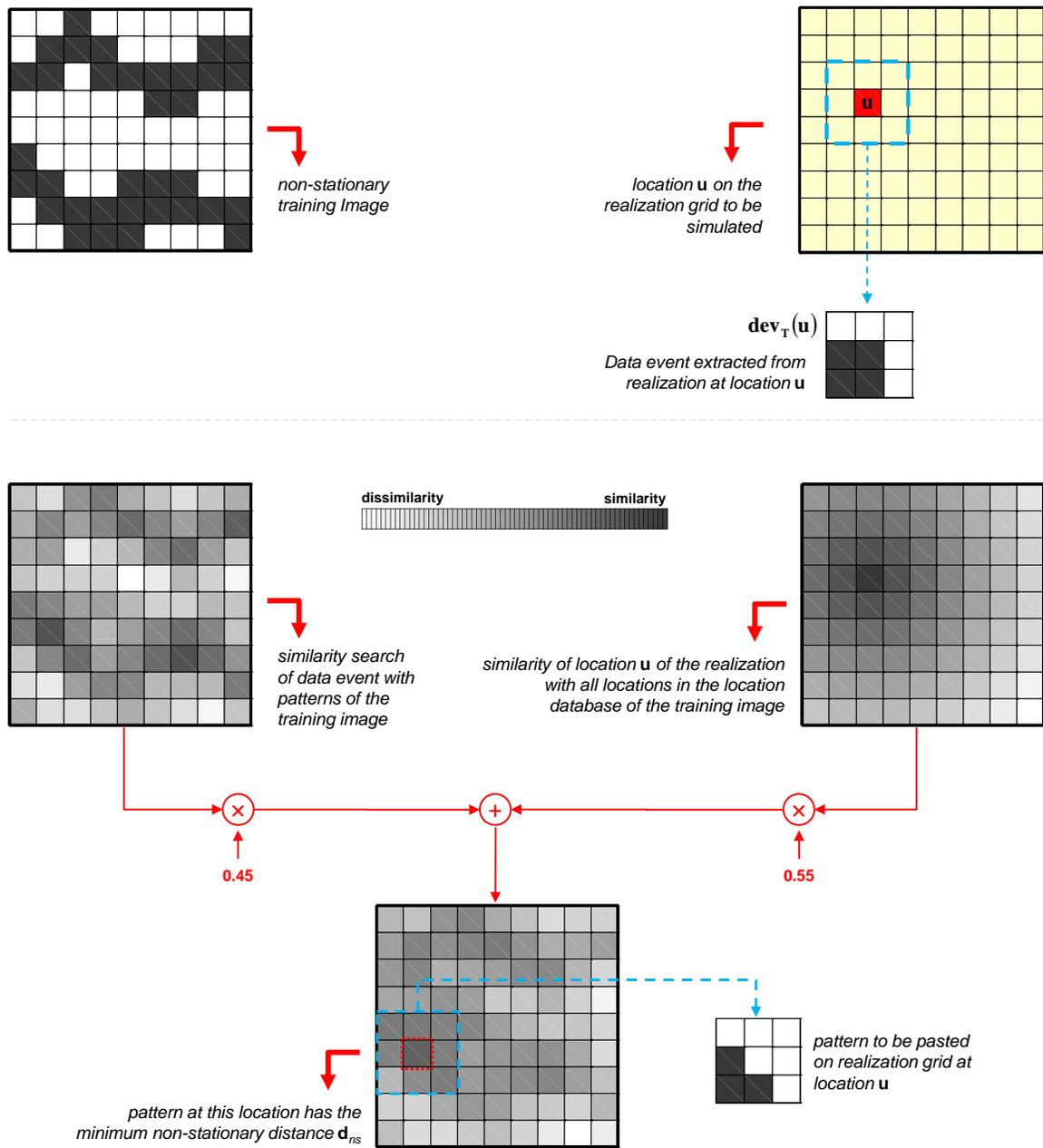


Figure 7.8: Example application of spatial-similarity method (SSM) is provided. A training image and a corresponding node location \mathbf{u} on the realization grid, that is to be simulated, are shown. The two distances for pattern similarity and location similarity are visually illustrated on the original grid G_{ti} . The most optimal pattern, to be pasted on the realization, is obtained from the minimum of \mathbf{d}_{ns} .

An example application of non-stationary modeling with the proposed SSM approach is depicted in Figure 7.9. The same fluvial fan-deposit conceptual training image of size 199×199 is used in this example. The template size is 15×15 , and $\omega_{\text{SSM}} = 0.75$. Four different realizations are shown. It can be observed that the realizations are indeed non-stationary, and there is spatial variability between them. For example, the thick channels originating from the top left corner of the realizations have varying spatial continuities. Moreover, pattern continuity is reasonably preserved in the long range.

Next, the effect of the non-stationarity weight ω_{SSM} on generated models will be investigated. One expects the lower value of ω_{SSM} to provide stationary results, and a value close to one to reproduce strict non-stationary models (similar to the training image). Three values for the non-stationary weight are explored: $\omega_{\text{SSM}} = \{0, 0.5, 1\}$. These three values will show two extreme performances of the method, and another one, with a compromise between those two. The realizations for each ω_{SSM} are shown in Figure 7.10.

Similarly, one can study the effect of ω_{SSM} on the E-type. 200 realizations are generated for each case, and the E-type is computed. Figure 7.11 shows the ensemble averages of the realizations. The E-type corresponding to the weight of zero shows no structure and is flat. However, as the weight increases to 0.1, some structures become apparent in the E-type. Further increasing the weight for spatial components reveals more of the non-stationarity of the training image. Finally, a value of one for ω_{SSM} leads to an E-type similar to the training image itself.

Finally, the process of spatial-similarity method is outlined in Algorithm 11 for reference. The embedding of the spatial components of patterns is the only major difference with a normal unconditional MPS simulation. However, the embedding of spatial components is the sole concept that makes non-stationary modeling feasible.

7.4 Neighborhood-Radius Method (NRM)

The spatial-similarity method (SSM), introduced in previous section, is a general technique that conforms to any non-stationary training image. This was demonstrated with the fluvial fan-deposit training image. However, one can improve the modeling for some specific non-stationary training images.

The main aspect in the proposed non-stationary modeling technique lies in the functional that embeds the spatial components into the simulation. In the SSM method, this functional

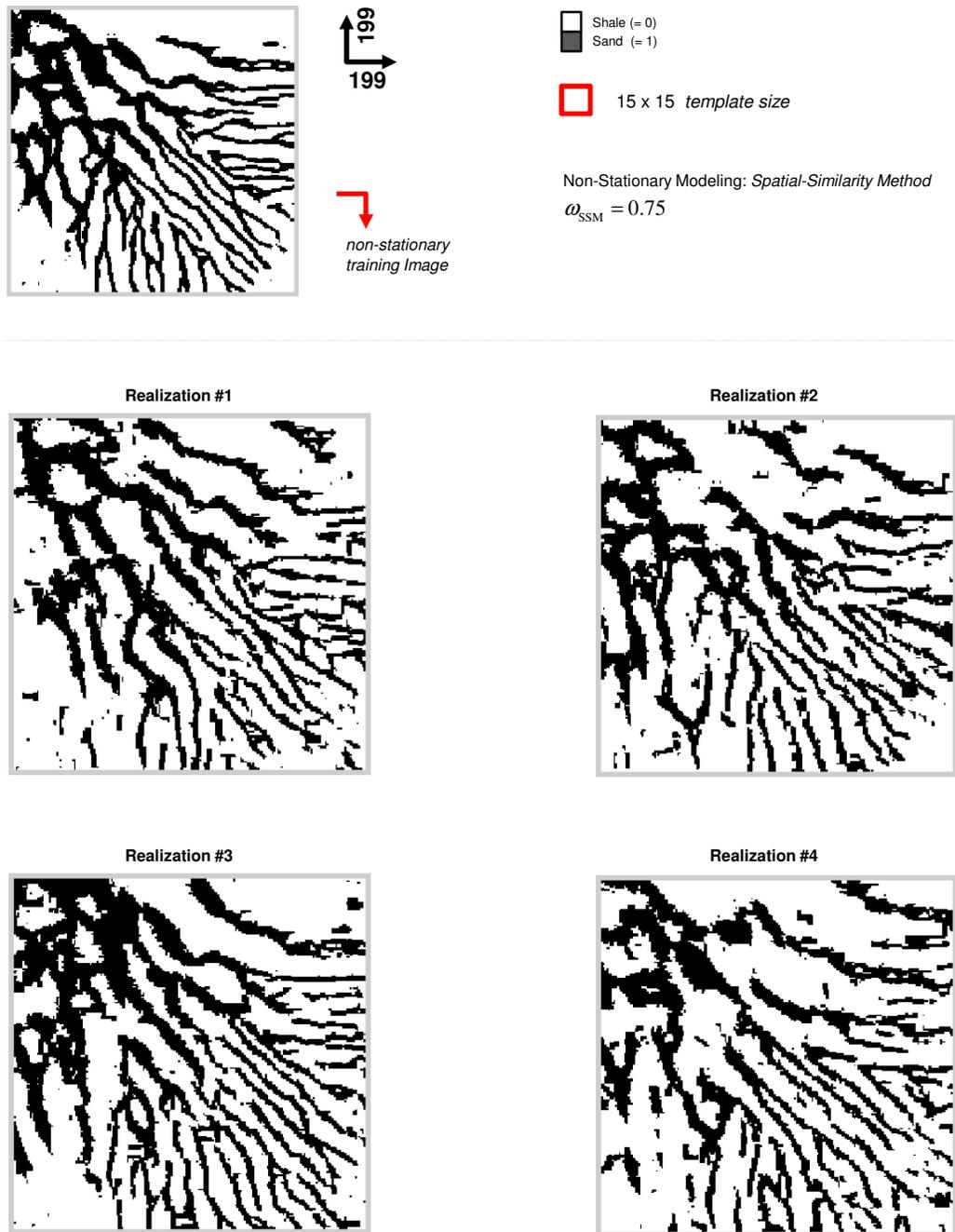


Figure 7.9: Application of Spatial-Similarity Method (SSM) on fluvial fan-deposit training image of size 199×199 . Template size is 15×15 with a multi-grid level of 3. The stationarity level is set to $\omega_{SSM} = 0.75$. Four realizations are shown to demonstrated non-stationarity and stochasticity of the proposed algorithm.

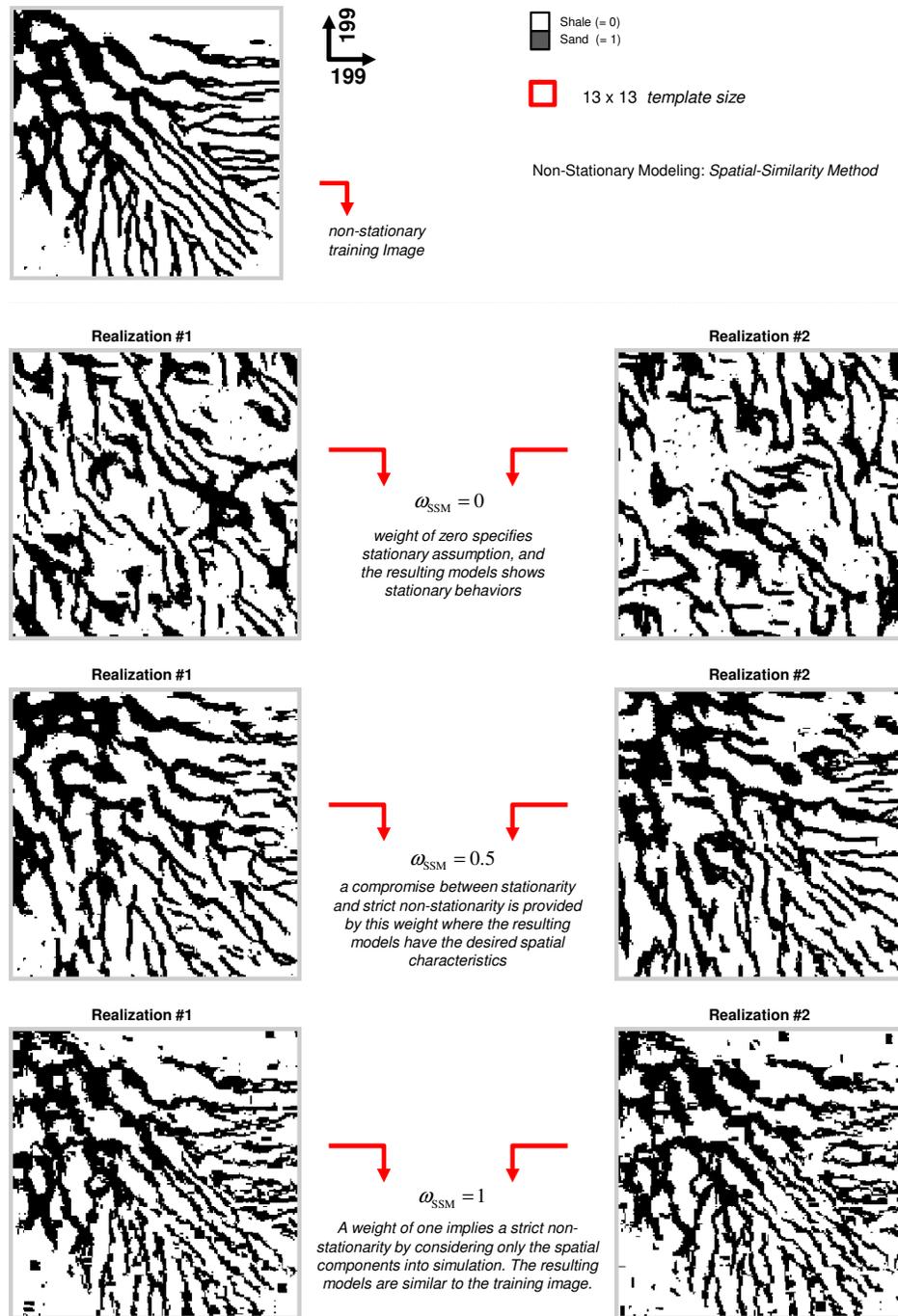


Figure 7.10: Sensitivity analysis of the non-stationarity level ω_{SSM} on the realizations generated by the application of Spatial-Similarity Method (SSM) on fluvial fan-deposit training image of size 199×199 . Weight of zero indicates stationarity, weight of 0.5 provides the desired results, and weight of one forces the exact reproduction of the training image.

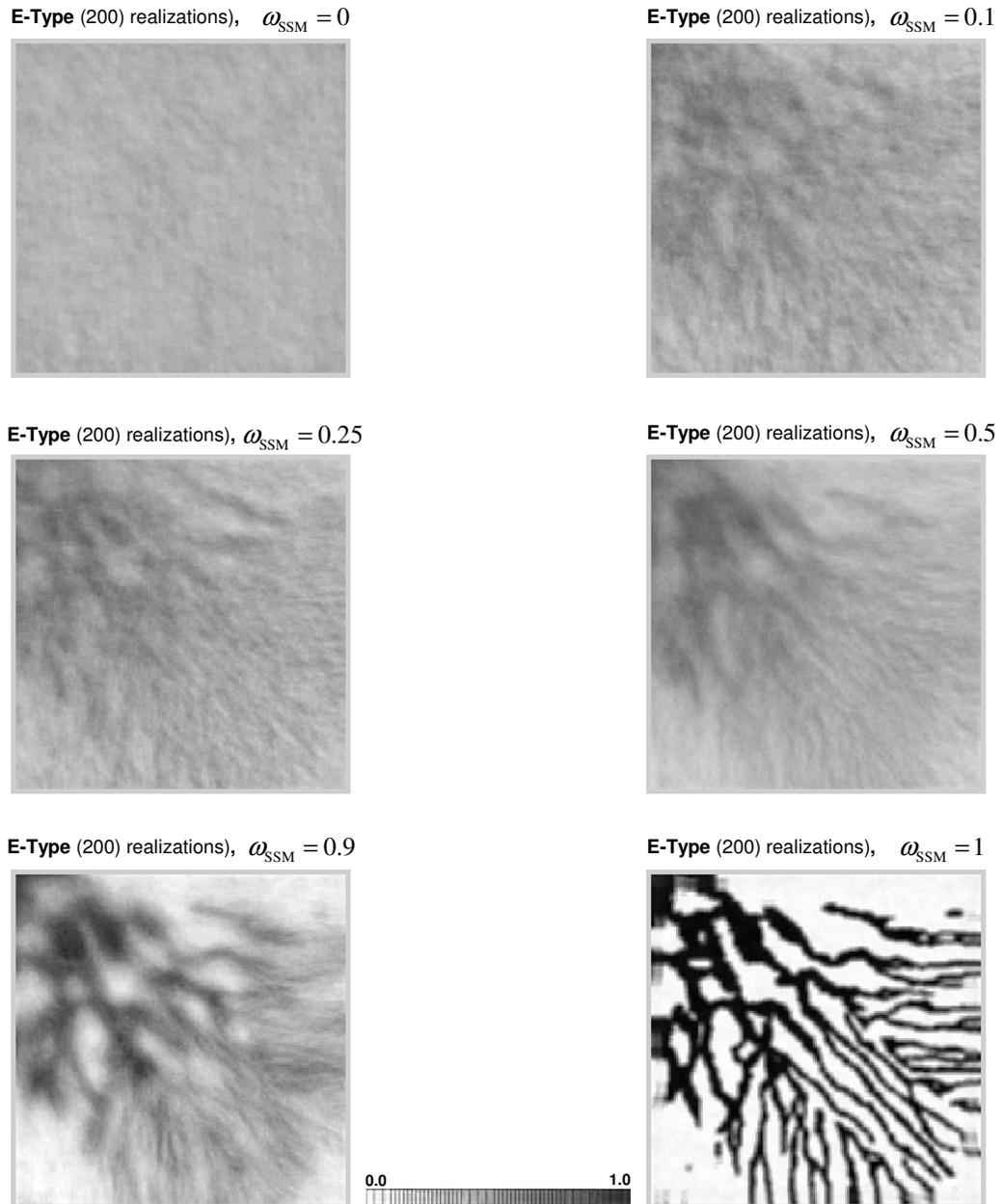


Figure 7.11: Sensitivity analysis of the E-types with respect to the non-stationary weight ω_{SSM} . The training image is the same fluvial fan-deposit shown in Figure 7.10. Higher weights lead to more non-stationarity in the resulting models. A value of zero provides no structural information due to stationary simulation, and the weight of one exactly reproduces the training image due to exclusion of pattern similarity searches.

Algorithm 11 Spatial-Similarity Method (SSM) Non-stationary Simulation

Require: Set template size \mathbf{T}

Require: Set non-stationary weight ω_{SSM}

- 1: Construct pattern database $\mathbf{patdb}_{\mathbf{T}}$
 - 2: Construct pattern database $\mathbf{locdb}_{\mathbf{T}}$
 - 3: Define a random path on the grid \mathbf{G}_{re} of realization.
 - 4: **for** each node \mathbf{u} along the random path **do**
 - 5: **if** $\mathbf{u} \neq \text{frozen}$ **then**
 - 6: Extract the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ from realization \mathbf{re} .
 - 7: **if** all nodes $\in \mathbf{dev}_{\mathbf{T}}(\mathbf{u}) = \text{uninformed}$ **then**
 - 8: Randomly select a pattern $\mathbf{pat}_{\mathbf{T}}^*$ from pattern database $\mathbf{patdb}_{\mathbf{T}}$
 - 9: **else**
 - 10: $\mathbf{d}_{pat} \leftarrow$ calculate the distances of $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ and patterns in $\mathbf{patdb}_{\mathbf{T}}$
 - 11: $\mathbf{d}_{loc} \leftarrow$ calculate the distances of location \mathbf{u} and all locations in $\mathbf{locdb}_{\mathbf{T}}$.
 - 12: compute non-stationary distance: $\mathbf{d}_{ns} = (1 - \omega_{\text{SSM}}) \cdot \frac{\mathbf{d}_{pat}}{\|\mathbf{d}_{pat}\|} + \omega_{\text{SSM}} \cdot \frac{\mathbf{d}_{loc}}{\|\mathbf{d}_{loc}\|}$
 - 13: $i \leftarrow \arg \min_i d_{ns}(i)$, find the index for minimum non-stationary distance.
 - 14: $\mathbf{pat}_{\mathbf{T}}^* = \mathbf{pat}_{\mathbf{T}}^i$
 - 15: **end if**
 - 16: Paste the pattern $\mathbf{pat}_{\mathbf{T}}^*$ on the realization \mathbf{re} .
 - 17: Freeze the nodes within the central inner patch neighboring location \mathbf{u} .
 - 18: **end if**
 - 19: **end for**
 - 20: **return** realization \mathbf{re} .
-

leads to a mapping from the spatial domain into a continuous weight value (as shown in Figure 7.7). The advantage of this functional is manifested in training images where the stationarity is hard to define; particularly, when distinctive regions cannot be explicitly recognized by the modeler. For example, imagine a training image where the features are gradually changing direction, while at the same time, new objects are slowly appearing and becoming larger. Defining regions for such a training image is impractical. The proposed functional in SSM method easily handles these complex non-stationarities. More examples will be provided in following sections.

However in some cases, the proposed functional limits the accuracy of pattern similarity comparisons by demoting the nearby patterns. It is believed that some geological phenomena can be assumed locally stationary. In situations where the local stationarity spreads over larger areas and with fewer gradual transitions, the impact of the SSM functional for integrating the spatial component can inherently decrease the admissible number of patterns for similarity searches. In other words, let us assume that a training image consists of two distinct stationary regions. The proposed SSM method easily handles such a training image. However, for each stationary region, instead of weighting the patterns according to their spatial distance to the simulated location, one can compare the data event invariably to all the patterns within that region. There are more patterns that fit the stationary assumption, and are thus admissible in the similarity search. The SSM method, on the other hand, intrinsically limits the search to the neighborhood of the location that is being simulated.

This situation happens only for the training images that can be partitioned into stationary regions. The Neighborhood-Radius Method (NRM) relaxes this assumption by providing another functional for incorporating the spatial components into the simulation. The main goal is to remove the spatial weights on the patterns, but instead, limit them to a specified radial range surrounding the simulated node. Fewer patterns will be compared with the data event, but at least, they will all ignore their spatial components. So there would be no loss of accuracy in the data event similarity searches caused by the spatial weights.

As an example of this issue, consider a non-stationary simulation that is currently visiting the node location $\mathbf{u} \in \mathbb{G}_{re}$. Consider three patterns on the training image: $\mathbf{pat}_{\mathbf{T}}^1 = \mathbf{ti}_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_1)$, $\mathbf{pat}_{\mathbf{T}}^2 = \mathbf{ti}_{\mathbf{T}}(\mathbf{u} + 2\mathbf{h}_1)$, and $\mathbf{pat}_{\mathbf{T}}^3 = \mathbf{ti}_{\mathbf{T}}(\mathbf{u} + 3\mathbf{h}_1)$, at increasingly farther locations $\mathbf{u} + \mathbf{h}_1$, $\mathbf{u} + 2\mathbf{h}_1$, and $\mathbf{u} + 3\mathbf{h}_1$ on the training image, correspondingly. In addition, assume

that the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$, extracted from the realization grid at node \mathbf{u} , is more similar to pattern $\mathbf{pat}_{\mathbf{T}}^3$ than to the rest. However, since this pattern is spatially farther from the rest, the calculated d_{ns} distance may be higher than another less similar candidate pattern; i.e., $\mathbf{pat}_{\mathbf{T}}^1$ is selected to be pasted on the realization grid even though it is not visually as similar to the data event. Therefore, an inaccuracy in pattern similarity search occurs. This is a rational behavior when the training image is strictly non-stationary. Nevertheless, if the local neighborhood of location \mathbf{u} is within a stationary region, then $\mathbf{pat}_{\mathbf{T}}^3$ should have been chosen for node \mathbf{u} , and any other choice would have decreased the quality of simulation. However, if we could make this assumption that, the spatial neighborhood of \mathbf{u} is stationary up to a radius of $3\mathbf{h}_1$, then one could analyze all the patterns within this neighborhood without any spatial weighting (and ignore all the patterns outside the stationary radius of $3\mathbf{h}_1$). This way, the correct pattern $\mathbf{pat}_{\mathbf{T}}^3$, in terms of similarity, will be selected for the node location \mathbf{u} . Neighborhood-Radius Method (NRM), proposed in this section, implements the same idea for non-stationary modeling. This method is thus more applicable to quasi-stationary training images.

Similar to the SSM approach, the training image \mathbf{ti} is processed, and both the pattern database $\mathbf{patdb}_{\mathbf{T}}$, and the location database $\mathbf{locdb}_{\mathbf{T}}$ are constructed. The pattern database is no more location-independent since the locations of patterns are now stored in the location database. The algorithm then proceeds by visiting all the nodes on the simulation grid. At each node location \mathbf{u} along the random path, the data event is extracted from the realization grid \mathbf{re} . The same procedure of SSM method for pattern selection will be adapted here. However, the functional that incorporates the spatial components changes. Two distances are calculated; one between the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ and all the patterns in the pattern database, and the other between the visited node location \mathbf{u} and all locations in the location database. These are the same formula used in SSM approach; repeated here for clarity:

$$d_{pat}\langle \cdot, \cdot \rangle = d\langle \mathbf{dev}_{\mathbf{T}}(\mathbf{u}), \mathbf{pat}_{\mathbf{T}}^k \rangle \quad (7.6)$$

for the pattern similarity searches with data events, and

$$d_{loc}\langle \cdot, \cdot \rangle = d\langle \mathbf{u}, \mathbf{loc}_{\mathbf{T}}^k \rangle \quad (7.7)$$

for the similarity searched of the spatial component. The functional that is used for obtaining the most appropriate pattern for non-stationary modeling in the NRM method is

provided below:

$$d_{ns}(k) = d_{pat}(k) \times \mathbf{I}_{\omega_{\text{NRM}}}^k(\mathbf{u}) \quad (7.8)$$

where ω_{NRM} defines the quasi-stationarity level of the simulation in the proposed *neighborhood-radius method* (as will be explained later). The indicator variable $\mathbf{I}_{\omega_{\text{NRM}}}(\mathbf{u})$ is defined:

$$\mathbf{I}_{\omega_{\text{NRM}}}^k(\mathbf{u}) = \begin{cases} 1, & \text{if } \|\mathbf{loc}_{\mathbf{T}}^k - \mathbf{u}\| \leq \omega_{\text{NRM}} \times r_{\mathbf{T}} \\ +\infty, & \text{otherwise.} \end{cases} \quad (7.9)$$

where $r_{\mathbf{T}}$ is the radius of the sphere enclosing the template \mathbf{T} , and defined as follows:

$$r_{\mathbf{T}} = \sqrt{n_{\mathbf{T}x}^2 + n_{\mathbf{T}y}^2 + n_{\mathbf{T}z}^2} \quad (7.10)$$

where $n_{\mathbf{T}x}$, $n_{\mathbf{T}y}$, and $n_{\mathbf{T}z}$ are the x , y , and z dimensions of pattern template \mathbf{T} .

The indicator function $\mathbf{I}_{\omega_{\text{NRM}}}^k(\mathbf{u})$ simply results in a value of one for locations in the proximity of the visited node \mathbf{u} , and infinity at distant locations. The proximity is defined through $\omega_{\text{NRM}} \times r_{\mathbf{T}}$, which is basically a multiplier on the template size. For example, a value of one for ω_{NRM} entails a neighborhood equal to the template size in either direction. This neighborhood concept is illustrated in Figure 7.12.

Therefore during the simulation, for each visited node location \mathbf{u} on the realization grid, one will calculate the distance vector \mathbf{d}_{ns} by computing the distances for all rows of $\mathbf{patdb}_{\mathbf{T}}$ and $\mathbf{locdb}_{\mathbf{T}}$ using Equation 7.31. The optimal pattern $\mathbf{pat}_{\mathbf{T}}^*$ to be pasted on the realization grid at node location \mathbf{u} is the one with the minimum value in the distance vector, \mathbf{d}_{ns} ,

$$\mathbf{pat}_{\mathbf{T}}^* = \mathbf{pat}_{\mathbf{T}}^i = \arg \min_i d_{ns}(i) \quad (7.11)$$

Thus, there are two differences between the functionals of NRM and SSM methods; the weighting scheme, or the way the spatial information are incorporated in the algorithm. The NRM method, introduced in this section, does not provide any weight to patterns at different spatial locations. All the pattern-similarity searches remain accurate and intact. On the other hand, the spatial influence of the NRM method is limited to a pre-defined neighborhood of the visited node \mathbf{u} . Only a limited set of patterns are taken into consideration. This is contrary to the SSM method where all patterns within the entire training image domain are included in the analysis. In a quasi-stationary training image, where distinct stationary

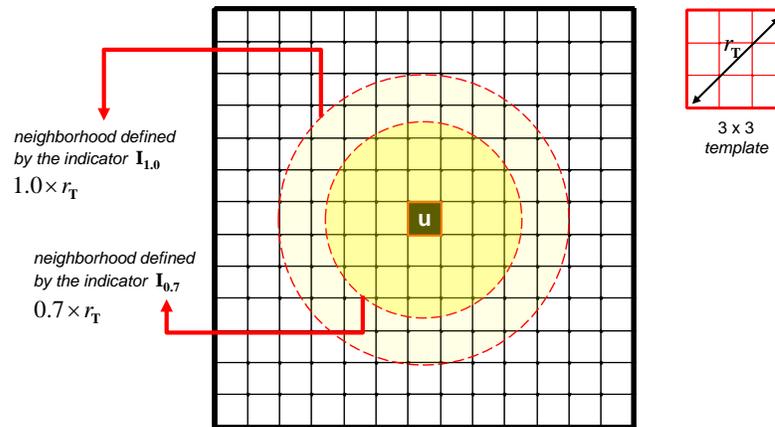


Figure 7.12: The concept of neighborhood-radius defined by $\omega_{\text{NRM}} \times r_T$ used in the indicator functional of $\mathbf{I}_{\omega_{\text{NRM}}}(\mathbf{u})$. Two different spatial weights of 1 and 0.7 are illustrated by their corresponding areas around location \mathbf{u} . Only this neighborhood area will be considered in the NRM method for modeling non-stationarity.

regions exist, this method will result in a more accurate stochastic simulation. Moreover, unlike the continuous functional of the SSM method (shown in Figure 7.7), the functional in this method is discontinuous. Figure 7.13 illustrates the functional that incorporates the spatial components for non-stationary simulations.

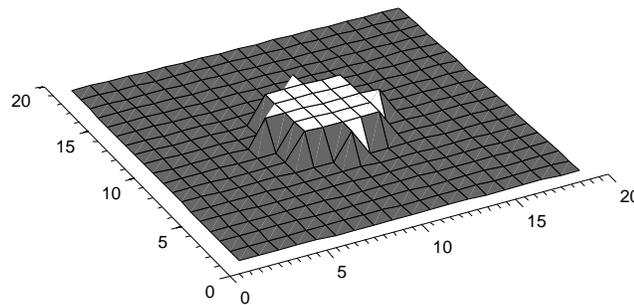


Figure 7.13: The non-smooth indicator functional of neighborhood-radius method for integration of spatial components into non-stationarity modeling. Patterns similarities are included for those in the local proximity of the visited node location \mathbf{u} .

A simple illustration of the NRM non-stationary modeling procedure is shown in Figure 7.14. The training image and the realization grid are shown on top. Two 2D visualizations of the functionals used in the NRM method are also shown in the middle; one representing the pattern similarities to the data event, and the other the indicator functional $\mathbf{I}_{\omega_{\text{NRM}}}(\mathbf{u})$ (where the white region denotes a value of one, and the black region, a value of infinity). The final non-stationary distance vector is obtained by point-wise multiplication of the two visualizations. The minimum value of the resulting distances indicates the most appropriate pattern to be pasted on the simulation grid.

The simple procedure in the neighborhood-radius methodology can be mentally visualized as a variant of the region-based approach. In traditional region-based non-stationary modeling techniques, the training image is partitioned into different stationary regions. Each region is simulated with the conventional stationary assumption. The NRM method is similar; in a sense that, each location belongs to a stationary region which is automatically set by a user-defined neighborhood. Each node location, thus, belongs to a distinct region. There are as many intersecting regions as nodes on the grid \mathbf{G}_{re} . One advantage of having many regions is that the uncertainty will not be reduced, contrary to the reduction caused by fixed region boundaries in traditional approaches. Not only do the regions change from one node location to the other, but due to the random path, the stochasticity will be preserved.

The non-stationarity level in this method is controlled by the weight ω_{NRM} . This weight can take any positive value, $\omega_{\text{NRM}} \in (0, +\infty)$. Low values, close to zero, indicate a small stationary neighborhood around each location \mathbf{u} . This entails a very strict non-stationary modeling of the training image; one that leads to an exact reproduction of the training image. For example, a value of $\omega_{\text{NRM}} \rightarrow 0$ is basically limiting the search to only $\mathbf{pat}_{\mathbf{T}}^* = \mathbf{ti}_{\mathbf{T}}(\mathbf{u})$; namely the pattern located at the same location on the training image. In other words, the data events are simply replaced with their spatial corresponding patterns on the training image. On the other hand, larger values for ω_{NRM} relax the strict non-stationary assumption. A large weight entails larger stationary regions ($\omega_{\text{NRM}} = 3$ is a good default value). The more the stationarity of the geological process, the larger the weight should be. However, there is a maximum value for this weight, which upon reaching, the simulation is turned entirely to a stationary one. The sensitivity on this parameter will be investigated later.

First, we investigate the application of neighborhood-radius method (NRM) on the same

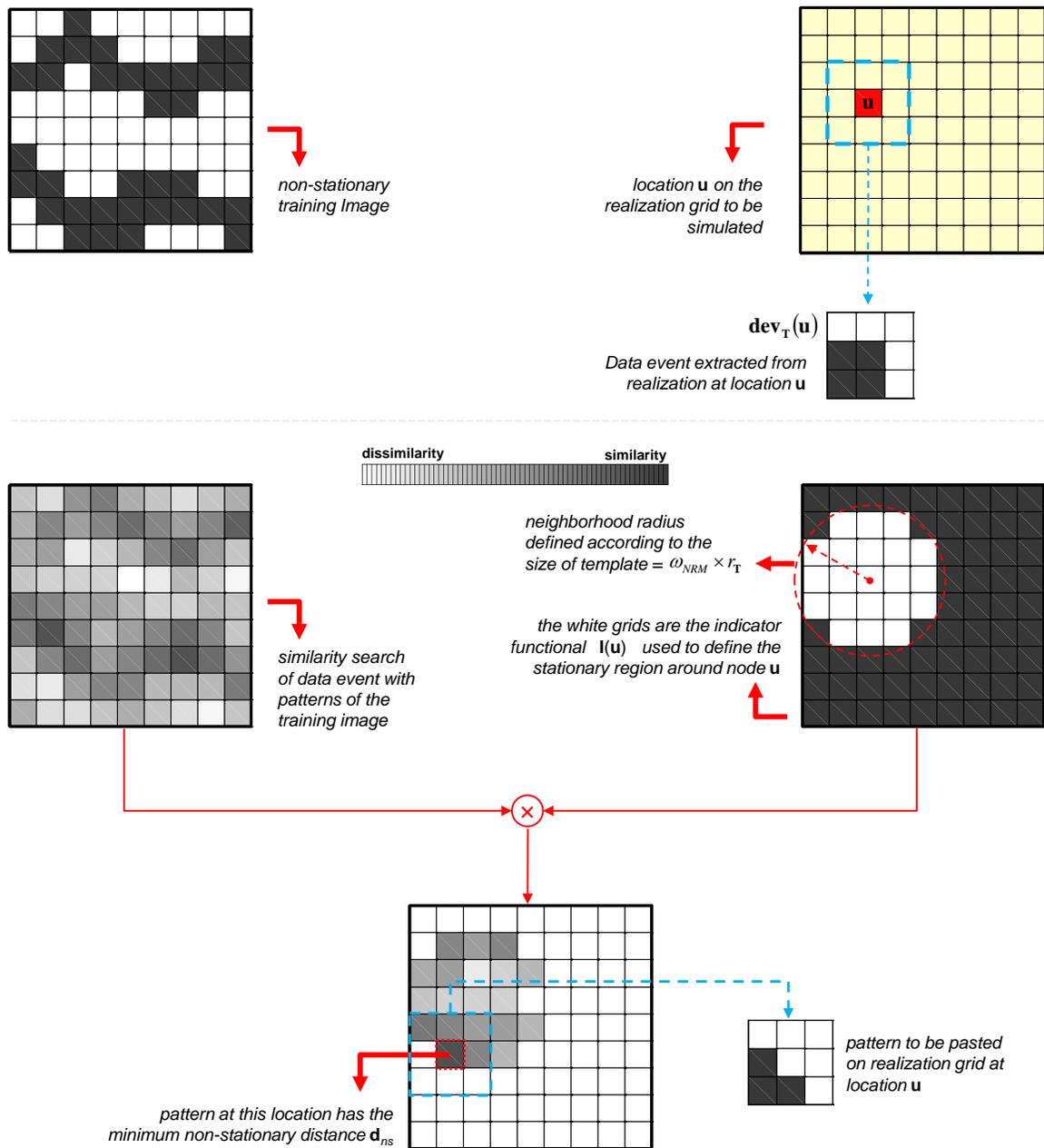


Figure 7.14: Example application of neighborhood-radius method (NRM) is provided. A training image and a corresponding node location \mathbf{u} on the realization grid, that is to be simulated, are shown. The distance for pattern similarity and indicator function $I_\omega(\mathbf{u})$ for location similarity are visually illustrated on the original grid \mathbf{G}_{ti} . The most optimal pattern, to be pasted on the realization, is obtained from the minimum of \mathbf{d}_{ns} .

fluvial fan-deposit training image of the previous section. Four realizations are depicted in Figure 7.15. One can observe that the proposed approach is able to generate non-stationary models. Furthermore, the region boundaries, intrinsically defined by the algorithm, are not discernible in the realizations, i.e., there are no distortions occurring at the boundaries. In addition, the long-range continuity of the non-stationary channels is preserved in the resulting models.

Next, the sensitivity of the algorithm on the weight ω_{NRM} is investigated. This weight, similar to the previous approach, controls the non-stationarity level of the method. Higher weights will decrease the non-stationarity (and vice versa). Figure 7.16 shows two realizations for each case of the weight, $\omega_{\text{NRM}} = \{0.2, 3, 30\}$. As expected, the weight of 0.2 considerably enforces non-stationarity, and the realizations become almost identical to the training image. A higher weight of 3 provides reasonable models; demonstrating both stochasticity and non-stationarity. However, the non-stationarity will disappear at a much larger weight of 30, and the stationarity assumption prevails. The stationary analysis obtained by $\omega_{\text{NRM}} = 30$ demonstrates an inconsistency with the conceptual geological models of fluvial fan-deposit.

Finally, for analyzing spatial variability between the realizations, one can generate the E-types. Figure 7.17 depicts six different E-types for a variety of weights from high to low. It demonstrates the effect of the weight on non-stationary assumptions inherent to the algorithm. A low weight of 0.2 displays an ensemble average that resembles the training image. As the weight increases, the non-stationarity decreases. A weight of 6 leads to models that have the desired spatial variability of the non-stationary image. On the other hand, a value of 40 for the weight does not display any spatial structures with regards to the location of channels. It is a constant field indicating stationary simulation. Therefore, by controlling the weight, the user is able to reach the desired level of non-stationarity deemed relevant to the geological model under study.

For reference, the proposed neighborhood-radius method (NRM) for non-stationary modeling is summarized in Algorithm 12. Due to the indicator variable $\mathbf{I}_{\omega_{\text{NRM}}}$, the proposed method is more suitable for quasi-stationary training images, where a local stationarity assumption is valid on the training image grid.

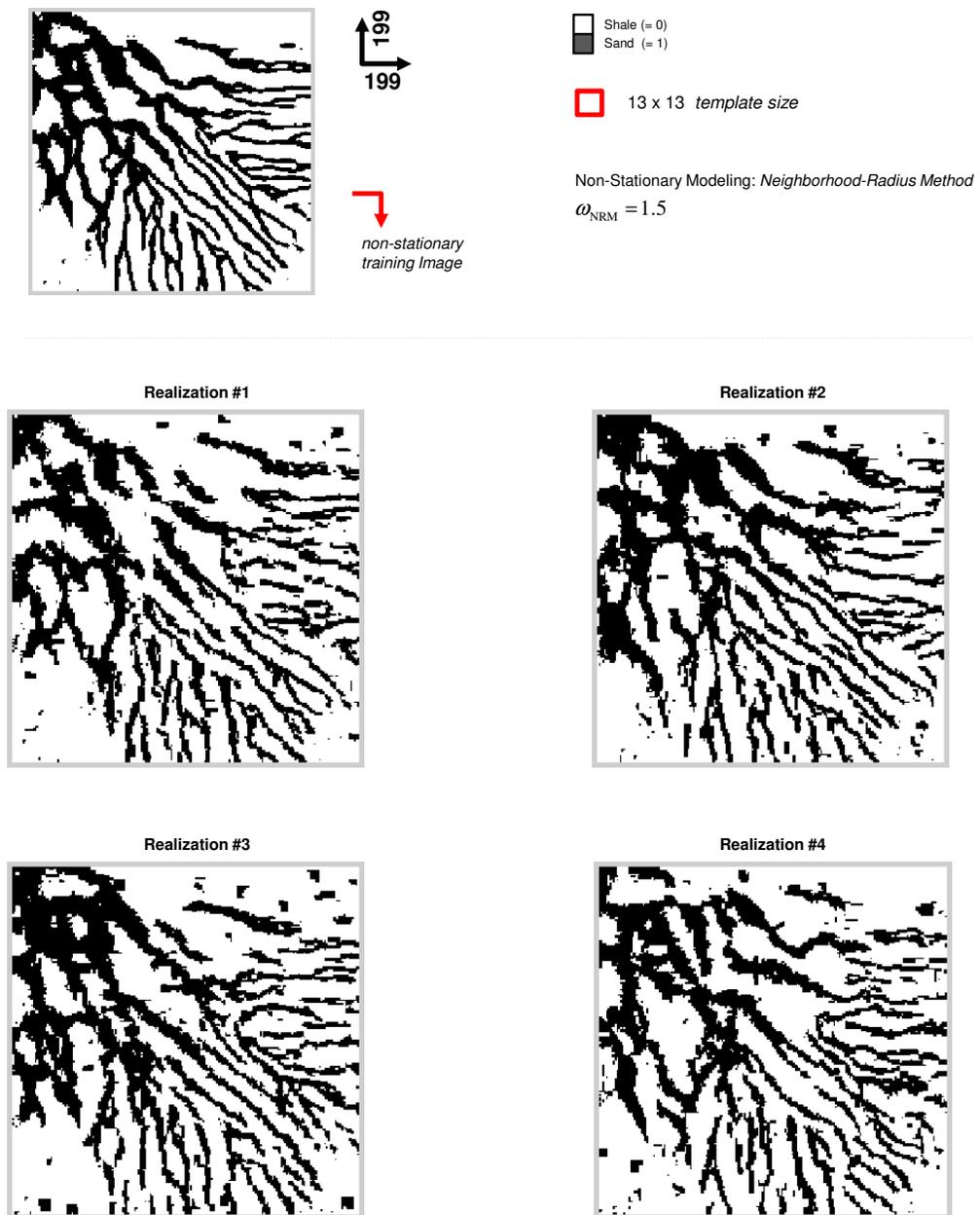


Figure 7.15: Application of Neighborhood-Radius Method (NRM) on fluvial fan-deposit training image of size 199×199 . Template size is 13×13 with a multi-grid level of 3. The stationarity level is set to $\omega_{\text{NRM}} = 1.5$. Four realizations are shown to demonstrate the non-stationarity and stochasticity of the proposed algorithm.

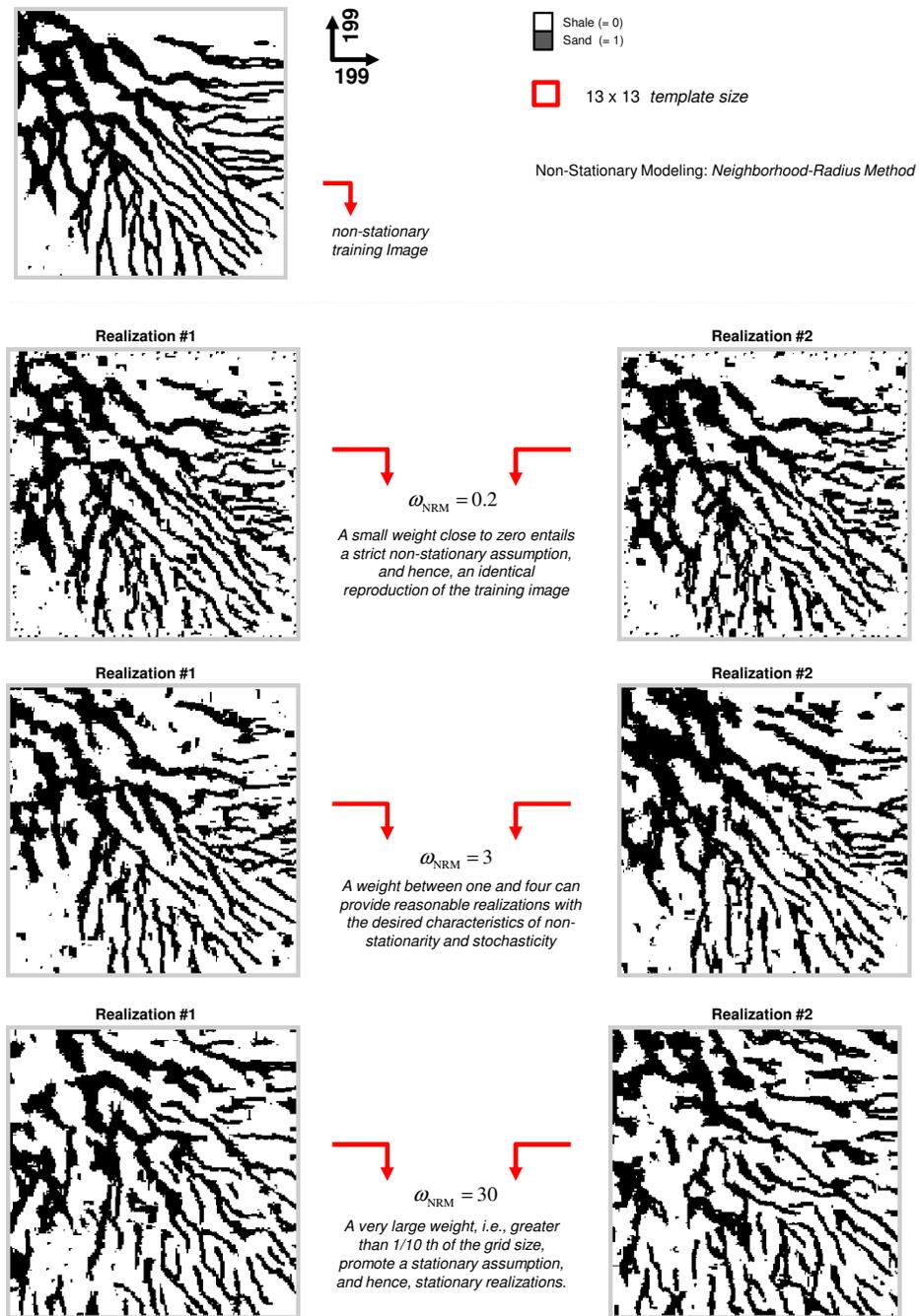


Figure 7.16: Sensitivity analysis of the non-stationarity level ω_{NRM} on the realizations generated by the application of Neighborhood-Radius Method (NRM) on fluvial fan-deposit training image of size 199×199 . Small weights indicate a strict non-stationarity of the training image, i.e., no locally noticeable stationary regions. As the weight increases, the stationarity assumption is reinforced. A large value of 30 is shown to produce stationary results.

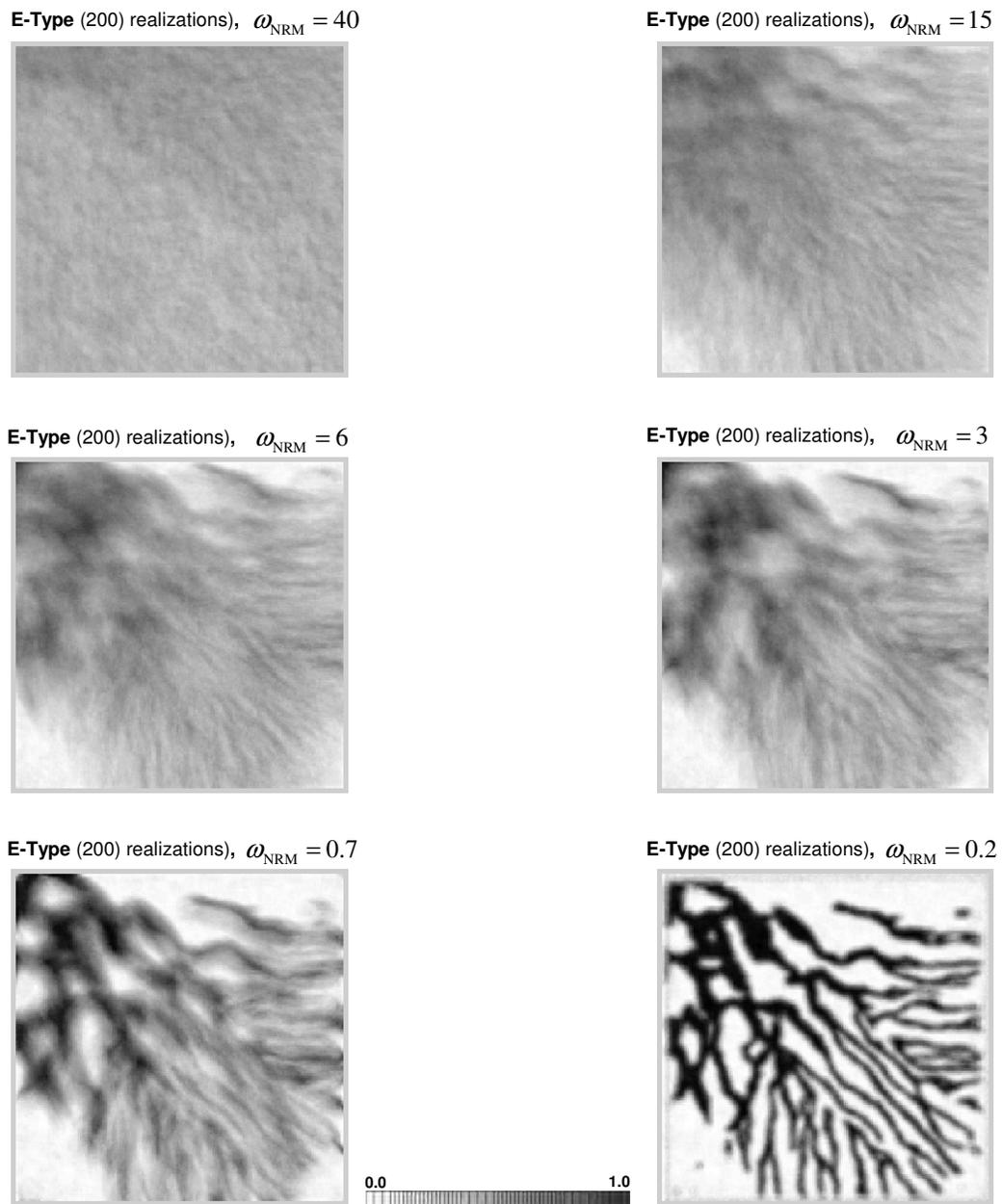


Figure 7.17: Sensitivity analysis of the E-types with respect to the non-stationary weight ω_{NRM} . The training image is the same fluvial fan-deposit shown in Figure 7.16. A template size of 13×13 is chosen for simulation. Lower weights lead to more non-stationarity in the resulting models. A very high value of 40 demonstrates uncertainty in the location of channels due to stationary simulation, and the weight of 0.2 exactly reproduces the training image due to extremely small neighborhood-radius.

Algorithm 12 Neighborhood-Radius Method (NRM) Non-stationary Simulation

Require: Set template size \mathbf{T}

Require: Set non-stationary weight ω_{NRM}

- 1: Construct pattern database $\mathbf{patdb}_{\mathbf{T}}$
 - 2: Construct pattern database $\mathbf{locdb}_{\mathbf{T}}$.
 - 3: compute template radius, $r_{\mathbf{T}} = \sqrt{n_{\mathbf{T}x}^2 + n_{\mathbf{T}y}^2 + n_{\mathbf{T}z}^2}$
 - 4: Define a random path on the grid \mathbf{G}_{re} of realization.
 - 5: **for** each node \mathbf{u} along the random path **do**
 - 6: **if** $\mathbf{u} \neq$ frozen **then**
 - 7: Extract the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ from realization \mathbf{re} .
 - 8: **if** all nodes $\in \mathbf{dev}_{\mathbf{T}}(\mathbf{u}) =$ uninformed **then**
 - 9: Randomly select a pattern $\mathbf{pat}_{\mathbf{T}}^*$ from pattern database $\mathbf{patdb}_{\mathbf{T}}$
 - 10: **else**
 - 11: $\mathbf{d}_{pat} \leftarrow$ calculate the distances of $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ and patterns in $\mathbf{patdb}_{\mathbf{T}}$
 - 12: $\mathbf{d}_{loc} \leftarrow$ calculate the distances of location \mathbf{u} and all locations in $\mathbf{locdb}_{\mathbf{T}}$.
 - 13: compute indicator functional, $\mathbf{I}_{\omega_{\text{NRM}}}(\mathbf{u}) = (\mathbf{d}_{loc} \leq \omega_{\text{NRM}} \times r_{\mathbf{T}})$
 - 14: compute non-stationary distance: $\mathbf{d}_{ns} = \mathbf{d}_{pat} \times \mathbf{I}_{\omega_{\text{NRM}}}(\mathbf{u})$
 - 15: $i \leftarrow \arg \min_i d_{ns}(i)$, find the index for minimum non-stationary distance.
 - 16: $\mathbf{pat}_{\mathbf{T}}^* = \mathbf{pat}_{\mathbf{T}}^i$
 - 17: **end if**
 - 18: Paste the pattern $\mathbf{pat}_{\mathbf{T}}^*$ on the realization \mathbf{re} .
 - 19: Freeze the nodes within the central inner patch neighboring location \mathbf{u} .
 - 20: **end if**
 - 21: **end for**
 - 22: **return** realization \mathbf{re} .
-

7.5 Automatic-Segmentation Method (ASM)

The last two techniques of the spatial-similarity method (SSM) and the neighborhood-radius method (NRM) were based on a pre-defined functional. One of the advantages of such a definition is the generality of these methods on any training image. Be as it may, the NRM method is more suitable for the quasi-stationary training images. If there are stationary regions in the training image, which could not be clearly partitioned, then, the NRM method would implicitly define subregions at each specific location. The NRM method would, indeed, provide more patterns for similarity calculations in each region. The NRM method is analogous to the ordinary kriging. In simple kriging, one has to provide the stationary mean of the distribution of the random variable, but in ordinary kriging, the mean is implicitly estimated at each unsampled location.

However, there are some training images that have visually distinct stationary regions. Although these training images are less prevalent in geological systems, but they need consideration. An example of such an image would be the one where the east half consists of only circles, and the west half of channels. A clear boundary can thus be defined in this training image. In order to take full advantage of all the stationary patterns within each region of this field, a new method, called ‘Automatic-Segmentation Method’ (ASM) is introduced in this section.

The ASM method is similar to the region-based approach, but with four distinctions. First, unlike region-based approaches, it can be applied on any training image, even if the image is strongly non-stationary. Second, the regions can be defined irrespective of the features of the non-stationary training image. Third, partitioning the training image is done automatically, without the need of a practitioner to provide an auxiliary variable. This advantage aids in removing the subjectivity that a modeler may introduce into MPS simulation. There are training images that have no clear boundaries between their stationary regions. It’s better to model such training images with the SSM and NRM methods. Nevertheless, if one insists on using an approach similar to the region-based modeling, then the subjectivity on choosing the region boundaries can be removed by an automatic segmentation of the training image. This technique finds the most optimal boundaries, which guarantees maximum stationarity of the subregions. And finally, similar to the SSM and NRM methods, it incorporates the spatial component into pattern modeling.

The idea behind this approach is to automatically partition the training image into

different regions, and then simulate them individually. The regions are then assumed to be stationary. Therefore, the partitioning should be able to account for different structures and geometries in the provided spatial model (the training image). On the other hand, defining regions entails taking into account the spatial components of the patterns into modeling. In other words, regions have to consist of connected volumes of cells. If the spatial component is not included in the analysis, then only the location-independent patterns are used for partitioning. This would lead to regions being scattered all over the training image (such as the one shown in Figure 7.4).

One solution to an automatic partitioning of a non-stationary training image is to use the distance-based pattern modeling framework. However, the dissimilarity distance function would not only include the patterns, but also their spatial components. This dissimilarity distance function can then be used to construct the distance matrix, and eventually, generate a representation in the Cartesian space using the MDS technique. This representation thus amounts to both the pattern variability and the spatial (contextual) variability. The contextual variability is necessary for having well-defined regions. This is especially true in the Earth sciences, where the geological phenomena gradually transition from one process to another. Therefore, a sense of stationarity prevails in specific areas.

Having obtained a set of points in the MDS space (representing both the pattern variability and the contextual variability), one can apply the kernel k -means algorithm to group them into similar clusters. Each cluster would then define a stationary region of the training image. The reason can be attributed to the fact that each cluster now includes similar patterns that are also located spatially close to one another; hence, a stationary region.

This distance-based modeling framework with the inclusion of spatial components is a powerful technique for most of the training images. However, there are some exceptions that can make it less effective. This is due to the non-stationarity of the training image. Non-stationary means that there is no repetitiveness of the patterns, nor there are any prior information on the expected features at any location. The geometrical sedimentary features can be of any size, any distribution, any shape, and any texture. Not knowing a-priori the essential features within each stationary region, one would not be able to accurately quantify the pattern variability. Strictly speaking, pattern templates cannot be defined a-priori. Due to non-stationarity, local statistics or local properties are not constant over the image. The previous technique of finding the optimal template size does not apply anymore. In other words, there is no specific template that can adequately capture the

entire non-stationarity. The diversity of geometrical features, in these situations, makes it meaningless to give a universal definition of a pattern. This situation is similar to the “Heisenberg uncertainty principle” in physics, which is a direct consequence of the wave-particle duality, that is, a wave describing a particle cannot be accurately measured in both position and momentum (Heisenberg, 1927). In similar terms, for the segmentation of a training image, both the region boundaries (position) and the features (momentum) have to be determined simultaneously. Finding the stationary regions is dependent upon correct feature size identification, and finding the correct feature size is dependent upon an accurate identification of each stationary region.

An example is shown in Figure 7.18. Three distinct regions are evident, one with very wide channels, one with small narrow fractures, and another with a mixture of slim and wide slabs. It can be observed that choosing one template size is not suitable for partitioning/segmenting the training image into stationary regions. Each region demands its own template size for accurate distance-based modeling. For example, the region, consisting of channels, needs a bigger template size than the region consisting of narrow fractures.

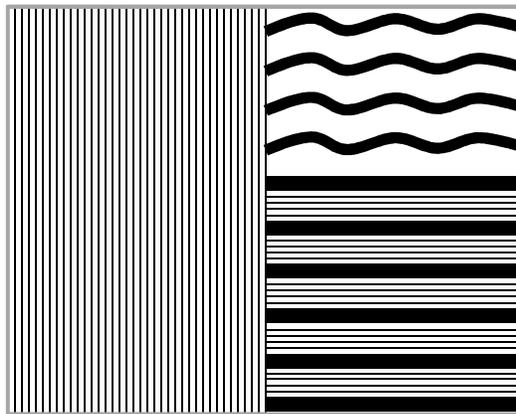


Figure 7.18: A non-stationary training image with three distinct features. Due to the scale differences between the features, there is no single pattern size that can be used to accurately segment/partition this training image.

In order to circumvent the scale differences (or any other non-stationary feature) in the training image, one can apply distance-based modeling, not with just one template size, but with a combination of many template sizes. For example, a 2D training image would be analyzed with an assortment of templates, having sizes (3×3) , (5×5) , (7×7) , \dots , $(30 \times$

30). Incorporating all these patterns in a distance-based framework, with the inclusion of the spatial component, can definitely resolve the issues for identifying the boundaries of stationary regions (the issues mentioned by the uncertainty principle). In this approach, each data in the distance-based framework would consist of a collection of patterns, and also, the spatial components. Assuming the input data matrix to be denoted by X , then, the i -th row vector, \mathbf{x}_i , representing node location \mathbf{u} on the training image \mathbf{ti} , can be formulated as follows:

$$\mathbf{x}_i = \left[\mathbf{ti}_{\mathbf{T}^1}(\mathbf{u}), \mathbf{ti}_{\mathbf{T}^2}(\mathbf{u}), \mathbf{ti}_{\mathbf{T}^3}(\mathbf{u}), \dots, \mathbf{ti}_{\mathbf{T}^n}(\mathbf{u}), u_x, u_y, u_z \right] \quad (7.12)$$

where $\mathbf{T}^1, \dots, \mathbf{T}^n$ represent the different template sizes used for this analysis, and each $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ represents the pattern centered at node \mathbf{u} in the training image \mathbf{ti} . Furthermore, u_x, u_y , and u_z denote the spatial coordinates of node \mathbf{u} . The input matrix, X , would then be used for distance calculations (in constructing the dissimilarity distance matrix and MDS mapping).

This distance-based approach for partitioning a training image into stationary regions is however infeasible due to the high computational complexity of the algorithm. In order to understand the reason, consider a 2D training image of size 250×250 . Also, assume pattern template sizes of $(3 \times 3), (7 \times 7), (11 \times 11), \dots, (47 \times 47)$. There are 12 different scales of analysis such that all possible feature sizes could be accurately captured. Each row vector of matrix X is a vector of size,

$$\begin{aligned} \text{size}(\mathbf{x}_i) &= \left[3^2 + 7^2 + 11^2 + \dots + (4m + 3)^2 \right]_{m=11} + 2 \\ &= \frac{1}{3}(m + 1)(16m^2 + 44m + 27) \Big|_{m=11} + 2 \\ &= 9788 \end{aligned} \quad (7.13)$$

which is a large vector. On the other hand, the number of rows of X is equal to the number of grid nodes of the training image, namely $250 \times 250 = 62500$. Contrary to the proposed MPS framework, one would not be able to use the concept of pattern skipping in this situation, since all training image grid nodes must be included for region-boundary identifications. Hence, the matrix X is of size 62500×9788 . Calculating all pair-wise distances would be computationally prohibitive. For this matter, we will use another powerful approach for automatically segmenting the training image.

The ‘Automatic-Segmentation Method’ (ASM) analyzes the training images by the application of a bank of even-symmetric Gabor filters for characterizing the pattern variability in the training image, and inclusion of the spatial component for the contextual variability. In the following sections, we will introduce Gabor filters and their definitions. Next, the automatic-segmentation methodology will be described. Finally, some examples for modeling non-stationary training images will be provided.

It should be noted that the ASM method will not automatically find the optimal number of stationary regions. It is the task of the modeler to provide the number of desired regions to the algorithm. Although such an automatic calculation would be possible by the technique of Section 5.3, but practically, it allows the modeler to control the non-stationarity level of the training image similar to the SSM and NRM methods. Choosing a value of one for the number of regions would result in one region, and thus, a stationary modeling. Higher number of regions would indicate more regions, and a much stricter non-stationarity.

7.5.1 Gabor Filters

According to the ‘uncertainty principle’ in signal processing, the product of the duration of a signal with its bandwidth cannot be less than a certain value; $(\Delta t)^2(\Delta \omega)^2 \geq \frac{1}{4}$, where Δt is the width of the signal in the time domain and $\Delta \omega$ is the width of the signal in the frequency domain. The uncertainty principle is simply stating that the spread of a signal and its Fourier transform are inversely proportional. Gabor described a function that can analyze signals by simultaneously capturing both the time and the frequency information (Gabor, 1946). This Gabor function is proven to be the most general function that is optimal in terms of this uncertainty principle (reaches the theoretical minimum). It is a Gaussian-modulated sinusoid given as follows:

$$g(t) = \exp\left(-\frac{(t - \mu)^2}{2\sigma^2}\right) \cdot \exp(iwt) \quad (7.14)$$

where σ is the resolution of the analysis, or simply, the spread of the Gaussian function whose center is located at t , and w is the position of the signal in the frequency domain. This function is illustrated in Figure 7.19, where a Gaussian function is multiplied by a periodic sinusoidal function to obtain the Gabor representation.

One of the advantages of this formulation is that any one dimensional function can be represented by a linear superposition of different Gabor functions. However, this is

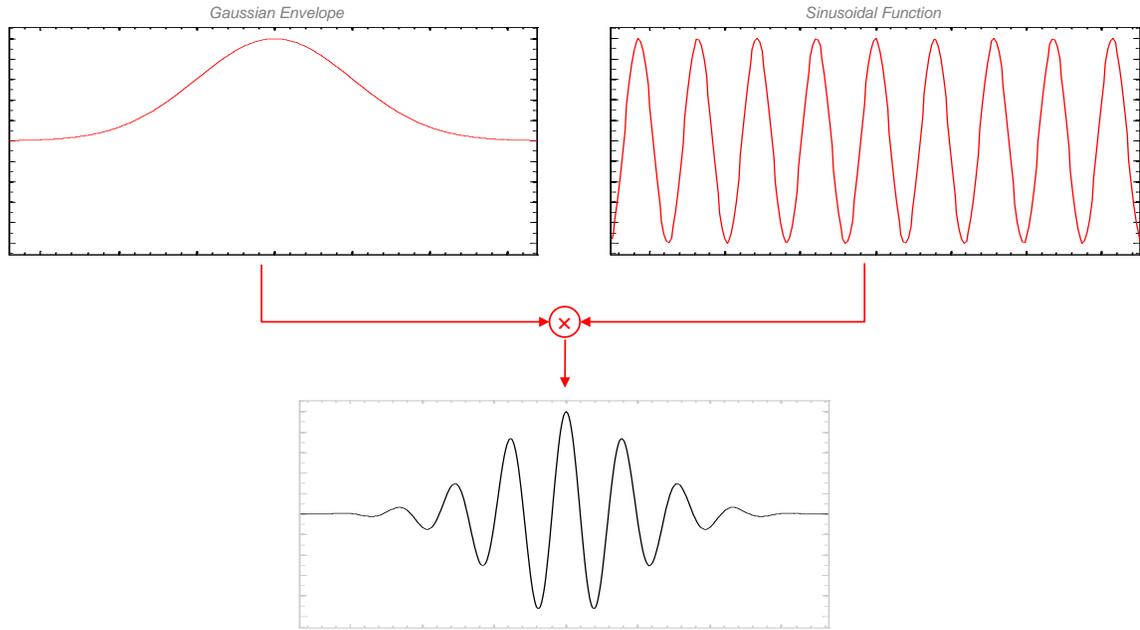


Figure 7.19: An elementary Gabor function obtained by multiplication of a Gaussian envelope, $\exp(-\frac{1}{2}t^2)$, with a sinusoidal function, $\cos(8t)$.

not bounded to one dimensional functions. Daugman (1980) proved Gabor's uncertainty principle for two-dimensional signals (i.e. images). Such Gabor elementary functions can represent two-dimensional information localized in both space and spatial frequency. Similar to 1D, two dimensional signals (images) can also be represented by a bank of Gabor functions (Gabor filters). A 2D Gabor filter in the spatial domain is defined as,

$$g(x, y) = \exp\left(-\frac{1}{2}\left[\frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2}\right]\right) \cdot \exp(i2\pi f x' + \psi) \quad (7.15)$$

where σ_x and σ_y represent the spatial variances of the Gaussian kernel in x and y axes, which basically determine the size of the support of Gabor filters. Also, f denotes the spatial frequency of the sinusoidal function, and ψ its phase frequency (which will be set to zero in our analysis). The parameters x' and y' represent the rotation of the Gabor filter as follows:

$$\begin{aligned} x' &= x \cos(\theta) + y \sin(\theta) \\ y' &= -x \sin(\theta) + y \cos(\theta) \end{aligned} \quad (7.16)$$

where θ is the orientation of the Gabor filter. Figure 7.20 illustrates an example of a Gabor filter in a two-dimensional spatial space.

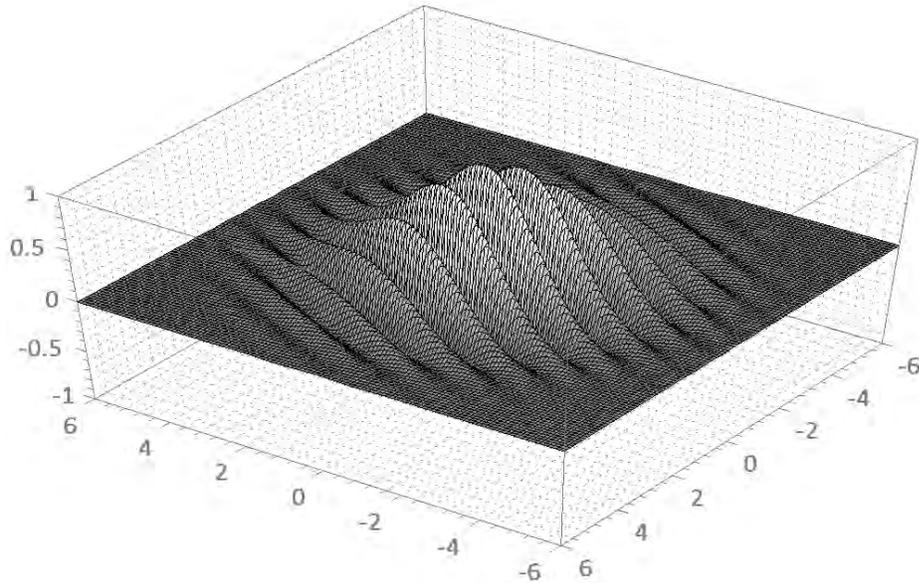


Figure 7.20: The real component of a 2D Gabor function, with parameters $\sigma_x^2 = \sigma_y^2 = 20$, $\theta = \frac{\pi}{6}$, and $f = 1$, plotted for $x, y \in [-6, +6]$.

Besides the advantages of Gabor filters in signal processing, there is considerable evidence on their physiological similarities with the visual processing mechanisms of several mammals, and also, the human spatial visual channels (Marcelja, 1980; Daugman, 1984, 1993). For example, Valois et al. (1982) provided evidence for the receptive fields of Macaque monkey visual cortex cells, and similarly, Jones and Palmer (1987) showed that the cat visual cortex measurements are based on the Gaussian-modulated sinusoids.

Due to such desirable characteristics, Gabor filters have been extensively used in various image and vision processing applications; such as, handwritten character recognition (Hamamoto et al., 1998), face recognition (Ayinde and Yang, 2002), image segmentation (Jain and Farrokhnia, 1991; Dunn and Higgins, 1995), feature detection (Manjunath et al., 1996), edge detection (Mehrotra et al., 1992), and so on.

In order to use Gabor filters for partitioning a non-stationary training image into stationary zones, one needs to suitably choose the corresponding parameters; such as the frequency, the orientation, or the Gaussian spread. Since these parameters are hard to define a-priori, we will use a bank of Gabor filters, similar to Jain and Farrokhnia (1991). A filter bank

generally consists of filters with different frequencies, and orientations. The scale/spread of the Gaussian in filter banks is then set as a constant, relative to other parameters.

For simplicity, in the proposed ASM method, only a set of real-valued even-symmetric filters will be considered due to the psychophysical justifications provided by Malik and Perona (1990). Moreover, the x and y scale of the Gaussian envelope is set to be equal (the circular filter: $\sigma_x = \sigma_y = \sigma$). The formulation of such 2D filters is as follows:

$$g(x, y) = \exp\left(-\frac{x'^2 + y'^2}{2\sigma^2}\right) \cdot \cos(2\pi f x') \quad (7.17)$$

where f is the frequency of the sinusoidal ($f = \frac{1}{\lambda}$), and x' and y' represent the rotated spatial components with orientation θ . The relation between the parameter σ and λ are obtained by setting the half-magnitude spatial frequency bandwidth, $b = \log_2\left(\frac{f_1}{f_2}\right)$, as follows:

$$b = \log_2 \frac{\frac{\sigma}{\lambda}\pi + \sqrt{\frac{\ln 2}{2}}}{\frac{\sigma}{\lambda}\pi - \sqrt{\frac{\ln 2}{2}}} \quad (7.18)$$

or equivalently,

$$\frac{\sigma}{\lambda} = \frac{1}{\pi} \sqrt{\frac{\ln 2}{2}} \cdot \frac{2^b + 1}{2^b - 1} \quad (7.19)$$

and the spatial frequency bandwidth, b , is set to a constant according to the experiments done on simple cells of visual cortex (Pollen and Ronner, 1983). Accordingly, b is set equal to 0.5, and the Gaussian scale σ can thus be obtained from λ .

Now, the only parameters remaining are the frequency f , and the orientation, θ . The combination of different frequencies and orientations generate the bank of Gabor filters, which will be used for training image analysis.

The choice on the orientations in 2D is trivial; the more, the better. In general, four orientations with a separation angle of 45° are sufficient for most training images. However, we use twice more, as follows:

$$\theta : \{0^\circ, 22.5^\circ, 45^\circ, 67.5^\circ, 90^\circ, 112.5^\circ, 135^\circ, 157.5^\circ\} \quad (7.20)$$

Furthermore, the choice on the frequency values, as recommended by Zhang et al. (2002),

is set as follows:

$$f = \{f_L, f_H\} \quad (7.21)$$

where,

$$\begin{cases} f_L(i) &= 0.25 - \frac{2^{i-0.5}}{N} \\ f_H(i) &= 0.25 + \frac{2^{i-0.5}}{N} \end{cases} \quad (7.22)$$

for $i = 1, 2, 3, \dots, \lceil \log_2 \left(\frac{N}{8} \right) \rceil$. Also, $N = \min\{n_x, n_y\}$, where n_x , and n_y denote the sizes of a 2D training image. This definition amounts to $2 \times \lceil \log_2 \left(\frac{N}{8} \right) \rceil$ frequencies. For example, a 250×250 training image leads to eight frequency values.

An example bank of Gabor filters is provided in Figure 7.21. There are 72 different filters (combination of nine frequency values with eight rotation values). Such filters can simultaneously provide fine distinctions among different features in the training image, and accurate identification of region boundaries.

It should be mentioned that the same analysis holds in three dimensions. The corresponding filter can be formulated as follows:

$$g(x, y, z) = \exp \left(-\frac{1}{2} \left[\frac{x'^2 + y'^2}{\sigma^2} + \frac{z'^2}{\sigma_z^2} \right] \right) \cdot \cos (2\pi f x' \sin(\phi) + 2\pi f_z z \cos(\phi)) \quad (7.23)$$

Two additional parameters of f_z and ϕ , respectively for the frequency bandwidth in the vertical direction, and for the orientation with respect to the z axis, are introduced in this 3D notation. Due to lateral depositional characteristics in the subsurface, all 3D filter banks are set to $\phi = 90^\circ$, which is comparable with a zero dip in variogram modeling. The Gaussian spread, σ_z , can also be obtained by using Equation 7.19 (with $\lambda = \frac{1}{f_z}$).

7.5.2 Methodology

The ASM methodology for segmenting a training image into stationary zones involves the application of Gabor filters for discriminating different non-stationary features. It involves three main procedures for segmenting the training image of: (1) applying the Gabor filter banks, (2) energy (feature) extraction, (3) clustering, and at the end of this procedure (4) the non-stationary simulation, which is similar to the NRM method but with a different functional. In this section, we will describe the four main procedures of the ASM

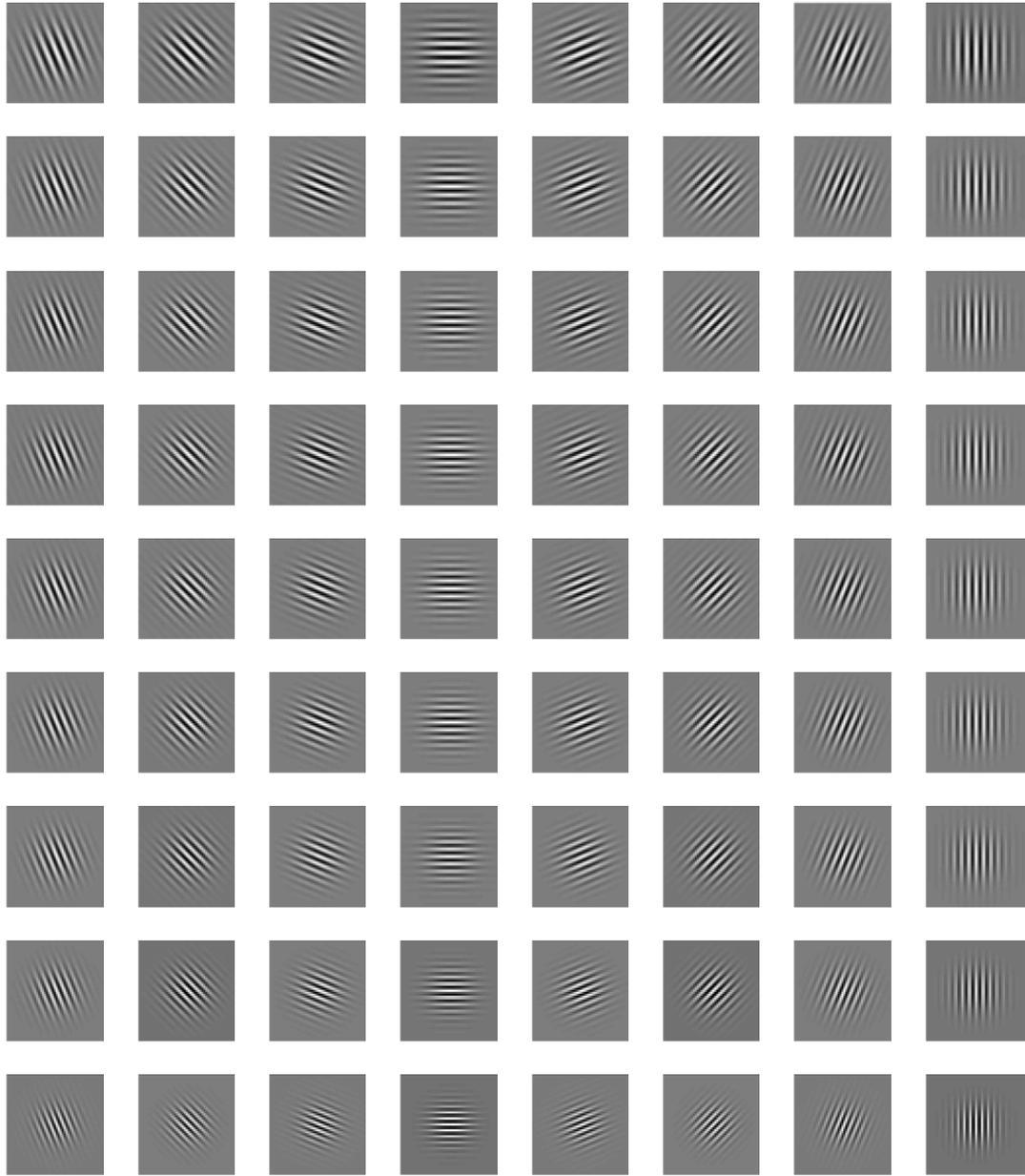


Figure 7.21: A bank of real-valued even-symmetric Gabor filters is shown. Eight different orientations with angular separation of 22.5° are shown horizontally, from left to right. Furthermore, nine different frequency values are shown in a descending order from top to bottom. This set of Gabor filters will be applied individually on the training image to accurately analyze its various features.

methodology with examples.

Applying the Gabor Filter Bank

Having defined the Gabor filter bank $\mathcal{G} = \{g_1, g_2, \dots, g_m\}$ (with m being the number of filters obtained with a combination of orientations and frequencies), they will be individually applied to the training image. The application of each Gabor filter involves the convolution of the filter with the training image, as follows:

$$o_i(x, y, z) = [g_i * ti](x, y, z) = \iiint_{\mathbb{R}^3} g_i(x, y, z) ti(x, y, z) dx dy dz \quad (7.24)$$

where o_i is the filter response obtained from the i -th filter in filter-bank \mathcal{G} , and $ti(x, y, z)$ represents the training image, and $(*)$ is the convolution operator.

Convolving each filter with the training image can be similarly applied by a multiplication in the Fourier domain. However, the only concern in convolution is how to process pixels at the boundary of the image. In the ASM methodology, we will reflect the training image at the boundaries to obtain an image with a bigger size. This allows us to obtain the filter responses at all locations on the training image grid, irrespective to the size of the filters themselves.

Finally, a set of output filtered training images are obtained: $\mathcal{O} = \{o_1, o_2, \dots, o_m\}$ (corresponding to the m initial Gabor filters). Each of the output filtered responses will then be normalized to $[-1, 1]$ to maximize the visibility of the features.

An example application of Gabor filter banks is performed on the non-stationary training image of fluvial fan-deposit (shown in Figure 7.1(a)). Only a limited set of filters and their responses are shown in Figures 7.22 and 7.23. The first figure shows four different orientations with 45° separation angles and a constant frequency of $f = 0.17, \lambda = \frac{1}{f} = 5.81$. The latter, shows the same orientations, but with another higher frequency of $f = 0.33, \lambda = \frac{1}{f} = 3.04$. In both figures, one can notice the ripple-like effect in the filter responses. This ripple structure is due to the sinusoidal component of the filter with positive and negative values.

Energy (Feature) Extraction

Having obtained the set of responses for all Gabor filters, one needs to extract their energy components. Due to the ripple-like structure of negative and positive values, we apply a

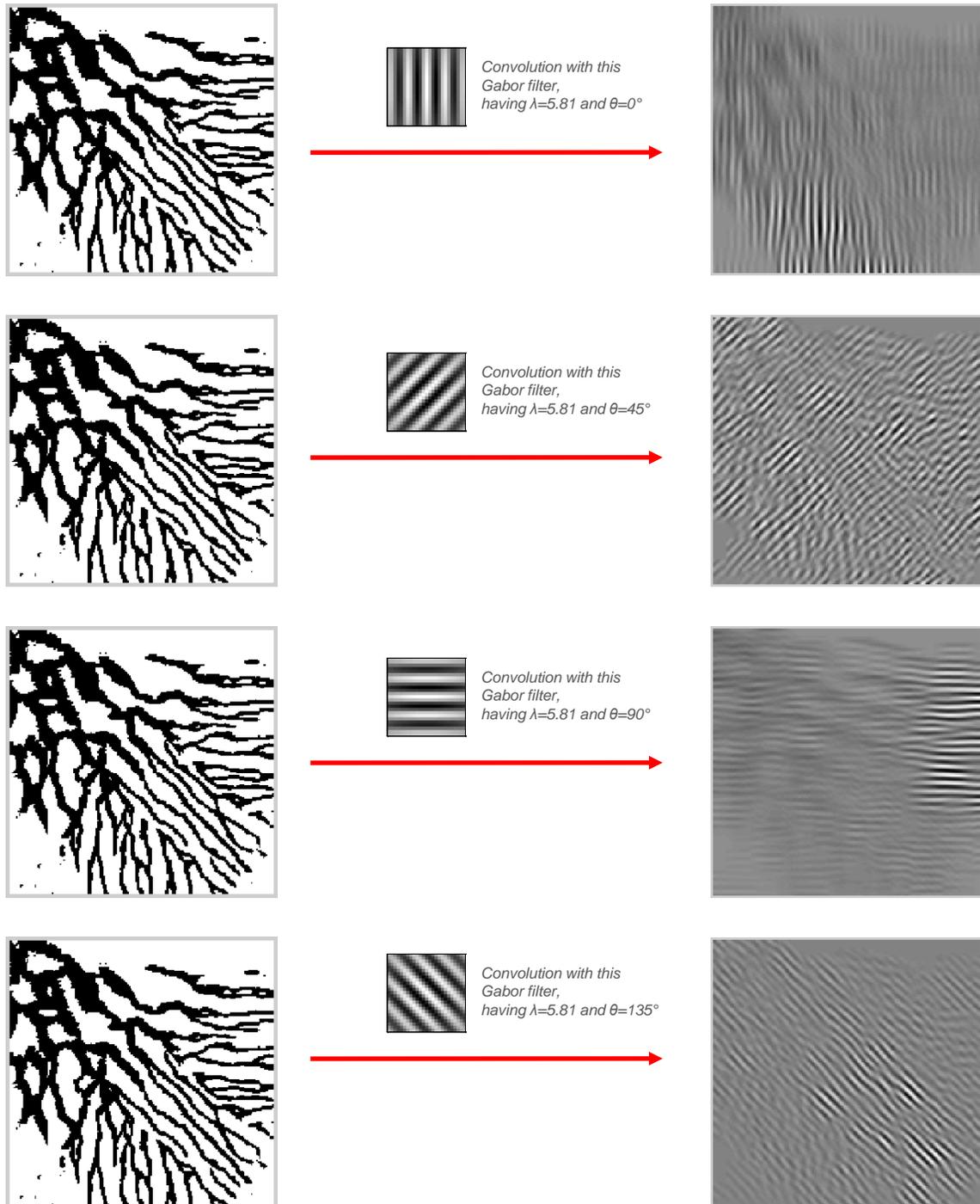


Figure 7.22: Application of four Gabor filters with orientation angles of $\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$, and a constant small frequency of $f = 0.17$. Each filter is distinguishing a different set of features in the training image.

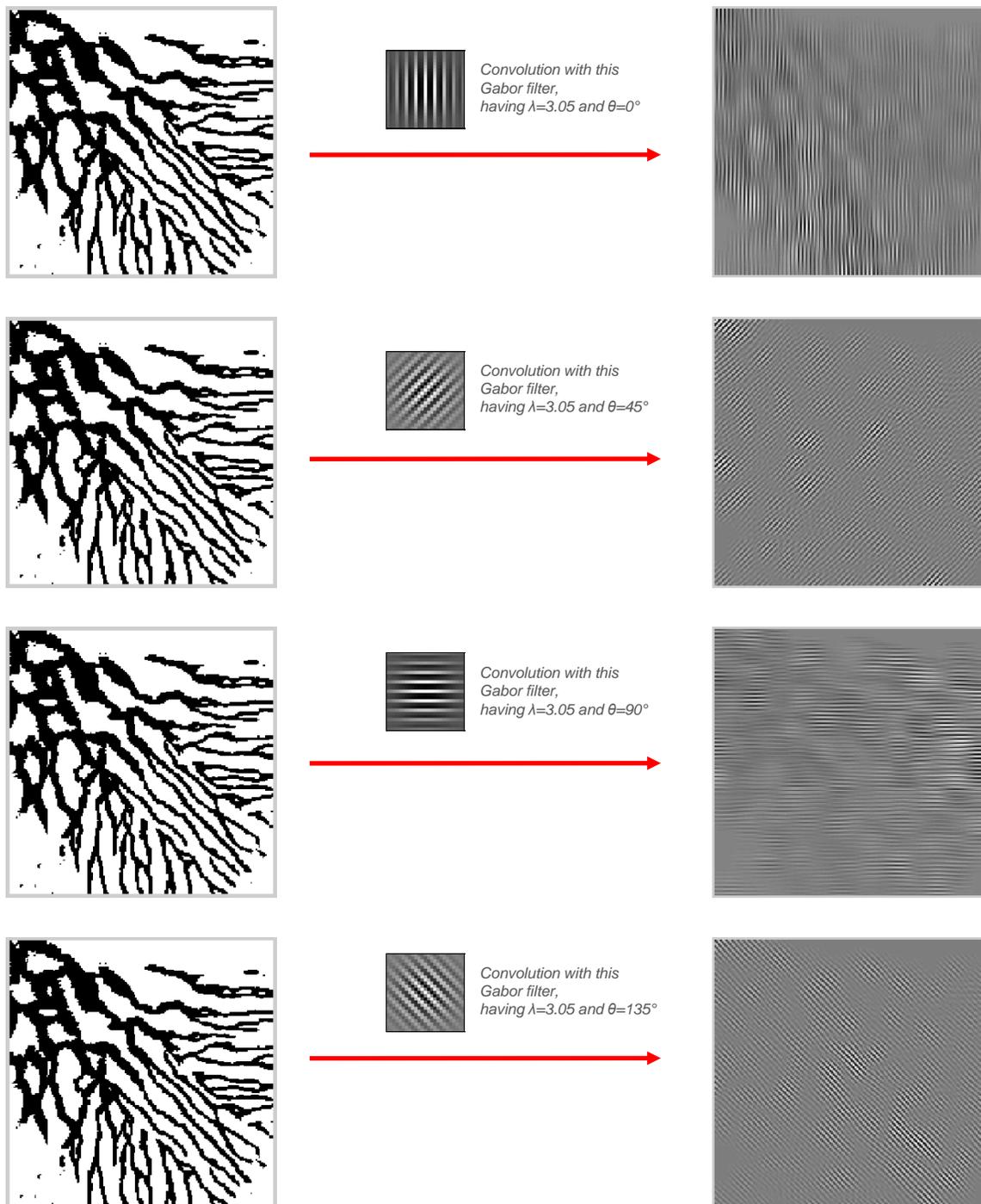


Figure 7.23: Application of four Gabor filters with orientation angles of $\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$, and a constant small frequency of $f = 0.33$. Each filter is distinguishing a different set of features in the training image, but it is more focused on small-scale details and is good for region-boundary identifications.

non-linear transformation on each filtered image, as follows:

$$\psi(t) = \tanh(\alpha t) = \frac{1 - e^{-2\alpha t}}{1 + e^{-2\alpha t}} \quad (7.25)$$

where t is the value of the filtered training image. This sigmoidal function rapidly saturates the filtered responses, and can be interpreted as blob (i.e. rectangles, ellipses, lines) detectors (Jain and Farrokhnia, 1991). In this analysis, we set the value of α to 0.25. The sigmoidal function is shown in Figure 7.24.

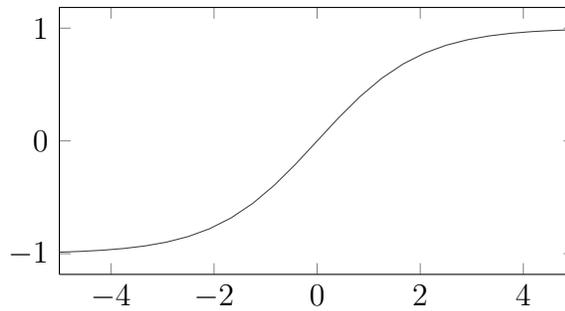


Figure 7.24: The sigmoidal function, $\tanh(0.25t)$, is illustrated, which, upon application, will make all values positive.

This transformation can be formulated for each filtered response, o_i , as follows:

$$o_i(x, y, z) = \tanh \left(0.25 \times o_i(x, y, z) \right) \quad \text{for } i = 1, 2, \dots, m \quad (7.26)$$

After non-linear transformation of the filtered images, we apply a Gaussian smoothing filter in order to extract the local energy for different features. The bandwidth of the Gaussian filter that is used for each result is related to the central frequency of the Gabor filter (also to the spread of the Gaussian filter in Equation 7.23). The Gaussian smoothing filter is computed for 3D, as follows:

$$s(x, y, z) = \exp \left\{ - \left(\frac{x^2 + y^2}{2\sigma_s^2} + \frac{z^2}{2\sigma_{sz}^2} \right) \right\} \quad (7.27)$$

and $\sigma_s = 3\sigma$ and $\sigma_{sz} = 3\sigma_z$, where σ and σ_z are the Gaussian spreads of their corresponding Gabor filters.

Finally, due to the local averaging performed using the Gaussian function, we will normalize all results to the range $[0, 1]$. The application of the energy extraction on some

of the filtered images is shown in Figures 7.25 and 7.26. Each of these local energy images defines a characteristic of the non-stationary training image; such as, the orientation of the channels, or the width of the objects. Specifically, different frequencies correspond to various details observed in the training image. This is clearly observed by a visual comparison of the local energy images between Figures 7.25 and 7.26. The latter figure distinguishes finer details than the former due to the higher frequency values of their corresponding Gabor filters.

The smoothing operation can be shown according to the formula below:

$$o_i(x, y, z) = [o_i * s_i](x, y, z) \quad \text{for } i = 1, 2, \dots, m \quad (7.28)$$

Clustering

The final task in training image segmentation involves two procedures. The first procedure is the integration of the spatial information (the coordinates) needed for a realistic partitioning. The spatial components will ensure adjacency in the final subregions. For a 3D training image, three additional images, each corresponding to a different spatial dimension, will be added to the pool of Gabor filter results. Therefore, each training image is now represented by a set of m local energy images for each filter bank, with the additional three spatial-component images, for x , y , and z directions. For 2D, only two images corresponding to the x and y spatial components are required. Figure 7.27 illustrates these spatial-component images for the previous non-stationary training image example.

It should be mentioned that the spatial components are always normalized to the $[0, 1]$ range for consistency with the results from the application of Gabor filters. However, a weight of 4 is applied for the spatial component due to the differences between the number of filters in a training image and the number of spatial dimensions. This weighting will ensure that spatial adjacency is not lost during clustering. Now, each node of the training image can be represented by a feature vector, as follows:

$$ti(x, y, z) : \left\{ o_1(x, y, z), o_2(x, y, z), \dots, o_m(x, y, z), 4\frac{x}{n_x}, 4\frac{y}{n_y}, 4\frac{z}{n_z} \right\} \quad (7.29)$$

where n_x , n_y , and n_z denote the sizes of the training image in x , y , and z directions.

The second procedure for non-stationary training image segmentation is to cluster the nodes into stationary regions. The clustering is applied on the feature vectors, defined for

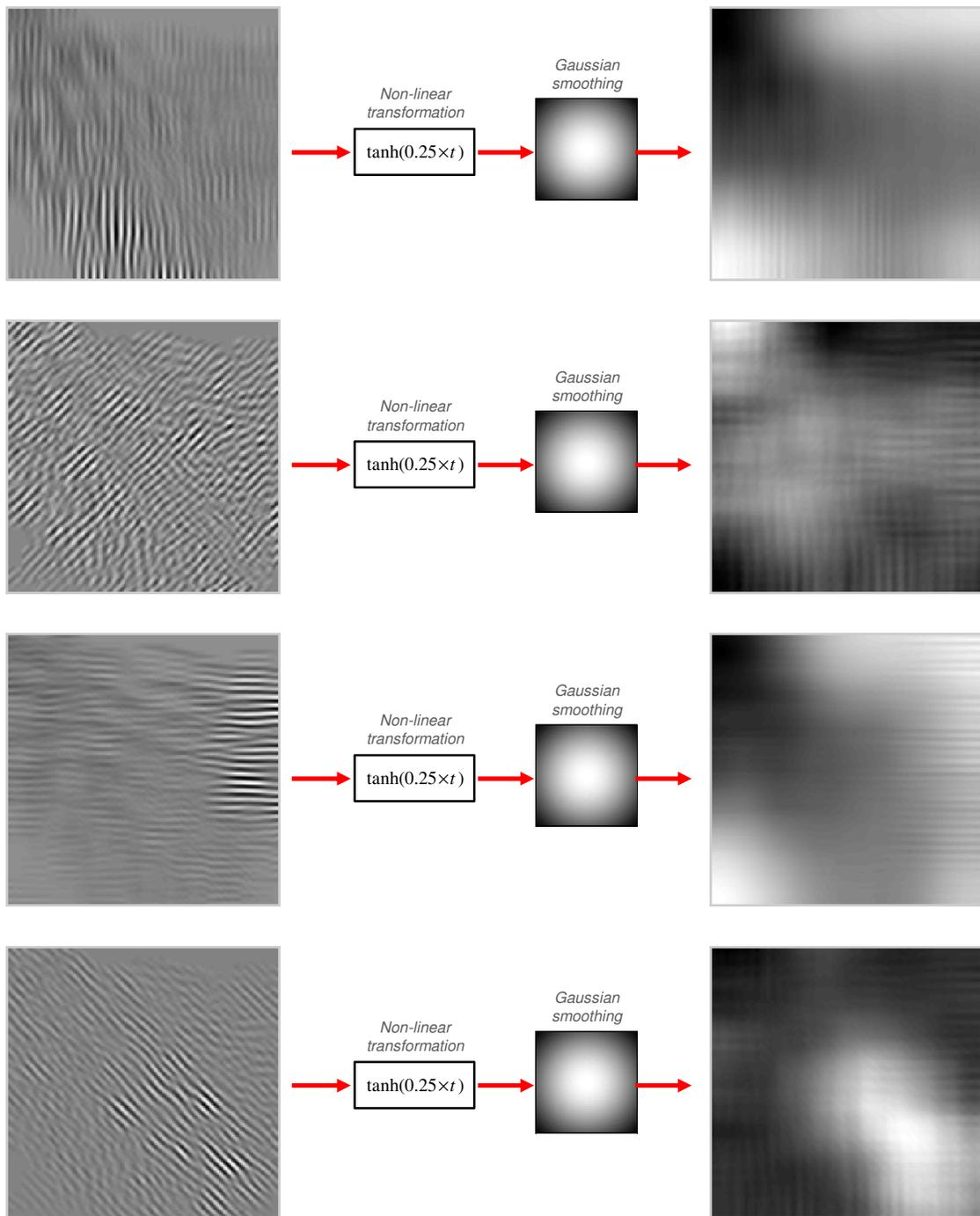


Figure 7.25: Application of energy extraction on the filtered images shown in Figure 7.22. It involves a non-linear transformation with sigmoidal function, and a smoothing filter with a Gaussian function that has three times bigger spread than the scale of its corresponding Gabor filter.

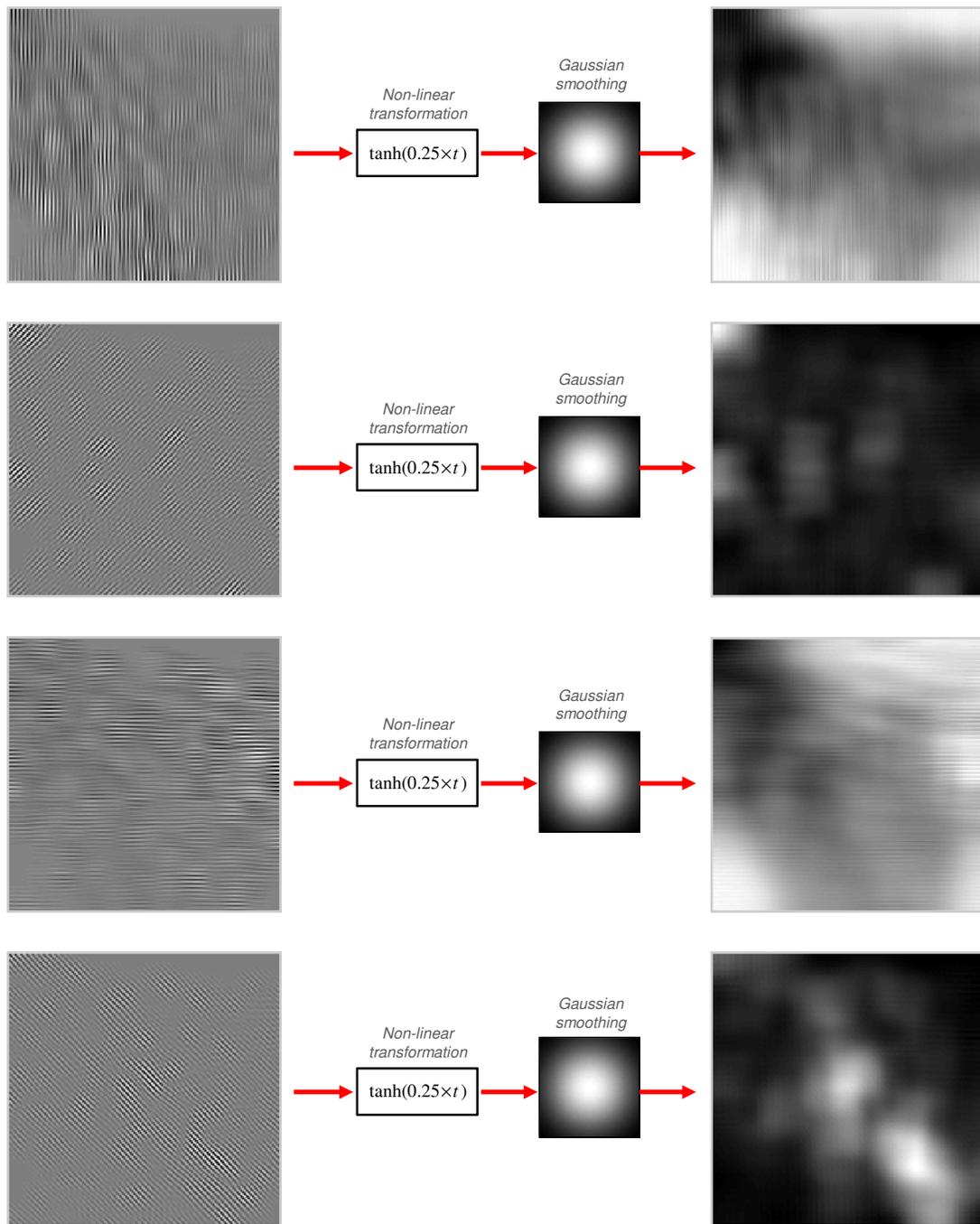


Figure 7.26: Application of energy extraction on the filtered images shown in Figure 7.23. It involves a non-linear transformation with sigmoidal function, and a smoothing filter with a Gaussian function that has three times bigger spread than the scale of its corresponding Gabor filter.

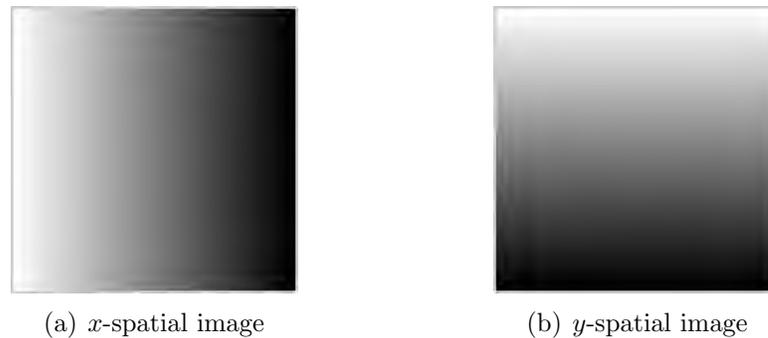


Figure 7.27: The spatial component images for a two-dimensional training image of size 199×199 is shown. The values are normalized to $[0, 1]$ range.

each node through Equation 7.29. We use the k -means clustering algorithm for this task. The final clustering result will assign each node to a different cluster. Therefore, all the nodes in each cluster are assigned to one stationary region. The resulting regions have two desired characteristics: (1) consisting of similar features (stationarity), and (2) lack of over-fragmentation for the stationary regions. All the corresponding features are shown in Figure 7.28. All these 88 features are defined at all node locations of the training image. Clustering them into similar groups will result in identification of stationary regions with accurate boundaries.

A flowchart of the previous steps for segmentation of a non-stationary training image is shown in Figure 7.29.

Application of k -means clustering is investigated for the non-stationary fluvial fan-deposit training image of Figure 7.1(a). The result for a choice of six stationary regions is shown in Figure 7.30. However, due to random initializations in the k -means clustering algorithm, each run leads to a different clustering result. Although the boundaries of each region may be different, however, they all have partitioned the training image into acceptable stationary regions. These subtle differences in the stationary regions, obtained because of the k -means convergence issues, are actually desirable, since they can introduce uncertainty into the boundary identification of the stationary regions and increase the stochasticity. Nevertheless, there are some non-stationary training images that consist of visually distinct stationary regions. In those images, the application of the k -means clustering algorithm will mostly converge to the correct solution with the strict (constant) region boundaries.

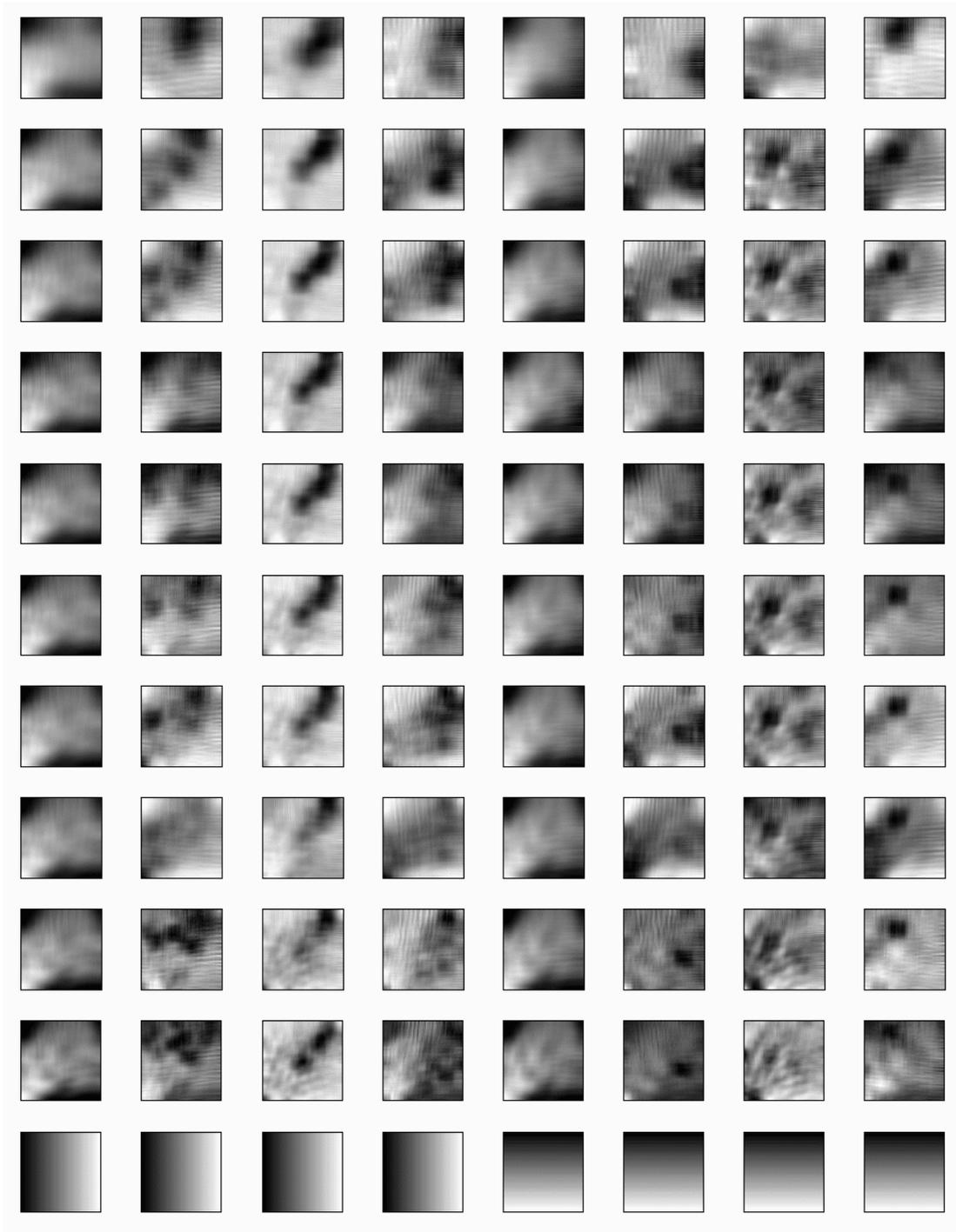


Figure 7.28: All 88 features obtained from the non-stationary training image shown in Figure 7.1(a). The final row depicts the spatial components that were embedded to ensure adjacency of the regions (each being repeated four times due to the employed weighting scheme).

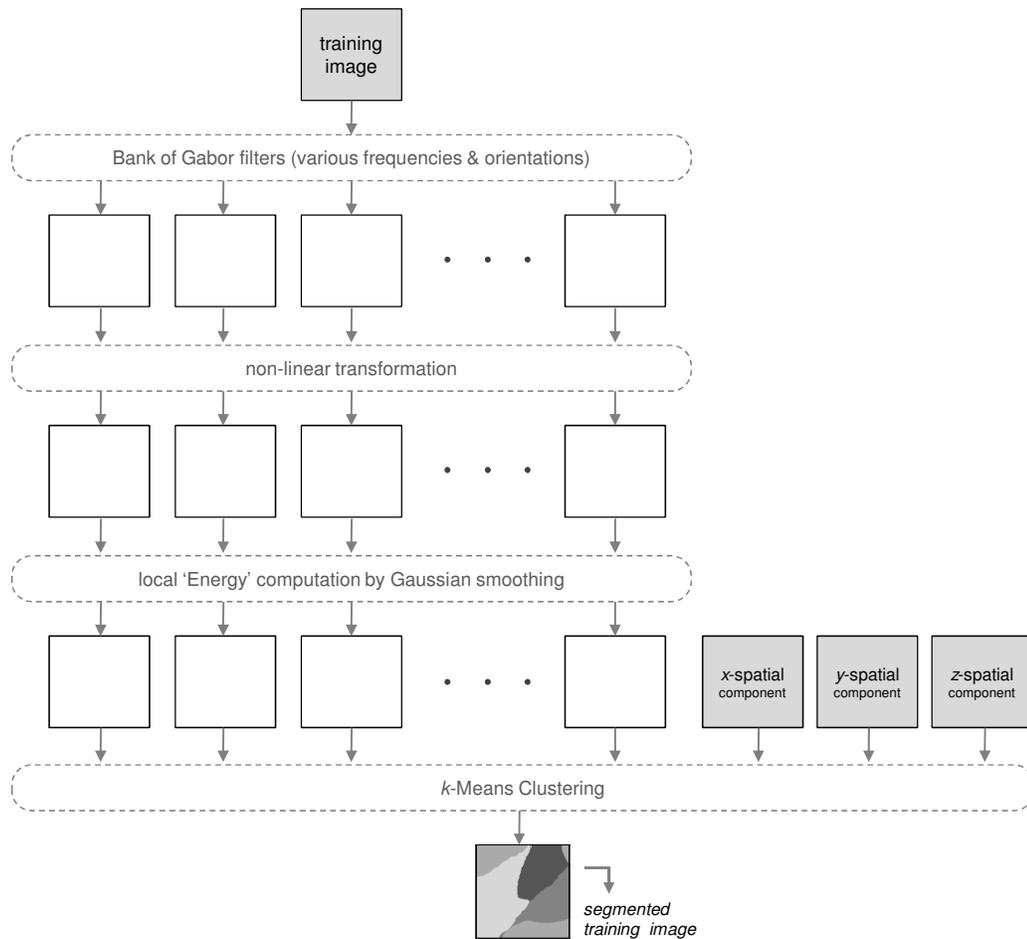


Figure 7.29: Flowchart of Automatic training image segmentation using a bank of Gabor filters, with the corresponding non-linearity filters and the Gaussian smoothing. The k -means clustering algorithm is used for defining the stationary regions.

ASM Simulation Algorithm

The automatic-segmentation methodology (ASM) is similar to the NRM method for non-stationary simulations. The ASM method proceeds by segmenting the training image \mathbf{t}_i into stationary subregions using the previously mentioned method of the Gabor filters bank. Simulation then continues with an empty realization grid. A random path is chosen to visit all the nodes of the realization. At each node location \mathbf{u} , a search for the most similar pattern is performed. However, this search is not limited to a neighboring radius of the node \mathbf{u} (as in the NRM method), but is allowed over a bigger stationary region that node \mathbf{u} belongs

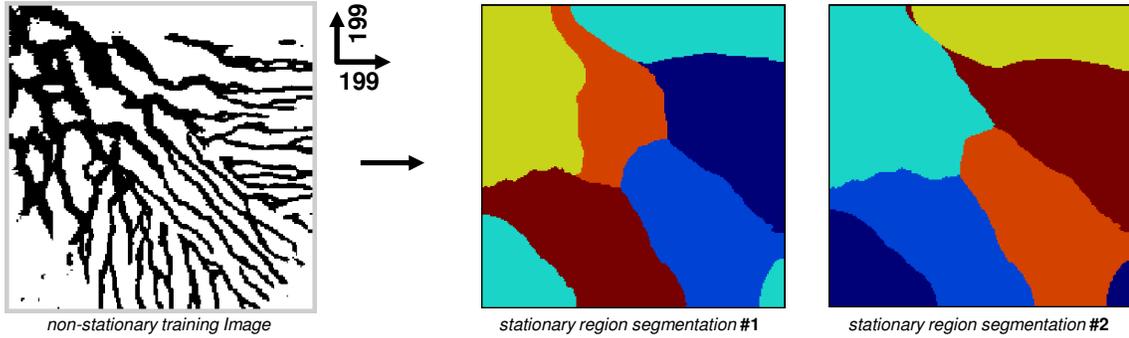


Figure 7.30: Application of segmenting a non-stationary training image. Each run of k -means clustering algorithm might lead to different subregions, due to the random initialization. It is a desirable characteristic due to the additional stochasticity.

to. In other words, each region is simulated individually by assuming its corresponding stationary training image.

Therefore, algorithmically speaking, the training image is used to construct both the pattern database $\mathbf{patdb}_{\mathbf{T}}$, and the region database, \mathbf{regdb} . The region database consists of the label defining the stationary subregions. If k subregions is required, i.e. $\mathcal{R} = \{r_1, r_2, \dots, r_k\}$, then, the following relationship defines the region database:

$$\mathbf{regdb}(\mathbf{u}) = i \iff \mathbf{u} \in r_i \quad (7.30)$$

where $1 \leq i \leq k$.

Having defined these two databases, the simulation starts by randomly visiting the realization grid nodes, and extracting their corresponding data events. For each data event, the same procedure of NRM method for pattern selection is adapted here. However, the functional that incorporates the spatial components is changed.

$$d_{ns}(t) = d_{pat}(t) \times \mathbf{I}^t(\mathbf{u}) \quad (7.31)$$

where the indicator variable $\mathbf{I}^t(\mathbf{u})$ is defined as follows:

$$\mathbf{I}^t(\mathbf{u}) = \begin{cases} 1, & \text{if } \mathbf{regdb}(\mathbf{u}_t) = \mathbf{regdb}(\mathbf{u}) \\ +\infty, & \text{otherwise.} \end{cases} \quad (7.32)$$

where \mathbf{u}_t represents the location of the t -th pattern.

Finally, during the simulation, for each visited node location \mathbf{u} on the realization grid, one calculates the distance vector \mathbf{d}_{ns} , and the optimal pattern $\mathbf{pat}_{\mathbf{T}}^*$ to be pasted on the realization grid at node location \mathbf{u} is the one with the minimum value in the non-stationary distance vector, \mathbf{d}_{ns} ,

$$\mathbf{pat}_{\mathbf{T}}^* = \mathbf{pat}_{\mathbf{T}}^i = \arg \min_i d_{ns}(i) \quad (7.33)$$

The simulation continues until all the nodes have been visited.

It is noteworthy to mention how the simulation algorithm changes for the two multi-scale methods of multi-grid and multi-resolution. In a multi-grid setting, the entire simulation follows the same procedure, such that, each node in higher multi-grid levels, $g > 0$, is assumed to belong to the same subregion defined at the same node location in the finest (original) scale of the training image. However, the situation is different for the multi-resolution setting. In the multi-resolution approach, one needs to find a correspondence between the subregions for each resolution. Due to the nature of regions, the most appropriate technique is the nearest-neighbor interpolation. This interpolation technique provides accurate region boundaries at all desired resolutions. Moreover, there is no need for thresholding the interpolated subregion grid. The simulation then proceeds as before.

An application of non-stationary modeling using the ASM method is depicted in Figure 7.31. Six stationary subregions have been determined using the Gabor filters. Four realizations are shown. One can observe the stochasticity obtained through such a partitioning. The ASM method even works on this non-stationary training image. However, it is more suited to those training images that can be categorized as quasi-stationary, or in other words, those that are fabricated by clearly distinct stationary regions.

In the ASM method, the practitioner can choose the desired non-stationarity level by defining the number of subregions a-priori to the algorithm. In order to investigate the behavior of the algorithm, three different region counts of $k = \{1, 6, 30\}$ are chosen. Two realizations are shown for each case in Figure 7.32. It can be observed that choosing only one stationary region for the entire training image grid will produce stationary realizations. One region is basically equivalent to assuming stationarity in the training image. On the other hand, a choice on six stationary regions is more realistic for the features observed in this training image. The two corresponding realizations show both the non-stationarity and the desired stochasticity. A higher value for the number of regions ($k = 30$) results in a

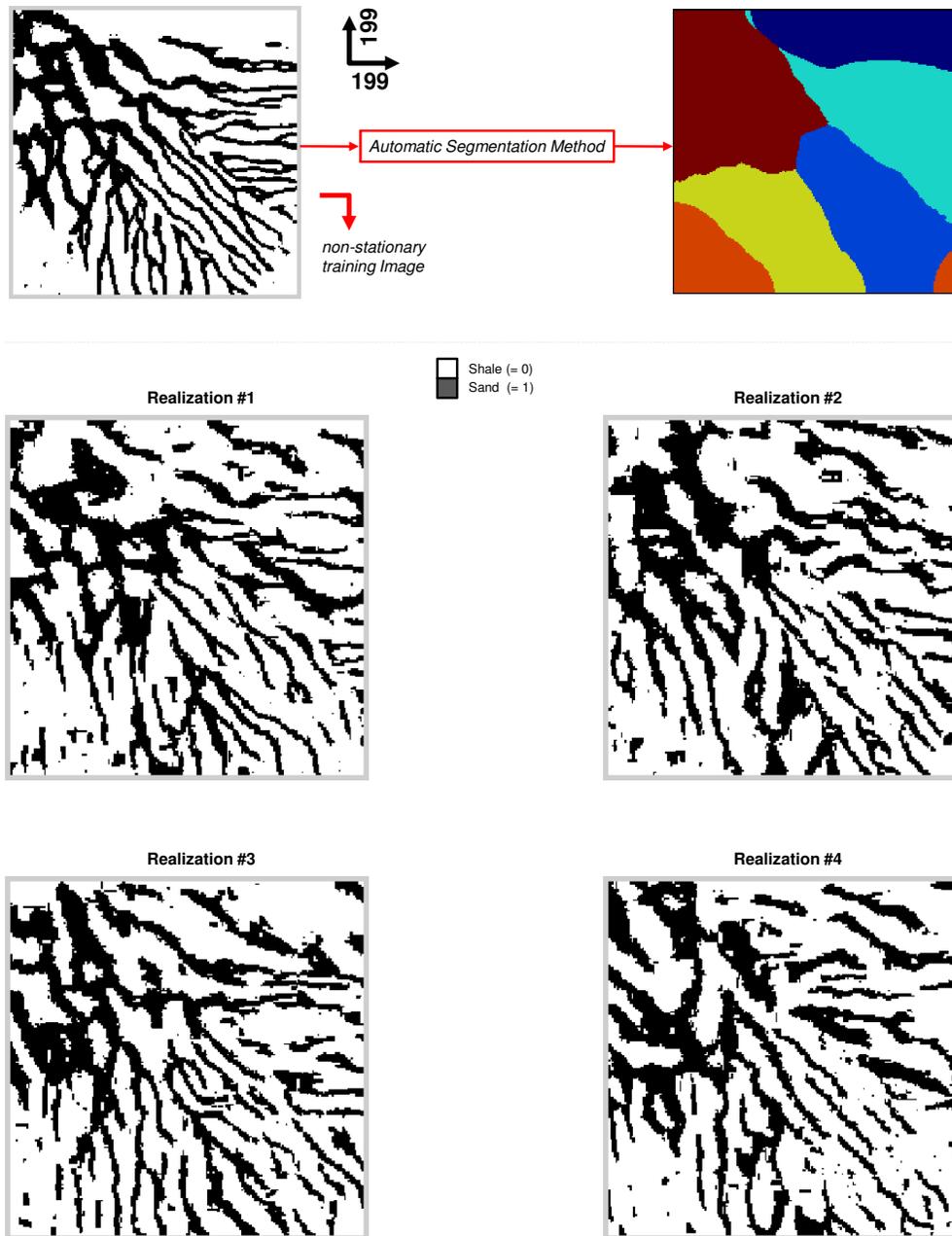


Figure 7.31: Application of Automatic-Segmentation Method (ASM) on fluvial fan-deposit training image of size 199×199 is investigated with a template size is 15×15 with a multi-grid level of three. There are six stationary regions defined for this training image. Four realizations are shown to demonstrate the non-stationarity and stochasticity of the proposed algorithm.

stricter non-stationary modeling of the training image. Hence, the realizations will exhibit a similar characteristic to the training image; perhaps more than desired.

Finally, an analysis is performed on the E-types of the realizations to observe the effect of the number of regions on the stochasticity of the resulting realizations of the ASM algorithm. One expects that by increasing the number of regions, the realizations resemble more to the training image, which entails less stochasticity. On the other hand, a lower number of regions increases the stochasticity in the realizations, and therefore, less spatial certainty in the E-types. Six different region-counts are investigated here: $k = \{1, 3, 6, 15, 50, 150\}$. The E-types (ensemble averages of 100 realizations) are shown in Figure 7.33. Clearly, a higher region-count amounts to less stochasticity, or stricter non-stationarity. And naturally, having only one stationary region leads to a flat (constant value) E-type.

Finally, for reference, the entire ASM methodology is outlined in Algorithm 13.

7.6 Examples

In this section, some examples of non-stationary modeling will be provided. Three different methods of SSM, NRM, and ASM are used throughout this section. An appropriate technique, depending on the non-stationarity of the example training image, will be adapted with clear explanations.

Non-Stationary Ellipses

The non-stationary training image in this example consists of ellipses that exhibit both rotation and elongation in the vertical direction. This training image is shown in Figure 7.34 (obtained from de Vries et al. (2009)). Clearly, the features in this training image are hard to quantify with auxiliary variables. One way would be to define one auxiliary variable for rotational angles, and one for scaling the features (but in only one direction to produce the desired elongations). However, such definitions and the amount of effort needed for algorithmic adjustments to work with such auxiliary variables make the traditional techniques inapplicable.

Due to the general non-stationarity of the training image, the most appropriate technique is the Spatial-Similarity Method (SSM). Four different realizations are shown in Figure 7.34 for a non-stationary weight of $\omega_{\text{SSM}} = 0.5$. It can be observed that all realizations produce the desired non-stationarity with minimum effort and minimum subjectivity.

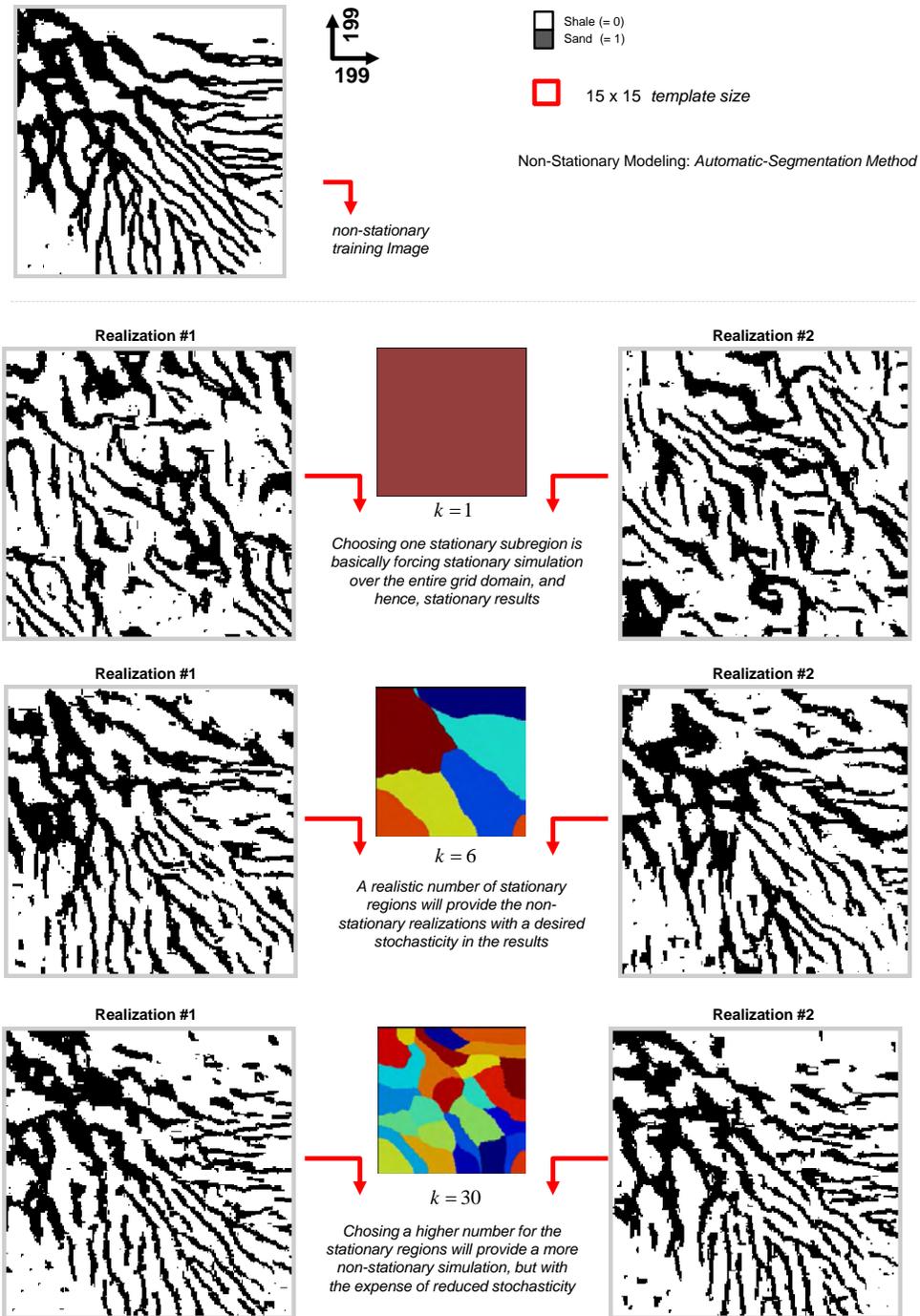


Figure 7.32: Investigation on the effect of the number of stationary regions on the generated realizations obtained by the ASM method. As can be seen, higher numbers indicate stricter non-stationarity in the results. Accordingly, choosing only one stationary region leads to stationary simulations, and hence, stationary realizations.

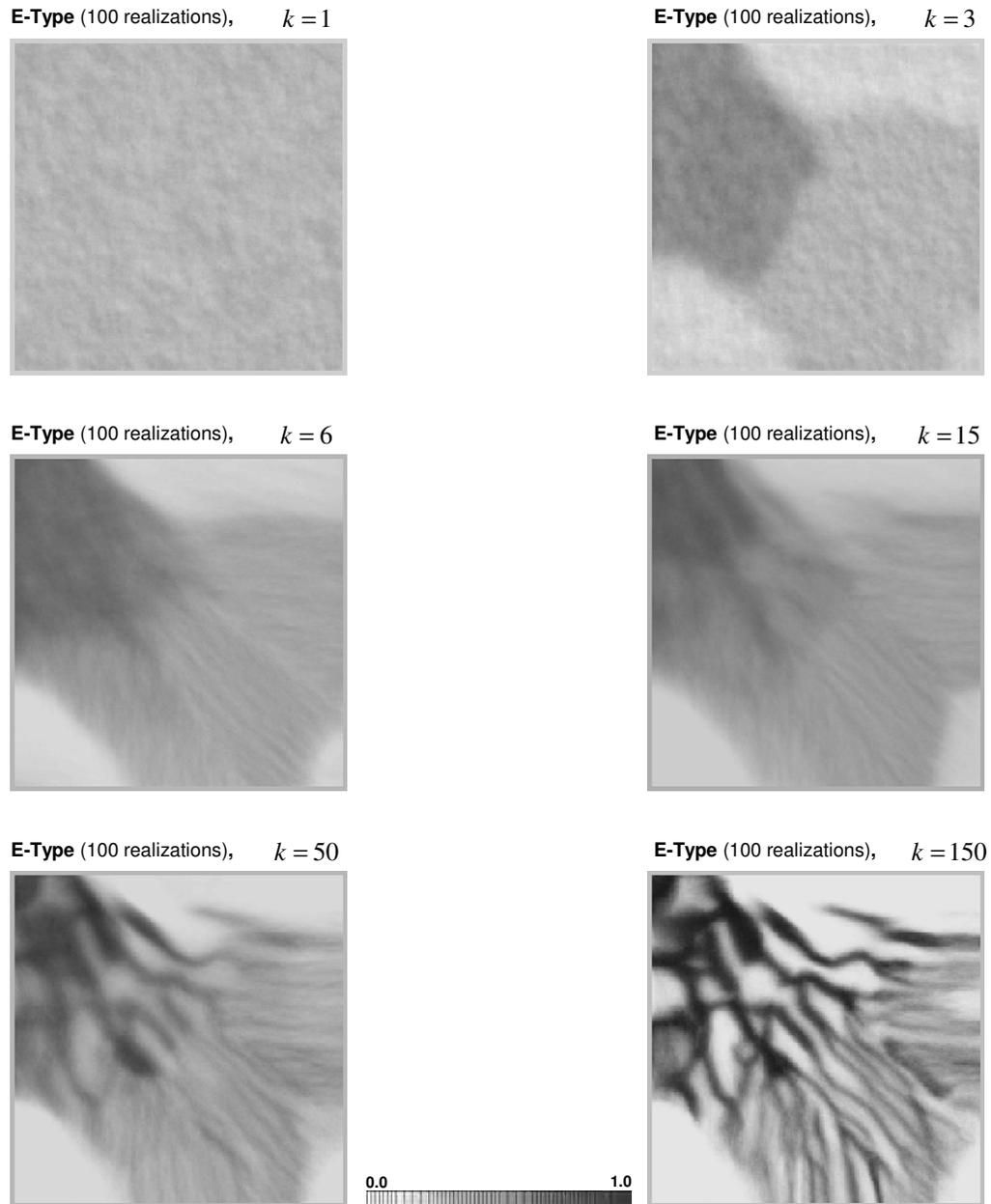


Figure 7.33: Investigation on the effect of the number of stationary regions on the generated realizations obtained by the ASM method. As can be seen, higher numbers indicate stricter non-stationarity in the results. A large number of 150 stationary realizations results in an E-type similar to the training image. Accordingly, choosing only one stationary region leads to stationary simulations, and hence, stationary realizations.

Algorithm 13 Automatic-Segmentation Method (ASM) Non-stationary Simulation

Require: Set number of stationary subregions k

- 1: obtain all possible frequencies f and f_z , and orientations θ for Gabor filters
 - 2: **for** each frequency $f_{zi} \in f_z$ **do**
 - 3: **for** each frequency $f_i \in f$ **do**
 - 4: **for** each orientation $\theta_i \in \theta$ **do**
 - 5: obtain the Gaussian spreads σ_i and σ_{iz} .
 - 6: construct the corresponding Gabor filter, g_i
 - 7: obtained the filtered training image by convolution: $o_i = g_i * ti$
 - 8: normalized the filtered image to $[-1, 1]$ range
 - 9: **end for**
 - 10: **end for**
 - 11: **end for**
 - 12: **for** each filtered image o_i **do**
 - 13: perform non-linear transformation by sigmoidal function, $o_i = \tanh(0.25 o_i)$
 - 14: perform Gaussian smoothing filter s_i with $\sigma_s = 3\sigma_i$ by convolution: $o_i = o_i * s_i$
 - 15: **end for**
 - 16: add spatial component x , y , and z as the feature vectors
 - 17: perform k -means clustering using all feature vectors of o_i and spatial components
 - 18: obtain the stationary subregions, and store them in region database **regdb**
 - 19: Construct pattern database **locdb_T**.
 - 20: Define a random path on the grid G_{re} of realization.
 - 21: **for** each node \mathbf{u} along the random path **do**
 - 22: **if** $\mathbf{u} \neq$ frozen **then**
 - 23: Extract the data event **dev_T(\mathbf{u})** from realization **re**.
 - 24: **if** all nodes \in **dev_T(\mathbf{u})** = uninformed **then**
 - 25: Randomly select a pattern **pat_T^{*}** from pattern database **patdb_T**
 - 26: **else**
 - 27: **d_{pat}** \leftarrow calculate the distances of **dev_T(\mathbf{u})** and patterns in **patdb_T**
 - 28: compute indicator functional, **I^t(\mathbf{u})** = (**regdb(\mathbf{u})** == **regdb(\mathbf{u}_t)**)
 - 29: compute non-stationary distance: **d_{ns}** = **d_{pat}** \times **I^t(\mathbf{u})**
 - 30: $i \leftarrow \arg \min_i d_{ns}(i)$, find the index for minimum non-stationary distance.
 - 31: **pat_T^{*}** = **pat_Tⁱ**
 - 32: **end if**
 - 33: Paste the pattern **pat_T^{*}** on the realization **re**.
 - 34: Freeze the nodes within the central inner patch neighboring location \mathbf{u} .
 - 35: **end if**
 - 36: **end for**
 - 37: **return** realization **re**.
-

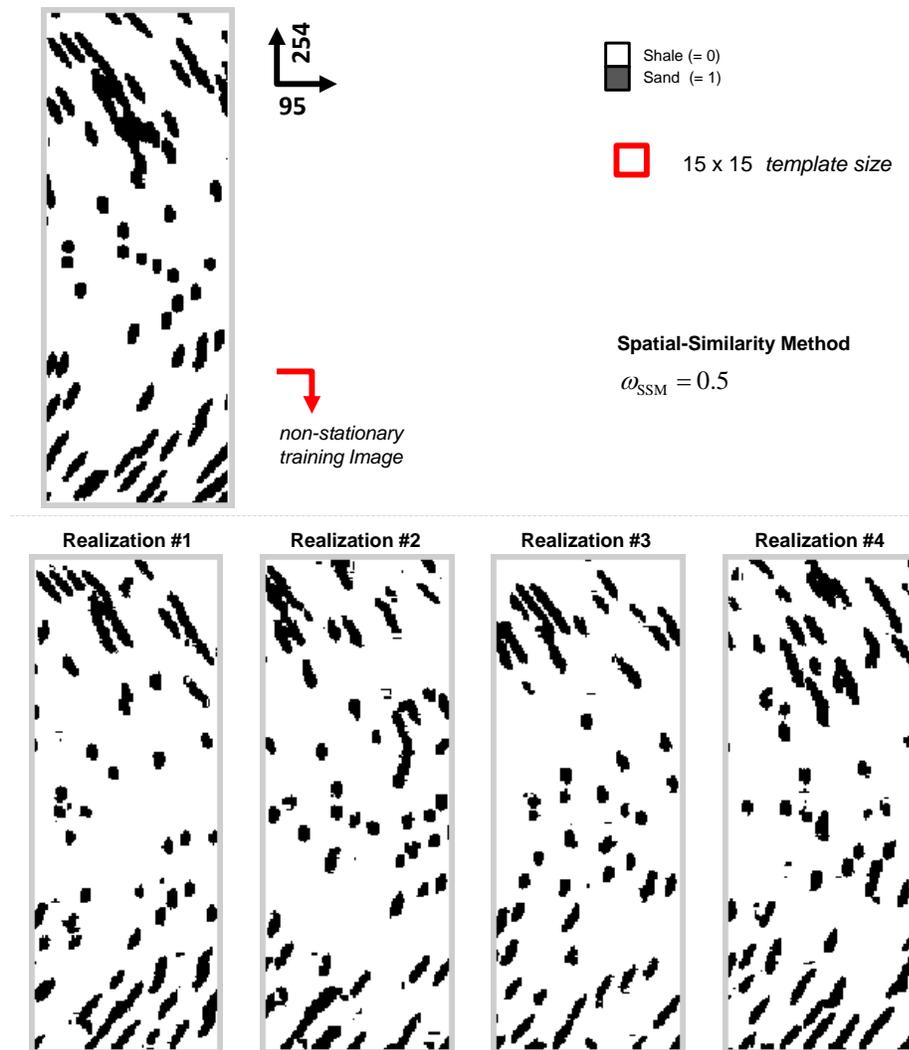


Figure 7.34: Application of the SSM method for non-stationary modeling of a training image consisting of elongated/rotated ellipses. Four realizations are shown by a template size of 15×15 , and a non-stationary weight of 0.5.

Non-Stationary Truncated-Gaussian

The training image in this case is obtained by truncating a continuous Gaussian simulation according to the proportion. The 100×200 training image is shown in Figure 7.35 (obtained from Chugunova and Hu (2008)). The non-stationarity in this case is simply the change in the proportion of the field. Therefore, only those traditional MPS algorithms that can take into account an auxiliary proportion field are able to model such a training image.

In this example, due to the quasi-stationary characteristic of this training image (weak non-stationarity), the most suitable technique is the Neighborhood-Radius Method (NRM). Although the ASM method of segmenting the training image into stationary regions would also be a good candidate for this example, but the NRM method is preferred because of the continuous E-W drift of the proportions. Four realizations are shown in Figure 7.35 with the non-stationary weight value of $\omega_{\text{NRM}} = 3$. One can observe the variability between the realizations, while honoring the input spatial variability in the proportions.

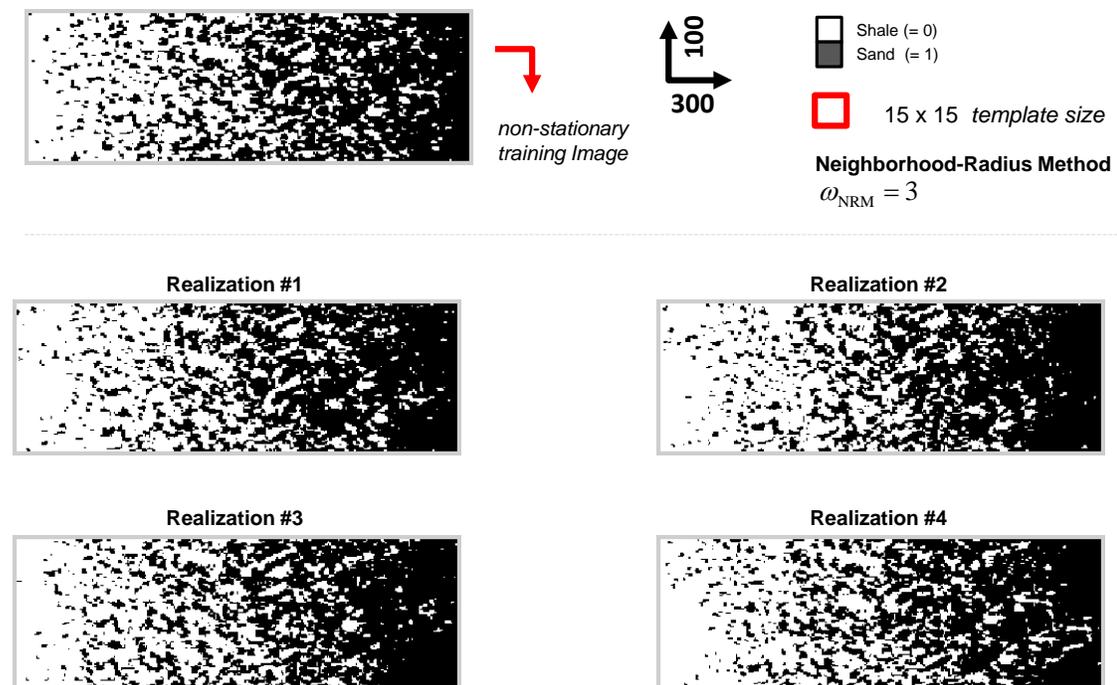


Figure 7.35: Application of the NRM method for non-stationary modeling of a training image constructed by truncating a Gaussian field according to the proportions between zero and one. Four realizations are shown by a template size of 15×15 , and a non-stationary weight factor of 3.

Non-Stationary Fracture Network 1

A fractured network is investigated in this example. The training image consists of fracture-like geometric features with different spatial orientations. Figure 7.36 illustrates such a training image (obtained from Chugunova and Hu (2008)). Traditional approaches need a

continuous map of orientations, or a set of subregions each with a pre-defined orientation angle, to model such a non-stationarity.

On the other hand, we can simply use the SSM methodology for modeling this non-stationary training image. It can be observed that the non-stationarity demands for a technique that can provide a continuous change in the orientations. This is in accordance with the continuous functional used in the SSM method. Accordingly, four realizations are generated using the non-stationary weight of $\omega_{SSM} = 0.5$, and with 15×15 template sizes. The SSM technique not only preserves the non-stationarity, but also keeps fracture continuity and generates stochasticity. These results demonstrate the versatility of the ASM technique on a variety of non-stationary training images.

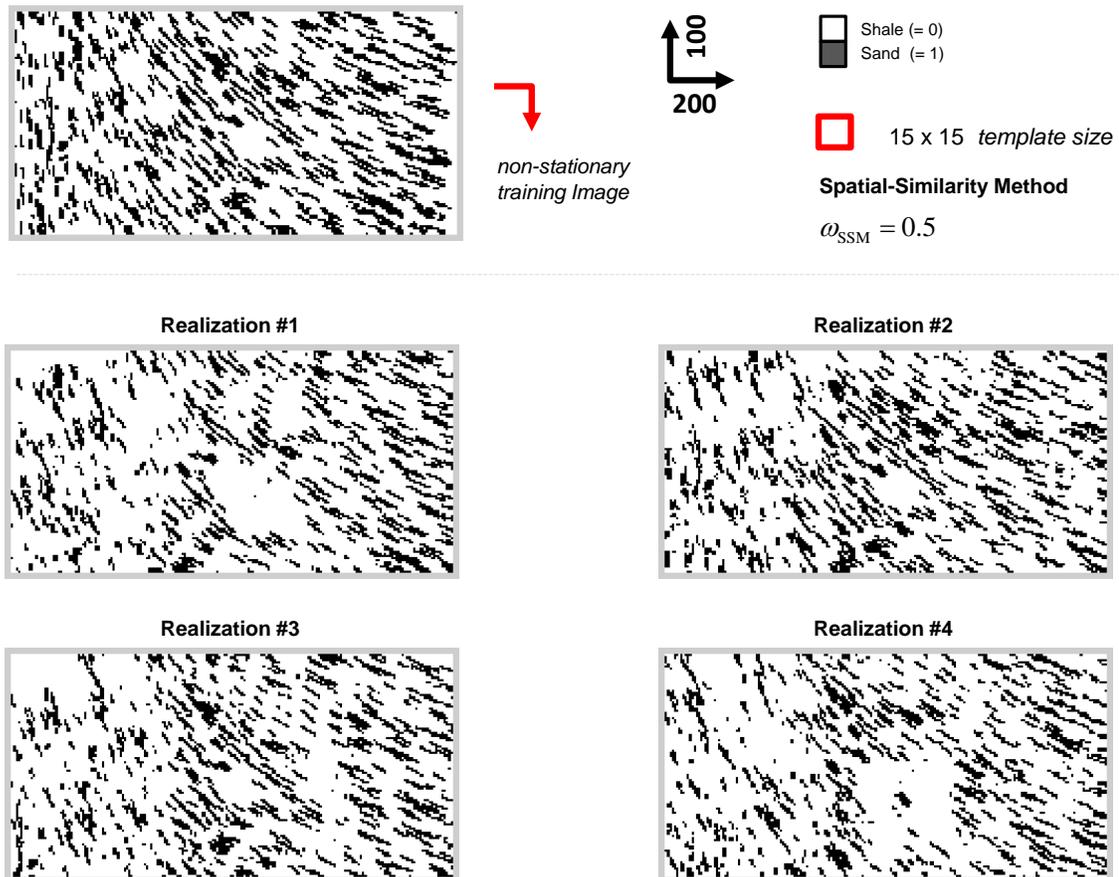


Figure 7.36: Application of the SSM method for non-stationary modeling of a fractured network. Four realizations are shown by a template size of 15×15 , and a non-stationary weight factor of 0.5.

Non-Stationary Fracture Network 2

The application of non-stationary modeling is analyzed on another fracture network. The corresponding training image is shown in Figure 7.37 (obtained from Boucher (2009)). The non-stationarity emerges from the three different fracture orientations. The quasi-stationarity of this training image clearly demands for the ASM simulation method. The ASM method is preferable in this specific case, since defining stationary subregions allows larger spatial variability in the generated realizations.

According to the training image, the number of stationary regions is set to three in the ASM method. The final segmentation is shown in the top-right corner of Figure 7.37. Clearly, this segmentation is in accordance to the expectations. Four different realizations are generated. It can be observed that the realizations exhibit a constant orientation in each subregion. There is also a good pattern reproductivity for each orientation in the fracture network. Therefore, these two fracture network applications demonstrate how the modeler can choose the appropriate non-stationary modeling technique for the geology at hand.

Non-Stationary Circles

Consider the non-stationary training image shown in Figure 7.38 (obtained from Strebelle and Zhang (2005)). It is a perfect candidate for the Automatic-Segmentation Method (ASM). There are clearly two distinct stationary regions in this training image. One region consists of elongated ellipses on 45° direction, and the other one in 135° direction. Although the SSM and NRM method can provide reasonable realizations, however, due to the quasi-stationarity of this training image, the ASM method is preferable.

In this example, two regions were selected a-priori in the algorithm. The resulting segmentation is shown in the top-right corner of Figure 7.38. It demonstrates the advantages of analyzing the training image at all scales using the Gabor filter bank. Otherwise, if all frequencies/orientations were not considered simultaneously, the partitioning would consist of scattered patches on the entire grid. The incorporation of the spatial components, in addition to the filter bank, leads to accurate partitioning into stationary regions.

Four realizations are shown in Figure 7.38. All realizations exhibit the desired non-stationary spatial behavior given by the training image. This example demonstrates the capabilities of the ASM methodology to handle quasi-stationary training images.

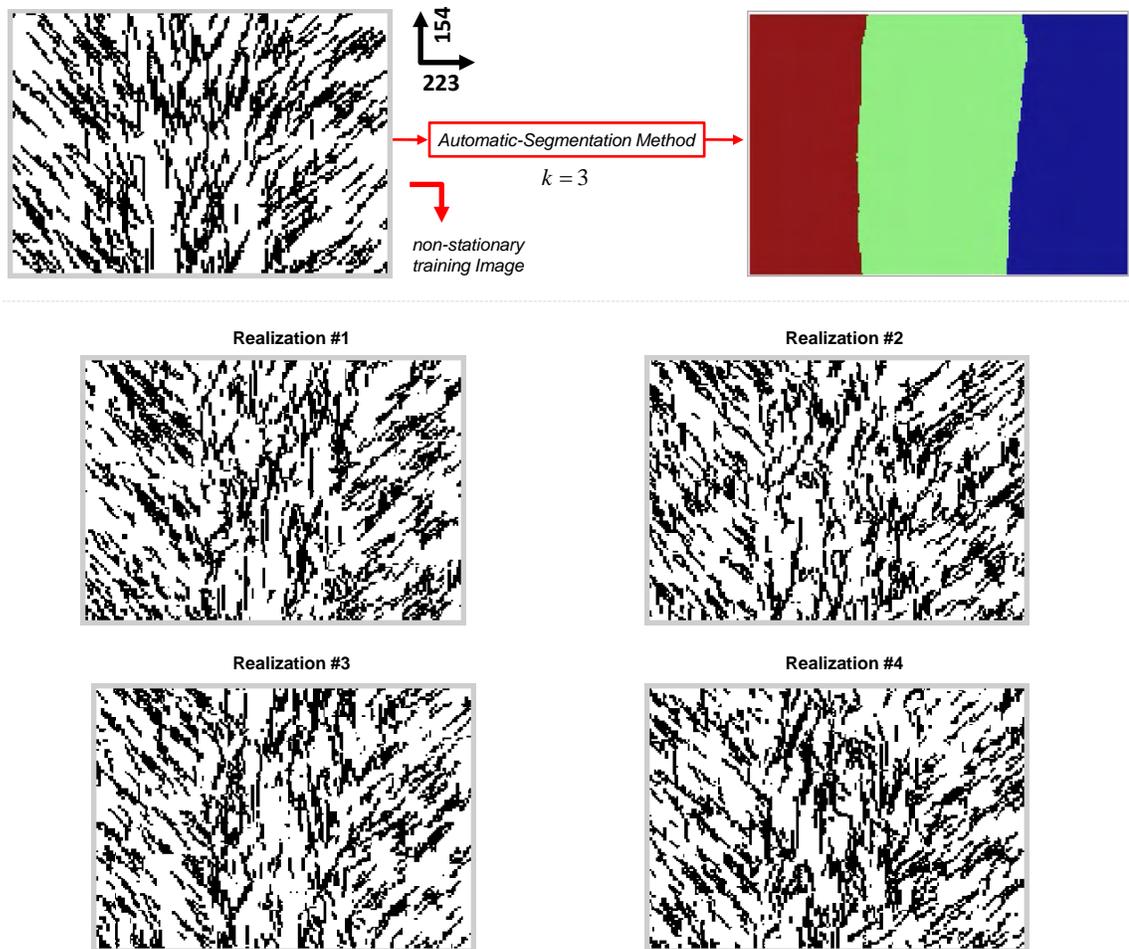


Figure 7.37: Application of the ASM method for non-stationary modeling of a fractured network. Three subregions are chosen for this training image, and the segmentation result is within expectation. Four realizations are shown, where a clear stochasticity is evident in every network. Furthermore, the patterns on the region boundaries conform to the two non-stationarities on each side.

Complex Non-Stationary Tidal System

In this example, a shallow-water tidal-dominated system will be simulated. The corresponding training image, shown in Figure 7.39, is very complex (obtained from Boucher (2009)). It consists of the shoreface, the lagoon, and the redbed. It is a categorical training image of seven different variables. The complexity of such a training image originates from the way the geological features are connected to each other, and the way the non-stationarity

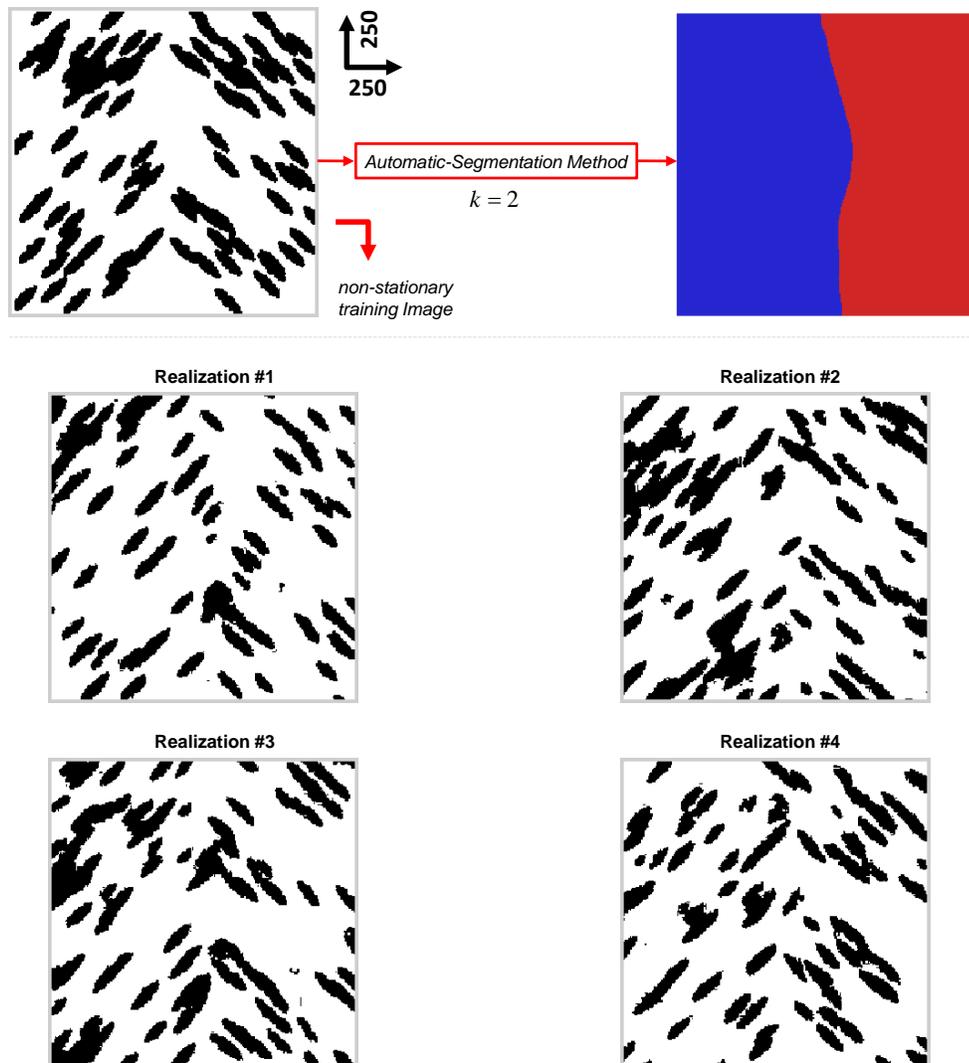


Figure 7.38: Application of the ASM method for non-stationary modeling. The training image consists of two visually distinct stationary regions. A correct segmentation is obtained by the application of Gabor filters on this training image. Four realizations are shown by a template size of 15×15 .

encompasses the grid.

Traditional MPS techniques are incapable of easily generating non-stationary realizations. The most promising technique would be a region-based approach. Defining the regions, however, is a subjective and a daunting task, plus regions are often assumed fixed. On the other hand, the non-stationary modeling techniques, introduced in this chapter, can

considerably facilitate this process.

Due to the complexity of this training image, one would choose the SSM technique to model this non-stationarity. As mentioned before, the SSM technique works on any training image. The non-stationarity weight ω_{SSM} can enforce the desired behavior of the simulation. In here, we have chosen $\omega_{SSM} = 0.5$, and generated six different realizations. The results are shown in Figure 7.39. One can notice the geological consistency in the generated realizations. The capability of the SSM method becomes evident with this complex training image.

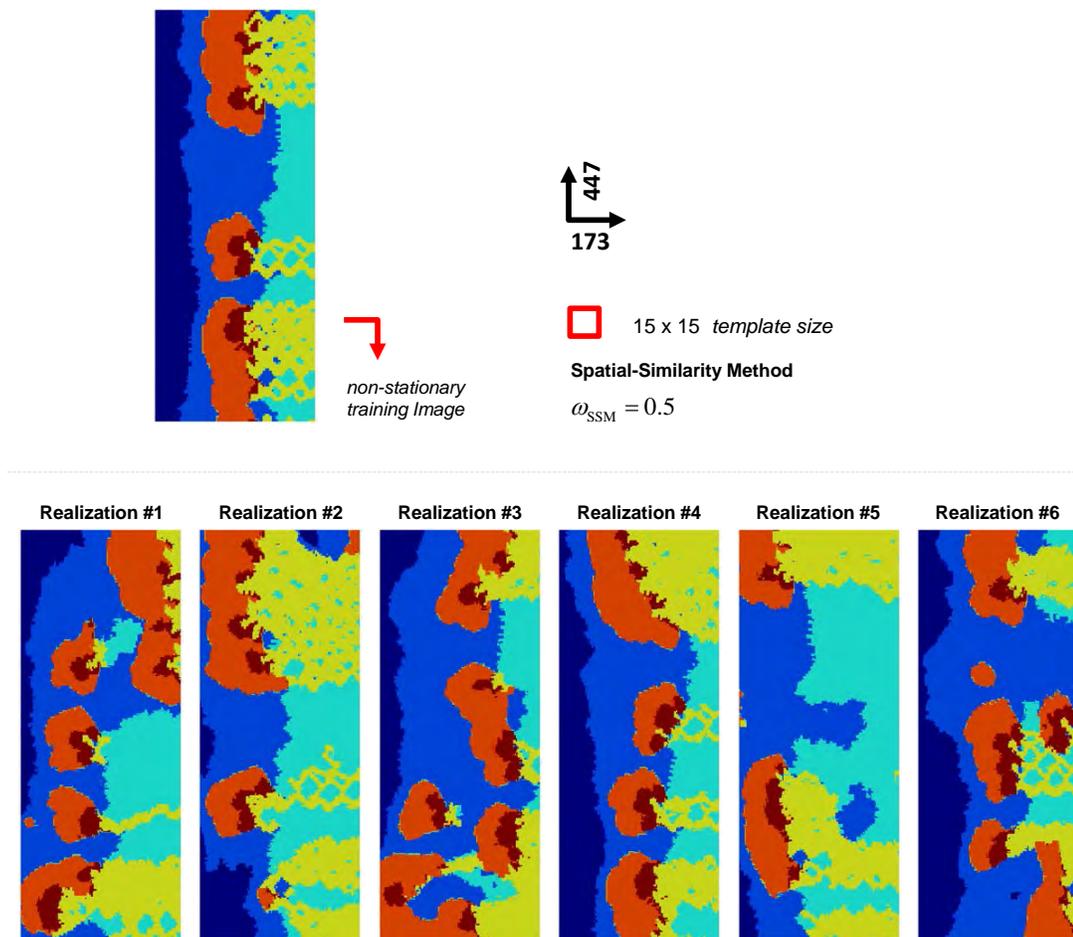


Figure 7.39: Application of the SSM method for non-stationary modeling of a complex tidal-dominated training image of size 447×173 . The SSM method can handle any form of non-stationarity. Six realizations are shown by a template size of 15×15 .

On the other hand, one can investigate the application of the ASM method to partition this training image into some stationary subregions. This is the decision of the modeler to choose which technique is more appropriate for the task at hand. If the modeler decides that this training image is quasi-stationary with distinct subregions, then the ASM method can be exercised. Therefore, the ASM method is performed by selecting $k = 5$ distinct subregions. The resulting realizations are shown in Figure 7.40. One can observe the variability between the realizations in terms of the boundaries, and the feature geometries and locations in the provided realizations. Conceptually, the ASM method can also handle any form of non-stationarity as long as an appropriate number of subregions can be determined.

7.7 Conclusion

In this chapter, we addressed the important issue of non-stationarity in MPS simulation. We then addressed the issues in the traditional techniques for modeling non-stationarity. In traditional methods, the introduction of auxiliary variables is seen as instruments to force the algorithm to generate the desired characteristic. Second, the stochasticity of the algorithms is affected by the subjectivity of the modeler in choosing the region boundaries, or the probability fields. And finally, the algorithmic adjustments, if possible, that are required for each training image make non-stationary modeling a complicated exercise.

The simple objective is to provide a non-stationary training image to the algorithm such that it returns several non-stationary realizations, without the need for any additional source of information.

Therefore, to eliminate these issues, we introduced three non-stationary modeling techniques; namely, the SSM, the NRM, and the ASM methods. All these techniques incorporate the spatial components into the simulation. Thereby, patterns cannot be anymore perceived as location-independent. This spatial component integration allows us to build non-stationary realizations without the need for any auxiliary variables.

All three techniques were explained with an example application. The Spatial-Similarity Method (SSM) is a general technique that applies to almost any training image. On the other hand, the NRM method is more suitable to quasi-stationary training image, where no visually distinct regions can be discerned. However, the ASM method was designed for

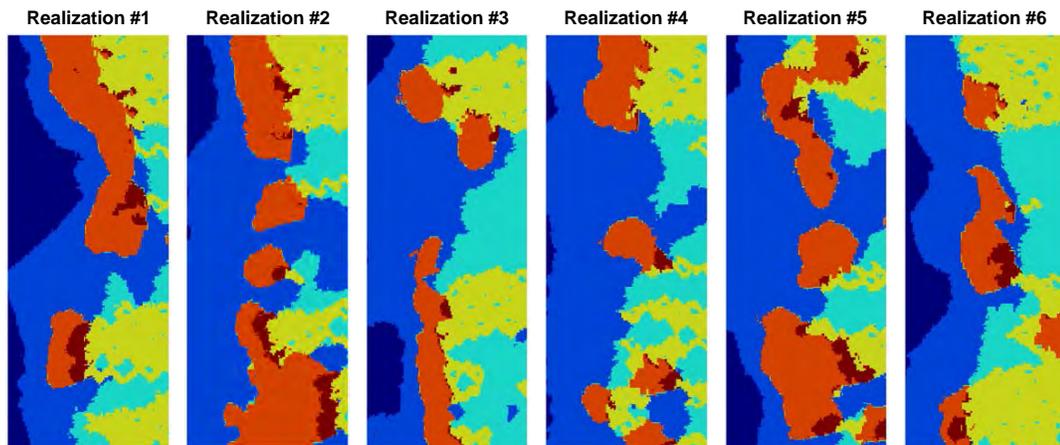
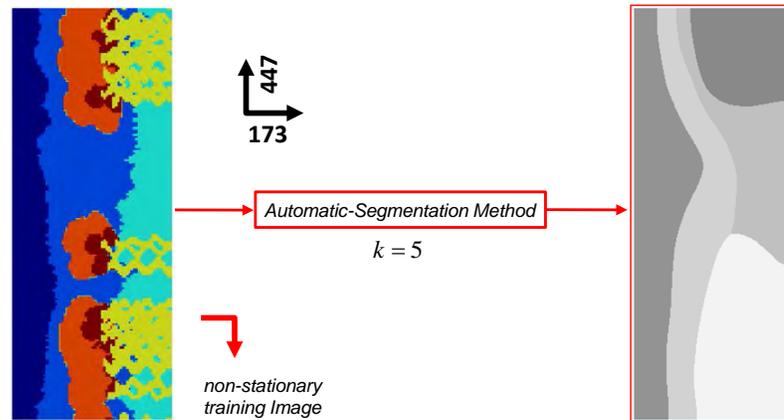


Figure 7.40: Application of the ASM method for non-stationary modeling of a complex tidal-dominated training image of size 447×173 . The training image is divided into five stationary regions for this simulation. Six different realizations are illustrated, where the desired spatial variability of the categories is evident.

quasi-stationary training images that consist of clear stationary subregions. In all these algorithms, a parameter is available to the modeler in order to control the non-stationarity behavior. By changing this parameter, the modeler can choose between strict non-stationarity, where the realizations resemble the training image, or stationarity, where the realizations ignore any spatial information.

Finally, a variety of training images were investigated. For each training image an appropriate non-stationary modeling technique was proposed. The results demonstrated

the powerful capabilities introduced by the proposed algorithms.

Chapter 8

Conclusion

The most difficult subjects can be explained to the most slow-witted man if he has not formed any idea of them already; but the simplest thing cannot be made clear to the most intelligent man if he is firmly persuaded that he knows already, without a shadow of doubt, what is laid before him.

LEO TOLSTOY (1828 - 1910)

Increased application of multiple-point geostatistical simulations requires a perfect understanding of the problem and a clear identification of the advantages and disadvantages of existing algorithms. Among those algorithms, pattern-based methods have been shown to have the best performance in terms of pattern reproduction as well as more general applicability (continuous variables) when applied to sequential pattern simulation, however, they may suffer from lesser CPU performance. Detailed analysis of the performance and memory requirements for these algorithms shows that searching for each potential pattern is the most computationally demanding step. The main focus should be on designing techniques that can handle the same pattern-based techniques in a more efficient way and with greater capabilities.

Better and more realistic models should not require an increase in user-set parameters (a disadvantage of filtersim) and a much simpler method should work just as well. We believe the same to be true for pattern analysis. Therefore, in this thesis, a new mathematical framework for modeling patterns of natural images, representing subsurface heterogeneities,

was obtained. A distance-based method was introduced for mapping the space of patterns through the mathematical theory of multi-dimensional scaling. In distance-based modeling, many of the tasks usually performed in multiple-point geostatistical algorithms can be carried out in a surprisingly simple yet powerful way.

As a result, a new multiple-point geostatistical algorithm using distance-based method and kernel mapping has been developed. The technique introduced in this thesis broadens the capabilities of multiple-point simulation processes. Reducing the dimensionality of the patterns, providing a filter-free classifier, and objectively learn the parameters are some prime examples introduced within this technique. Specifically, the better pattern classification capability of the proposed method is one of the main advantages provided by the distance-based pattern modeling framework. In conclusion, the proposed method provides, in the minimum theoretically and potentially also practically, an improvement over all previous MPS algorithms such as *snemis*, *simpat* and mainly *filtersim* in terms of speed, pattern reproductivity and easy integration of hard data and soft data.

8.1 Summary of the Study

The contributions of this thesis can be summarized as follows:

- A distance-based pattern modeling framework was introduced to analyze the spatial variability among the patterns of a training image. The methodology is inspired by the human brain and how it understands the surroundings; that is, the ability to observe and organize patterns. Using this logic, we adapted a simple, but powerful, approach for calculating the pair-wise distances between all the patterns of a training image. These distance calculations allow representing the pattern variability in a Cartesian space using multi-dimensional scaling. Each pattern is then represented by a point in the MDS space. This representation provides a framework to apply different mathematical algorithms on the patterns of a training image. Then, we transformed these patterns into a higher-dimensional feature space, where they become more linearly separable. We demonstrated the capability of the proposed approach to categorize the patterns into different clusters. Many examples and their comparisons with the previous approach of *filtersim* showed how this distance-based analysis often leads to a higher classification quality. Another advantage of such an

approach is the independence of the algorithm on filters or other statistical measures to analyze the patterns or features of the conceptual geological model. Consequently, a unifying framework for pattern recognition was obtained.

- We described a pattern-based MPS simulation framework for generating realizations that have similar multiple-point statistics as the input spatial model (the training image). We argued how this algorithm can reduce the subjectivity inherent to MPS algorithms.
- A new technique for quantifying the spatial variability among the models, using the Jensen-Shannon divergence, was introduced. This technique analyzes the multiple-point histograms between two realizations, or even between a realization and the training image. It allows for quantitative assessment of the behavior of any MPS algorithm. We used this technique to compare the proposed method to filtersim. Better multiple-point statistical reproduction of training image features was observed in our results.
- A multi-scale MPS modeling framework was introduced to enable the reproduction of long range pattern continuity. The traditional approach of the multi-grid simulation was revisited, and a new coarse-grid rescaling method was developed for improving the pattern connectivity in the realizations. Furthermore, a novel multi-resolution simulation technique was proposed. This technique is inspired by the human front-end visual system, and how we see things at multiple scales simultaneously. The multi-resolution approach simulates a realization by proceeding with a low resolution training image, and successively fine-tuning the realizations in higher resolutions. We illustrated how such an approach can increase the pattern reproductivity in the realizations. A series of unconditional simulations was performed on different training images and the improvement in pattern reproduction and large-scale continuity was observed in both categorical and continuous cases.
- Hard and soft data conditioning algorithms were introduced for both the multi-grid and the multi-resolution approaches. For the multi-grid approach, the existing data conditioning capability was improved by a ‘Smart-Vibrating Data Relocation’ method. In this approach, an enhanced data relocation algorithm was introduced which, by intelligibly relocating the data, preserves their spatial information, as much

as possible. It was shown how such an approach can considerably improve the conditioning in dense hard datasets. Moreover, the internal inconsistency, existing in the traditional data relocation approaches, was resolved. Hard data conditioning was examined by obtaining the ensemble averages of realizations. The enhanced data conditioning established a promising future for this method.

A new ‘distance-fusion’ approach was also introduced for soft data conditioning. This approach takes advantage of the similarities between the soft data and the cluster prototypes in terms of their blurred representations of the geological features. Integration of the data event with a soft data event by using the proposed distance-fusion technique enabled conditioning of the soft probability cubes to the subsurface models. Several examples demonstrated the strength of this distance-based approach. Furthermore, we obtained an analogy between our approach and the ‘permanence of ratios’ approach in the Tau model. We argued why such a simple distance-fusion approach is a non-convex algorithm, and how it can select the correct class prototype without the need for accurate probabilistic calculations. Finally, we illustrated the consequence of different confidence levels on soft data to the generated realizations.

- new methodologies on parameter selection were introduced. These techniques minimize the modeler’s subjectivity in specifying correct parameters, and at the same time eliminate the need for tedious trial-and-error computations.

First, an algorithm on selecting an optimal template size was developed. This algorithm relies on the concept of entropy to analyze the information content within the patterns at each template size. Then, using the proposed ‘maximum log-likelihood profile estimation’, an appropriate template size is selected. This provides a robust methodology that captures the essential features of any conceptual training image. The algorithm was exhaustively tested for its accuracy. We also showed how such an approach behaves on non-stationary spatial models.

Second, an approach for selecting a reasonable number of clusters was introduced. An appropriate number of clusters is a compromise between the computational efficiency of the MPS algorithm, and the quality of the generated realizations. We stated how the readily-available kernel matrix can be used to find a structure among the patterns. By analyzing this structure with an eigenvalue decomposition of the kernel matrix, we were able to quantify the distribution compactness of the clusters, and hence, find

the optimal cluster count.

- Novel techniques for modeling non-stationary training images were introduced. We expressed the concerns related to the way traditional algorithms handle non-stationary geological phenomena; that is, by providing auxiliary variables which control the modeling process. We also stated how the traditional approaches can reduce spatial uncertainty, trigger potential subjectivity in auxiliary variables, or discourage a straightforward implementation. We thus introduced three new techniques by integrating the spatial component of the patterns into MPS simulations. The techniques are as follows:

1. Spatial-Similarity Method (SSM), which is applicable to any non-stationary training image.
2. Neighborhood-Radius Method (NRM), which is more suitable for quasi-stationary training images where visually distinctive regions cannot be distinguished.
3. Automatic-Segmentation Method (ASM), which is well-qualified for training images having clearly identifiable stationary subregions.

In all these techniques, a non-stationary weight was incorporated in the distance calculations to allow the modeler to select the level of desired non-stationarity for the simulation. We provided several sensitivity analysis on the weight parameters and its effect on the stochasticity of the generated realizations.

We demonstrated the capabilities of these approaches on a variety of non-stationary training images, and showed how each algorithm can be beneficial for each scenario. The competency of the proposed algorithms not only facilitates the modeling process, but also provides a general framework for simulating *any* complex spatial model.

- A software implementation (DisPAT) of the proposed distance-based pattern modeling framework has been provided as a DLL plugin in SGeMS (Stanford Geostatistical Modeling Software). We compared the computational time of the DisPAT algorithm with the existing MPS techniques, and showed significant speed improvements in both 2D and 3D training images.

8.2 Future Work

In this section, several ideas on multiple-point geostatistical modeling will be described. These ideas involve improving the DisPAT method, introducing new functionalities, or the future avenues in MPS modeling.

Better dissimilarity distance functions

One of the most important components in the proposed distance-based modeling approach is the dissimilarity distance function. An unsuitable distance function results in poor pattern reproduction. For example, if we had used the absolute difference between the mean of the patterns as the dissimilarity function, low classification accuracy would have undoubtedly prevailed. In this dissertation, we used proximity transform distance function. It was illustrated how such a function can better distinguish the patterns, more inline with the human expectation.

However, there is not a single distance function that can provide the best measure of similarity for all possible cases. In order to obtain a high accuracy in pattern classification, we need to combine different distances into a composite dissimilarity measure. Each distance function, in such an approach, is capable of distinguishing some aspects of the features observed in the patterns. Some distance functions may provide erroneous measures between specific patterns, but it is believed that an ensemble of distance functions can decrease these errors. For example, if we have five distance functions, and one of them is providing an inaccurate measure between two specific patterns, the four remaining functions can compensate for this error, such that, the composite measure obtained by this set of functions accurately evaluates the pair-wise dissimilarity between those patterns.

So as a future work, one can combine distance functions and evaluate the classification quality of the patterns. One simple way to combine these functions is to average their normalized distances. In other words, each distance function will be used to construct a distance matrix. Then, each distance matrix will be normalized to one such that the distances lie in the zero to one range. The final distance matrix used for MDS mapping is the sum of all these distance matrices. If the set of m distance matrices can be represented by $\mathcal{D} = \{D_1, D_2, \dots, D_m\}$, then the dissimilarity distance function used in the DisPAT method can be computed as follows:

$$D = \frac{D_1}{\|D_1\|_\infty} + \frac{D_2}{\|D_2\|_\infty} + \cdots + \frac{D_m}{\|D_m\|_\infty} \quad (8.1)$$

Other combination strategies can also be adapted. For example, one can multiply the distance functions instead of summing them, as follows:

$$D = \frac{D_1}{\|D_1\|_\infty} \times \frac{D_2}{\|D_2\|_\infty} \times \cdots \times \frac{D_m}{\|D_m\|_\infty} \quad (8.2)$$

Another method for combining the distances is to use a linear combination, where the weights can be obtained by a genetic algorithm optimization technique. The objective function, to be minimized, can be a measure of variability among the patterns in the MDS space, or the final clustering quality (i.e. cluster sharpness). For example, a simple objective function could be the maximization of the median of the distances, or a measure to maximize the correlation between the distance functions. Some examples of distance functions, to be included in the DisPAT algorithm, are:

- Hausdorff distance (measuring how far two subsets are from each other).
- The Euclidean distance between the Radon transform of the patterns.
- The Euclidean distance between the Hough transform of the patterns (detection of lines, circles, ellipses).
- Distance between the Gabor filters.
- Distance between the SIFT feature descriptors (Scale-Invariant Feature Transform).
- Distance based on edge detection algorithms (Canny, Sobel, Prewitt).
- Distance based on corner detection, such as Harris operator.
- Distance based on blob detection, such as DOG (difference of Gaussian), or DOH (determinant of Hessian).

Better cluster prototype definitions

In the proposed methodology, we rely on the cluster prototypes to categorize the data event into one of the pattern classes. The ability to find the suitable class has a significant effect on the final simulated realizations. This classification ability is dependent upon two factors:

the distance measure used between the data event and the prototypes, and the prototype definitions themselves. The first factor is always picked as a simple measure of absolute differences.

However, a detailed analysis needs to be performed on the prototype definition. In this thesis, the prototypes are obtained by averaging of all the patterns within a cluster. Nevertheless, such an approach leads to a bias towards selecting patterns with higher proportions (in the binary case for example). In other words, due to possible errors in classification, the higher proportion patterns will be more prevalent among the patterns of each cluster. Despite this bias, for prototype calculations, we should try to be consistent with the distance function that was used for pattern classification; namely, the proximity transform distance function. Using another function for calculating the prototypes (i.e., a simple Euclidean averaging of the patterns in our case) may degrade the classification quality.

Therefore, other methods for constructing a prototype (a representative image of the patterns within a cluster) are required. Some of these techniques have already been tested. For example, we tested the application of the k -medoid of the cluster as prototype. However, this method was unsuccessful and pattern reproductivity was reduced in the generated realizations. This is due to the distinct features of the selected patterns used as the prototype. For instance, in a binary image, the prototypes consist of only ones and zeros, and do not contain any information regarding the rest of the patterns in their clusters. Therefore, the similarity search with the data event may fail to find the most optimal pattern.

Another method of inverse-distance weighted linear averaging was tested, but found unsuccessful. As a future work, one needs to find better methods to find the prototypes which are most representative of their clusters. One such technique is to use the pre-image problem (Kwok and Tsang, 2004) to construct the prototype. The pre-image problem involves finding the inverse mapping Φ^{-1} from the kernel-induced feature space to the input pattern space. In other words, after finding the cluster centroids in the feature space using kernel k -means algorithm, one can find the pre-image and use that as the cluster prototype. The pre-image obtained by the Gaussian kernel is basically a linear average of all the patterns, and therefore, informative of the pattern in its cluster. The application of this technique may result in more suitable prototypes and more precise similarity searches.

Better kernel functions

We have used the radial basis function (Gaussian kernel) for mapping the patterns from MDS space to the feature space. The choice of this kernel function is because of its mathematical simplicity and general classification capability. However, we need to evaluate other kernel functions. Some of the relevant kernel functions are:

- Polynomial kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + d)^d$
- Hyperbolic tangent kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\lambda \mathbf{x}_i \cdot \mathbf{x}_j + c)^d$
- Laplacian kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\sigma |\mathbf{x}_i - \mathbf{x}_j|)^d$

For example, the polynomial kernel corresponds to a map Φ into a feature space spanned by the products of the input entries. Some pattern analysis techniques have shown the effectiveness of the polynomial kernel for recognition (Zhao et al., 2007). In our pattern-based MPS algorithm, we can analyze the classification accuracy obtained with different kernel functions. Moreover, we can combine multiple kernels in order to achieve higher accuracies. It can be done simply by presuming the kernels, or learning the mixture of kernels using techniques such as MKL (Multiple Kernel Learning), or PMK (Pyramid Match Kernel) (Lanckriet et al., 2004; Grauman and Darrell, 2007; Bosch et al., 2007).

Analysis on the effect of number of clusters on pattern reproduction

Changing the number of clusters affects the MPS simulation. Increasing this number results in higher computational costs due to the larger number of similarity searches required between the data event and the prototypes. Indeed, more accurate classifications are obtained due to the higher sharpness index of the prototypes. On the other hand, decreasing the number of clusters causes lower sharpness indexes, and therefore, lesser classification capability in finding the optimal cluster to the data event.

Therefore, a study needs to be performed to understand the effect of the number of clusters on the quality of the generated realizations. This study is not about the speed of the algorithm, but on the multiple-point reproduction of the training image statistics. One expects that selecting the maximum possible number of clusters entails the most accurate pattern similarity search, which means that the realizations have the highest possible multiple-point reproduction of the patterns of the training image. Whereas a lower number

of clusters decreases the pattern reproductivity. By quantifying the variability between the realizations and the training image (using the technique of Section 3.3.2), one can characterize the performance of the MPS simulation with respect to the number of clusters. We can then plot this measure of pattern reproduction according to the number of clusters. We expect this plot to have an elbow such that increasing the number of cluster above it shows insignificant effects on the multiple-point statistics of the realizations. This study, hopefully, aids in obtaining a heuristic measure on the number of clusters. Furthermore, it can provide an upper bound on the number of clusters. Overall, we will be able to have a better understanding of the effect of the number of clusters on the MPS simulation.

Better kernel k -means using spectral clustering

Kernel k -means is a clustering method introduced in Section 2.5. It performs clustering in the kernel-induced feature space to increase the classification accuracy. However, similar to all clustering algorithms, it suffers from the possible misclassifications introduced by the initialization. Convergence of the kernel k -means algorithm to the local minima highly depends on the initialization of the centroids.

In order to reduce the possibility of trapping in the local minima, one needs to obtain better initialization algorithms. We have already implemented an algorithm which works better than random initialization. The algorithm proceeds by random selection of the first centroid, and sequentially, it picks the next centroid as the point farthest away from all previous centroids. This method provides a better initialization for the centroids. In spite of this attempt, one requires more robust initialization algorithms.

Spectral clustering is a method that analyzes the eigen-structure of the similarity matrix (distance matrix) to group the points into similar clusters. It is less prone to the initialization. However, the eigenvalue decomposition of the distance matrix is computationally intensive. For example, a $n \times n$ distance matrix, requires $O(n^3)$ operations. Therefore, their application on large-scale problems, such as pattern-based MPS analysis, is limited. To circumvent this computational issue, one can use the Nyström method to extrapolate the solution using only a small set of the input patterns (Fowlkes et al., 2004). In other words, the Nyström method solves the problem for a small random subset of the input, and then, extrapolates them to the full dataset. Using this method, one would be able to take advantage of the appealing features of the spectral clustering to find a reasonably good clustering, which does not suffer from the initialization issue. Subsequently, one uses the

obtained centroids for the initialization of the kernel k -means algorithm.

Smarter path in lower multi-scale levels

Another interesting study is decreasing the randomness of the sequential path in the lowest multi-scale level. In MPS simulations, the highest multi-scale levels (multi-grid or multi-resolution) are the main stages for defining the spatial variability in the realizations. The final multi-scale level, on the other hand, is only for refining the small-scale features.

It has been observed that a random path in the lowest multi-scale level may produce artifacts, such as channel discontinuities, decrease in the lower proportion categories, or even low hard data conditioning capability. Therefore, one needs to search for a smarter path during the simulation. This path visits the nodes in such a way as to reduce the possible misclassification errors. For example, it can store the errors between the data events and the cluster prototypes in the previous multi-scale level, and then visit the nodes in an ascending order of the errors. Furthermore, it can give precedence to the hard data locations such that a better conditioning capability could prevail.

Hard data conditioning in the multi-resolution method

The novel approach of the multi-resolution simulation provides better pattern reproductivity and long range spatial continuity than the multi-grid setting. However, in the multi-resolution simulation, there is no one-to-one correspondence between the grid nodes of one scale with the other scales. One requires a mapping (transformation) between the scales. In our analysis, we use cubic interpolation in two or three dimensions. However, this method is not applicable to sparsely-sampled hard data.

To find an allocation for the hard data at different resolutions, we used the inverse-distance weighted interpolation for populating the lower resolution realization grids, $\mathbf{re}_{r>0}$. We demonstrated the applicability of such a technique in data conditioning. However, it would be beneficial to study other interpolation techniques. One simple example, as mentioned before, is to use simple kriging to interpolate a hard data value for the unsampled location, which is defined by the center of the neighboring grid node at the lower resolution. This technique takes care of the redundancy between the data, and can de-cluster the hard data.

Another idea for hard data interpolation in the multi-resolution setting is to use thin-plate splines (TPS) as the solution to the bi-harmonic equation (Wahba, 1990). For example in 2D, it provides a smooth surface with the minimum bending energy needed for matching the known hard data. Then, the values of the TPS surface can be assigned to the multi-resolution grid at their collocated grid nodes. Generally speaking, more research needs to be conducted on hard data conditioning in the multi-resolution setting.

Better template size selection

An automatic template size selection was introduced in Section 5.2. This method relies on the concept of entropy to characterize the information content of the patterns. It aids in finding the optimal template size for stationary training images.

The effectiveness of the methodology was demonstrated using a variety of training images. However, one needs to improve upon the concept of entropy in order to get a more robust measure of stationarity. There are two different ideas that can be followed for such a goal.

The first idea is to use the multiple-point entropy, instead of the one-point entropy measure. However, the exact computation of the multiple-point entropy is challenging. There are not enough replicates of each pattern configuration in the training image to allow inference of their probabilities. One needs to relax the statistics of the patterns to be able to calculate entropies (i.e., by assuming similar patterns to have the same statistics). In other words, calculating the entropy is easy if we consider only pixel-wise information, but it becomes harder when probabilities need to be computed for a multiple-point pattern. For example, in a two-point scenario, the probabilities used in the calculation of entropy should be obtained from the multiple-point histogram, using a two-point template. The existence of higher order statistics makes probability inference unattainable due to smaller occurrences of such statistics. One measure to alleviate this situation is to use the same pattern similarity measures (distance functions) to infer these probabilities. For each template size, we set a threshold such that all patterns whose similarity measures are above that threshold are assumed to fall into the same multiple-point category (or the same bin in a multiple-point histogram). This way of grouping the patterns allows inferring statistics from them, and therefore, calculating their multiple-point entropies. The procedure on selecting the optimal template size then follows the same steps as before; that is, a plot of entropy with respect to the template size is made, and the appropriate template size is revealed by the corresponding

elbow in the plot.

Generally speaking, entropy of the patterns of a training image provides a textural description of its features. The second idea is to go beyond entropy and characterize the training image with not just one, but a variety of textural descriptors. We have previously used entropy as the relevant feature, describing the input spatial model. In order to combine other textural properties, with their statistical distributions, we can take advantage of the ‘grey-level co-occurrence matrix’ (Haralick et al., 1973). These matrices, denoted by GLCM, simply show the distribution of co-occurring values at specific offsets. It shows the statistical properties of the training image similar to the way the search-tree of the snesim algorithm characterizes the MPS information. For a 2D $n \times m$ binary training image \mathbf{ti} , the elements of the GLCM matrix for offsets $\Delta \mathbf{x}$ and $\Delta \mathbf{y}$ are as follows:

$$C_{\Delta x, \Delta y}(i, j) = \sum_{p=1}^n \sum_{q=1}^m \begin{cases} 1, & \text{if } ti(p, q) = i \text{ and } ti(p + \Delta x, q + \Delta y) = j \\ 0, & \text{otherwise} \end{cases} \quad (8.3)$$

Therefore, the co-occurrence matrix characterizes the distribution of random variables in the local neighborhood of each point. We can thus generate the GLCM matrix for all the offsets (neighborhood); including all template sizes. This provides us with the means for analyzing the texture of the training image to find the optimal template size. There are several textural parameters, which quantify different information about the training images (analogous to Connors et al. (1984)). Some examples of textural properties that can be obtained from a $(L + 1) \times (L + 1)$ GLCM matrix G are listed in the table below:

Statistical Texture Feature	Definition
Uniformity (first order)	$f_1 = \frac{1}{N_c} \sum_{i=0}^L G(i, j)$
Uniformity (second order)	$f_2 = \frac{1}{N_c^2} \sum_{i=0}^L G(i, j)^2$
Homogeneity	$f_3 = \frac{1}{N_c} \sum_{i=0}^L \sum_{j=0}^L \frac{G(i, j)}{1+(i-j)^2}$
Correlation	$f_4 = \frac{1}{N_c} \sum_{i=0}^L \sum_{j=0}^L ijG(i, j)$
Energy	$f_5 = \frac{1}{N_c} \sum_{i=0}^L \sum_{j=0}^L G(i, j)^2$
Entropy	$f_6 = -\frac{1}{N_c} \sum_{i=0}^L \sum_{j=0}^L G(i, j) \log \left(\frac{G(i, j)}{N_c} \right)$

Table 8.1: Example of textural properties used to find the optimal template size.

By incorporating these measures for each size L of the GLCM matrix, one can characterize a plot where the elbow signifies the optimal template dimension for capturing the

stationary essence of the training image. One should note that, this analysis is now multi-dimensional. Therefore, to obtain a 2D plot of textural information versus the template size, one needs to find a one-dimensional representation of this multi-dimensional data using either the PCA or the kernel PCA method. In general, such an approach where all the textural properties are explored simultaneously is more informative than a simple entropy measure for template size selection.

Besides the improvements in template size selection, a methodology on obtaining an anisotropic template is required. Up to now, only isotropic templates in 2D are considered (i.e. the template size is always a square in the horizontal plane). One requires a methodology to characterize the stationarity of the training image in different directions, separately. One straightforward, but computationally expensive approach is to first find the square template size, and then refine the selection by analyzing varying anisotropies by the expansions or contractions around that template.

Finding the appropriate number of multi-scale levels

Both the multi-grid and the multi-resolution settings aid in reproducing the long range continuity of the features of the training image. Currently, the practitioner has to choose the number of multi-scale levels. There are no algorithms that can intelligibly recognize an appropriate number of levels for the multi-scale simulation.

Therefore, we require an algorithm that can analyze the geometrical structures of the training image, and select the optimal multi-scale level according to the given template size. In general, it appears that choosing three multi-scale levels is sufficient for most training images, and blindly choosing this number will be suitable for the MPS simulation. However, this convenience should not stop us from searching for the most appropriate level for each specific training image. This would bring us a step closer to our goal of having a parameter-free MPS algorithm.

One idea on how to learn this parameter is using the entropy function as a measure of complexity of the training image patterns at different resolutions. The entropy of the patterns at each resolution, with the given template size, may help in identifying the required scale of analysis for the specified training image.

Enhancements to the pattern similarity search

In an unconditional sequential simulation, a data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ is defined as the set of previously simulated values found in the template \mathbf{T} centered on the visited location \mathbf{u} where \mathbf{T} is the same template used to scan the training image.

Searching for the optimal pattern to these data events is the most time-consuming part of the DisPAT algorithm. In the proposed methodology, one sequentially visits the nodes of the simulation grid and extracts the data event. Then, the most similar prototype to the given data event determines the cluster, from which a pattern is randomly selected to be pasted on the realization grid. This process was shown to provide good pattern reproductivity in the generated realizations. However, more study needs to be conducted on different methods for finding the appropriate patterns.

One such technique that can significantly decrease the computational time of the similarity search is the PCA approach. This approach has been implemented in the DisPAT algorithm in SGeMS. The PCA-based approach is exactly similar to the original approach. The only difference is on searching for the most similar cluster prototype. In the beginning of the algorithm, using principal component analysis, the dimensionality of the cluster prototypes are significantly reduced. The appropriate dimensions are chosen with the maximum profile log-likelihood method (introduced in Section 2.3.3). Then, during the simulation, instead of searching in the original pattern space, one maps the data event to the PCA principal vectors; and only searches in the lower-dimensional space, which represents both the data event and the prototypes. This search is significantly faster due to the reduction in the dimensionality of the patterns.

However, such an approach only works when the data events are fully informed. The DisPAT implementation of PCA approach is only applied in the lowest multi-scale level, where the ‘coarse-grid rescaling’ method (Section 4.1.2) has fully populated all the nodes of the realization grid. Incomplete patterns (not fully informed data events) cannot be handled with this approach. No mapping is defined for such incomplete data events. Moreover, the quality of the realizations (in terms of pattern reproductivity and connectedness) is reduced with the PCA-based method. The reason is that the distance function, with different weights assigned to the nodes of the data event, is no more applicable in the PCA approach. One cannot force larger weights for the frozen nodes or the nodes containing the hard data. The algorithm must assume a constant weight for all the nodes in the data event; frozen or not. Therefore, in the future, one needs to study other possible avenues for enhancing the search

for the optimal pattern.

One such approach that can handle the incomplete data events and has a similar style to the original clustering-based approach is explained hereafter. However, only the preliminary experimental results are provided, but an extensive evaluation in a simulation framework is required in the future.

As mentioned before, an essential aspect for improvement in the proposed methodology is a procedure to handle these incomplete data events. Incomplete, in simulation terms, refers to a case where some of the nodes within a chosen pattern template have not been visited; hence, their values remain unknown. This situation is encountered when some nodes in a data event are already determined as being zero or one (in the binary case), but the rest of the nodes are yet unresolved. The idea is to take advantage of the readily-available lower-dimensional MDS representation of the patterns for the similarity search. Therefore, a solution to mapping an incomplete pattern to MDS space needs to be found. This will take advantage of the dimensionality reductions during the simulations in the same manner as the PCA-based approach, but with the ability to apply different weights for the nodes.

One simple idea for the binary case is to assign the prior proportion (the mean of the training image) for the unknown nodes right before mapping them into the MDS space. For the continuous case however, one can opt to calculate the distances only between the known nodes, since it approximately provides the same mapping to the MDS space.

During the simulation, the algorithm aims at finding the closest pattern to a specific data event. If one tries to map an incomplete pattern to MDS space using all the patterns in the pattern database, a different configuration needs to be found not only for the additional data event, but also for the patterns in the database. Hence, during every search for the most similar pattern, one new MDS mapping has to be executed over the entire database with the inclusion of the additional incomplete data event. Although the MDS technique has been tremendously improved in terms of computational speed, mapping an incomplete pattern for every single node in a large 3D simulation grid makes this approach infeasible.

One simple solution for this problem is to use the same idea of *SEQ*-MDS technique (introduced in Section A.1) to map only the new incomplete data event. This is done by randomly selecting a number of patterns (greater than the dimensionality of the MDS space) from the pattern database and combining them with the incomplete data event into one new set, and then, performing the *SEQ*-MDS method over this set. However, assuming the set to be composed of n patterns, each mapping will consist of n^2 distance calculations,

which is computationally expensive. In order to reduce the number of distance calculations in each loop of the algorithm, a pre-defined set of patterns needs to be found, such that their pair-wise distances can be stored prior to simulation. This way, the number of distance calculations will decrease dramatically. A random selection of these pre-defined patterns does not ensure an accurate MDS mapping. The pre-selected patterns need to effectively span the lower-dimensional space. Therefore, a fast k -medoid algorithm is applied on all points in the MDS space to find a representative set of patterns for future mappings. This is similar to finding the cluster prototypes, as in the previous approach. Algorithm 14 summarizes the procedure on handling the incomplete data events in MDS mapping.

Algorithm 14 Incomplete Pattern Mapping

Require: Pattern database $\mathbf{patdb}_{\mathbf{T}}$

- 1: Map the patterns $\in \mathbf{patdb}_{\mathbf{T}}$ $\xrightarrow{\text{MDS}}$ points in \mathbb{R}^d
 - 2: A subset of patterns $\mathcal{S} : \{s_1, \dots, s_{2d}\} \leftarrow$ using k -medoid algorithm in MDS space \mathbb{R}^d
 - 3: Store $D_{\mathcal{S}}^{2d \times 2d}$, dissimilarity distance Matrix, of set \mathcal{S}
 - 4: **for all** $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$: Incomplete data events during the course of simulation **do**
 - 5: Unknown nodes $\leftarrow 0.5$
 - 6: $\mathcal{S}' \leftarrow \mathcal{S} \cup \mathbf{dev}_{\mathbf{T}}(\mathbf{u})$
 - 7: Construct $D'_{\mathcal{S}}$ by using $D_{\mathcal{S}}$ and \mathcal{S}'
 - 8: SEQ-MDS mapping of $\Delta'_{\mathcal{S}}$
 - 9: obtain $\mathbf{dev}_{\mathbf{T}}^{MDS}(\mathbf{u})$
 - 10: Find the closest pattern by a search in the lower-dimensional MDS space
 - 11: Paste it on the simulation grid
 - 12: **end for**
-

In summary, after mapping all the patterns to MDS space, a set of representative points will be selected by the k -medoid algorithm, and future mappings of data events will be carried out by the SEQ-MDS technique; using the small representative subset of the patterns and the data event.

In general, selecting a subset of patterns for mapping will result in an approximation as compared to using the entire database. However, it is believed that the selected subset of patterns is the one that will minimize the global reconstruction error between the mapped patterns using the entire dataset, and the mapped patterns using the representative set.

In order to test the effectiveness of this algorithm, the channelized 2D training image (sand/shale) has been used. 2209 patterns of size 9×9 were mapped to a 12-dimensional MDS space, and then, the k -medoid algorithm was applied to find a subset of 60 representative patterns. In an ideal case, by selecting a complete pattern from the database and

applying the proposed algorithm, the true coordinates of the mapped point is obtained (the ‘true coordinates’ refers to the case when the whole dataset is used for mapping). In order to test this technique, a random pattern is selected from the database and is mapped to the MDS space using the proposed methodology, and its reconstruction error is examined. The results are shown in Figure 8.1. Here, in order to visualize the 12-dimensional MDS space, a parallel coordinate is plotted where the x -axis represents the coordinates. The gray lines represent the entire database of patterns, and the black line and red line show the coordinates of the original pattern and the approximate reconstructed coordinates of the resulting pattern, respectively. Three different test patterns were chosen, and their mapped MDS coordinates are shown. Clearly, the proposed algorithm provides a fairly good approximation in terms of MDS mapping. It should be mentioned that the first coordinates (with smaller x values) have greater significance on the results. This is because of the larger influence of the largest eigenvalues representing the first few dimensions, as shown in Figure 8.1.

Another experiment is conducted to assess how the proposed algorithm handles incomplete data events. Some nodes in the middle region of a pattern are set as uninformed. Accordingly, a value of 0.5 is assigned to these uninformed nodes. The applicability of the method is assessed by mapping the incomplete data events to MDS space with the proposed approach. Next, the search for the closest point is carried out in the lower-dimensional MDS space. Then, the closest pattern corresponding to the closest point is selected as the approximate outcome of the mapping. Figure 8.2 illustrates two examples exhibiting different uninformed set of nodes. It is observed that even in an incomplete case, one can reconstruct the missing nodes reasonably well. It should be kept in mind that the approximate mapping does not affect the simulation capability of the MPS method. This is because the previous MPS procedure, first, tries to find the closest cluster prototype to the data event, and then a pattern is randomly selected from that cluster. As a result, there is also a significant approximation on the optimal pattern to be pasted on the realization grid.

In the future, one needs to evaluate how the approach works in a simulation framework, and how good the pattern reproductivity of the realizations will be. This is an essential avenue for future research; to have better and faster pattern similarity searches.

Novel method for generating patterns

Future research should also be focused on automatically finding the most suitable pattern for a data event. These patterns need not exist in the pattern database, but should be similar

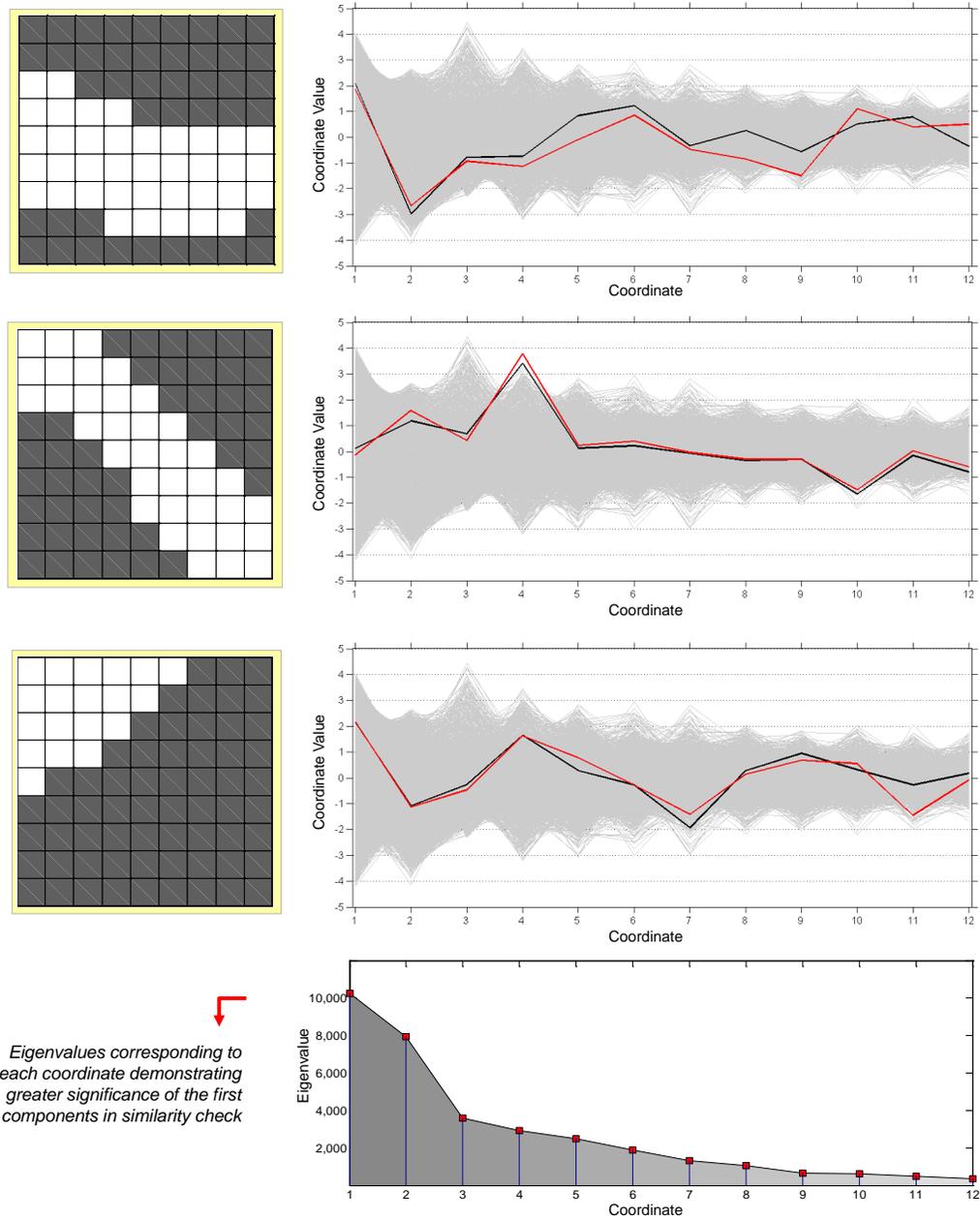


Figure 8.1: The application of new pattern mapping (data event) to MDS space using a subset of patterns by the proposed *SEQ*-MDS method. The patterns from the database are shown on the left, and the 12-dimensional MDS coordinates are shown on the right. Gray lines represent the entire database, black line represents the true coordinates, and red line represents the approximate reconstructed coordinate.

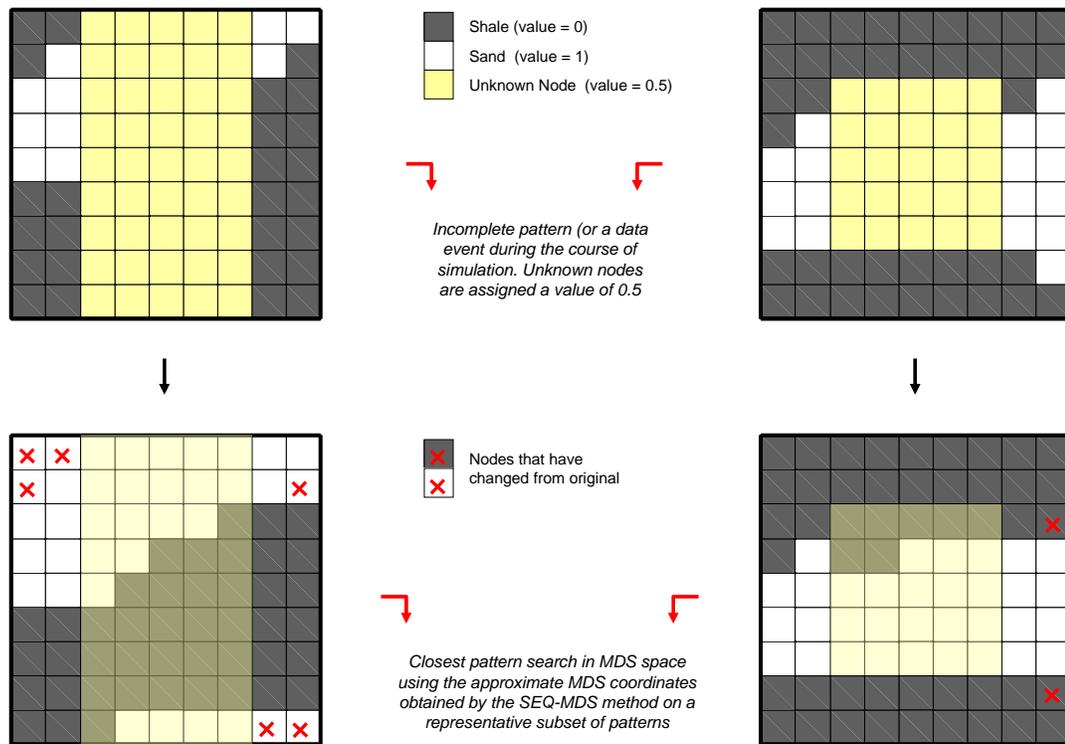


Figure 8.2: The application of incomplete pattern mapping to MDS space using a subset of patterns. Uninformed nodes are represented by yellow, and the closest pattern to the obtained MDS point is shown.

to the structures seen in the training image. It is our opinion that any improvement on these algorithms has to start by considering ways to extend the pattern database by generating new patterns. The generated pattern should have similar multiple-point statistics to the ones given by the training image. In these approaches, the pattern database is not entirely eliminated. It will be used to learn the variability among the patterns, so that new patterns can be generated.

In principle, having a pattern generating framework is desirable. However, one should also analyze the effectiveness of such approaches. The question is how beneficial they are in terms of multiple-point pattern reproduction. Upon finding a technique to generate new patterns, a quantitative analysis should be made with respect to the multiple-point histograms of the realizations and the training image. One may reach the conclusion that such an approach is redundant, and a simple use of the original patterns is adequate for

accurately modeling the desired spatial variability. Preliminary studies on this subject show insignificant improvements, which may just be due to the inferior performance of the tested algorithms.

Therefore, new pattern generation methodologies should be developed in the future. Three different ideas are outlined below. All these ideas are based on generating the pattern from a set of existing ones (from the patterns in the database). None of these methods attempt on learning any statistical function of spatial continuity. Instead, they morph the patterns in order to generate new ones. In other terms, they provide a *continuous interpolation* over the pattern space.

1. The first algorithm works by generating a new pattern from three existing patterns in its vicinity in MDS space. The methodology proceeds by extracting the data event and finding the most similar cluster prototype to it. Next, we sample a point from that cluster. Here, a point represents a location in the MDS space which belongs to the same cluster. In other words, considering the representation of those patterns in the MDS space, one can use a sampling method (i.e. Latin Hyper-cubic Sampling) to obtain a new point.

Next, we need to find a pattern that maps to that point. In other words, we need a technique for back-mapping from the MDS space to the original pattern space. There are a variety of techniques that can aid in such a mapping. One suggestion is to use Radon transformation. The idea behind back-mapping comes from the fact that the features in this newly generated pattern should comply with the features present in the patterns existing in the database. However, in order for this new pattern to be mapped to that specific point, it should have the same features existing in the patterns closest to that point. The closest points to our sampled point in the MDS space correspond to the patterns which have features similar to the new pattern. On that account, the three closest points to the sampled point are chosen for pattern generation. Their Radon transformations are obtained. Then, using an inverse-distance weighting scheme, their transformations are averaged out into a new transformed image. Finally, using inverse Radon transformation (filtered back-projection algorithm), one can obtain a pattern corresponding to the newly sampled point. For details of such an algorithm and examples, please refer to Honarkhah and Caers (2008).

2. The second algorithm takes advantage of the kernel space representation of the patterns. This method, however, works differently than the previous algorithm. One can eliminate the need for any cluster prototype for generating a new pattern. It works by calculating the distances between the data event and all the patterns of the training image. It then uses these distances, similar to the proposed distance-based modeling framework of DisPAT, to map the data event into the kernel-induced feature space. Therefore, a new point in feature space, which represents the most appropriate pattern for the given data event, is obtained. Similar to the previous approach, one needs a technique to find the corresponding pattern by mapping back from the feature space to the original pattern space. For this matter, we can use the pre-image problem in kernel space. In solving the pre-image problem, one finds directly the pattern-based on the distance constraints in the feature space. This provides a non-linear set of weights on the existing patterns that are used to combine the patterns into a new one. For binary images, a thresholding of the final pattern will also be required.

This approach is however very time-consuming. The reason lies in the fact that, to obtain a point in the feature space, corresponding to the data event, an exhaustive search over all the patterns in the database is to be performed. So even though the method is sound in theory, in practice, it suffers from high computational complexity of the distance computations at each grid node. In order to make the method applicable, one needs to try other approximate kernel methods. One such approach is the reduced set methods (Burgess and Schölkopf, 1997; Osuna and Girosi, 1999; Zeng et al., 2006). These methods try to approximate the kernel matrix by a reduced set of inputs, that is, the dimensionality of the kernel matrix is reduced. The selected set should therefore be representative of the entire domain. They form the most optimal basis for the feature space; such that the distance calculations on this reduced set are sufficient to accurately map the data event into the feature space. In mathematical terms, they find the appropriate $\{\beta_k, \mathbf{z}_k\}$, such that, $\sum_{j=1}^N \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}) \approx \sum_{k=1}^M \beta_k \kappa(\mathbf{z}_k, \mathbf{x})$, with $M < N$ could be a good approximation.

This way, the calculation related to the feature space, such as the kernel mapping and the pre-image problem, can be cast within the reduced set. Hence, the desired speed of the MPS simulation can be obtained with such a pattern generating approach.

3. The third algorithm is similar to the first proposed algorithm using the Radon transformation. However, in this algorithm, one attempts to directly morph the patterns into each other. This morphing will generate new patterns that can better match the data events at hand. The algorithm uses non-rigid image transformation to obtain elastic deformable patterns. In other words, one would be able to morph one pattern into another. One suggestion is to use the spline-based registration methods where a set of control points are used between the source and the target pattern to find a correspondence between all the data nodes. One popular spline-based method is the Thin-Plate Spline (TPS) transformation (Bookstein, 1989). The Thin-Plate Spline warping algorithm, in 2D, tries to find a smooth transformation which results in the minimum curvature over the deformed image surface. The poly-harmonic function can be used for three dimensional cases.

The algorithm thus proceeds, as before, by extracting the data event and finding the most similar cluster to it. From within that cluster, two patterns are randomly selected. These two patterns are used for the morphing using the TPS formulation. However, as mentioned before, the TPS interpolation technique requires a set of control points for learning the smooth mapping. This allows the algorithm to apply the mapping on all the nodes of the pattern template, for obtaining the desired elastic deformation.

For this method, we can rely on a general pre-specified set of control points. For example in 2D, they can be defined by five points scattered regularly on the pattern template (i.e. one located at the center, and the other four at the corresponding centers of the four quadrants of the pattern template). Such a general set of control points allows one to morph two patterns into each other. The most optimal pattern to be pasted on the realization grid will be the morphed pattern, which leads to the minimum dissimilarity distance with the data event. Therefore, not only new patterns can be generated, but also the pattern searches to the data events will have the least possible error.

Other smarter techniques on finding the control points can also be developed. Overall, this technique is applicable in pattern-based MPS modeling and needs to be investigated extensively.

Improvements on the non-stationary modeling

We introduced novel methods on modeling non-stationarity in training images. Future work on non-stationarity can follow a similar procedure of embedding the spatial components of the patterns into MPS simulations.

The first improvement can be achieved by incorporating all the three methods (the SSM, NRM, and ASM methods) jointly into a single new method. This method will benefit from the strong capability of the SSM method on modeling any non-stationary phenomena, while at the same time, the advantages of the automatic segmentation in the ASM method are exploited for increasing the stochasticity of the generated realizations.

Conceptually, the method works by defining a new set of functional for integrating the spatial components. One simple solution is to relax the ASM method. In the ASM method, each location belongs to a specific region, and only that region is used for the pattern similarity search. The new method should relax this assumption and search over all the patterns in the training image. However, the functional provides higher weights for the patterns located in their equivalent regions, and reduces the weight of other regions (according to the similarity of those regions with the principal region). This task can be achieved by computing a functional that provides different weights for each region (at each visited node during the simulation), and multiplying it by the functional of the SSM method. The integration of the proposed non-stationary modeling technique into one unifying framework is essential in future research.

Another similar enhancement is to integrate the NRM method with the ASM method. In the NRM method, a pre-defined neighborhood radius is used for defining a subregion to confine the pattern similarity search, at each visited node location. Mixing The NRM technique with the automatic-segmentation method (ASM) entails an approach, where instead of having a pre-defined region area at each location, an equivalent stationary region, whose size is not subjectively pre-determined, is considered. This method thus has both the non-stationary capability of the NRM method, and the quasi-stationary assumption of the ASM method.

Finally, one can develop techniques that can automatically select the appropriate non-stationary modeling framework according to the provided training image. These avenues of research should be deeply investigated.

Appendix A

Algorithmic Enhancements

A.1 *SEQ*-MDS Algorithm

One of the challenges that might make the multi-dimensional scaling method inapplicable is the computational cost associated with the MDS mapping. Improving the multi-dimensional scaling algorithm is essential for achieving reasonable run-times in MPS simulations. Moreover, the huge number of patterns in a 3D training image renders the MDS algorithm extremely memory-demanding. The matrix calculations involved in this mapping, which are at least quadratic in nature, have an adverse effect on the computational efficiency of MPS methods. Therefore, the number of analyzed patterns is restricted by the high computational complexity of MDS. These limitations can be enumerated as follows:

- Distance matrix takes huge amount of memory as the number of patterns increases by the increase in the size of training images.
- Singular Value Decomposition (SVD) of matrices becomes prohibitive by the increase in pattern dimensions or number of patterns.

In order to make metric MDS applicable for large training images, a new metric MDS method is developed. This technique will resolve both the limitations mentioned above. The starting idea for improving MDS speed is the anchor-point MDS method (Buja et al., 2008, 2001). In this method, data are grouped into clusters, some being anchors, and some floaters. Anchors provide the coarse-scale output layout, and floaters refine the fine-scale structures. The mapping is then performed sequentially on each cluster. The intermediate

steps consist on dividing the dataset into many clusters and employing each one separately, where the anchor points ensure a correct coarse layout of the points. In order to increase the speed to the greatest extent possible and also reduce the memory issues inherent in an anchor-point multi-dimensional scaling algorithm, the mapping has to be performed sequentially (*SEQ*-MDS method). In next sections we will first provide the mathematical formulation of this method, and then, state the conditions needed for accurate mapping and the computational complexity associated with it. Next, an example on the applicability of the technique will be given and compared to traditional metric MDS mappings.

A.1.1 Methodology

In order to show the methodology, we define $\mathbf{x}_i \in \mathbb{R}^{n_T}$ as the pattern coordinates (according the template \mathbf{T}), for all N patterns ($i = 1, \dots, N$) in $\mathbf{patdb}_{\mathbf{T}}$, where $n_T \ll N$. Applying multi dimensional scaling on large number of patterns is computationally demanding. In order to simplify the methodology, we assume that the patterns are divided into two sets $\mathbf{x}_i^1 \in \mathcal{S}_1$ and $\mathbf{x}_i^2 \in \mathcal{S}_2$, in such a way that, at least, n_T of the individual patterns belong to both sets. Therefore, these two sets of \mathcal{S}_1 and \mathcal{S}_2 are overlapping, and $\mathcal{S}_1 \cap \mathcal{S}_2 \neq \emptyset$. The main idea in *SEQ*-MDS approach is to individually apply MDS to each of the sets to get their representation in the lower-dimensional MDS space, and then, use the coordinates of the overlapping patterns $\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathcal{S}_1 \cap \mathcal{S}_2\}$ to combine the two representations into one.

One of the challenges in classical MDS on large pattern databases is the size of the dissimilarity distance matrix. However, by splitting the pattern database into smaller subsets, one only needs to store the distance matrix of those subsets. For example, a pattern database of size N would unequivocally need $N \times N$ storage space for the distance matrix. By just splitting the pattern database into two equally-sized subset of size $\frac{N}{2}$, one only needs $\frac{N}{2} \times \frac{N}{2}$ storage space during the *SEQ*-MDS method (we assumed non-overlapping sets for the sake of computation). Moreover, while the classical approach required N^2 distance calculations, the *SEQ*-MDS approach can obtain similar results with only $2 \left(\frac{N}{2}\right)^2$ calculations. This computational advantage brought upon by the *SEQ*-MDS technique is of paramount importance for handling large pattern databases in the proposed pattern modeling framework.

The question still remains on how to combine any two subsets into one. Assume that

each of the two subsets of \mathcal{S}_1 and \mathcal{S}_2 are mapped to MDS space. We denote their configurations in the d dimensional MDS space as follows: $\mathbf{y}_i^1 \in \mathbb{R}^d$ for \mathcal{S}_1 , and $\mathbf{y}_i^2 \in \mathbb{R}^d$ for \mathcal{S}_2 . In order to combine the two sets, we would use the coordinates of the anchor points. Since anchor points exist in both sets, we can find an affine transformation that will try to map the anchor points within one set to their own configuration in the other set. This affine mapping can be represented by $\mathbf{y} \mapsto \mathbf{A}\mathbf{y} + \mathbf{b}$ such that $\forall \mathbf{y}_k \in (\mathcal{S}_1 \cap \mathcal{S}_2) : \mathbf{y}_k^1 \simeq \mathbf{A}\mathbf{y}_k^2 + \mathbf{b}$.

In order to obtain \mathbf{A} and \mathbf{b} , we can re-write the affine transformation equation as follows:

$$\mathbf{A} (\mathbf{y}_k^1 - \mathbf{y}_k^2) = \mathbf{b} \quad (\text{A.1})$$

This is a system of linear equation. In order to find the least square solution for this system, we can use QR decomposition (Wahba, 1990). Assume \mathbf{Y}_1 and \mathbf{Y}_2 are matrices whose columns are the coordinates of the anchor points obtained by applying MDS to the two data sets, and $\bar{\mathbf{Y}}_1$ and $\bar{\mathbf{Y}}_2$ represent the means of \mathbf{Y}_1 and \mathbf{Y}_2 , respectively. The centered data in each of the sets would result in same orthogonal basis for representation. Therefore, we can apply QR decomposition to the centered data: $\mathbf{Y}_1 - \bar{\mathbf{Y}}_1\mathbf{1}^T = \mathbf{Q}_1\mathbf{R}_1$ and $\mathbf{Y}_2 - \bar{\mathbf{Y}}_2\mathbf{1}^T = \mathbf{Q}_2\mathbf{R}_2$. With this configuration, the upper triangular matrices of \mathbf{R}_1 and \mathbf{R}_2 should be similar. Therefore,

$$\mathbf{Q}_1^T (\mathbf{Y}_1 - \bar{\mathbf{Y}}_1\mathbf{1}^T) = \mathbf{Q}_2^T (\mathbf{Y}_2 - \bar{\mathbf{Y}}_2\mathbf{1}^T) \quad (\text{A.2})$$

Solving this equation will provide us with the affine operator $\mathbf{A} = \mathbf{Q}_1\mathbf{Q}_2^T$ and the shifting operator $\mathbf{b} = -\mathbf{Q}_1\mathbf{Q}_2\bar{\mathbf{Y}}_2 + \bar{\mathbf{Y}}_1$.

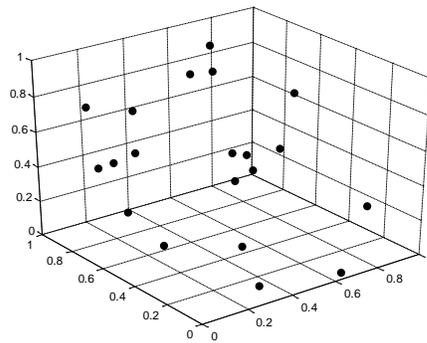
Before going into mathematical details, let's illustrate the technique with simple visualization. Assume that we have 20 points (Figure A.1(a)) in \mathbb{R}^3 , and we would like to map them to a 2D MDS space, \mathbb{R}^2 . In *SEQ*-MDS procedure, for simplicity, we divide the 20 data points into two clusters such that each of them share four similar anchor points. We thus have two overlapping sets of $\{\mathbf{x}_1, \dots, \mathbf{x}_{12}\} \in \mathcal{S}_1$ and $\{\mathbf{x}_9, \dots, \mathbf{x}_{20}\} \in \mathcal{S}_2$, and the intersection of these two sets is $\{\mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\}$. These two sets are shown in Figure A.1(b) and Figure A.1(c), where black dots represent the set itself and the red dot represents the overlapping points. All the red dots in that figure correspond to the same four anchor points of $\{\mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\}$. Applying MDS to each of these sets will provide a 2-dimensional representation for that set as shown in Figure A.1(d) and Figure A.1(e). Indeed, the red dots represent the overlapping set (the anchor points). Therefore, we can now use these

anchor points to rigidly map (affine transformation) one overlapping set representation to the other one. After learning the affine mapping, it will be applied to non-anchor points in the corresponding set. This results in the coordinates of all initial 20 points within the MDS space (Figure A.1(f)). As one can observe, not only there is a limitation in the accuracy of the affine mappings between the anchor points, but also, it can have a huge impact on the final resulting MDS coordinates if errors propagate. In the next subsection we will discuss the conditions for an accurate mapping using *SEQ*-MDS method.

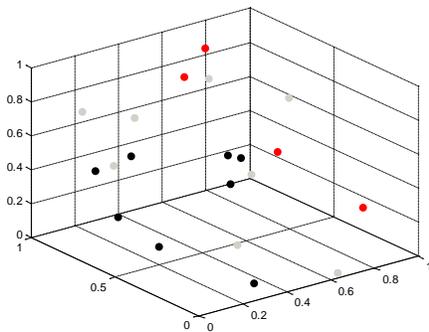
A.1.2 Condition for Accuracy

In the previous section, we discussed the procedure for applying MDS technique on a data set by splitting them into two intersecting subsets. In pattern-based approaches, the objects are patterns. This can lead to a huge number of data for MDS mapping. Splitting into two subsets would still not provide us with an efficient algorithm. In these cases, where large data sets are involved, the data are split into several overlapping subsets instead of just two. For example, according to the number of data and their dimensionality, Data are divided into k overlapping subsets $\{s_1, \dots, s_k\}$, such that $s_i \cap s_{i+1} \neq \emptyset$. The same procedure for *SEQ*-MDS approach of previous section can then be applied on k subsets. We assume the first subset, s_1 , as the central subset, and all the rest $\{s_2, \dots, s_k\}$ are separately selected to be combined with s_1 by using the *SEQ*-MDS method and mapped into lower-dimensional space. This procedure is repeated until all data are combined within the MDS space.

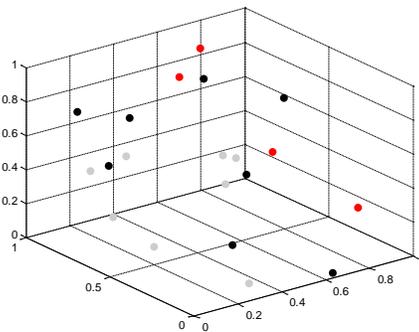
In order to understand the factors that affect the performance and accuracy of *SEQ*-MDS method, one needs to understand its approximations and limitations when data are split into k overlapping subsets. One of the main factors is the ability to find a correct affine mapping to transform the data from one subset to another. The accuracy of this transformation greatly affects the resulting lower-dimensional configurations. For example, if the anchor points existing in two different subsets cannot be matched together by the affine transformation, the resulting layout of the points would not be what the classical MDS provides. Therefore one needs to choose the anchor points to have maximal separation in space to increase the possibility of finding efficient mappings. The other factor that is crucial to the algorithmic performance of *SEQ*-MDS method is the number of anchor points. Let's assume that we would like to map the data into a three dimensional MDS space, \mathbb{R}^3 . If we choose only one anchor point for finding an affine mapping, not just the QR decomposition



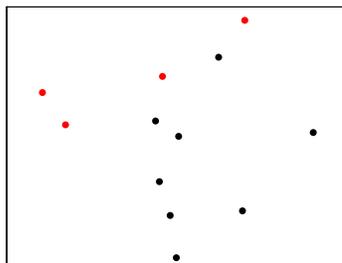
(a) Original set of 20 points $\in \mathbb{R}^3$



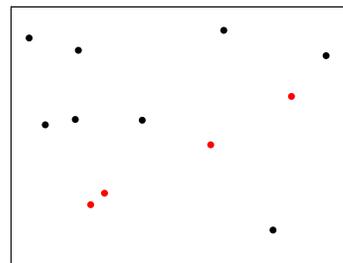
(b) Set s_1



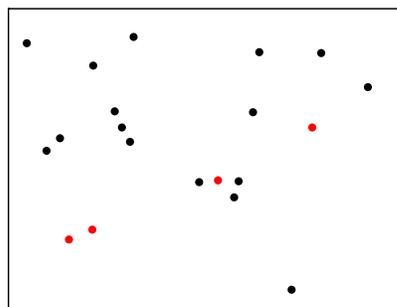
(c) Set s_2



(d) 2-d MDS of s_1



(e) 2-d MDS of s_2



(f) 2-d MDS space of all 20 points

Figure A.1: Simple procedure of *SEQ*-MDS method with four similar red points as anchors.

will fail to provide sufficient rank for the transformation, intuitively speaking, it would not even be able to provide a unique map. Having one anchor point between two datasets in \mathbb{R}^3 , one would only be able to resolve the translational component of affine mapping. However, rotational and scaling transformations of the data points remain arbitrary and unknown. The same argument holds true for the case where we have chosen two anchor points for transformation. In that case, two points represent a line. Therefore by finding a mapping from one line in \mathbb{R}^3 into another line, one would only be able to understand the translational and scaling components, and the rotational component of the mapping is set for only one orthogonal direction. The other orthogonal rotational component would still remain unknown and unresolved. Therefore, for *SEQ*-MDS algorithm to work and provide accurate mappings, the number of anchor points needs to be greater than the MDS dimension. The two conditions needed for accurate *SEQ*-MDS mapping is summarized as follows:

- Good spread of the selected anchor points over all dataset.
- Number of anchor points being greater than the MDS dimension, d .

A.1.3 Computational Complexity

Assume N to be the number of patterns in \mathbf{patdb}_T of a training image, N_a be the number of patterns used as anchor points between subsets, and N_s be the number of patterns in each subset. By this definition, the number of distinct patterns in each subset is equal to $N_s - N_a$, and assuming k overlapping subsets, we have $k = \frac{N - N_a}{N_s - N_a}$.

According to *SEQ*-MDS procedure, we split the data into k overlapping subsets, and apply classical MDS technique to each of them individually. Classical MDS has a computational complexity of $O(N_s^3)$ to find the configuration of points within each subset in MDS space. This will result in $k \times O(N_s^3) = \frac{N - N_a}{N_s - N_a} O(N_s^3)$ operations. On the other hand, for each subset of $\{s_2, \dots, s_k\}$, one needs to apply *QR* factorization to obtain the corresponding affine transformation for the anchor points. This has a computational complexity of $O(N_a^3)$, which has to be performed $(k - 1)$ times. Thus *QR* decomposition will need an overall $(k - 1)O(N_a^3)$ operations. Therefore the total computational time is approximately:

$$O(\text{SEQ} - \text{MDS}) \simeq \frac{N - N_a}{N_s - N_a} O(N_s^3) + \frac{N - N_s}{N_s - N_a} O(N_a^3) \quad (\text{A.3})$$

And assuming the number of anchor points being equal to the dimensionality of the final MDS Space, \mathbb{R}^d , we can approximate $N_a = d$. Moreover, assuming $N_s = \alpha N_a$ and $d \ll N$, we can simplify the computational complexity to:

$$O(SEQ - MDS) \simeq \frac{N-d}{(\alpha-1)d} O(\alpha^3 d^3) + \frac{N-\alpha d}{(\alpha-1)d} O(d^3) \simeq O(d^2 N) \quad (\text{A.4})$$

Therefore the computational complexity of *SEQ*-MDS is $O(d^2 N)$, instead of $O(N^3)$ in the classical algorithm. However, the computational complexity can be further reduced by using a faster classical MDS algorithm, which can be accomplished by a faster eigenvalue decomposition of the data matrix using the Arnoldi method (Lehoucq and Sorensen, 1996). The reason for this speed improvement lies behind the fact that, in classical MDS, one calculates all the eigenvalues of the covariance matrix of the dataset. However, since the dimensionality of the MDS space (\mathbb{R}^d) is assumed known a-priori, there is no need to calculate all the eigenvalues for each subset; only the d biggest values that are needed for the calculations are retained. This further reduces the computational complexity of *SEQ*-MDS from $O(d^2 N)$ to $O(dN)$.

Another improvement in computational complexity originates from the reduction in the number of distance calculations. For classical MDS, one needs to calculate the pair-wise distances between all the patterns in the database. Assuming N patterns, each being n_T dimensional, constructing the distance matrix requires $\frac{N(N-1)}{2} \times n_T$ calculations, minimum. On the other hand, the *SEQ*-MDS method divides the pattern database into k overlapping sets. Therefore, pair-wise distance calculations should only be performed on each subset individually. If, for simplicity of calculations, we assume zero anchor points between subsets, one requires $k \times \frac{N_s(N_s-1)}{2} \times n_T$ calculations only. *SEQ*-MDS method provides an improvement by a factor of k over the classical MDS technique for constructing the necessary distance matrices.

The methodology for the sequential MDS method is outlined in Algorithm 15 as a reference.

A.1.4 Experimentation

In order to experiment the accuracy of this approximate, but fast, MDS algorithm, the correlation coefficient between the distances of the original set of patterns and the dimensionality-reduced set of data points will be analyzed. A comparison will be made with the classical

Algorithm 15 *SEQ*-MDS mapping**Require:** Set s containing all the patterns

- 1: $s_1 \leftarrow$ randomly select N_s patterns from s
- 2: $X_1 \leftarrow$ randomly select N_a anchor patterns from s_1
- 3: $Y_1 \leftarrow$ Do MDS on s_1
- 4: **for** $k = 2, \dots, N$ **do**
- 5: $s_k \leftarrow$ randomly select $(N_s - N_a)$ patterns from $s \cap \left(\bigcup_{i=1}^{k-1} s_i\right)$
- 6: $s_k \leftarrow s_k \cup X_1$
- 7: $Y_k \leftarrow$ Do MDS on all $X_k \in s_k$
- 8: $Y_k \leftarrow$ Do affine transformation using $\{\forall j : Y_{1,j}, Y_{k,j} | X_j \in (s_1 \cap s_k)\}$
- 9: **end for**
- 10: **return** Y as the MDS mapping

MDS algorithm (exact method) and will be used to quantify the exactitude of the *SEQ*-MDS methodology. The mapped coordinates with classical MDS can be used to construct a distance matrix. The same distance matrix can also be constructed using the coordinates obtained through *SEQ*-MDS method. The correlation coefficient is thus obtained by finding the correlation between the distance values within these two matrices. A high correlation coefficient means that the distances obtained through *SEQ*-MDS method are in agreement with the classical method, and a low correlation coefficient reveals the inaccuracies produced through the anchor point locations and affine mappings.

The training image shown in Figure 2.3(a) is used for this experiment. A set of 2209 patterns are obtained by scanning the training image of size 101×101 with a template of size 9×9 . After some preliminary analysis, the most efficient set of values for each grouping, N_s , and for the number of intersection point, N_a , were chosen as follows: $N_s = 200$ and $N_a = \frac{3}{2}d$, where d is the dimension of the MDS space ($d = 12$ for this training image). The correlation coefficient is calculated according to the formula below:

$$\rho_{D_X, D_Y} = \frac{E(D_X D_Y) - E(D_X) E(D_Y)}{\sqrt{E(D_X^2) - E^2(D_X)} \sqrt{E(D_Y^2) - E^2(D_Y)}} \quad (\text{A.5})$$

where, D_X and D_Y indicate the values of dissimilarity distance matrices on the sets X and Y , respectively. X represents the original patterns and Y represents the set of points in MDS space.

The result of the mapping using both the classical CMDS and *SEQ*-MDS are shown in Figure A.2. Clearly, the cloud of data points in both mappings are perceptually identical.

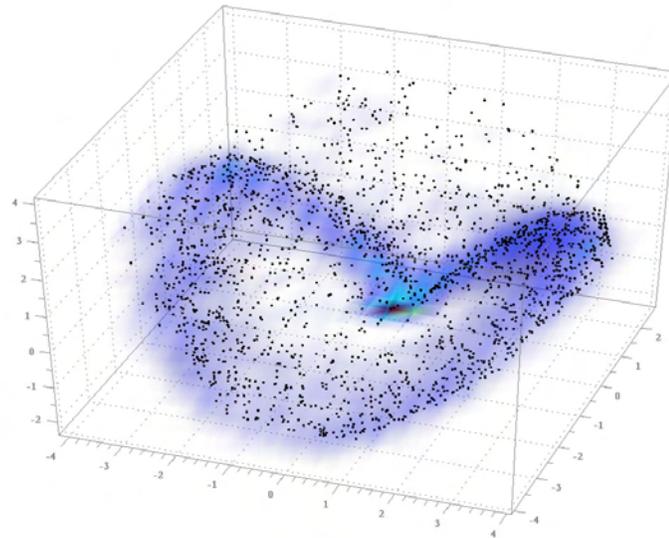
It should be noted that these data are originally 12 dimensional, but only the first three principal components are shown in this plot. The correlation coefficient obtained using classical MDS is $\rho_{\text{CMDS}} = 98.98$ and for sequential MDS is $\rho_{\text{SEQ-MDS}} = 94.5$. Thus, there is a reasonable agreement between the exact and approximate methods. A better measure for comparison would be *Kruskal's* goodness of fit index (STRESS). The formula for STRESS is as follows:

$$\text{STRESS} = \sqrt{\frac{\sum_{i,j} (d_{i,j} - \tilde{d}_{i,j})^2}{\sum_{i,j} d_{i,j}^2}} \quad (\text{A.6})$$

where \tilde{d} represent the distances calculated from the configuration of points in the lower-dimensional MDS space, instead of the original high-dimensional one.

The STRESS of computing errors for *SEQ-MDS* is only 0.1008 and the STRESS for classical CMDS is 0.099. As can be seen, the errors are infinitesimal. Therefore, the proposed *SEQ-MDS* methodology is extremely fast and reasonably accurate.

Next, the speed of *SEQ-MDS* is compared with classical MDS (CMDS). A set of random vectors in a 81-dimensional space is created with the sample sizes ranging from 300 to 3000 points. Because of the hardware limitation to process CMDS in Matlab, a maximum set of 3000 sample points is used to compare *SEQ-MDS* performance with CMDS. In Figure A.3, the black line (CMDS) is produced by Matlab default script, and the red line refers to the *SEQ-MDS* method. It is noticeable that the simulation result matches the theoretical analysis. In Figure A.3(b), it can be verified that the slope of the two lines correspond to the computational complexity of the methods. For CMDS, the algorithm is approximately $O(N^{3.1})$ and *SEQ-MDS* has $O(N)$ complexity, which is a significant performance improvement for distance-based pattern modeling. For instance, the number of patterns in a 3D training image of size $100 \times 100 \times 100$ is approximately 10^6 . MDS mapping in Matlab over this large pattern database would take around 13 hours to finish (if there were no memory limitations), but using the proposed technique, it would take less than 3 minutes on the same machine.



(a) Classical MDS

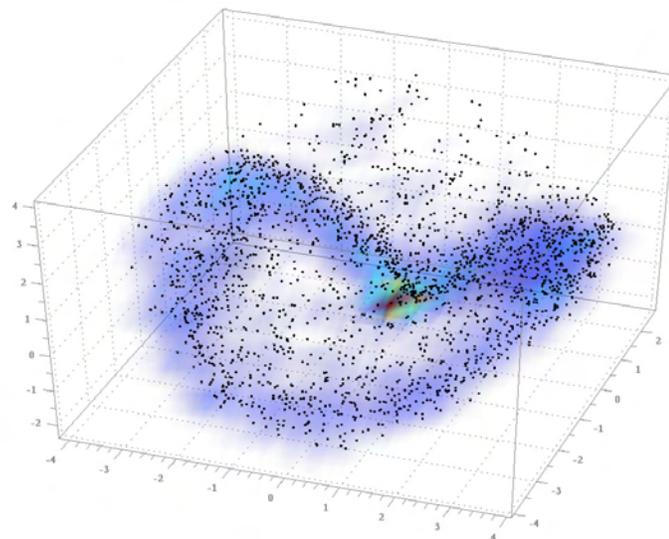
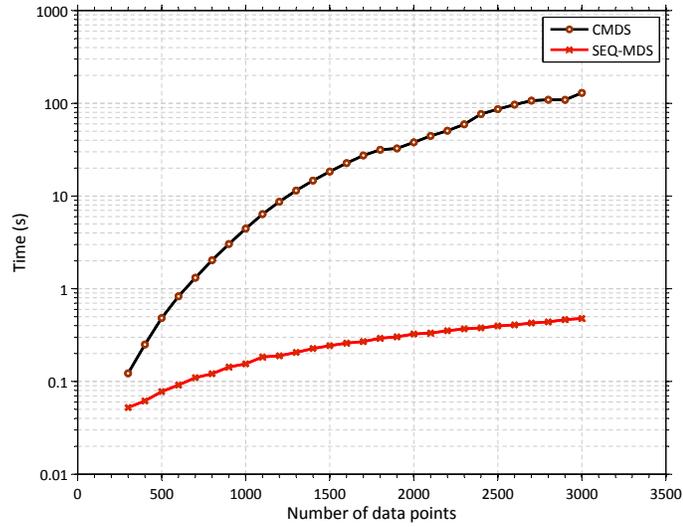
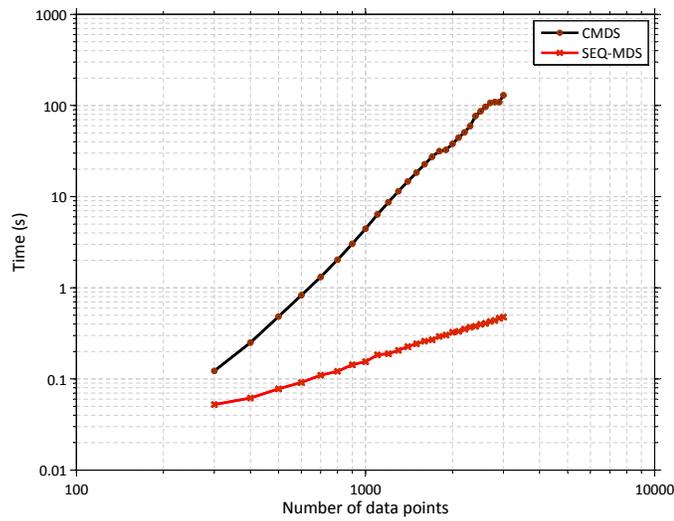
(b) *SEQ*-MDS

Figure A.2: mapping 2209 patterns with the dimensionality of 9×9 , to a 12 dimensional MDS space, using (a) classical MDS and (b) *SEQ*-MDS algorithms. This mapping shows the similarity in the layout of MDS produced by the sequential MDS method to the classical MDS layout.



(a) Semi-Log Plot



(b) Log-Log Plot

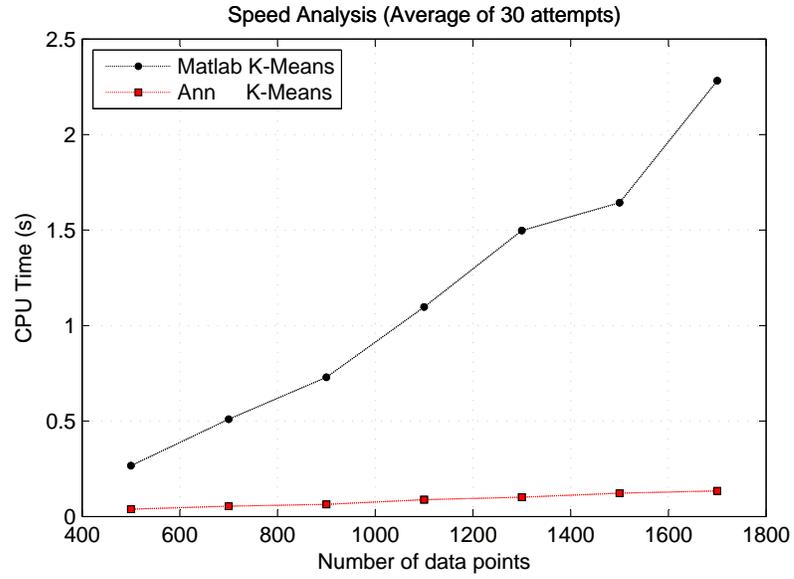
Figure A.3: Speed analysis of CMDS and *SEQ*-MDS with respect to the number of sample points. The dimensionality is reduced from 81 to 12. (a) the semi-log behavior, (b) the log-log behavior, indicating the time complexity of both methods. The mapping is performed using a PC with CPU 2.66GHz and 2GB of memory.

A.2 H-Ann k -means

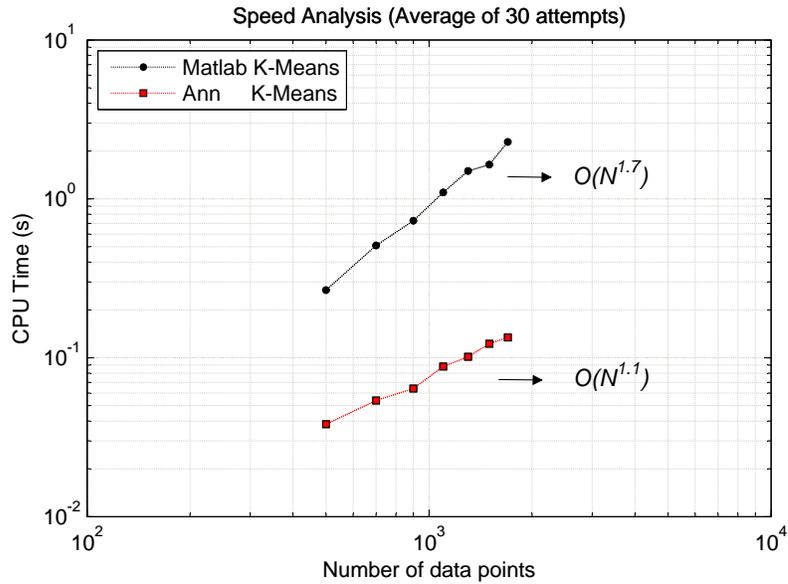
The huge amount of patterns stored in the database increases the need for an effective analysis method. The k -means method has been shown to be effective in producing good clustering results for many practical applications. However, k -means algorithm requires CPU time proportional to the product of the number of patterns and number of clusters per iteration. This will be computationally expensive, especially, for large databases, i.e., large 3D training images. An efficient implementation method for k -means clustering is proposed here.

The k -means algorithm for clustering finds a local optimum of the squared Euclidean distances between points in any subset and their center of mass. It accomplishes this by keeping track of centroids of the subsets, and issuing a large number of nearest-neighbor queries. First improvement over the classical k -means algorithm is to reduce the time spent on the nearest-neighbor queries. Kd-trees can be used to reduce the number of nearest-neighbor queries in k -means. A kd-tree is a data structure for storing a finite set of points from a finite-dimensional space (Bentley, 1980). The nodes in kd-tree can represent a large number of points; hence, they can be used to accelerate nearest-neighbor queries. We could use them transparently in the k -means inner loop (a method called Ann K-means). For this method to work, the centers are stored in the kd-tree and used as the starting point for nearest-neighbor searches. The use of kd-trees will bring huge improvements over the standard k -mean algorithm. There has been many attempts on using kd-trees in k -means clustering (Alsabti, 1998; Kanungo et al., 2000). In order to understand the advantages of using kd-trees in real situation, a speed comparison is made with Matlab k -means algorithm. The results are shown in Figure A.4. The computational improvement is evident in the plot. The computational complexity of Matlab k -means algorithm is approximately $O(N^{1.7})$ and for Ann k -means is $O(N^{1.1})$.

The speed comparison shows tremendous improvements. The use of kd-trees is favorable in the context of multiple-point geostatistics since we have mapped the high-dimensional patterns into a low-dimensional MDS space. This provides us with the ability to take advantage of kd-trees since they guarantee improvements only in the case of low-dimensional data. Therefore, Ann k -means algorithm would only work when we have our patterns mapped into MDS space. If we had applied it on the original high-dimensional patterns, not only we would not get the desired computational advantage, but we would also get less



(a) Speed Comparison



(b) Log-Log scale

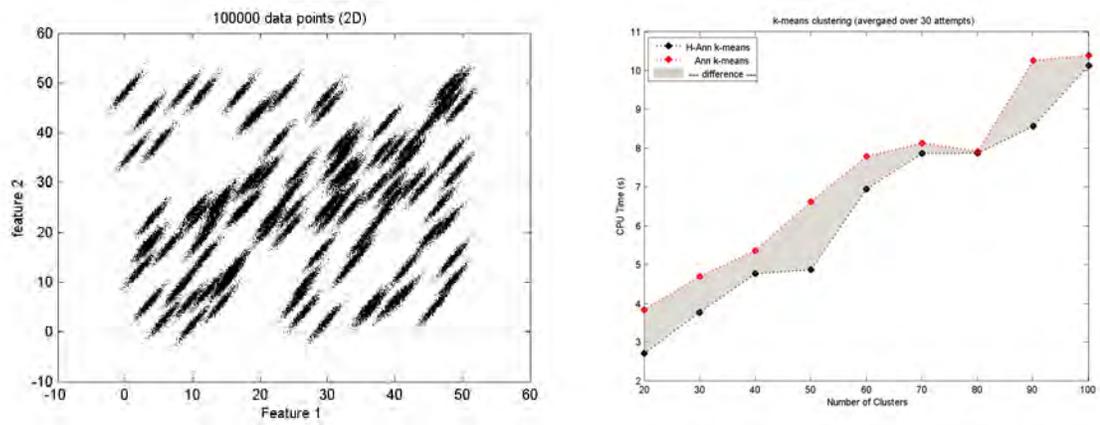
Figure A.4: (a) Speed comparison of Matlab k -means algorithm and Ann k -means algorithm, showing clear improvement of Ann k -means method, (b) Log-Log scale of the speed improvement showing the computational complexity of both methods being $O(N^{1.7})$ and $O(N^{1.1})$ for Matlab and Ann k -means respectively.

accurate nearest-neighbor queries. With the aid of kd-trees, one tries to find approximate nearest neighbors in each iteration (Ann k -means). For pattern classification required for the proposed methodology, we will adapt an even faster algorithm by proposing a hierarchical Ann k -means (H-Ann k -means) algorithm.

Hierarchical Ann k -means algorithm has a very simple concept. Since the computational complexity of the k -means algorithm depends on good initial centroids and the number of data points, the hierarchical k -means finds approximately-correct initial centroids for k -means algorithm by applying Ann k -means on a subset of the sample points. It then uses the obtained centroids as an initial guess for another Ann k -means algorithm which performs over the whole dataset. The improvement becomes more evident on larger datasets. This hierarchical application of Ann k -means over the dataset will ensure a good initial centroid for the next hierarchy. Good initial centroids not only help with the accuracy of the final clustering results, but also, ensure minimum number of iterations required for convergence to a local minimum.

In order to effectively test the performance of H-Ann k -means algorithm, one needs to find an optimal subset of data points for the initial hierarchical k -means application. Since a smart initialization can also take some CPU time, a random sampling of points is assumed to work reasonably well. However, one needs to find the proportion of points from the entire dataset that reduces the computational efficiency the most. In our analysis, a reduction factor of 10 over the pattern database size is chosen for the first Ann k -means hierarchy. The final centroids, obtained from the reduced dataset, are given as the initial centroids for the entire dataset. In order to compare two methods, a dataset of 100000 points are generated in \mathbb{R}^2 with different discernible cluster distributions as shown in Figure A.5(a). The comparison of Ann k -means with H-Ann k -means is shown in Figure A.5(b). It is noticeable that H-Ann k -means is more efficient with same number of clusters. However, this strongly depends on the configuration of the points and also on the true number of clusters.

In general, H-Ann k -means algorithm has two advantages: (1) providing a more robust initialization for k -means, and (2) reducing the computational complexity of clustering.



(a) Data points

(b) Speed comparison

Figure A.5: (a) 100000 data points used for clustering comparison, (b) Speed results for the two different methods of Ann k -means and H-Ann k -means, showing the improvement with the hierarchical approach.

Bibliography

- Alabert, F., Massonat, G., 1990. Heterogeneity in a complex turbiditic reservoir: Stochastic modeling of facies and petrophysical variability. In: 65th SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers, pp. 775–790.
- Alsabti, K., 1998. An efficient k-means clustering algorithm. In: Proceedings of IPPS/SPDP Workshop on High Performance Data Mining.
- Anderson, J. A., 1974. Diagnosis by logistic discriminant function: Further practical problems and results. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 23 (3), pp. 397–404.
- Arpat, G., Caers, Jef, Feb. 2007. Conditional simulation with patterns. *Mathematical Geology* 39 (2), 177–203.
- Arpat, G. B., 2005. Sequential simulation with patterns. Ph.D. thesis, Stanford University, Stanford, CA, USA.
- Ayinde, O., Yang, Y.-H., 2002. Face recognition approach based on rank correlation of gabor-filtered images. *Pattern Recognition* 35 (6), 1275 – 1289.
- Bentley, J. L., April 1980. Multidimensional divide-and-conquer. *Commun. ACM* 23, 214–229.
- Besag, J., 1974. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)* 36 (2), 192–236.
- Bookstein, F. L., 1989. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (6), 567–585.

- Borg, I., Groenen, P., 2005. Modern multidimensional scaling: theory and applications. Springer series in statistics. Springer.
- Borgefores, G., 1984. An improved version of the chamfer matching algorithm. In: 7th International Conference on Pattern Recognition. Montreal, P.Q., Canada, pp. 1175–1177.
- Borgefores, G., June 1986. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing* 34 (3), 344–371.
- Bosch, A., Zisserman, A., Munoz, X., 2007. Representing shape with a spatial pyramid kernel. In: Proceedings of the 6th ACM international conference on Image and video retrieval. CIVR '07. ACM, New York, NY, USA, pp. 401–408.
- Boucher, A., 2007. Downscaling of satellite remote sensing data - application to land cover mapping. Ph.D. thesis, Stanford University.
- Boucher, A., 2009. Considering complex training images with search tree partitioning. *Computers and Geosciences* 35 (6), 1151 – 1158.
- Bratvold, R., Maksimov, M., Galushko, V., Kutcherov, A., Surguchev, L., Feb. 28-Mar. 3 1993. Parallel nested factorization algorithms. In: 12th SPE Symposium on Reservoir Simulation. New Orleans, LA., paper SPE 25242.
- Buja, A., Swayne, D., Littman, M., Dean, N., Hofmann, H., 2001. Xgvis: Interactive data visualization with multidimensional scaling. Tech. rep., ATT research.
- Buja, A., Swayne, D. F., Littman, M. L., Dean, N., Hofmann, H., Chen, L., Jun. 2008. Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics* 17 (2), 444–472.
- Burges, C. J. C., Schölkopf, B., 1997. Improving the accuracy and speed of support vector learning machines. In: Mozer, M., Jordan, M., Petsche, T. (Eds.), *Advances in Neural Information Processing Systems 9*. MIT Press, Cambridge, MA, pp. 375–381.
- Caers, J., 2008a. Distance-based random field models and their applications. In: Proceedings of 8th International Geostatistical Congress, J.M. Ortiz and X. Emery (eds). Gecamin, Santiago, Chile, pp. 109–118.

- Caers, J., 2008b. Distance-based stochastic modeling: theory and applications. In: 21st SCRF Annual Meeting Report.
- Caers, J., 2010. Modeling Uncertainty in the Earth Sciences. Wiley-Blackwell.
- Caers, J., 2011. On internal consistency, conditioning and models of uncertainty. In: 24th SCRF affiliate meeting. Stanford University.
- Caers, J., Hoffman, T., Strebelle, S., Wen, X.-H., 2006. Probabilistic integration of geologic scenarios, seismic, and production data—a west africa turbidite reservoir case study. *The Leading Edge* 25 (3), 240–244.
- Caers, J., Journel, A. G., September 1998. Stochastic reservoir simulation using neural networks trained on outcrop data. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers, New Orleans, LA, pp. 321–336.
- Caers, J., Ma, X., 2002. Modeling conditional distributions of facies from seismic using neural nets. *Mathematical Geology* 34 (2), 143–167.
- Caers, J., Ma, X., 2006. Modeling conditional distributions of facies from seismic using neural nets. In: 19th SCRF affiliate meeting. Stanford University.
- Caers, J., Park, K., Sep. 2008. A distance-based representation of reservoir uncertainty: the metric enkf. In: Proceedings of 11th European Conference on the Mathematics of Oil Recovery. Bergen, Norway.
- Caers, J., Scheidt, C., Park, K., 2010. Modeling uncertainty of complex Earth systems in metric space. In *Handbook of Geomathematics*, W. Freeden et al. (eds). Springer, pp. 877–901.
- Caers, J., Z. T., 2004. Multiple-point geostatistics: A quantitative vehicle for integrating geologic analogs into multiple reservoir models. In: *Integration of outcrop and modern analogs in reservoir modeling*, grammer m, harris pm, eberli gp (eds) Edition. Vol. 80 of American Association of Petroleum Geology Memoir.
- Caine, G., Caine, R. N., 1994. Making Connections: Teaching and the Human Brain. Dale Seymour Publications.

- Calinski, R. B., Harabasz, J., 1974. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods* 3 (1), 1–27.
- Castro, S., Caers, J., Otterlei, C., Gomel, P., 24-27 September 2006. A probabilistic integration of well log, geological information, 3d/4d seismic, and production data: Application to the oseberg field. In: *SPE Annual Technical Conference and Exhibition*. San Antonio, Texas, USA.
- Chater, N., Vitányi, P. M., 2001. The generalized universal law of generalization. *Journal of Mathematical Psychology* 47, 346–369.
- Chatterjee, S., Dimitrakopoulos, R., Mustapha, H., 2011. Dimensional reduction of pattern-based simulation using wavelet analysis. *Mathematical Geosciences* Work to be submitted.
- Chiles, J. P., Delfiner, P., 1999. *Geostatistics Modeling Spatial Uncertainty*. Wiley, New York.
- Chugunova, T., Hu, L., 2008. Multiple-point simulations constrained by continuous auxiliary data. *Mathematical Geosciences* 40, 133–146.
- Clemensten, R., Hurst, A., Knarud, R., Omre, H., 1990. A computer program for evaluation of fluvial reservoirs. In: *North Sea Oil and Gas Reservoirs II*. Buller et al. (Eds.). Graham and Trotman, London, UK, pp. 372–385.
- Connors, R. W., Trivedi, M. M., Harlow, C. A., 1984. Segmentation of a high-resolution urban scene using texture operators. *Computer Vision, Graphics, and Image Processing* 25 (3), 273 – 310.
- Cover, T. M., 1965. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *Electronic Computers, IEEE Transactions on EC-14* (3), 326–334.
- Cover, T. M., Thomas, J. A., 1991. *Elements of information theory*. Wiley-Interscience, New York, NY, USA.
- Cox, T. F., Cox, M. A. A., 2001. *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC, London.

- Daly, C., 2005. Higher order models using entropy, markov random fields and sequential simulation. In: Leuangthong, O., Deutsch, C. V. (Eds.), *Geostatistics Banff 2004*. Vol. 14 of *Quantitative Geology and Geostatistics*. Springer Netherlands, pp. 215–224.
- Daly, C., Knudby, C., 2007. Multipoint statistics in reservoir modelling and in computer vision. In: *EAGE Petroleum Geostatistics*. Cascais, Portugal.
- Damsleth, E., Tjolsen, C., Omre, K., Haldorsen, H., Apr. 1992. A two-stage stochastic model applied to a north sea reservoir. in , pp. 791802. *Journal of Petroleum Technology* 44 (4), 402–408.
- Daugman, J. G., 1980. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision Research* 20 (10), 847 – 856.
- Daugman, J. G., 1984. Spatial visual channels in the fourier plane. *Vision Research* 24 (9), 891–910.
- Daugman, J. G., 1993. An information-theoretic view of analog representation in striate cortex. MIT Press, Cambridge, MA, USA, pp. 403–423.
- de Vries, L., Carrera, J., Falivene, O., Gratacós, O., Slooten, L., 2009. Application of multiple point geostatistics tonon-stationary images. *Mathematical Geosciences* 41, 29–42.
- Deutsch, C., 1992. Annealing techniques applied to reservoir modeling and the integration of geological and engineering (well test) data. Ph.D. thesis, Stanford University, 306pp.
- Deutsch, C., 2002. *Geostatistical Reservoir Modeling*, 1st Edition. Oxford University Press, New York.
- Deutsch, C., Gringarten, E., 2000. Accounting for multiple-point continuity in geostatistical modeling. In: of Southern Africa, G. A. (Ed.), *6th International Geostatistics Congress*. Vol. 1. Cape Town, South Africa, pp. 156–165.
- Deutsch, C., Wang, L., 1996. Hierarchical object-based stochastic modeling of fluvial reservoirs. *Mathematical Geology* 28, 857–880.
- Deutsch, C., Wen, X., 2001. Integrating large-scale soft data by simulated annealing and probability constraints. *Mathematical Geology* 32 (1), 49–68.

- Deutsch, C. V., Journel, A. G., 1997. *GSLIB: Geostatistical Software Library and User's Guide*, 2nd Edition. Oxford University press, New York.
- Deutsch, C. V., Journel, A. G., 1998. *GSLIB: Geostatistical Software Library and Users Guide*, second edition Edition. Oxford, New York.
- Deutsch, C. V., Lewis, R. W., 1992. Advances in the practical implementation of indicator geostatistics. In: *Proceedings of the 23rd International APCOM Symposium*. in Kim, Y. C., edition. Society of Mining Engineers, Littleton, Tuscon, AZ, pp. 133–148.
- Dimitrakopoulos, R., Mustapha, H., Gloaguen, E., 2009. High-order statistics of spatial random fields: Exploring spatial cumulants for modeling complex non-gaussian and non-linear phenomena. *Mathematical Geosciences* 42 (1), 65–99.
- Domingos, P., Pazzani, M., 1996. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In: *Machine Learning*. Morgan Kaufmann, pp. 105–112.
- Doyen, P. M., 1988. Porosity from seismic data: A geostatistical approach. *Geophysics* 53 (10), 1263–1275.
- Dubuisson, M. P., Jain, A. K., 1994. A modified hausdorff distance for object matching. In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. IEEE Comput. Soc. Press, pp. 566–568.
- Duda, R. O., Hart, P. E., 1973. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc.
- Dujardin, B., Wu, J., Journel, A., 2006. Sensitivity analysis on filtersim. In: *19th SCRF affiliate meeting*. Stanford Center For Reservoir Forecasting, Stanford University.
- Dunn, D., Higgins, W., Jul. 1995. Optimal gabor filters for texture segmentation. *IEEE Transactions on Image Processing* 4 (7), 947–964.
- Endres, D. M., Schindelin, J. E., Jul. 2003. A new metric for probability distributions. *IEEE Transactions on Information Theory* 49 (7), 1858–1860.
- Evensen, G., 1994. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research* 99, 10143–10162.

- Fält, L., Henriquez, A., Holden, L., Tjelmeland, H., May 1991. Moheres, a program system for simulation of reservoir architecture and properties. In: 6th European Symposium on Improved Oil Recovery. pp. 27–39.
- Fang, J., Wang, P., 1997. Random field generation using simulated annealing vs. fractal-based stochastic interpolation. *Mathematical Geology* 29, 849–858.
- Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., 1996. *Advances in knowledge discovery and data mining*. American Association for Artificial Intelligence, Menlo Park, CA, USA.
- Forgy, E., 1965. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics* 21, 768–780.
- Fowlkes, C., Belongie, S., Chung, F., Malik, J., feb. 2004. Spectral grouping using the nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (2), 214 –225.
- Friedman, J., Tukey, J., 1974. A projection pursuit algorithm for exploratory data analysis. *Computers, IEEE Transactions on C-23* (9), 881 – 890.
- Friedman, J. H., Fayyad, U., 1997. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery* 1, 55–77.
- Gabor, D., 1946. Theory of communication. *J. Inst. Electr. Engineering* 93, 429–457.
- Gardner, H. E., Apr. 1993. *Frames Of Mind: The Theory Of Multiple Intelligences*, 10th Edition. Basic Books.
- Geman, S., Bienenstock, E., Doursat, R., January 1992. Neural networks and the bias/variance dilemma. *Neural Comput.* 4, 1–58.
- Geman, S., Geman, D., Nov. 1984. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (6), 721–741.
- Georgsen, F., Omre, H., 1993. Combining fibre processes and gaussian random functions for modeling fluvial reservoirs. In: *Geostatistics Troia 1992*. Vol. 2 of Soares, A., Editor. Kluwer, Dordrecht, pp. 425–440.

- Gershenfeld, N., 1999. *The nature of mathematical modeling*. Cambridge University Press, New York, NY, USA.
- Gersho, A., Gray, R. M., 1991. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA.
- Girolami, M., 2002. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks* 13, 780–784.
- Goovaerts, P., 1996. Stochastic simulation of categorical variables using a classification algorithm and simulated annealing. *Mathematical Geology* 28 (7), 909–921.
- Goovaerts, P., 1997. *Geostatistics for Natural Resources Evaluation*. Oxford Press, New York.
- Grauman, K., Darrell, T., May 2007. The pyramid match kernel: Efficient learning with sets of features. *J. Mach. Learn. Res.* 8, 725–760.
- Gregory, R., September 1974. *Concepts and Mechanisms of Perception*, 1st Edition. Gerald Duckworth & Co., London.
- Guardiano, F., Srivastava, R., 1993. Multivariate geostatistics: beyond bivariate moments. In: Soares, A. (Ed.), *Geostatistics Troia*. Vol. 1. Kluwer Academic Publications, pp. 133–144.
- Haldorsen, H. H., Damsleth, E., 1990. Stochastic modelling. *Journal of Petroleum Technology* 42 (4), 404–412.
- Haldorsen, H. H., Lake, L. W., August 1984. A new approach to shale management in field-scale models. *SPE Journal* 24 (4), 447–457.
- Hamamoto, Y., Uchimura, S., Watanabe, M., Yasuda, T., Mitani, Y., Tomita, S., 1998. A gabor filter-based method for recognizing handwritten numerals. *Pattern Recognition* 31 (4), 395 – 400.
- Haralick, R. M., Dinstein, Shanmugam, K., Nov. 1973. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics SMC-3*, 610–621.

- Harbaugh, J. W., Bonham-Carter, G., 1970. Computer simulation in geology. John Wiley & Sons, New York.
- Harding, A., Strebelle, S., Levy, M., Thorne, J., Xie, D., Leigh, S., Preece, R., Scamman, R., 2005. Reservoir facies modelling: New advances in mps. In: Leuangthong, O., Deutsch, C. V. (Eds.), Geostatistics Banff 2004. Vol. 14 of Quantitative Geology and Geostatistics. Springer Netherlands, pp. 559–568.
- Hatløy, A., 1995. Numerical facies modeling combining deterministic and stochastic method. In: Stochastic Modeling and Geostatistics: Principles, Methods, and Case Studies, yarus, j.m., chambers, r.l. (eds.) Edition. Vol. 3 of AAPG Computer Applications in Geology. Ch. 10, p. 109120.
- Heisenberg, W., Mar. 1927. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. Zeitschrift für Physik 43 (3-4), 172–198.
- Henriquez, A., Tyler, K., Hurst, A., September 1990. Characterization of fluvial sedimentology for reservoir simulation modeling. SPE Formation Evaluation, 211–216.
- Holden, L., Hauge, R., Skare, A., Skorstad, A., 1998. Modeling of fluvial reservoirs with object models. Mathematical Geology 30 (5), 473–496.
- Honarkhah, M., Caers, J., 2008. Classifying existing and generating new training image patterns in kernel space. In: 21st SCRF affiliate meeting. Stanford University.
- Hong, S., Ortiz, J., Deutsch, C., 2008. Multivariate density estimation as an alternative to probabilistic combination schemes for data integration. In: Geostats 2008 Proceedings of the Eighth International Geostatistics Congress. Vol. 1 of Ortiz, J.M., Emery, X. (Eds.). Gecamin Ltda., Santiago, Chile, pp. 197–206.
- Hove, K., Olsen, G., Nilsson, S., Tonnesen, M., Hatløy, A., October 1992. From stochastic geological description to production forecasting in heterogeneous layered reservoirs. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers, Washington, D.C.
- Indow, T., 1994. Metrics in color spaces: Im kleinen und im groen. In: Contributions to mathematical psychology, psychometrics, and methodology. In G. H. Fischer & D. Laming (Eds.). Springer, New York.

- Isaaks, E., 1990. The application of monte carlo methods to the analysis of spatially correlated data. Phd. thesis, Stanford University, USA.
- Jain, A. K., Farrokhnia, F., 1991. Unsupervised texture segmentation using gabor filters. *Pattern Recognition* 24 (12), 1167 – 1186.
- Jones, J. P., Palmer, L. A., 1987. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology* 58 (6), 1233–1258.
- Journel, A., 1983. Nonparametric estimation of spatial distributions. *Mathematical Geology* 15 (3), 445–468.
- Journel, A., 1985. The deterministic side of geostatistics. *Mathematical Geology* 17, 1–15.
- Journel, A., 2002. Combining knowledge from diverse sources: An alternative to traditional data independence hypotheses. *Mathematical Geology* 34, 573–596.
- Journel, A., 2005. Multiple point statistics. In: *Proceedings of IAMG05, The Annual Conference of the International Association for Mathematical Geology*. in Cheng, Qiuming, Graeme Bonham-Carter (Eds). Toronto.
- Journel, A., Alabert, F., 1988. Focusing on spatial connectivity of extreme-valued attributes: stochastic indicator models of reservoir heterogeneities. In: *63rd Annual Technical Conference and Exhibition of the Society of Petroleum Engineers*. Society of Petroleum Engineers, Richardson, Texas, p. 12.
- Journel, A., Deutsch, C., 1993. Entropy and spatial disorder. *Mathematical Geology* 25, 329–355.
- Journel, A., Zhang, T., 2006. The necessity of a multiple-point prior model. *Mathematical Geology* 38 (5), 591–610.
- Journel, A. G., 1974. Geostatistics for conditional simulation of ore bodies. *Economic Geology* 69 (5), 673–687.
- Journel, A. G., Huijbregts, C. J., 1978. *Mining Geostatistics*. Academic Press, London.
- Kanungo, T., c, M., Mount, D. M., Piatko, C., Netanyahu, N. S., Wu, A. Y., Silverman, R., 2000. The analysis of a simple k-means clustering algorithm.

- Kjøsberg, H., Kolbjørnsen, O., 2008. Markov mesh simulations with data conditioning through indicator kriging. In: Proceedings of the Eighth International Geostatistics Congress. Santiago, Chile.
- Koenderink, J., 1984. The structure of images. *Biological Cybernetics* 50, 363–370.
- Krige, D. G., 1951. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of Chemical Metall and Mineralogy Society South Africa* 52 (6), 119–139.
- Krishnan, S., 2008. The tau model for data redundancy and information combination in earth sciences: Theory and application. *Mathematical Geosciences* 40, 705–727.
- Krishnan, S., Boucher, A., Journel, A., 2005. Evaluating information redundancy through the tau model. In: Leuangthong, O., Deutsch, C. V. (Eds.), *Geostatistics Banff 2004*. Vol. 14 of *Quantitative Geology and Geostatistics*. Springer Netherlands, pp. 1037–1046.
- Kwok, J. T. Y., Tsang, I. W. H., Nov. 2004. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks* 15 (6), 1517–1525.
- Lampert, C. H., March 2009. Kernel methods in computer vision. *Found. Trends. Comput. Graph. Vis.* 4, 193–285.
- Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., Jordan, M. I., December 2004. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.* 5, 27–72.
- Lantuejoul, C., 2002. *Geostatistical Simulation: Models and Algorithms*, 1st Edition. Springer, Berlin.
- Lehoucq, R., Sorensen, D. C., 1996. Deflation techniques for an implicitly re-started Arnoldi iteration. *SIAM J. Matrix Anal. Appl* 17, 789–821.
- Lindeberg, T., 1994. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA.
- Lopez, J. L., Rappold, P. M., Ugueto, G. A., Wieseneck, J. B., Vu, C. K., 2004. Integrated shared earth model: 3d pore-pressure prediction and uncertainty analysis. *The Leading Edge* 23 (1), 52–59.

- Lyster, S., Deutsch, C., 2007. Artifacts near conditioning data in mps gibbs sampling. In: Centre for Computational Geostatistics. No. 9.
- Lyster, S., Deutsch, C., 2008. Mps simulation in a gibbs sampler algorithm. In: Proceedings of the Eighth International Geostatistics Congress. Santiago, Chile.
- Mackay, D. J. C., Jun. 2003. Information Theory, Inference & Learning Algorithms, 1st Edition. Cambridge University Press.
- MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In: Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability. Vol. 1. University of California Press, Berkeley, pp. 281–297.
- Maitre, H., Campedel, M., Moulines, E., Datcu, M., 2008. Feature selection for satellite image indexing. In: ESA-EUSC: Image Information Mining. Frascati, Italy.
- Malik, J., Perona, P., 1990. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America* 7, 923–932.
- Mallat, S., Sep. 1999. *A Wavelet Tour of Signal Processing, (Wavelet Analysis & Its Applications)*, 2nd Edition. Academic Press.
- Manjunath, B. S., Shekhar, C., Chellappa, R., 1996. A new approach to image feature detection with applications. *Pattern Recognition* 29 (4), 627 – 640.
- Marcelja, S., Nov. 1980. Mathematical description of the responses of simple cortical cells. *Journal of the Optical Society of America* 70 (11), 1297–1300.
- Mariethoz, G., Renard, P., Apr. 2010. Reconstruction of incomplete data sets or images using direct sampling. *Mathematical Geosciences* 42 (3), 245–268.
- Mariethoz, G., Renard, P., Straubhaar, J., 2010. The direct sampling method to perform multiple-point geostatistical simulations. *Water Resources Research* 46 (11), 12.
- Matheron, G., 1971. The theory of regionalized variables and its applications. No. 5 in *Les cahiers du Centre de morphologie mathématique de Fontainebleau*. Ecole nationale supérieure des mines, Paris.

- Matheron, G., 1973. The intrinsic random functions and their applications. *Advances in Applied Probability* 5 (3), 439–468.
- Mehrotra, R., Namuduri, K., Ranganathan, N., 1992. Gabor filter-based edge detection. *Pattern Recognition* 25 (12), 1479 – 1494.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., 1953. Equations of state calculations by fast computing machines. *J. Chem. Phys.* 21, 1087–1091.
- Milligan, G. W., Cooper, M. C., 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika* 50, 159–179.
- Mondt, J. C., 1993. Use of dip and azimuth horizon attributes in 3d seismic interpretation. *SPE Formation Evaluation* 8 (4), 253–257.
- Mustapha, H., Dimitrakopoulos, R., 2010a. High-order stochastic simulation of complex spatially distributed natural phenomena. *Mathematical Geosciences* 42 (5), 457–485.
- Mustapha, H., Dimitrakopoulos, R., 2010b. A new approach for geological pattern recognition using high-order spatial cumulants. *Computers & Geosciences* 36 (3), 313 – 334.
- Norrena, K., Deutsch, C., April 2000. Using the critical temperature to improve the speed of geostatistical applications of simulated annealing. In: *GEOSTATS 2000, Geostatistical Congress*. Cape Town, South Africa.
- Omre, H., 1992. Heterogeneity models. In: *SPOR Monograph: Recent Advances in Improved Oil Recovery Methods for North Sea Sandstone Reservoirs*. Norwegian Petroleum Directorate, Norway, pp. 141–153.
- Ortiz, J., Deutsch, C., 2004. Indicator simulation accounting for multiple-point statistics. *Mathematical Geology* 36 (5), 545–565.
- Ortiz, J. M., Emery, X., 2005. Integrating multiple-point statistics into sequential simulation algorithms. In: Leuangthong, O., Deutsch, C. V. (Eds.), *Geostatistics Banff 2004*. Vol. 14 of *Quantitative Geology and Geostatistics*. Springer Netherlands, pp. 969–978.
- Osuna, E. E., Girosi, F., 1999. Reducing the run-time complexity in support vector machines. MIT Press, Cambridge, MA, USA, pp. 271–283.

- Otsu, N., Mar. 1979. A threshold selection method from gray level histograms. *IEEE Trans. Systems, Man and Cybernetics* 9, 62–66.
- Park, K., Caers, J., September 10-14. 2007. History matching in low-dimensional connectivity vector space. In: EAGE Petroleum Geostatistics conference. Cascais, Portugal.
- Park, K., Scheidt, C., Caers, J., Jun. 2008a. Ensemble Kalman filtering in distance-based kernel Space. In: *Proceedings of EnKF Workshop 2008*. Voss, Norway.
- Park, K., Scheidt, C., Caers, J., 2008b. Simultaneous conditioning of multiple non-Gaussian geostatistical Models to highly nonlinear data using distances in kernel Space. In: *Proceedings of 8th International Geostatistical Congress*, J.M. Ortiz and X. Emery (eds). Gecamin, Santiago, Chile, pp. 247–256.
- Parra, A., Ortiz, J., 2009. Conditional multiple-point simulation with a texture synthesis algorithm. In: *International Association of Mathematical Geosciences Conference*. Stanford University.
- Peredo, O., Ortiz, J. M., 2010. Parallel implementation of simulated annealing to reproduce multiple-point statistics. *Computers & Geosciences In Press*, Corrected Proof.
- Pollen, D., Ronner, S., 1983. Visual cortical neurons as localized spatial frequency filters. *T-SMC* 13, 907–916.
- Polyakova, E., Journel, A., 2007. The nu expression for probabilistic data integration. *Mathematical Geology* 39, 715–733.
- Ravenne, C., Beucher, H., 2-5 Oct. 1988. Recent development in description of sedimentary bodies in a fluvio deltaic reservoir and their 3d conditional simulations. In: *63rd Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, Houston, pp. 463–476.
- Remy, N., Boucher, A., Wu, J., 2009. *Applied Geostatistics with SGeMS: A User's Guide*. Cambridge University Press, Cambridge.
- Rosenblatt, M., 1985. *Stationary sequences and random fields*. Birkhuser, Boston.

- Rudkiewicz, J. L., Guerillot, D., Galli, A., 1990. An integrated software for stochastic modelling of reservoir lithology and property prediction with an example from the yorkshire middle jurassic formation. In: North Sea oil and gas reservoirs II. A. T. Buller, E. Berg, O. Hjelmeland, J. Kleppe, O. Torster, and J. O. Aasen, eds. Graham and Trotman, London.
- Russ, J. C., 2002. Image Processing Handbook, Fourth Edition, 4th Edition. CRC Press, Inc., Boca Raton, FL, USA.
- Scheidegger, A. E., 2004. Morphotectonics, 1st Edition. Springer, Berlin, New York.
- Scheidt, C., Caers, J., 2009a. A new method for uncertainty quantification using distances and kernel methods. Application to a deepwater turbidite reservoir. SPE Journal 14 (4), 680–692.
- Scheidt, C., Caers, J., 2009b. Representing spatial uncertainty using distances and kernels. Mathematical Geosciences 41 (4), 397–419.
- Scheidt, C., Park, K., Caers, J., 2008. Defining a random function from a given set of model realizations. In: Proceedings of 8th International Geostatistical Congress, J.M. Ortiz and X. Emery (eds). Gecamin, Santiago, Chile, pp. 469–478.
- Schölkopf, B., Smola, A. J., Dec. 2001. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning), 1st Edition. The MIT Press, Cambridge, MA, USA.
- Shannon, C. E., 1948. A mathematical theory of communication. Bell system technical journal 27, 379–423.
- Shawe-Taylor, J., Cristianini, N., 2004. Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA.
- Shepard, R. N., 1987. Toward a universal law of generalization for psychological science. Science 237, 1317–1323.
- Srivastava, R. M., 1992. Reservoir characterization with probability field simulation. SPE Formation Evaluation 7 (4), 927–937.

- Stanley, K., Jorde, K., Raestad, N., Stockbridge, C., 1990. Stochastic modeling of reservoir sand bodies for input to reservoir simulation, snorre field, northern north sea. In: North Sea Oil and Gas Reservoirs II. Buller et al. (Eds.),. Graham and Trotman, London, UK, pp. 91–103.
- Stien, M., Kolbjørnsen, O., 2011. Markov mesh model specification for facies modeling. Tech. Rep. SAND/09/10, Norwegian Computing Center.
- Stoyan, D., Kendall, W., Mecke, J., 1987. Stochastic Geometry and Its Applications. John Wiley & Sons, New York.
- Straubhaar, J., Renard, P., Mariethoz, G., Froidevaux, R., Besson, O., 2011. An improved parallel multiple-point algorithm using a list approach. *Mathematical Geosciences* 43 (3), 305–328.
- Strebelle, S., 2000. Sequential simulation drawing structures from training images. Ph.D. thesis, Stanford University.
- Strebelle, S., 2002. Conditional simulation of complex geological structures using multiple-point statistics. *Mathematical Geology* 34 (1), 1–22.
- Strebelle, S., Zhang, T., 2005. Non-stationary multiple-point geostatistical models. In: Leuangthong, O., Deutsch, C. V. (Eds.), *Geostatistics Banff 2004*. Vol. 14 of *Quantitative Geology and Geostatistics*. Springer Netherlands, pp. 235–244.
- Suzuki, S., Caers, J., 2006. History matching with an uncertain geological scenario. In: *SPE Annual Technical Conference and Exhibition*. San Antonio, Texas, USA, SPE 102154-MS.
- Suzuki, S., Caers, J., 2008. A Distance-based prior model parameterization for constraining solutions of spatial inverse problems. *Mathematical Geosciences* 40 (4), 445–469.
- Suzuki, S., Caumon, G., Caers, J., 2008. Dynamic data integration into structural modeling: model screening approach using a distance-based model parameterization. *Computational Geosciences* 12 (1), 105–119.
- Tang, H., Wang, F., Kurnianwan, B., June 2007. Geostatistic modelling and flow simulation of a mixed-influenced deltaic reservoir. In: *Canadian International Petroleum Conference*. Calgary, Alberta.

- Tenenbaum, J. B., Griffiths, T. L., 2001. Generalization, similarity, and bayesian inference. *Behavioral and Brain Sciences* 24, 629–640.
- Tjelmeland, H., 1996. Stochastic models in reservoir characterization and markov random fields for compact objects. Ph.D. thesis, Norwegian University of Science and Technology, Trondheim, Norway.
- Tjelmeland, H., Eidsvik, J., 2004. Directional metropolis-hastings updates for posteriors with non linear likelihood. In: Leuangthong O, D. C. (Ed.), *Geostatistics, Banff 2004*. Springer, Dordrecht, p. 95104.
- Tran, T. T., 1994. Improving variogram reproduction on dense simulation grids. *Computers & Geosciences* 20 (7-8), 1161 – 1168.
- Tyler, K., Henriquez, A., Georgsen, F., Holden, L., Tjelmeland, H., June 1992. A program for 3d modeling of heterogeneities in a fluvial reservoir. In: *3rd European Conference on the Mathematics of Oil Recovery*. Delft, pp. 31–40.
- Valois, R. L. D., Albrecht, D. G., Thorell, L. G., 1982. Spatial frequency selectivity of cells in macaque visual cortex. *Vision Research* 22 (5), 545 – 559.
- Van Wagoner, J., Mitchum, R., Campion, K., , Rahmanian, V., 1990. *Siliciclastic Sequence Stratigraphy in Well Logs Cores, and Outcrops*. No. 7 in *Methods in Exploration*. American Association of Petroleum Geologists.
- Wahba, G., 1990. *Spline models for observational data*. Vol. 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.
- Wu, J., 2007. 4d seismic and multiple-point pattern data integration using geostatistics. Ph.D. thesis, Stanford University.
- Xu, W., 1996. Conditional curvilinear stochastic simulation using pixel-based algorithms. *Mathematical Geology* 28, 937–949.
- Xu, W., Tran, T., Srivastava, R., Journel, A., October 1992. Integrating seismic data in reservoir modeling: The collocated cokriging alternative. In: *67th SPE Annual Technical*

- Conference and Exhibition. Society of Petroleum Engineers, Washington, D.C., pp. 833–842.
- Zeng, Z.-Q., Gao, J., Guo, H., 2006. Simplified support vector machines via kernel-based clustering. In: Sattar, A., Kang, B.-h. (Eds.), *AI 2006: Advances in Artificial Intelligence*. Vol. 4304 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pp. 1189–1195.
- Zhang, H., 2004. The optimality of naive bayes. In: Barr, V., Markov, Z. (Eds.), *FLAIRS Conference*. AAAI Press.
- Zhang, J., Tan, T., Ma, L., 2002. Invariant texture segmentation via circular gabor filters. In: *16 th International Conference on Pattern Recognition*. Vol. 2. Quebec City, QC, Canada, pp. 901–904.
- Zhang, T., 2002. Multiple-point simulation of multiple reservoir facies. Master's thesis, Stanford University, Stanford, CA.
- Zhang, T., 2006. Filter-based training pattern classification for spatial pattern simulation. Ph.D. thesis, Stanford University.
- Zhao, L.-H., Zhang, X.-L., Xu, X.-H., nov. 2007. Face recognition base on kpca with polynomial kernels. In: *International Conference on Wavelet Analysis and Pattern Recognition, 2007. ICWAPR '07*. Vol. 3. pp. 1213 –1216.
- Zhu, H., Journel, A., 1993. Formatting and integrating soft data: Stochastic imaging via the markov-bayes algorithm. In: Soares, A. (Ed.), *Geostatistics Troia92*. Kluwer Academic Publishers, Dordrecht, Holland, pp. 1–12.
- Zhu, M., Ghodsi, A., 2006. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics and Data Analysis* 51, 918–930.