Classifying Existing and Generating New Training Image Patterns in Kernel Space

MEHRDAD HONARKHAH AND JEF CAERS

Stanford Center for Reservoir Forecasting
Stanford University

May 8, 2008

Abstract

Advent of Multiple-Point Geostatistics helped to integrate complex subsurface geological structures and features into the model by the concept of training images. Initial algorithms such as *snesim* generated geologically consistent realizations by using these training images to obtain the distributional properties and probabilities needed in a stochastic simulation framework. Later on, pattern-based geostatistical algorithms such as *simpat* and *filtersim* came along, in which patterns, instead of covariances or probabilities, were used in an image construction process in order to generate geologically realistic models. In these approaches, the training image which represents the spatial continuity of the subsurface structure is used to construct a pattern database, and consequently, the sequential simulation will be carried on by selecting a pattern from that database and pasting it onto the simulation grid. One of the shortcomings of the present algorithms is the lack of randomness, in other words, the strong similarity between the generated realizations and the input training image patterns. Hence, the main issue for both *simpat* and *filtersim* is caused by drawing from the same limited pattern database, no interpolation nor extrapolation is done. In this study a whole different approach will be taken towards pattern analysis and generation. The main purpose is to introduce a better pattern classification algorithm, by using distance methods. This method is more flexible than the previous methods and can be tailored to any application at hand. Additionally, we will also deal with the shortcomings of previous algorithms. That is, to generate new patterns from the same training image, using the existing pattern database. In other words, given a data event, the most similar pattern, which does not necessarily exist in the training image, will be obtained. This new and promising pattern generation methodology will bring more randomness into training images by spanning a more diverse and much bigger pattern space.

1 Introduction

Sequential simulation is one of the most popular methods for generating multiple reservoir models constrained to geologic, seismic and production data. Algorithms such as SGSIM and SISIM (Deutsch and Journel, 1998) have been developed and are widely used in practice. These variogram-based techniques are simple, fast and robust. However, outcrop analogs and geological structures such as channels, faults, and curvilinear shapes can not be reproduced by these methods. By describing merely the correlation between

only two spatial locations, a variogram can not capture the complexity of features. Moreover, it can not describe any strong connectivity within a reservoir. To overcome these drawbacks, multiple point geostatistics (MPS) was introduced (Guardiano and Srivastava, 1993). Training image replaces the variogram in multiple point geostatistics as a measure for geological heterogeneity; it contains multiple point information and is much more intuitive since one can observe, prior to any geostatistical estimation and simulation, the patterns that will be reproduced in a set of models.

SNESIM was the first algorithm proposed for the application of MPS (Strebelle, 2000). In this method, the conditional probability is sampled from the training image by looking for replicates of the data event. In contrast to classical algorithms based only on two-point variogram model, snesim allows reproduction of complex patterns. Such patterns include the traditional two-point statistics which are also reproduced even though they are not explicitly modeled. But one of the shortcomings of snesim is that the training image needs to be categorical. Also, the snesim algorithm can only capture the stationary features of the training image. However, most geological structures contain also non-stationary features. These non-stationary patterns are unique and non-repetitive. Additionally, any type of trend, such as vertical or areal proportion change needs to be explicitly enforced.

Therefore, SIMPAT as an alternative pattern-based algorithm was proposed (Arpat, 2004). The sequential simulation method of SIMPAT replaces the traditional probability framework of drawing from conditional probability distributions with the calculation of similarity between patterns. The strong assumption of stationarity in the previous methods is now somewhat reduced. This method was also improved by summarizing multiple-point spatial patterns with a few general linear filters (Zhang, 2004). In this method, called FilterSim, a low-dimensional representation of patterns was provided and classification was carried out according to their filter scores. Hence, in filtersim, one attempts to group local training image patterns into classes and identifies the most similar class to the available conditioning data event. Randomness/variability is introduced through randomly sampling a pattern from that class, and then, pasting it on the realization. However, it should be kept in mind that these patterns originate from the same training image which has a limited/finite pattern database. Hence, in both *simpat* and *filtersim*, patterns are drawn from a pattern database, and no interpolation or extrapolation is made. As a result, simulated realizations include only the patterns that existed in the training image, and also, a strong similarity between the generated realizations and the input training image patterns will be observed.

This paper consists of two parts. In the first part, a new methodology on pattern classification will be introduced. The patterns in a training image will be analyzed using the new concept of distance metrics. Distance methods are more general than the standard (fixed) filters used in *filtersim* as they can be tailored to the application at hand. The proposed methodology can be stated intuitively as follow: All the patterns in the training image will be represented by points in a Euclidean space. Classification phase will then be as straightforward as clustering these points into some groups. Undoubtedly, clustering these points results in classifying their corresponding patterns. However, the main advantage of our method lies in the way patterns are mapped into Euclidean space and represented by a point. This transformation is done by calculating the distances between available patterns, and then, using multi-dimensional scaling (MDS) to map

them as a point in Euclidean space. Hence, flexibility of the methodology comes into play by the ability of choosing different distance methods for different applications. Moreover, the better pattern classification is brought about by kernel k-means algorithm (relying on the same distances), in which data are mapped into a higher dimensional space that allows to get non-linear separation surfaces in the data.
The second part of the study focuses on generating a new pattern that does not exist in the pattern database of the training image. This methodology introduces randomness in the pattern database, and resolves the shortcomings inherent in the *filtersim* or *simpat* algorithms. It is simply done by using the previous concepts. In other words, as the patterns are now represented by a point in Euclidean space, this pattern generation methodology can be intuitively seen as a random sampling of a point in that space. Indeed, this point has to be mapped back into a pattern. In this study, three different methods will be investigated for the pattern generation part. The resulting pattern will automatically be constrained to have similar features as the patterns in the database.
The outline of this report goes as follow. Section 2 introduces the concept of distance methods in similarity measurement, and different distance functions will be introduced and analyzed. In section 3, multi-dimensional scaling method will be explained. In Section 4 clustering algorithms will be discussed and the kernel method will be introduced. In section 5 some pattern classification results using different distance methods will be illustrated, and also the effect of pattern skipping will be analyzed on a binary (sand/shale) case study. Finishing the classification part of the study, we will proceed with the pattern generation phase in section 6, where simple random sampling in kernel space will be introduced with some examples. Finally, the pattern generation methodology with a morphological concept will be introduced and some examples will also be provided to substantiate the merit of this paper.

2 Distance Methods

2.1 Notation

For clarity, this section introduces the required notation for explaining the proposed algorithm on a binary (sand/shale) case. Some of the notations and definitions are identical to the notations used by Arpat, 2004.

2.1.1 Training image

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

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

$\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ indicates 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:

$$\mathbf{ti}_{\mathbf{T}}(\mathbf{u}) = \left\{\mathbf{ti}(\mathbf{u}+\mathbf{h}_1), \mathbf{ti}(\mathbf{u}+\mathbf{h}_2), \ldots, \mathbf{ti}(\mathbf{u}+\mathbf{h}_\alpha), \ldots, \mathbf{ti}(\mathbf{u}+\mathbf{h}_{n_T})\right\}$$

Where $\mathbf{h}_\alpha$ vectors are vectors defining the geometry of the $n_{\mathbf{T}}$ nodes of template $\mathbf{T}$ and $\alpha=1,\ldots, n_{\mathbf{T}}$.

2.1.2 Pattern

The processing of the training image **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 $\mathbf{patdb_T}$. 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 $\mathbf{u} \in \mathbf{G}_{ti}$ of a pattern; only the content of the pattern is stored. Hence, k^{th} pattern can be denoted like this:

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

where $k = 1,\ldots, n_{Pat_T}$, and all n_{Pat_T} patterns are defined on the same template $\mathbf{T}$.

2.1.3 Dissimilarity distance Matrix

Having a pattern database $\mathbf{patdb_T}$ of a training image, the dissimilarity 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 Δ will be a $n_{Pat_T} \times n_{Pat_T}$ size matrix, where each element δ_{ij} of matrix Δ will represent the distance between i^{th} and j^{th} pattern. This distance is calculated using a dissimilarity distance function, which will be defined later. By this definition, the dissimilarity matrix should be symmetric with zero-diagonal elements.

2.2 Dissimilarity distance function

In the first part of the study, we want to classify the patterns in $\mathbf{patdb_T}$ in similar groups, called clusters. In other words, we want to match the patterns that are similar to each

other, in one cluster. To this end, we need a measure of similarity between the patterns. This measure, which is a function that calculates the dissimilarity between the patterns, is called the dissimilarity distance function. This concept was first introduced by Arpat (2005), and Suzuki and Caers (2006).
This dissimilarity distance measure should have two properties:

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

For this reason, different distance functions of interest to our work are introduced and analyzed according to the properties mentioned above.

2.2.1 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 $\mathbf{pat}_{\mathbf{T}}^{\mathbf{m}}(\mathbf{u})$, $\mathbf{pat}_{\mathbf{T}}^{\mathbf{n}}(\mathbf{u})$ be two patterns from the pattern database $\mathbf{patdb_T}$. The Euclidean distance $d_E(x,y)$ is given by:

$$\mathbf{d}_{\mathbf{E}}\left\langle \mathbf{pat}_{\mathbf{T}}^{\mathbf{m}}(\mathbf{u}), \mathbf{pat}_{\mathbf{T}}^{\mathbf{n}}(\mathbf{u}) \right\rangle = \sum_{\alpha=1}^{\mathbf{n_T}} \left(\mathbf{pat}_{\mathbf{T}}^{\mathbf{m}}(\mathbf{h}_\alpha) - \mathbf{pat}_{\mathbf{T}}^{\mathbf{n}}(\mathbf{h}_\alpha) \right)^2$$

As can be seen in formula above, the Euclidean distance is only a summation of the pixel-wise differences. However, according to the analysis made in this study, this distance function may not provide much discriminative power for complex patterns.

2.2.2 Hausdorff and Modified Hausdorff distances

In this distance function, each fixed-value $\mathbf{pat}_{\mathbf{T}}^{\mathbf{k}}(\mathbf{h}_\alpha)$ in a specific pattern $\mathbf{pat}_{\mathbf{T}}^{\mathbf{k}}(\mathbf{u})$ is treated as a point in a set, the "shape" of the template. The Hausdorff measure provides a means for determining the resemblance of one point set to another, by examining the fraction of points in one set that lie near the points in the other set. In other words, the Hausdorff distance is the minimum number r such that the closed r-neighborhood of any point "a" in A contains at least one point "b" of B and vice versa. For instance, if we draw a sphere around each point in the first pattern, the minimum sphere radius that results in covering all the points in the other pattern is the Hausdorff distance between the patterns. This is visually illustrated in Figure 1. This measure is different from the previous pixel-based matching because there is no pairing of pixels in the two patterns being compared. Additionally, another distance function named the Modified Hausdorff distance, is also a metric distance function used in our analysis. It is just a modification made to the original formula in order to bring more discriminatory power to the distance measure. For the formulation of the Hausdorff and Modified Hausdorff distance functions refer to appendix A.

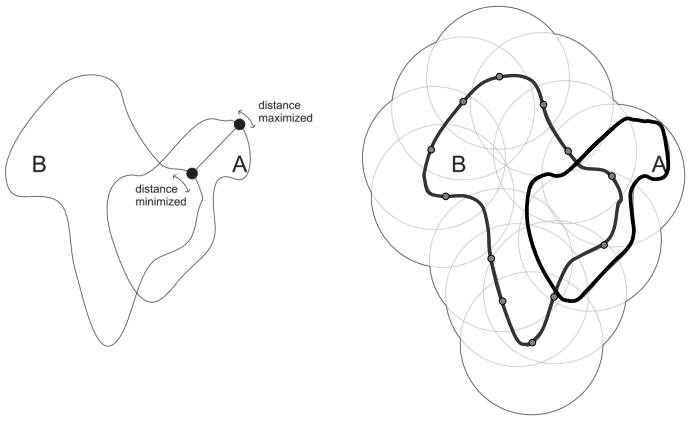


Figure 1: ***A visual depiction of Hausdorff distance. Left: Picking the largest of the minimum distances between two shapes. Right: Intuitive visual meaning is to pick the smallest expansion of B to cover A.***

2.2.3 Hough/Radon transform distance

These are two different distance methods that are based on the Hough transform and Radon transform. Basically, the dissimilarity distance between two patterns is calculated by taking the Euclidean distance between their corresponding transformations. For clarity, a brief explanation for each of these transformations will be given here.

The Hough transform is a general technique for identifying the locations and orientations of certain types of features, such as straight lines, in a pattern. The characteristics of each line will be represented by two values of distance and angle (of a perpendicular line drawn from origin to it). Hence, a line in a two-dimensional pattern will produce a point in the Hough transform, where the coordinates are distance and angle.

Radon transform, or x-ray transform is to some extent similar to Hough transform. We will not dive into details of these transformations, but generally speaking, they are able to transform two-dimensional images with lines into a domain of possible line parameters, where each line in the image will give a peak positioned at the corresponding line parameter. An illustrative example of the result of applying these two transformations to a simple pattern, consisting of a line, is given in Figure 2. For further information refer to Deans, Stanley R., (1983).

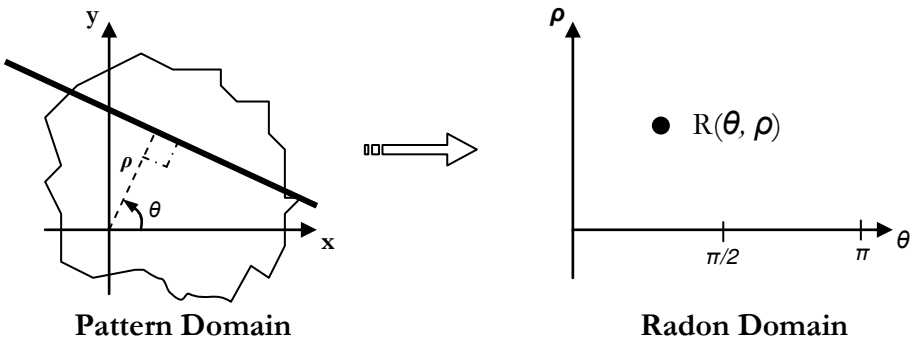


Figure 2: ***An illustrative example of Radon/Hough transform. It can be observed that in both methods, a line is mapped into another feature space and is denoted by a point.***

2.2.4 Distance/Proximity transform

Many image analysis applications require the measurement of objects, the components of objects or the relationship between objects. One technique that may be used in a wide variety of applications is the distance transform or Euclidean distance map (Russ, 1995). 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 and a background, the distance transform assigns each pixel its distance to the closest object. This process of converting a binary image to an approximate distance image is called distance transformation (DT). Refer to Appendix A for the formulation of DT. In this study, the "Chamfer 3-4 algorithm" is used to easily and efficiently calculate this distance transform by approximating the Euclidean metric (Borgefores, 1986 and Borgefores, 1984).
Another useful transformation is the proximity transform, which is just an inverse normalized transformation of the distance transform. So, after this transformation, each node **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.

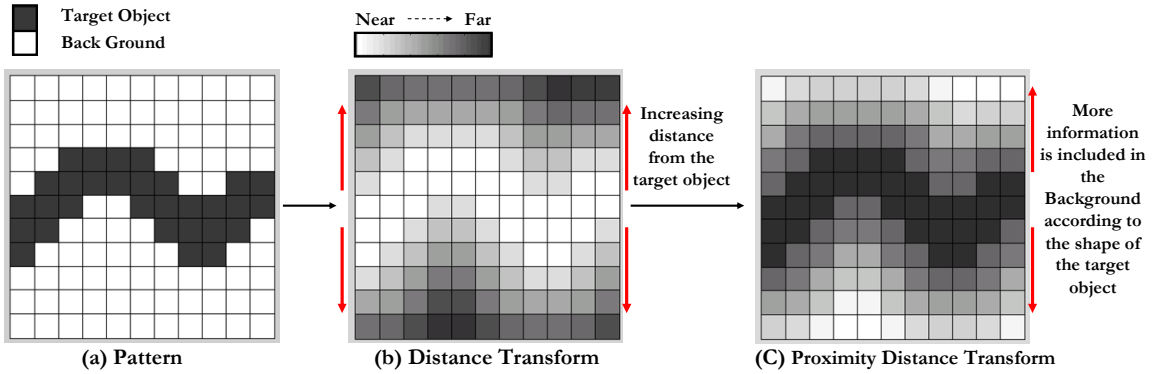


Figure 3: ***illustration of distance transformation and proximity distance transformation on a pattern***

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 distance measures.
It should be noted that the distance transform method, using chamfer's algorithm, will assign infinity to every pixel in a situation where the patterns only consist of background. Instead, the maximum possible value of distance transform, $\sqrt{2}\cdot\left(\sqrt{n_{\Gamma}}-1\right)$, is manually assigned to all the pixels of the pattern.

2.3 Distance Functions Comparison

One of the mostly used dissimilarity functions is the Euclidean distance or similarly the Manhattan distance. According to the analysis made by Arpat (2004), 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 less dissimilar patterns (For further examples regarding this issue refer to Arpat, 2004). Hence, in our study we will not use this dissimilarity measure.

Additionally, the classification obtained by distance transform method has a slightly better performance than the one with proximity transform. Therefore, one may think that proxy distance transform, which is just the inverse normalized outcome of distance transform, is a poor dissimilarity measure for pattern classification. However, the same argument of visual human expertise comes into play. To clarify the point, we have analyzed some of the patterns in a training image (with dimensions of 41×41, using a 9×9 template). Some of the results of dissimilarity measurements using proximity transform and distance transform are shown in Figure 4. Visually speaking, a human expert would categorize pairs of patterns with decreasing similarity as follows:

$$\mathbb{A}\leftrightarrow\mathbb{C} > \mathbb{B}\leftrightarrow\mathbb{C} > \mathbb{A}\leftrightarrow\mathbb{B}$$

with $\leftrightarrow$ signifying similarity. However as visually observed in Figure 4, 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 our visual intuition. This is clearly seen in the dissimilarity function values obtained with these two methods.

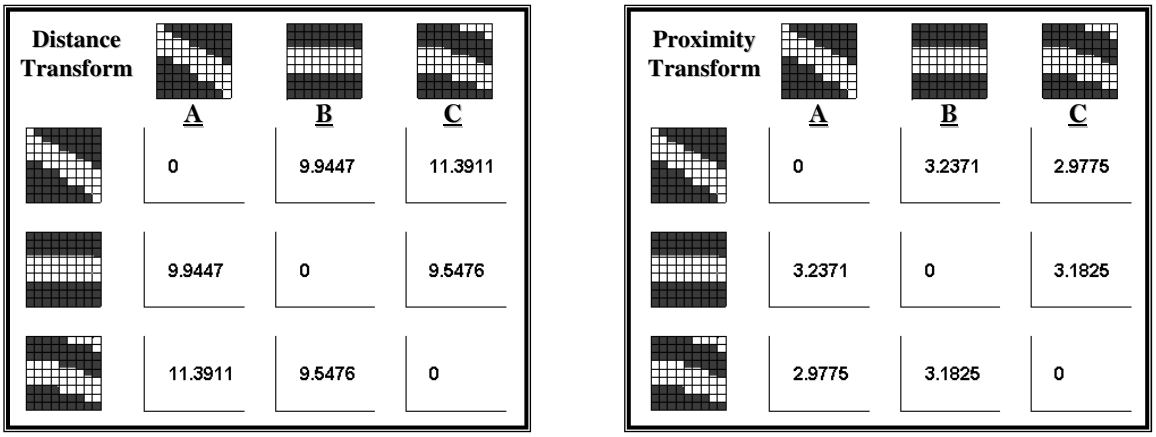


Figure 4: ***Distance transform and Proximity transform dissimilarity capabilities' comparison, showing correct similarity measurements between pairs of patterns for proximity transform, and wrong measurements for distance transform.***

The same analysis was also made for Hough and Radon transform distance functions. For both, there was a mismatch with visual perception. However, their result are somewhat accurate as the dissimilarity achieved was: $\mathbb{A}\leftrightarrow\mathbb{B} > \mathbb{A}\leftrightarrow\mathbb{C} > \mathbb{B}\leftrightarrow\mathbb{C}$, 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 best distance function, we will choose proximity transform as the one for performing classification.

3 Multidimensional scaling

The first purpose of the study is to classify the patterns of a training image. We introduced a measure of similarity between patterns as a method to cluster patterns. However, by just constructing the dissimilarity Matrix, Δ, we can not easily achieve this goal. We need a technique that can reveal the hidden structures underlying the patterns. Multidimensional scaling (MDS) encompasses a collection of methods which allow gaining insight in the underlying structure of relations between patterns by providing a geometrical representation of their similarities. The main assumption in MDS is that the patterns can be described by values along a set of dimensions that places these patterns as points in multidimensional space and that the dissimilarity between the patterns is related to the distances of the corresponding points in the multidimensional space. In other words, the inter-point distances in MDS space are related to the obtained similarities by some distance function. Additionally, MDS reveals the inner dimensions in the patterns that can meaningfully describe them. This multidimensional representation is evidently useful as a basis for building a mathematical model for categorization that we will later use in pattern classification. Thus mathematically speaking, this technique translates the dissimilarity matrix into a configuration of points in n-D Euclidean space (Borg and Groenen, 1997).

In this study, we assume that the dissimilarity matrix displays metric properties. Hence, classical MDS will be used. An illustration of an application of classical MDS is represented in Figure 5. For illustration purposes, a 2D Euclidean space was chosen for mapping of all the patterns in 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 verified in Figure 5.

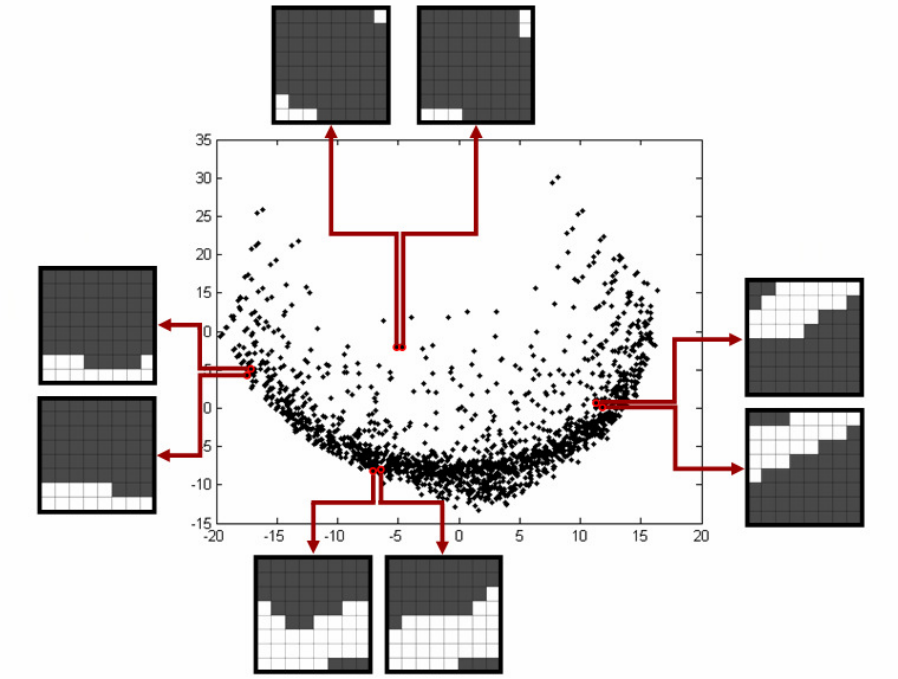


Figure 5: ***Multidimensional scaling sample result using a 8 x 8 pattern template. Any two patterns that are similar map as two close points in Euclidean space.***

An important issue in MDS is choosing the number of dimensions for the scaling solution (2D in Figure 5). A configuration with a high number of dimensions achieves very low errors, but it can result in computationally expensive mathematical analysis later on. On the other hand, a solution with too few dimensions might not reveal enough of the structures in the data.

The methodology we used to select the dimensionality stems from the correlation between the inter-point distances in Euclidean space and their corresponding dissimilarity distance values. Here, we plot the correlation coefficient 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 does not increase substantially. The workflow for choosing the dimensionality of the MDS space is shown in Figure 6. An example is also provided for this workflow in Figure 7, where a 64 dimensional pattern (8×8 pattern template) can be represented in a lower 5-dimensional space with a correlation coefficient of 0.9992, which means that the original dissimilarity distance can be very accurately represented by a 5D Euclidean distance.

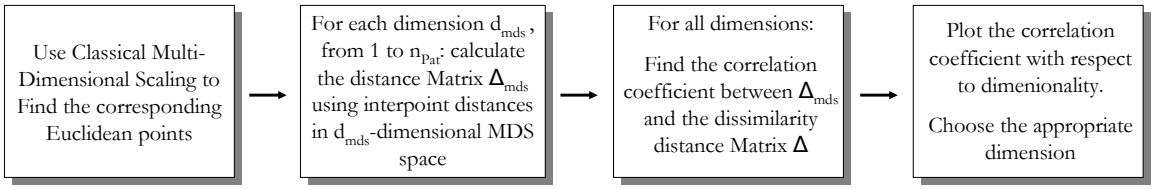


Figure 6: ***Workflow to choose an appropriate MDS space dimension using MDS***

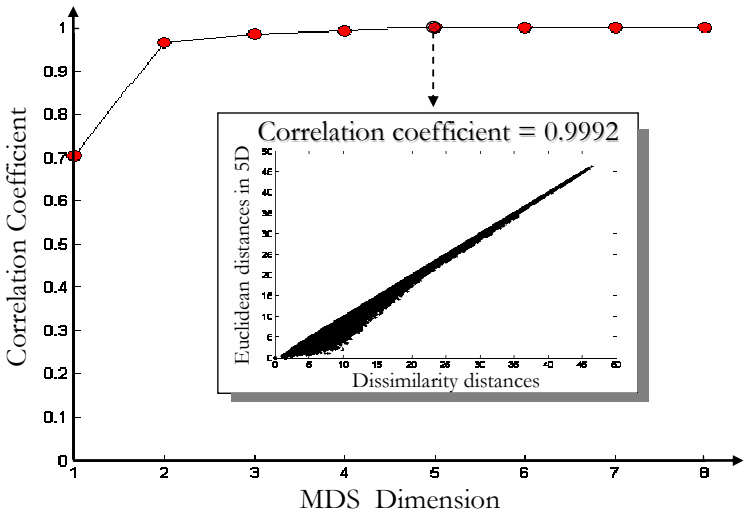


Figure 7: ***An example using a*** 100×100 ***training image with a*** 8×8 ***pattern template, using distance transform as the dissimilarity measure. Plot shows an intact dimension reduction from 64 to 5***

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, as a big dimension reduction is easily achieved according to different experiments. In fact, this method can also be interpreted as a dimensionality reduction algorithm applied on the original high-dimensional patterns.

4 Pattern clustering and Kernel method

4.1 Clustering methods

So far, we have mapped each pattern 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. To this end, we will need to employ clustering methods. Clustering will organize the collection of patterns in the pattern database (represented by points in a multidimensional 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. An illustrative example of clustering is depicted in Figure 8. The input patterns, represented by points are shown in Figure 8 (a), and the desired clusters are shown in Figure 8 (b). As can be observed, points belonging to the same cluster are given the same label.

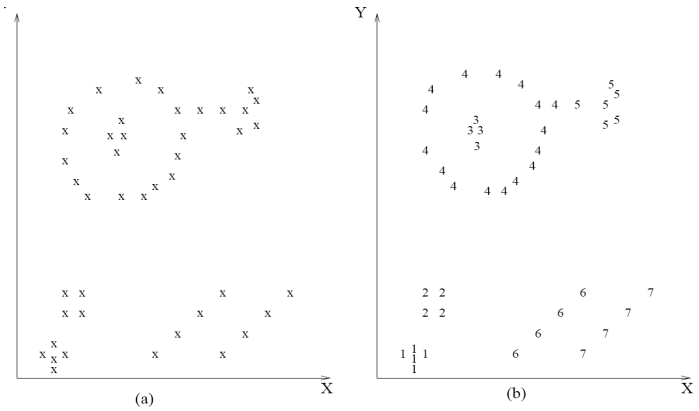


Figure 8: ***Clustering illustration***

Before introducing the clustering methods, some notations employed in this research are first provided for clarity.

4.1.1 Notation

Suppose that all patterns have been mapped into a d-dimensional MDS space as points. Hence, we can now show each pattern by a data point $\mathbf{x_k}$ in a d-dimensional row vector as follow:

$$\mathbf{x_k} = \{\mathbf{x_{k1}}, \mathbf{x_{k2}}, \ldots, \mathbf{x_{kd}}\}, \mathbf{x} \in \mathbf{R^d}$$

A set of $N\left(= n_{Pat_T}\right)$ points is then represented by Matrix $\mathbf{X} = \{\mathbf{x_k} \,|\mathbf{k} = 1, 2, \ldots, \mathbf{N}\}$ which is represented as a $N \times d$ matrix as follow:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \cdots & \mathbf{x}_{1d} \\ \mathbf{x}_{21} & \mathbf{x}_{21} & \cdots & \mathbf{x}_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{N1} & \mathbf{x}_{N2} & \cdots & \mathbf{x}_{Nd} \end{bmatrix}$$

Here, the rows of X represents the coordinates of points in the MDS space. Also, d is the MDS dimension, and N is the number of patterns in **patdb$_T$**.

4.1.2 K-Means

One of the most popular methods for clustering is K-Means algorithm. 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 set is $\ell = \{l_1, \ldots, l_N\}$ with $l_i \in \{1, \ldots, k\}$ where k is the number of clusters. This objective of this method is to allocate each pattern (data point) to one of the k clusters in order to minimize the within-cluster sum of squares:

$$\text{Minimize} \quad \sum_{i=1}^{k} \sum_{p \in A_i} \left\| \mathbf{x}_p - \mathbf{v}_i \right\|_2$$

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

$$\mathbf{v}_i = \frac{\sum_{p=1}^{N_i} \mathbf{x}_p}{N_i}, \quad \mathbf{x}_p \in \mathbf{A}_i$$

Where N_i is the number of patterns in A_i. It should be noted that the reason behind choosing a Euclidean norm in the minimization phase of the algorithm is simply because of the specific metric distance used in multidimensional scaling. In mapping the dissimilarity distances between patterns to MDS space, we used Euclidean distance 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, we can now classify patterns in different clusters.

4.1.3 Number of Clusters

One of the main issues in any classification is the decision that should be made by the user on the number of clusters, prior to the algorithm. In order to automate this phase, and most importantly, to find the most appropriate number of clusters, two different methods will be introduced hereafter. In both methods, an index is used to validate the best number of clusters.

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 follow:

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

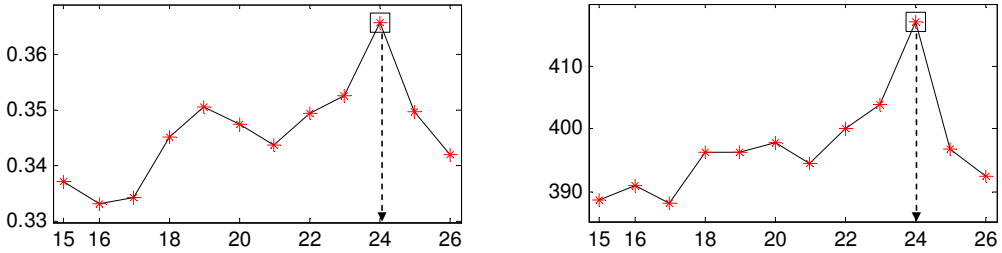
Where a_i is the average distance from the i-th point to the other points in its cluster, and b_i is the minimum (over all clusters) of the average distance from the i-th point to the points in another cluster. The silhouette value for each point is a measure of how similar that point is to points in its own cluster compared to points in other clusters, and ranges from -1 to +1. The average of this index for all the 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 provide excellent recovery in terms of selection of the number of clusters (Milligan, G.W., & Cooper, M.C., 1985).

$$\text{Calinski Harabasz} = CH = \frac{SSB / (k-1)}{SSW / (N-k)}$$

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 9 for a 30×30 TI, and a 8×8 pattern template. As can be seen, both indexes 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 within that constraint.



[a] Silhouette (sil) index [b] Calinski Harabasz (CH) index

Figure 9: ***Silhouette and Calinski-Harabasz (CH) indexes indicating 23 as the best number of clusters***

4.2 Kernel Methods

4.2.1 Definition

Considering the amount of available patterns in the pattern database of a training image, we are interested in using simple, fast and efficient clustering algorithms such as K-means method to classify them. K-means suffers from several drawbacks. For instance, its results strongly depend on the initialization process and it cannot adapt to any cluster shape. That is to say, 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 of our methodology. In order to solve this problem, data will be mapped by a transformation φ into a new feature space $\boldsymbol{F}$ where samples become linearly separable (Shawe-Taylor and Cristianini, 2004). In other words, by mapping the data points to that higher dimensional space $\boldsymbol{F}$, we can capture the nonlinear relationship among the information provided by the data. A sample representation of this process is shown in Figure 10 (Scholkopf and Smola, 2001) where a non-linear dataset is linearized in feature space. As a consequence, in this feature space, classical clustering algorithms such as K-Means will perform better, and so, a stronger and more accurate pattern classification will result. We will next introduce the formulation to achieve this non-linear transformation.

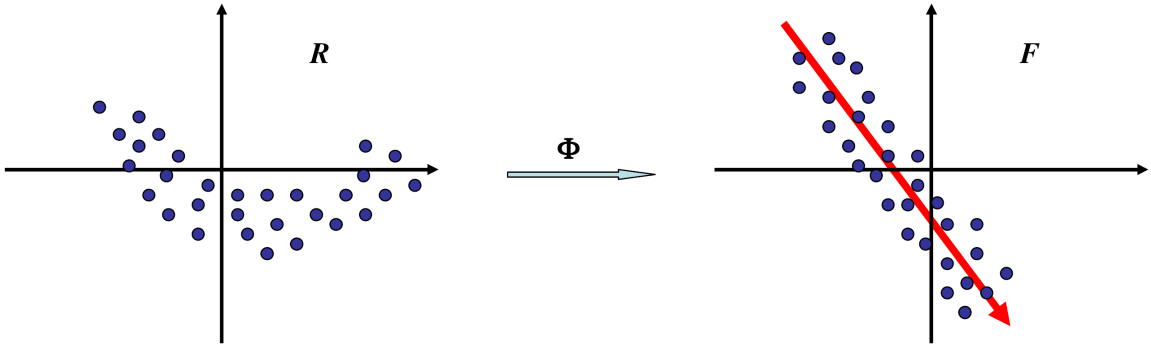


Figure 10: ***Kernel transformation, and the principal component in the linear kernel space***

Generally, by transformation, we want the similarity between the data to be preserved in the feature space. One particularly simple yet surprisingly useful notion of this similarity - the one that we used in this study – was derived from embedding the patterns into a Euclidean space and utilizing geometrical concepts. Likewise, this dissimilarity can be measured by dot product in some high-dimensional feature space $\boldsymbol{F}$. This leads to one of the crucial ingredients of this formulation, "the kernel trick", for the computation of this dot product in the high-dimensional feature space using simple functions defined on pairs of input patterns. Therefore, the patterns are first mapped into $\boldsymbol{F}$ using $\boldsymbol{\Phi : X \to F,\ x \mapsto \phi(x)}$, and then compared using a dot product $\langle \boldsymbol{\phi(x_k), \phi(x_1)} \rangle$.

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 nonlinear function $K(\cdot)$ in the input space, i.e. by means of a kernel trick.

$$\mathbf{K}(\mathbf{x}_k, \mathbf{x}_l) = \langle \phi(\mathbf{x}_k), \phi(\mathbf{x}_l) \rangle$$

The success of this approach is related to the fact that using a kernel is equivalent to defining a feature space transform. In other words, the advantage of the kernel trick is that instead of explicitly determining the coordinates of the data vectors in the feature space, the distance computation in ***F*** can be efficiently performed in $\mathcal{X}$ (input space) through a kernel function. Hence, $\mathbf{K}(\mathbf{x}_k, \mathbf{x}_l)$ is the inner product not of the coordinate vectors $\mathbf{x}_k$ and $\mathbf{x}_l$ in $\mathbf{R}^d$ but of vectors $\phi(\mathbf{x}_k)$ and $\phi(\mathbf{x}_l)$ in higher dimensions. This trick allows the formulation of nonlinear variants of any algorithm that can be cast in terms of dot products, K-Means and PCA being the most prominent examples. Hence, the solution to a better pattern classification can be obtained by using the Kernel K-means algorithm (H. Maitre M. Campedel, 2005) as an alternative to K-Means.
The most frequently used kernel function, as used in our study, is the Gaussian radial basis function, which is given by:

$$\mathbf{K}(\mathbf{x}_k, \mathbf{x}_l) = \mathbf{e}^{-\frac{\|\mathbf{x}_k - \mathbf{x}_l\|^2}{2\sigma^2}}$$

The incorporation of this kernel function enables the K-Means algorithm to explore the inherent data pattern in the new feature space ***F***. An example of the beneficial result of clustering using kernel k-means (in linear feature space) in comparison with k-means (in non-linear Euclidean space) is depicted in Figure 11.

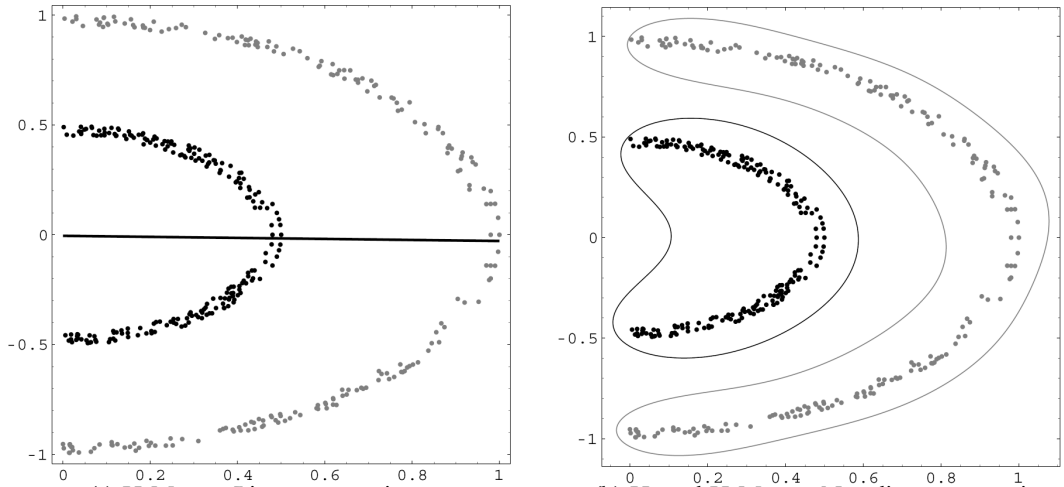


Figure 11: ***(a) K-Means on dataset. The solid line indicates the linear separation line determined by K-Means. (b) Kernel K-Means on dataset. The region delimited by black line identifies the first cluster which has been properly separated from the second data cluster, identified by the gray line.***

4.2.2 Number of Clusters

Similar to the previous case of K-Means, we need to find the most suitable number of clusters automatically, without user interaction. However, as we are using the kernel k-means method, another procedure should be undertaken to account for the feature space transformation. In this procedure, the optimal number of clusters is selected on the basis of minimum description length (MDL) principle (Rissanen, 1983), in a trade-off between the likelihood of the model to the given data and the complexity of the model itself. In order to understand intuitively the MDL principle we will provide an example for model selection and overfitting.

Consider the points in Figure 12. We would like to learn how the y-values depend on the x-values. To this end, we may want to fit a polynomial to the points. Straightforward linear regression will provide the leftmost polynomial - a straight line that seems overly simple: it does not capture the regularities in the data well. Since for any set of n points there exists a polynomial of the $(n - 1)$st degree that goes exactly through all these points, simply looking for the polynomial with the least error will give us a polynomial like the one in the second picture. This polynomial seems overly complex: it reflects the random fluctuations in the data rather than the general pattern underlying it. Instead of picking the overly simple or the overly complex polynomial, it seems more reasonable to prefer a relatively simple polynomial with small but nonzero error, as in the rightmost picture. Intuitively, if one naively fits a high-degree polynomial to a small sample (set of data points), then one obtains a very good fit to the data. Yet if one tests the inferred polynomial on a second set of data coming from the same source, it typically fits this test data very badly in the sense that there is a large distance between the polynomial and the new data points. We say that the polynomial overfits the data. Indeed, all model selection methods that are used in practice either implicitly or explicitly choose a tradeoff between goodness-of-fit and complexity of the models involved. In practice, such tradeoffs lead to much better predictions of test data than one would get by adopting the "simplest" (one degree) or most "complex" (n−1 degree) polynomial. MDL provides one particular way of achieving such a tradeoff.

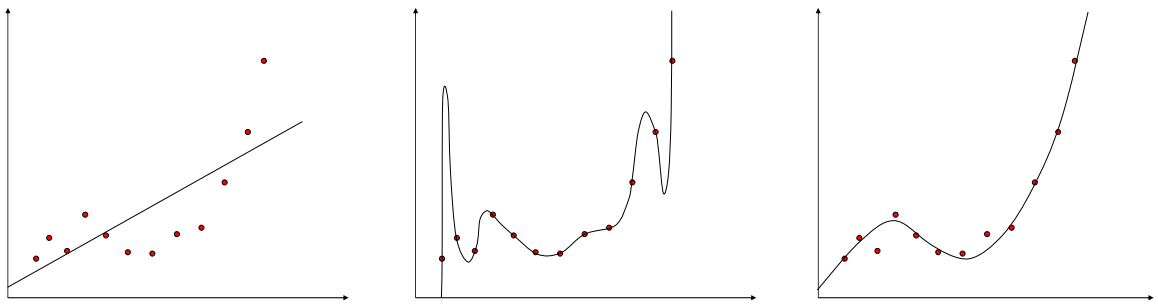


Figure 12: ***A simple, complex and a tradeoff (third-degree) polynomial***

An intuitive explanation of MDL aids in formulating a procedure for selecting an appropriate amount of clusters. The MDL formulation for kernel k-means consists of plotting a metric, KMDL, versus the number of clusters k as follow:

$$\mathrm{KMDL}(k) = -\sum_{l=1}^{k} n_l \log\left(\frac{n_l^2}{S_l^d}\right) + k\left(d^2 + 3d + 2\right)\log(N)/2$$

Where, N is the number of patterns n_{Pat_T}, d is the dimension of the points in the MDS space, n_l is the number of points in l-th cluster and S_l is the sum-squares distance from patterns to their corresponding l-th cluster centroid. For interested reader, the mathematical background of this formulation is presented in Appendix B. The minimum value in the KMDL curve plotted with respect to the number of clusters k will provide the appropriate number of clusters. Hence, using this number we can classify the patterns in the feature space.

Having explained all the steps involved in the pattern classification process, next we provide a case study on a training image example. In the next section, we will first introduce the concept of pattern skipping. And then, by providing some means of pattern classification validity index, different distance functions for dissimilarity measurement will be investigated in the context of the final classification capability. Our results will be compared with the FilterSim method.

5 Case Study

5.1 Pattern skipping Concept

Due to 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. Also, the distance calculations between all the patterns imply at least $\frac{1}{2} n_{Pat_T}^2$ similarity calculations and their storage. In this study, a new idea of pattern skipping is introduced to avoid this drawback. 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 2 lags, every other pattern is skipped. That is, if one pattern is located at location **u** as its center point, the next pattern to be considered will be located at **u+2h** (**h** being the distance between grid cells). An illustration of pattern skipping is given in Figure 13. By this method, the size of the pattern database, and also, the dimensions of the dissimilarity matrix will be reduced by a factor of (skip-size)^2, which can 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 would still cover almost every possible pattern.

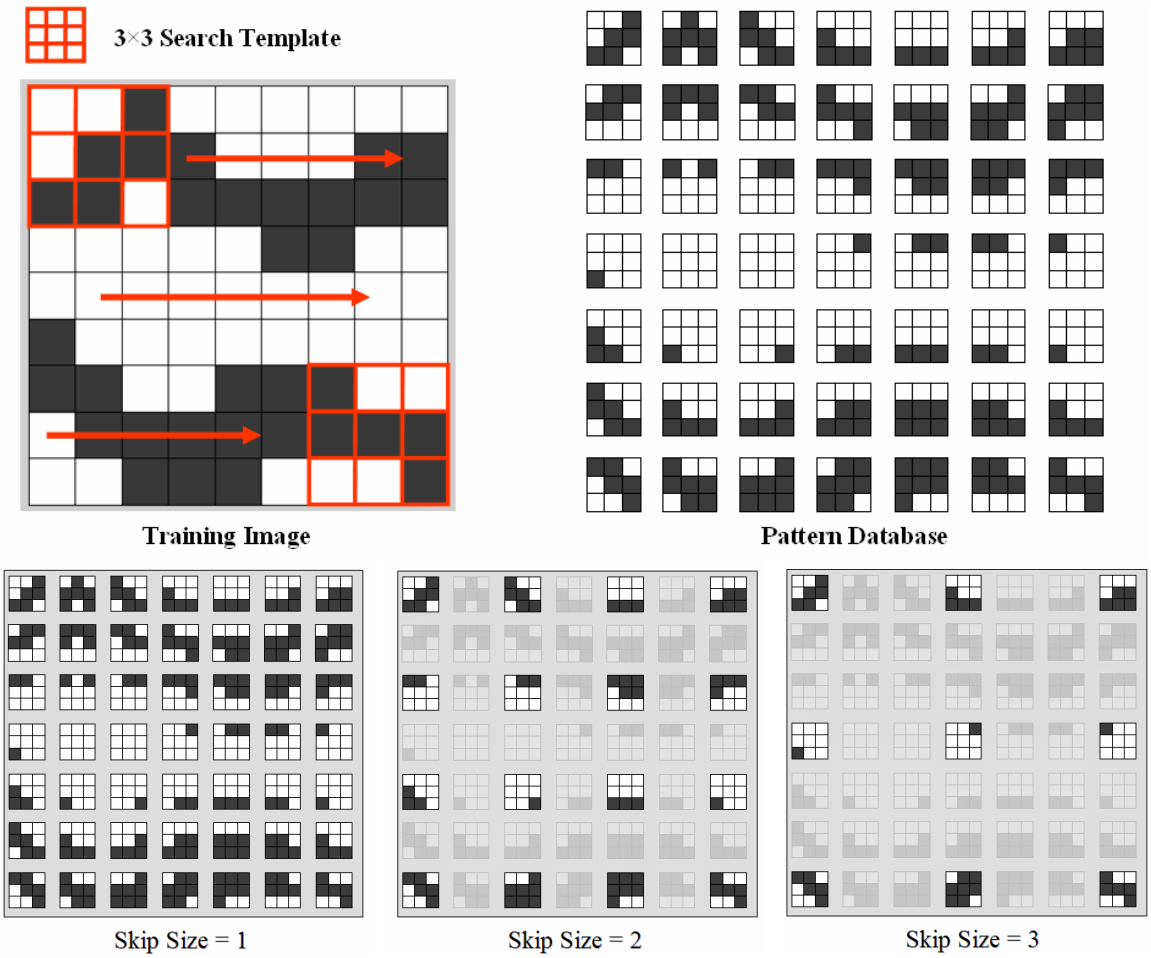


Figure 13: ***Showing training image of 9×9, scanned with a 3×3 template resulting in all possible patterns. Concept of pattern skipping is shown for skip-sizes of 1 (None), 2 and 3***

5.2 Training Image

The training image that was used in this study has binary sand/shale channel structures. Figure 23 shows this training image. The workflow of the pattern classification part of this study goes as follow:

1: *Extract all existing patterns from the training image, with the specified template, and collect them in the pattern database.*

2: *Construct the dissimilarity Matrix using a specific distance function.*

3: *Use MDS to map the patterns to points in a Euclidean space* R^d*, with its dimension, d, chosen such that the resulting Euclidean distance has a desired correlation coefficient with the original distance function.*

4: *Map the data into the kernel space (implicit in algorithm).*

5: *K-means clustering in kernel space for pattern classification*

Here, we have used a 100×100 training image, with a template size of 8×8. The results are shown in Figure 24 to Figure 28. Different skip-sizes and multiple-grid values were tested as well as different distance functions. The classification result shown here is for the case of distance transform method with 12-dimensional feature space and 25 clusters. It is notable that by having 12 dimensions, which is a huge reduction from the original 64-dimensional pattern, a correlation coefficient of 0.99996 is obtained, as can be seen in Figure 24.

The benefit of MDS space to feature space transformation is illustrated in Figure 25. As one can observe, the unstructured scatter of data points in the Euclidean space are now distributed in a more structures fashion in the feature space, shown using the first 3 principal components in that space.

Finally, the prototypes of the resulting clusters are shown in Figure 26. 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 should 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 a sample pattern within 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 1 pixel in one pattern will be zero 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 26 is considered sharp, while cluster 20 is not. According to these qualitative visual concepts, we introduced a quantitative sharpness index which can help in validating the classification accuracy. In simple words, if a pixel in a prototype is close to 0.5, which indicates poor clustering, we rate it as zero and for the pixels close to 1 or zero, which indicates similarity of the patterns within that cluster, we rate them as 1. The sharpness index is then just an averaged sum of all these values. The formula for sharpness index is as follow:

$$\text{Sharpness Index} = \text{SI} = \frac{\sum_{L_i \in L} \left| 2 \times \mathbf{v_i} - 1 \right|}{n_T} \in (0,1) \qquad \text{, where } \mathbf{v_i} = \frac{\sum_{k \in L_i} \mathbf{pat_T^k(u)}}{n_{L_i}}$$

However, suppose that a pattern classification, with 25 clusters, resulted in 24 clusters having only 1 pattern and the last cluster contains all the remaining patterns in the database. This results in a high sharpness index, because the 24 perfect clusters, each with SI=1, are averaged with one cluster having $\text{SI} \approx 0.5$. 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 result of this, the situation described above would result in a small sharpness index, indicating poor classification. The following formula is therefore proposed:

$$\text{weighted SI} = \text{WSI} = \frac{1}{n_T} \sum_{\text{clusters}} \left(\frac{n_{L_i}}{n_{Pat_T}} \right) \times \left| 2 \times \mathbf{v_i} - 1 \right| \qquad \text{, where } \mathbf{v_i} = \frac{\sum_{k \in L_i} \mathbf{pat_T^k(u)}}{n_{L_i}}$$

One shortfall of this formula is produced by the pixel-wise averaging technique used in obtaining the cluster prototypes. They utilize the same Euclidean distance comparison method that was deemed unreliable. That is to say that a "sharp" prototype results from averaging the patterns that are similar pixel-wise. This has a misleading effect when comparing different distance functions for their final classification capability. To avoid this misinterpretation, two other sharpness indexes were also calculated accordingly. For both of them, 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.

The results of all sharpness indexes for different distance methods are given in Figure 27. In this figure, the x-axis represents skip-size in pattern skipping. To some extent, they all give similar results. This is a satisfactory result in the sense that we are free to use any distance method applicable to our study, and the results will reflect similar sharpness indexes. The pattern database obtained from a multiple grid approach is also analyzed and the classification results are plotted in Figure 28. This figure shows that when having a greater pattern complexity (resulted from multiple gridding), the sharpness indexes for different distance methods still remain the same. The decline in SI with patterns from coarser grids is anticipated because of the fact that we have more diverse and dissimilar patterns in the database.

5.3 FilterSim Comparison

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 6 different filters to map the patterns into a 6-dimensional space, where the pattern 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 our algorithm, a sample training image is chosen, and the pattern clustering capabilities of both methods is compared in terms of sharpness indexes.

The training image used here is shown in Figure 29. As can be seen, this looks like the previous one. However, the dimensions of this training image is 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 that more dissimilarities will be introduced between the patterns in the database, and hence, the more difficult it will become to classify them. This can help us observe the differences more clearly. We have chosen 20 clusters for the final results. In Filtersim, for the sake of comparison, the k-means algorithm is also used for classification.

Different ways of comparison are introduced hereafter. The first one is the graphical portrayal of the clustering results. This is achieved by showing the training image grid, and then, assigning to each grid location $\mathbf{u}$, a value which represents the cluster label that the pattern, $\mathrm{pat}_{\mathrm{T}}(\mathbf{u})$, belongs to. Therefore, the resulting figure can be a way to visually evaluate the classification method. Here, if such "cluster image" shows clear structures, then this may indicate a good classification. The "cluster image" of a high-resolution training image is given in Figure 30. One can note the analogy between the training image and its corresponding cluster image. It mimics the trends of channels in the training image, which justifies a good pattern classification. Evidently, for the lower resolution training image, which is used here for classification comparisons, the cluster image may not perfectly mimic the structures of the training image. However, the closer they look to the high-resolution cluster image, the better the classification algorithm is. The cluster images using our method and the FilterSim method are shown in Figure 31. Accurately inspecting the structures in these figures, one can undoubtedly observe that our algorithm achieves a better classification.

Besides these cluster representations, we can also look at the cluster prototypes. As presented previously, the sharpness index of a prototype is a good measure for the accuracy of pattern classification. To this account, Figure 32 and Figure 33 show the prototypes of each cluster obtained by FilterSim and our method. The sharpness index of each prototype is indicated below them. Visual perception indicates that a better classification is obtained with our algorithm. Nevertheless, numerically speaking, Table 1 summarizes the average and weighted average sharpness indexes for each case. The huge difference between these values substantiates the powerful classification capability of our method in comparison with the method employed in FilterSim.

Table 1: ***FilterSim comparison with our method in terms of sharpness index***

	Sharpness Index	Weighted Sharpness Index
FilterSim Method	0.395735	0.399662
Proposed Algorithm	0.523792	0.523600

Concluding the effectiveness of our procedure, we are now going to introduce a method for pattern generation in the following section.

6 Pattern Generation

Neither FilterSim, nor Simpat introduce or create patterns that are different from the database. 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. This is the purpose of this section.

To this end, suppose that we have a data event. In previous algorithms, a search for the most similar pattern to this data event will be performed. In FilterSim method, this search normally starts by comparing the data event with all the prototypes. After finding the most similar prototype to the data event, a pattern from that cluster is randomly selected and pasted onto the simulation grid. In our method, we follow the same steps. However, instead of randomly selecting one of the patterns within that cluster, a random pattern will be generated. This new pattern does not exist in the pattern database, but it is similar to the patterns in that specific cluster. The overall methodology on how this is done is explained below and is also illustrated in Figure 14.

1: *Using a measure of similarity, obtain the most similar prototype to our data event.*
2: *Find the patterns within the cluster of that prototype.*
3: *Randomly sample a point from the multivariate distribution of the cluster points in kernel space.*
4: *map the sample point back into the feature space (MDS space) using the pre-image method.*
5: *Generate a new pattern using that point in MDS space by one of the following methods*:

I : *Morph the most similar patterns (being closest in MDS space) to the sampled point in order to obtain a new pattern. (Presented in the next section)*

II: *Map the sampled point from MDS space back to the pattern space. (By solving the pre-image problem from MDS space back to the initial pattern space)*

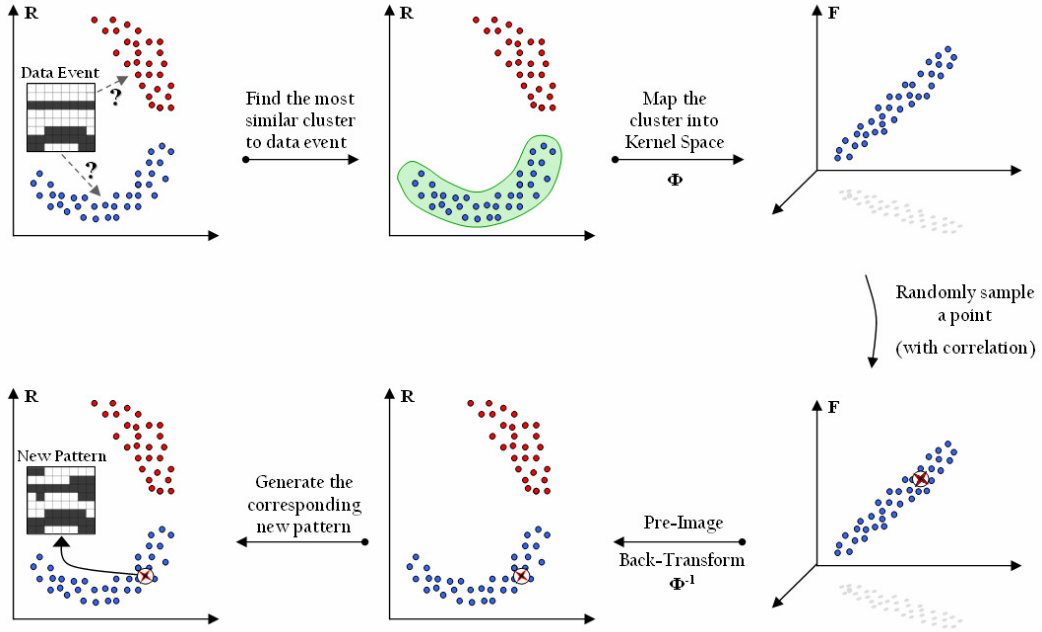


Figure 14: ***The methodology for random sampling and pattern generation process***

6.1 Sampling

In order to clarify the proposed methodology in Figure 14 we present it on an example dataset. Suppose that a cluster of patterns is shown in a 2-D MDS space by their corresponding points. For consistency with the methodology, assume that these points represent the class of patterns having a prototype most similar to our data event. Figure 15 shows these data points which are scattered non-linearly around a hypothetical circle. As stated in the methodology, after finding the cluster, the next step in the pattern generation algorithm is to randomly sample a point that would belong to the same cluster. We expect the sampled data to be located within the span of the original dataset, for instance in this sample test, around the perimeter of the circle. However, traditional random sampling from this non-linear dataset would result in the sampled data to unfavorably cover the whole area of the circle as shown in Figure 16. In order to circumvent this obstacle, we have to sample from a better structured dataset. Indeed, mapping the data from the MDS space to the feature space increases the linear separability of the patterns within the transformed space and therefore simplifies the associated data structure. Then, by using Kernel PCA, we can satisfactorily perform the desired sampling with a linearly structured data.

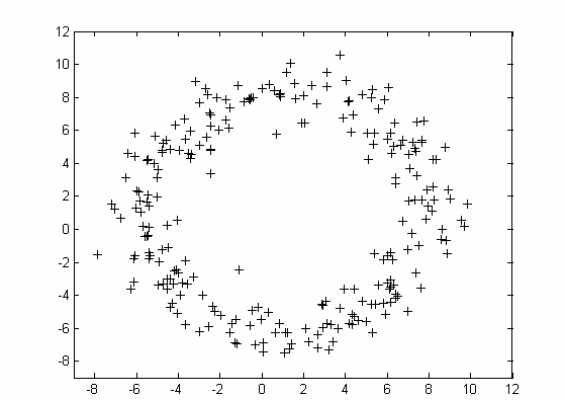


Figure 15: ***Sample test dataset, a circular shape data points with standard deviation of 1 for noise***

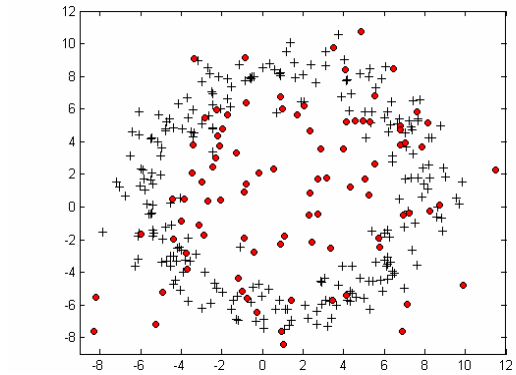


Figure 16: ***100 randomly sampled data (red) from the original non-linear (circular) data points (black)***

In this study, in order to sample a point, a multivariate normal distribution for data point vectors is assumed. By using simple random sampling, we form a random vector from

prescribed probability distribution, and consequently, a randomly sampled point is obtained. The algorithm has to consider also the correlation among the data. Otherwise, the samples will be uncorrelated in the feature space, causing the same problem as was shown for the MDS space. This is accomplished by using LU decomposition method proposed by Davis (1987) to incorporate the correlation. However, the sampled point is still in kernel space and to retrieve the new pattern, it needs to be mapped back into MDS space. This reverse mapping process is called the pre-image problem (Schölkopf et al., 2002). To summarize, by projecting the points to a high-dimensional feature space and randomly sample a point in that space, and then, back-mapping it into the original MDS space, we have a new point which corresponds to a new pattern. To illustrate the end results, 80 data points using simple random sampling (with correlation) are generated on the previous circle dataset and is shown in Figure 18. Their distribution follows the same non-linear behavior inherent in the initial data, as desired.

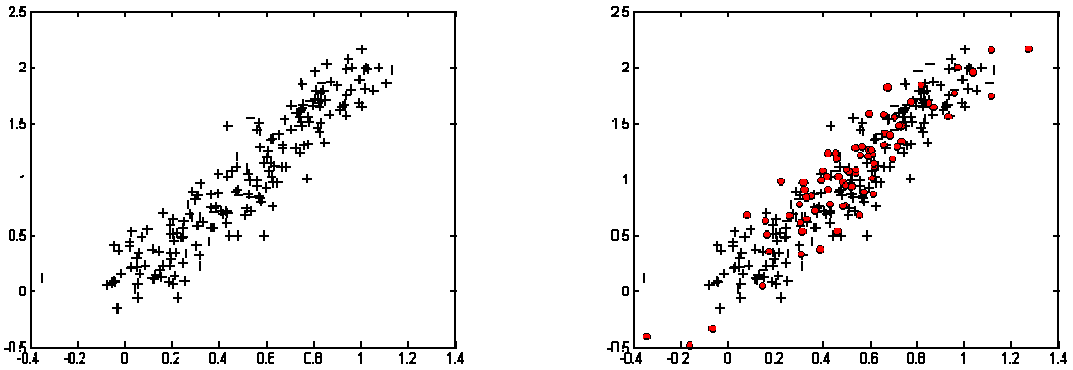


Figure 17: ***data points in these plots are assumed to be in kernel space. Left figure: original data points, showing linear behavior/ right figure: simple random sampling with correlation in the feature space.***

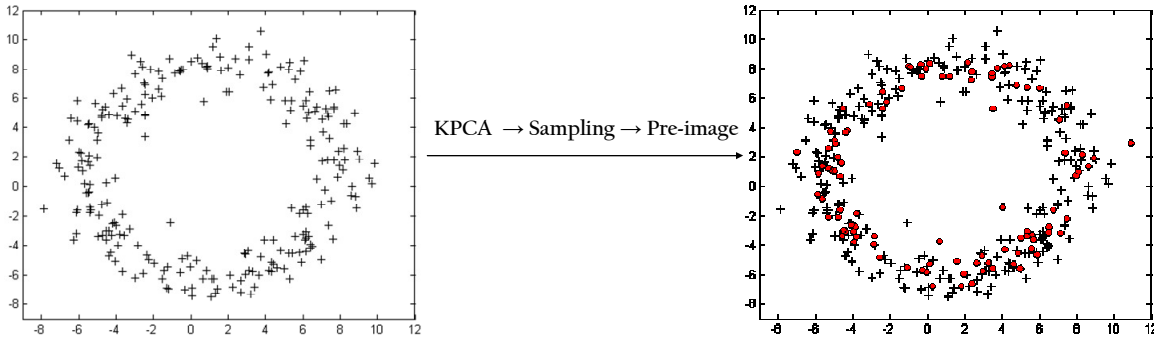


Figure 18: ***Simple random sampling by: (1) mapping into kernel space, (2) random sampling, (3) back-mapping into original input space***

Having finished the random sampling part, we are now at the last step of the methodology which is to use the sampled data point (in the MDS space) in order to generate its corresponding pattern. In this study, only the first technique that was outlined in the methodology is investigated and presented hereafter.

6.2 Pattern Generation

So far, we have presented a methodology for obtaining a new point in the MDS space. This point belongs to a specific cluster whose corresponding prototype is closest to the given data event. As mentioned before, patterns of a training image are represented by points in the MDS space. Thus, a new randomly sampled point in this space should represent a new pattern. In other words, there exists a pattern that if mapped to the MDS space would result in that randomly sampled point. Hereafter, a back-mapping methodology for finding the pattern, which corresponds to a point in MDS space, is presented with some examples.

The idea behind back-mapping comes from the fact that the features in this new pattern should comply with the features present in some other patterns 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. In other words, 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 3 closest points to the sampled point are chosen for pattern generation. The methodology is provided below and is illustrated in Figure 19.

1: *Sample a new point in the MDS space (obtained according to the previous section).*
2: *Find the 3 closest points to the sampled data point.*
3: *Find the corresponding patterns and their relative distances to the sampled point.*
4: *Calculate the Radon Transform of the selected patterns.*
5: *Using inverse distance weighting of their radon transforms to obtain a new radon transformed image.*
6: *Use inverse radon transform (Filtered back-projection algorithm) on this new image and apply a*

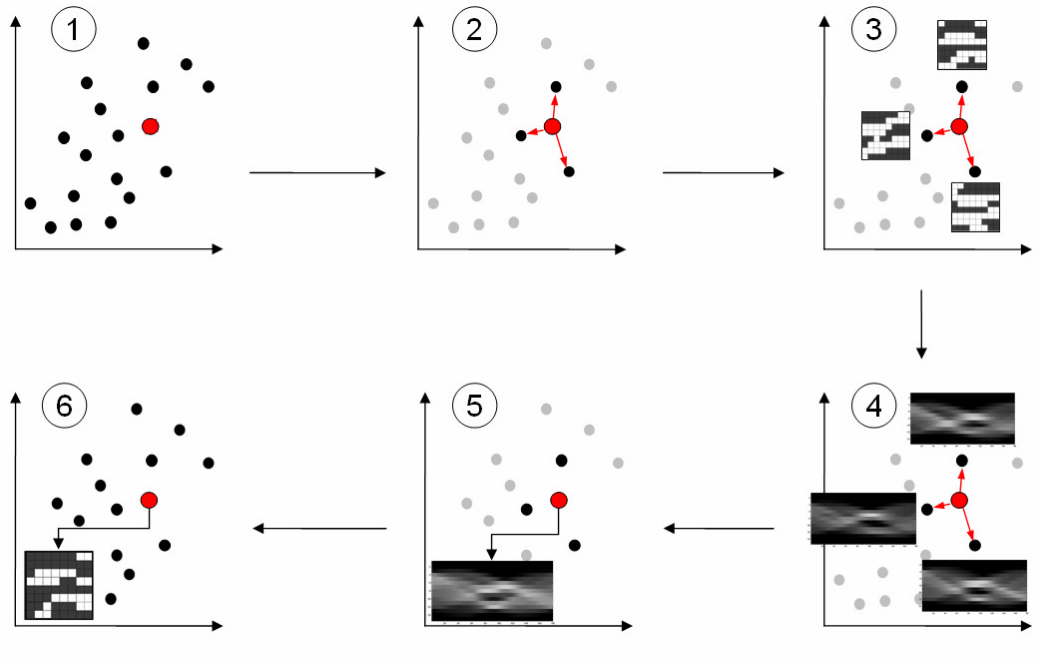


Figure 19: ***New pattern generation methodology***

It should be noted that the Radon transform captures the features in a pattern. Thus, by averaging those radon transforms, the new transformed image conforms to the same features. The Radon transform, as explained before, is a set of 1-D projections along different angles. Therefore, assuming a 2-D pattern as a two-variable function $f(x,y)$, then the projection function $R(r,\alpha)$ can be written as:

$$R(r,\alpha)\left[f(x,y)\right] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\delta(r - x\cos\alpha - y\sin\alpha)\,dx\,dy$$

where r is the perpendicular distance from a line to the origin and α is the angle formed by the distance vector. The collection of all $R(r,\alpha)$ at all angles α is called the Radon transform of the image $f(x,y)$. In order to reconstruct the pattern, we used what is known as the Fourier Slice Theorem. According to this theorem, we can simply take the 2D inverse Fourier Transform to obtain the unknown pattern. More specifically, we can apply an inverse radon transform to reconstruct the pattern. This inverse problem is called filtered back-projection and is formulated as follow:

$$f(x,y) = c\int_{0}^{\pi}\int_{-\infty}^{\infty} \mathcal{F}_{\omega,\alpha}\left[R\left[f(\omega,\alpha)\right]\right]|\omega|\,e^{i\omega(x\cos\alpha + y\sin\alpha)}\,d\omega\,d\alpha$$

where $\mathcal{F}$ is the Fourier Transform. This filtered back-projection is an explicit and computationally efficient inversion algorithm for the two-dimensional Radon transform. Having this algorithm, we can easily reconstruct the pattern that has a specified Radon transformation. Indeed, a threshold value is also chosen according to the proportions in the 3 original patterns, in order to transform the resulting continuous pattern into a binary one.

The effectiveness of this methodology to generate new patterns will be shown here with an example. Three patterns are first selected from one of the pattern clusters, as shown in Figure 20. Next, according to the position of the sampled point to the corresponding patterns in MDS space, new patterns are generated. For illustration purposes, only the points lying between each pair of patterns is considered in this example. As a consequence, by traversing the line between each two points, a new set of patterns can be constructed. Also, we expect these new patterns to smoothly morph from one pattern to the other. An illustration of this methodology is provided in Figure 21 and Figure 22. The first figure shows the locations considered for the sampled points, and the second one shows the smooth transition from one pattern to the other one.

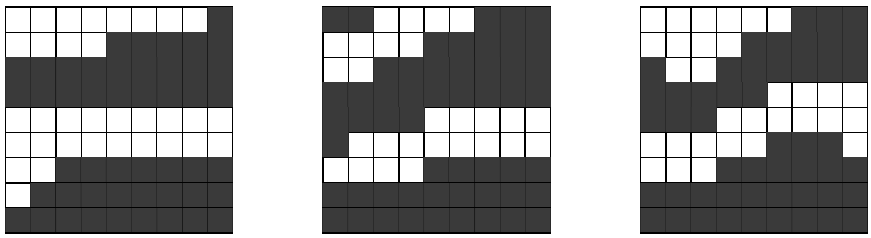


Figure 20: ***Three different patterns selected as the basis for pattern generation.***

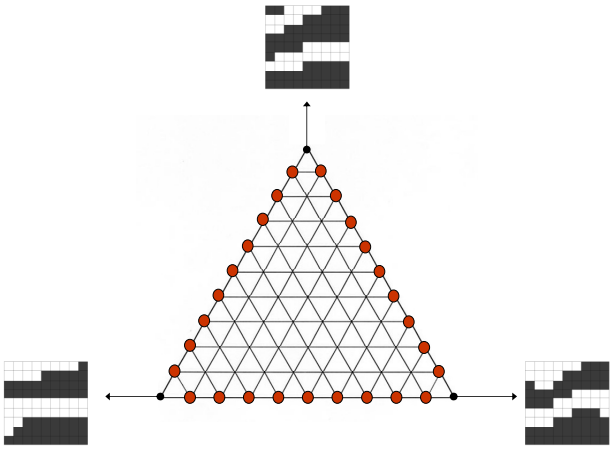


Figure 21: ***Red points indicate the sampled points in MDS space used in pattern generation algorithm.***

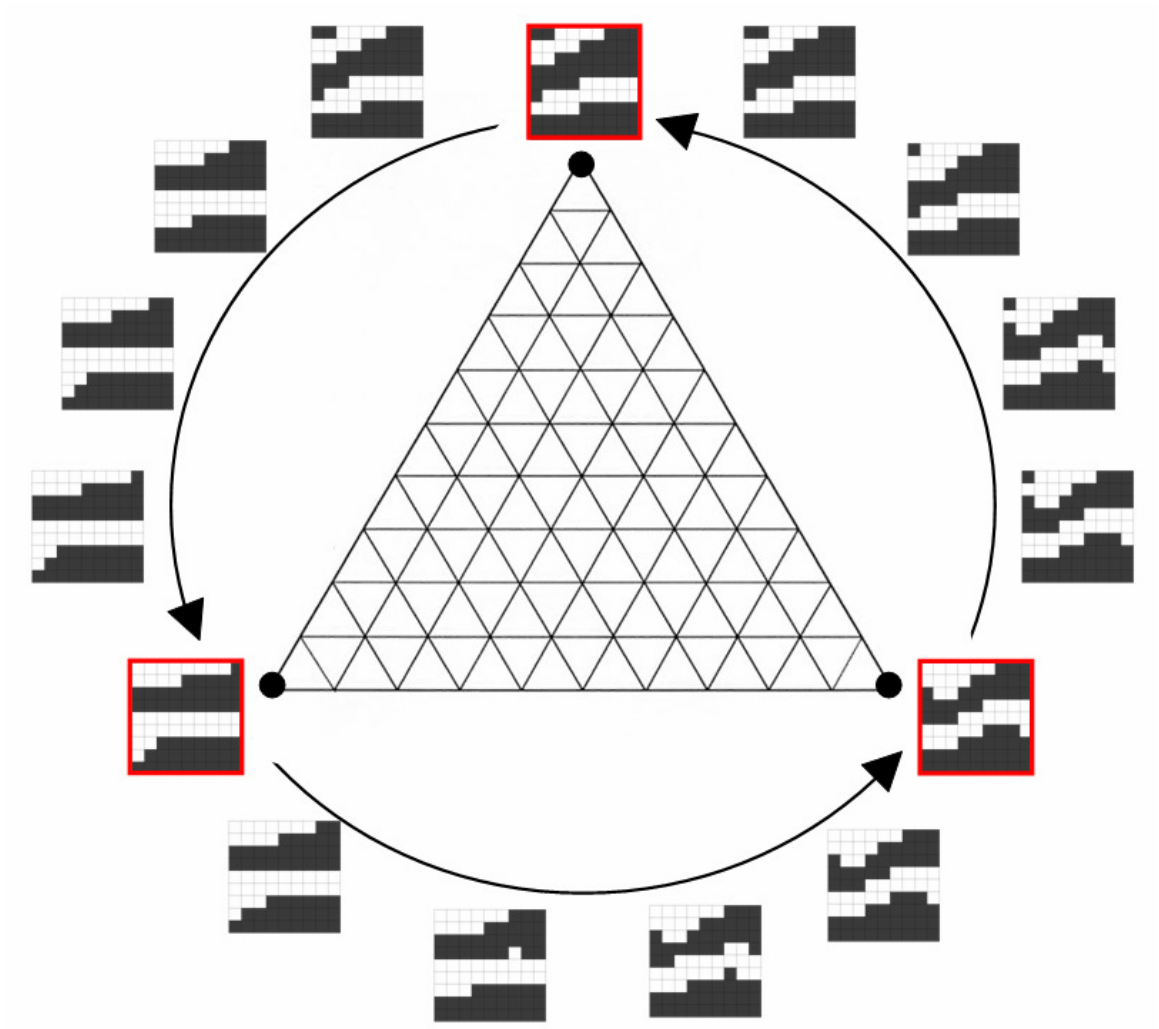


Figure 22: ***Transitional new patterns generated between the original ones, showing smooth morphing from one pattern to the next one. New generated patterns have similar features as their neighboring patterns.***

Conclusions and Future Work

Different algorithms for building geologically realistic reservoir models have been proposed before. They all suffer from some shortcomings such as poor pattern classification, similarity of the generated realizations to the training image, and most importantly being restricted in their applications. In this paper, a new methodology has been proposed. Distance-based methods are used in the context of patterns in a training image. According to this, by using different distance functions, pattern analysis can be easily tailored to the application at hand. In other words, the pair-wise distances between all of the patterns are calculated according to a specific dissimilarity distance function. Then by using the calculated dissimilarities, patterns are mapped into a lower-dimensional MDS space, where all sorts of mathematical algorithms can be applied on their corresponding points, instead of the patterns themselves. On this account, Kernel methods are used to map them into a higher-dimensional feature space where the non-linear structure inherent in the data points can be easily captured. Finally, patterns are classified by applying clustering methods in this space. The proposed methodology provides a better pattern classification in comparison to the algorithm used in the FilterSim.

Additionally in the previous methods, during the sequential simulation, patterns were selected from the pattern database of a training image and patched on the simulation grid. In this study, a new pattern generation algorithm is proposed. According to this, patterns which are pasted on the simulation grid may no longer exist in the pattern database. In other words, more randomness is introduced by generating new patterns that have similar features as the patterns in the training image. To this end, a new point is randomly sampled in the feature space. Then, by back-mapping it to the original MDS space a point in the non-structured dataset will be obtained. This new point should correspond to a pattern. Hence, by using Radon and inverse Radon transformations, a new pattern is generated such that it will be mapped to the exact same point in the MDS space. The generated pattern is then pasted on the grid, bringing less resemblance between the simulated realizations with the input training image. A sample illustration of the pattern generation algorithm, by using three closest patterns, demonstrated the effectiveness of the proposed methodology.

The methodology is in its initial phase and needs more improvements. Some of the future works can be briefly stated as follow:

- Distance-based methods should also be applied in the context of continuous variables, additional to the categorical case presented here.
- In the pattern generation algorithm, soft/hard data conditioning should be taken into account.
- Finally, a comparison in a simulation framework should be made with the previous methods in order to see the accuracy and the amount of randomness introduced by the methodology.

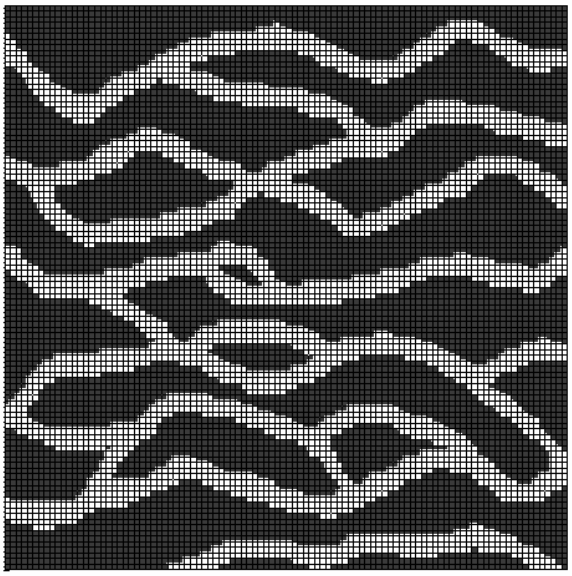


Figure 23: ***Training Image used in this study having*** 100×100 ***dimensions with*** 8×8 ***template***

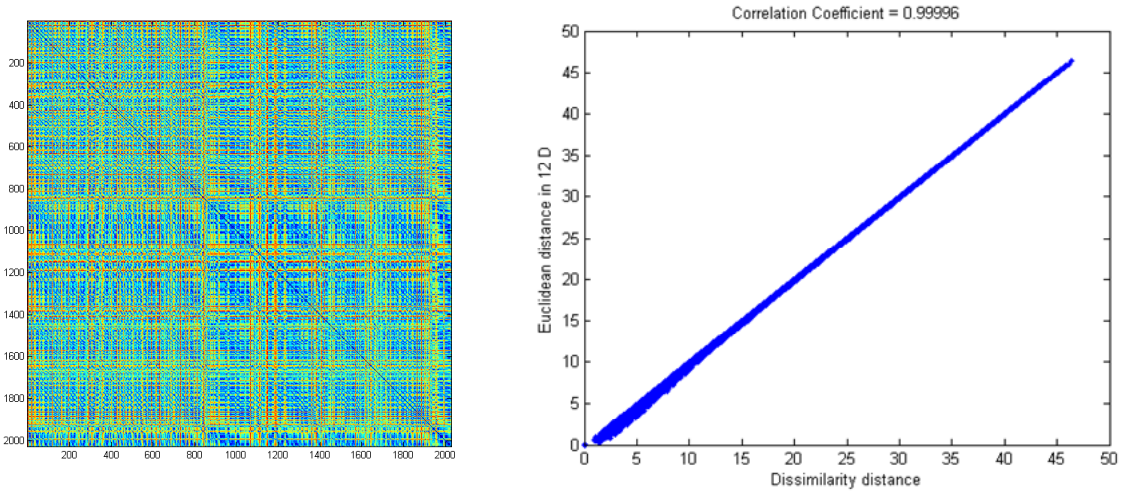


Figure 24: ***Dissimilarity Matrix (left figure) and the correlation between dissimilarity distance and MDS distances being 0.99996 (right figure)***

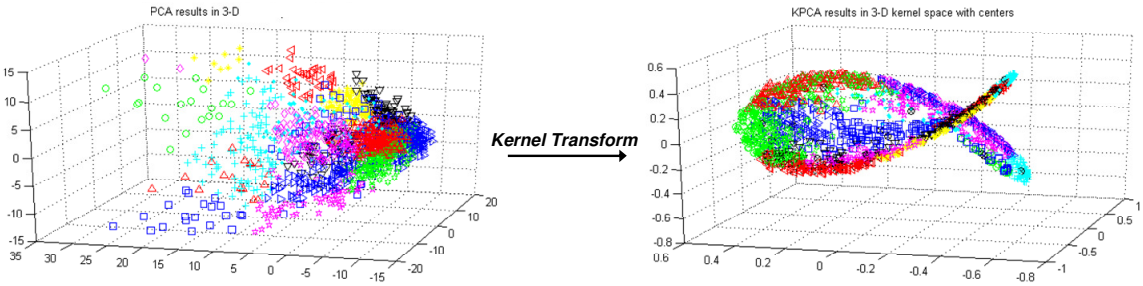


Figure 25: ***patterns represented by points in the Euclidean space (MDS space) with their final classified results shown in different colors (left figure), and feature space representation of the same points having more linear behavior in a higher-dimensional space, but shown in 3 dimensions (right figure)***

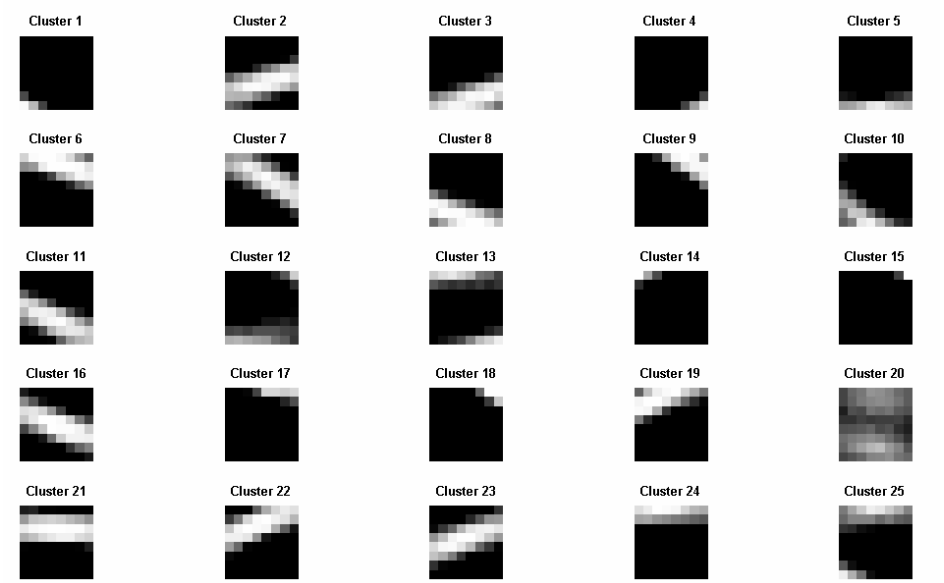


Figure 26: ***Cluster prototypes for all 25 clusters, more sharpness means better clustering***

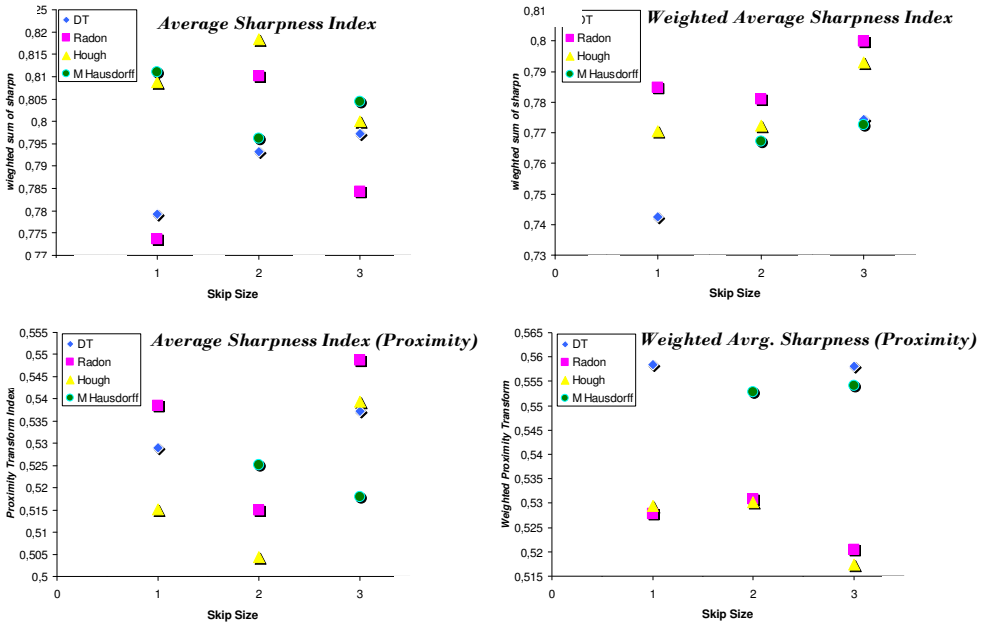


Figure 27: ***different sharpness indexes results for different distance methods: distance transform, Radon transform, Hough transform and Modified Hausdorff; and for different step-sizes (presented on x-axis).***

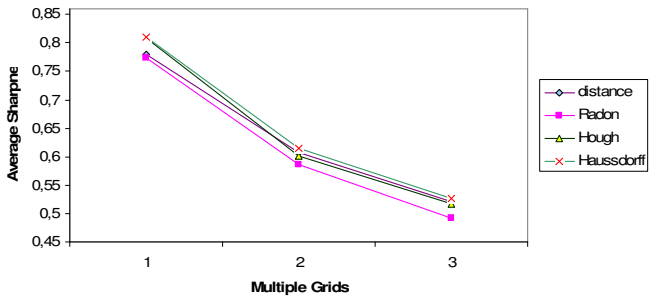


Figure 28: ***Effect of multiple-grid on classification results, showing similar behavior for all methods***

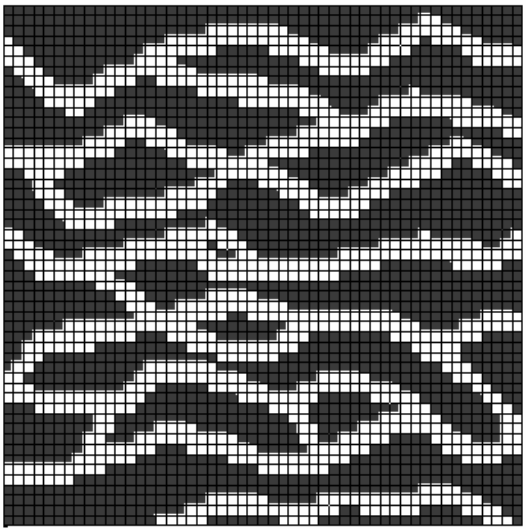


Figure 29: 51×51 ***Training Image used for FilterSim comparison with*** 9×9 ***template***

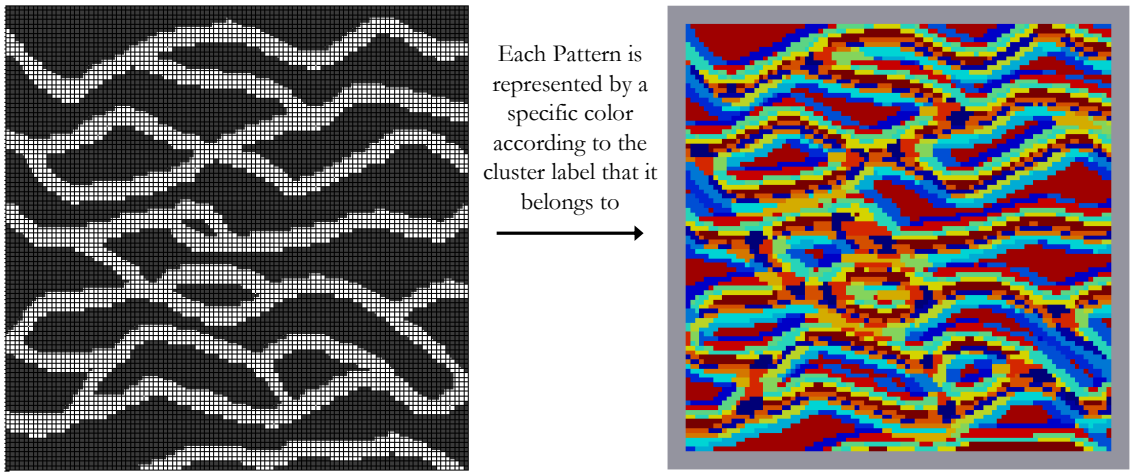


Figure 30: ***Original*** 100×100 ***training image, and its corresponding clustering representation***

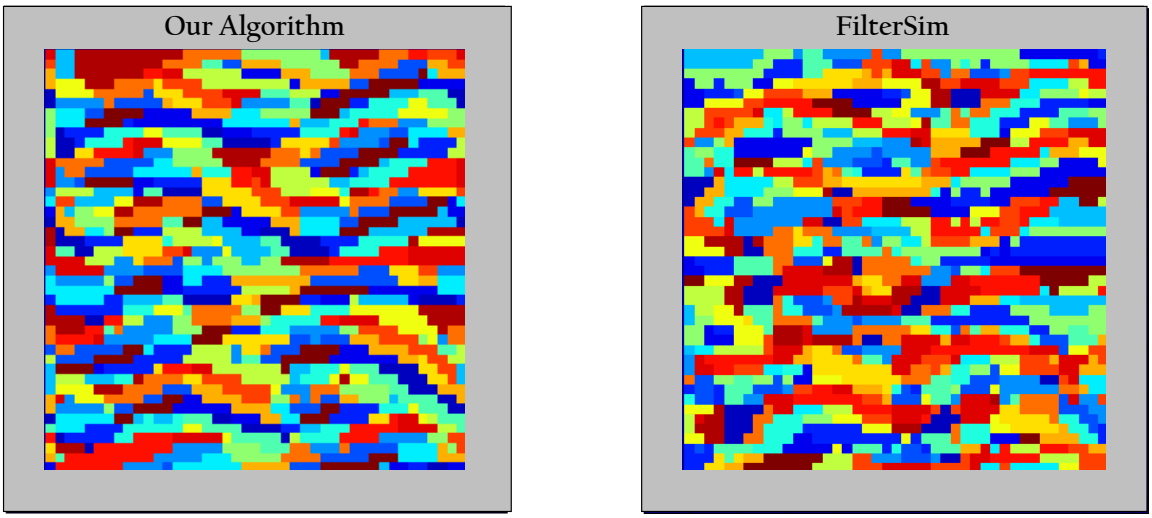


Figure 31: ***Cluster image of patterns in our case (left figure) and FilterSim (right figure)***

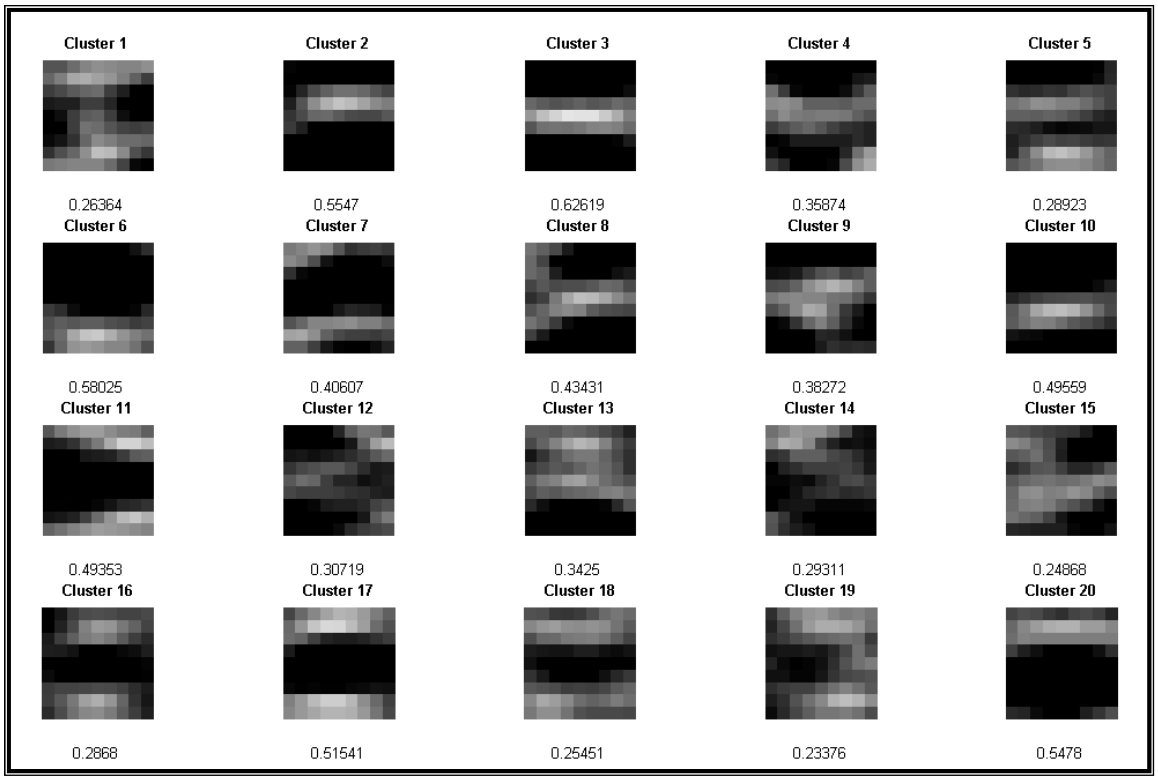


Figure 32: ***FilterSim classification results; each sub-figure shows the prototype of each cluster***

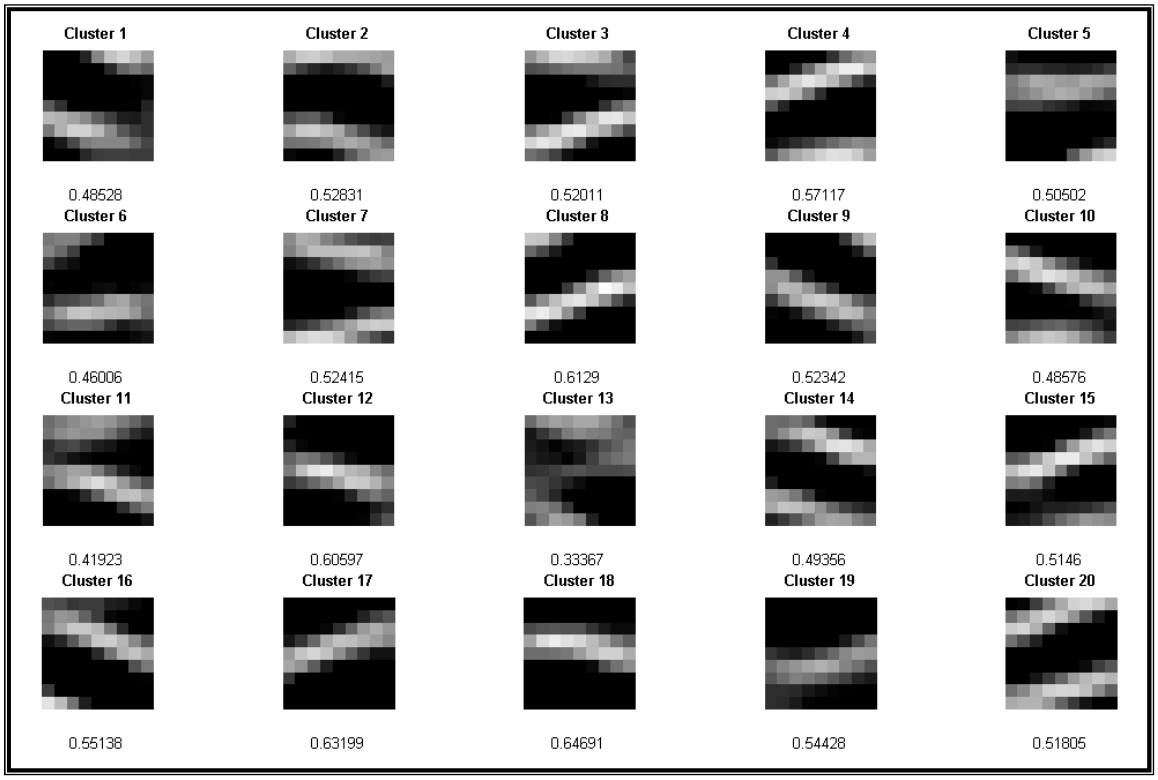


Figure 33: ***Better classification results of our algorithm (improved sharpness of the results in comparison to FilterSim shown above); each sub-figure shows the prototype of each cluster***

References

G. Borgefors. Distance transformations in digital images. Computer Vision, Graphics and Image Processing, 34(3):344–371, June 1986.

G. Borgefores, “An improved version of the chamfer matching algorithm,” in 7th Int. Conf, Pattern Recognition , Montreal, P.Q.,Canada, pp. 1175-1177, 1984.

Calinski, R.B., & Harabasz, J. (1974). A dendrite method for cluster analysis. Communications in Statistics, 3, 1-27.

David J. C. MacKay. Information Theory, Inference, and Learning Algorithms. Cambridge University Press, 2003.

Davis, M.W., 1987. Production of conditional simulations via the LU triangular decomposition of the covariance matrix. Mathematical Geology 19, 91-98.

Deans, Stanley R. (1983). The Radon Transform and Some of Its Applications. New York: John Wiley & Sons.

Gomez-Hernandez J.J., Journel A.G.: Joint sequential simulation of multiGaussian fields. (1993) Geostatistics Troia '92. Vol. 1,, pp. 85-94.

D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance", IEEE Trans. PAMI, vol. 15, pp. 850-863, 1993.

Iman, R.L., Conover, W.J., 1982. A distribution-free approach to inducing rank correlation among input variables. Communications in Statistics B11, 311-334.

I. O. Kyrgyzov, O. O. Kyrgyzov, H. Maître and M. Campedel. Kernel MDL to Determine the Number of Clusters, MLDM, pp. 203-217, 2007.

G. McLachlan and D. Peel, Finite Mixture Models. John Wiley & Sons, 2000.

H. Maitre M. Campedel, E. Moulines and M. Datcu. Feature selection for satellite image indexing. In ESA-EUSC: Image Information Mining, 2005.

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.

Pebesma, E.J., Heuvelink, G.B.M., 1999. Latin hypercube sampling of Gaussian random fields. Technometrics 41, 303-312.

J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, 2004.

Rissanen, K.: A universal Prior for integers and Estimation by Minimum decription Length. Ann, Stat. 11 (1983) 416-431.

Bernhard Scholkopf and Alexander J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA, 2001.

Ross, Sheldon M. 1990. A Course in Simulation. New York: Macmillan.

J.C. Russ, "Image Processing Handbook", 2nd edition, CRC Press, Boca Raton, Florida (1995).

Suzuki, S., Caers, J. [2006] History matching with an uncertain geological scenario. SPE Annual Technical Conference and Exhibition, SPE 102154.

Appendix A

Hausdorff distance

Having the definitions outlined in the text, the distance between two patterns $\mathbf{Pat}_T^m(\mathbf{h}_\alpha)$ and $\mathbf{Pat}_T^n(\mathbf{h}_\beta)$ is defined as the Euclidean distance:

$$\mathbf{d}\left\langle \mathbf{Pat}_T^m(\mathbf{h}_\alpha), \mathbf{Pat}_T^n(\mathbf{h}_\beta) \right\rangle = \left\| \mathbf{h}_\alpha - \mathbf{h}_\beta \right\|$$

The distance between a point $\mathbf{Pat}_T^m(\mathbf{h}_\alpha)$ and a set of point $\mathbf{Pat}_T^n(\mathbf{u})$ is commonly defined as:

$$\mathbf{d}\left\langle \mathbf{Pat}_T^m(\mathbf{h}_\alpha), \mathbf{Pat}_T^n(\mathbf{u}) \right\rangle = \min_{\mathbf{Pat}_T^n(\mathbf{h}_\beta) \in \mathbf{Pat}_T^n(\mathbf{u})} \left\| \mathbf{h}_\alpha - \mathbf{h}_\beta \right\|$$

Hence, having these notations, Hausdorff distance between two point sets (patterns) $\mathbf{Pat}_T^m(\mathbf{u})$ and $\mathbf{Pat}_T^n(\mathbf{u})$ is defined (according to Huttenlocher, 1993) as:

$$\mathbf{d}_H\left\langle \mathbf{Pat}_T^m(\mathbf{u}), \mathbf{Pat}_T^n(\mathbf{u}) \right\rangle = \max\left(\max_{\text{all } h_\alpha} \mathbf{d}\left\langle \mathbf{Pat}_T^m(\mathbf{h}_\alpha), \mathbf{Pat}_T^n(\mathbf{u}) \right\rangle, \max_{\text{all } h_\beta} \mathbf{d}\left\langle \mathbf{Pat}_T^m(\mathbf{u}), \mathbf{Pat}_T^n(\mathbf{h}_\alpha) \right\rangle \right)$$

As by this definition, the Hausdorff distance can result in limited number of results; therefore their power in pattern similarity measurement will be diminished. Hence, Hausdorff distance is modified using the formula below:

$$\mathbf{d}_{MHD}\left\langle \mathbf{Pat}_T^m(\mathbf{u}), \mathbf{Pat}_T^n(\mathbf{u}) \right\rangle = \max\left(\frac{1}{\mathbf{n}_{T_m}} \sum_{\text{all } h_\alpha} \mathbf{d}\left\langle \mathbf{Pat}_T^m(\mathbf{h}_\alpha), \mathbf{Pat}_T^n(\mathbf{u}) \right\rangle, \frac{1}{\mathbf{n}_{T_n}} \sum_{\text{all } h\beta} \mathbf{d}\left\langle \mathbf{Pat}_T^m(\mathbf{u}), \mathbf{Pat}_T^n(\mathbf{h}_\beta) \right\rangle \right)$$

Distance Transform

Let the pixels of a pattern $\mathbf{Pat}_T^k(\mathbf{u})$ be subdivided into two categories of object pixel ($\mathbf{Pat}_T^k(\mathbf{h}_{Ob}) = \mathbf{1}$, shale/black) and background pixel ($\mathbf{Pat}_T^k(\mathbf{h}_{Bg}) = \mathbf{0}$, sand/white).

$$\mathbf{Pat}_T^k(\mathbf{u}) \in \{\mathbf{Ob}, \mathbf{Bg}\}$$

The distance transform of this pattern, $\mathbf{E}_{\mathrm{DT}}\left(\mathbf{Pat}_{\mathrm{T}}^{\mathrm{k}}(\mathbf{u})\right)$ labels each background pixel of this binary pattern with the distance between the pixel and the nearest object pixel. Mathematically it is written like this:

$$\mathbf{E}_{\mathrm{DT}}\left(\mathbf{Pat}_{\mathrm{T}}^{\mathrm{k}}(\mathbf{u})\right)=\begin{cases}\mathbf{0} & \mathbf{Pat}_{\mathrm{T}}^{\mathrm{k}}\left(\mathbf{h}_{\beta}\right)\in\mathbf{Ob}\\ \min\limits_{\forall\alpha}\left(\left\|\mathbf{h}_{\alpha}-\mathbf{h}_{\beta}\right\|,\ \forall\mathbf{Pat}_{\mathrm{T}}^{\mathrm{k}}\left(\mathbf{h}_{\beta}\right)\in\mathbf{Ob}\right) & \mathbf{Pat}_{\mathrm{T}}^{\mathrm{k}}\left(\mathbf{h}_{\alpha}\right)\in\mathbf{Bg}\end{cases}$$

where $\|\bullet\|$ is the Euclidean distance metric. Hence, after transformation, each white node ($\mathbf{Pat}_{\mathrm{T}}^{\mathrm{k}}\left(\mathbf{h}_{\mathrm{Ob}}\right)=\mathbf{0}$) of the pattern is mapped into a continuous value denoting its distance to the target object (Arpat, 2004).

Appendix B

The idea behind this formulation is the assumption that the pattern data point $\mathbf{x}$ belonging to the cluster l follows the multivariate mixture normal distribution, known as Gaussian mixture models (GMM). First, before formulating MDL, mixture models are going to be introduced.
According to previous notations, patterns in Euclidean space are represented by a $\mathrm{N}\times\mathrm{d}$ matrix **X**, where each row represents a pattern. This set of rows, $\mathbf{x}_{\mathrm{i}}$, are modeled by a finite mixture model consisting of two parts:

1. The prior probability $\mathbf{P}\left(\mathbf{x}_{\mathrm{i}}\in 1\middle|\theta_{\mathrm{l}}\right)=\alpha_{\mathrm{l}}$ that every pattern $\mathbf{x}_{\mathrm{i}}$ is a member of only one mixture component (cluster) l, $l=\{1,\ldots,k\}$, where $\alpha_{\mathrm{l}}=\mathrm{n}_{\mathrm{l}}/\mathrm{N}$ (n_{l} denoting the number of patterns belonging to the cluster l);
2. The conditional probability modeling each cluster l by the parametrized probability density function (pdf) $\mathbf{P}_{\mathrm{l}}\left(\mathbf{x}_{\mathrm{i}}\middle|\theta_{\mathrm{l}}\right)$, where θ_{l} denotes the parameter set.

Now let $\mathbf{P}_{\mathrm{l}}\left(\mathbf{x}_{\mathrm{i}}\middle|\theta_{\mathrm{l}}\right)$ denote the class-probability of observing pattern $\mathbf{x}_{\mathrm{i}}$ conditional to $\mathbf{x}_{\mathrm{i}}$ belonging to the cluster l. The finite mixture model expresses the probability of observing the pattern $\mathbf{x}_{\mathrm{i}}$ as a sum of pdf:

$$\mathbf{P}\left(\mathbf{x}_{\mathrm{i}}\middle|\theta\right)=\sum_{\mathrm{l}=1}^{\mathrm{k}}\alpha_{\mathrm{l}}\mathbf{P}_{\mathrm{l}}\left(\mathbf{x}_{\mathrm{i}}\middle|\theta_{\mathrm{l}}\right)$$

For the Gaussian mixture model, we have to use the multivariate Gaussian distribution defined as follow:

$$\mathbf{P}_l\left(\mathbf{x}_i\left|\theta_l\right.\right)=\mathcal{N}\left(\mathbf{x}_i\left|\mu_l,\sum_l\right.\right)=\frac{\mathbf{e}^{-\frac{1}{2}\left((\mathbf{x}_i-\mu_l)^{\mathbf{T}}\sum_l^{-1}(\mathbf{x}_i-\mu_l)\right)}}{(2\pi)^{\mathbf{D}/2}\left|\sum_l\right|^{1/2}}$$

Where μ_l and $\sum_l$ are the mean and covariance matrix of the l^{th} cluster, respectively. These estimates are obtained as:

$$\mu_l=\frac{1}{\mathbf{n}_l}\sum_{l=1}^{\mathbf{n}_l}\mathbf{x}_l$$

$$\sum_l=\frac{1}{\mathbf{n}_l}\sum_{l=1}^{\mathbf{n}_l}(\mathbf{x}_l-\mu_l)^{\mathbf{T}}(\mathbf{x}_l-\mu_l)$$

where $\mathbf{x}_l\in\mathbf{l}$.

And with the assumption that patterns are data instances $\mathbf{x}_i$ are independently distributed, the joint probability (probability of observing data $\mathbf{x}$ or likelihood function) is the product of the individual instance probabilities:

$$\mathbf{P}(\mathbf{x}|\theta)=\prod_{\mathbf{i}=1}^{\mathbf{N}}\sum_{\mathbf{l}=1}^{\mathbf{k}}\alpha_l\mathbf{P}_l\left(\mathbf{x}_i\left|\theta_l\right.\right)$$

The purpose of pattern classification is to simplify their representation in the feature space by replacing each pattern by a generic class which is likely to express all the properties of the patterns. However, when substituting a pattern by its model, an error is introduced. The more complex the model, the less the error. The model complexity is well expressed by the number of parameters needed to build the model. In the mixture of Gaussians case where every cluster is given by its mean and its covariance matrix, the more clusters are used, the more complex the model is, and the less error between data and model. A method to choose the optimal number of clusters consists in selecting the number that provides the shortest description when representing the patterns using models and the errors to the model. This method, named Minimum Description Length (MDL), is defined like this (Rissanen, 1984):

$$\min_{\mathbb{K},\theta}\ -\log\mathbf{p}(\mathcal{X}|\theta)+\frac{1}{2}\mathbb{K}\log(\mathbf{N}),$$

where $\log\mathbf{p}(\mathcal{X}|\theta)$ is the log-likelihood of the mixture model and $\frac{1}{2}\mathbb{K}\log(\mathbf{N})$ is the penalty function with $\mathbb{K}$ parameters. This is the most commonly used selection criterion. By substituting the mixture normal distribution for $\log\mathrm{p}(\mathcal{X}|\Theta)$ and finding $\mathbb{K}$, we will arrive at this equation for MDL:

$$\mathbf{MDL}(\mathbf{k})=-\sum_{\mathbf{l}=1}^{\mathbf{k}}\mathbf{n}_l\log\left(\frac{\mathbf{n}_l^2}{\left|\sum_l\right|}\right)+\mathbf{k}\left(\mathbf{d}^2+3\mathbf{d}+2\right)\log(\mathbf{N})/2$$

This MDL depends on the determinants of $\left|\sum_l\right|$ matrices which describe the model to data error. This is the error function for sample $\mathbf{x}_i$ being represented by the l^{th} cluster (for instance, the distance between $\mathbf{x}_i$ and the mean of the cluster l). Subsequently, for kernel k-means this error can be determined in the original space **X**, as well as in the feature space **F** after kernel transformation. The simplest error function is the Euclidean distance which can be calculated using kernel K. Therefore, the sum-squares distance from patterns to their corresponding l^{th} cluster centroid, as given by (Shawe-Taylor and Cristianini, 2004) is:

$$\mathbf{S}_l = \frac{1}{\mathbf{n}_l \cdot \mathbf{d}} \sum_{k \subseteq l} \left(\mathbf{K}(\mathbf{X}_k, \mathbf{X}_k) - \frac{1}{\mathbf{n}_l} \sum_{m \subseteq l} \mathbf{K}(\mathbf{X}_k, \mathbf{X}_m) \right)$$

If we supposing that the variances of a cluster are equal for each dimension, the determinant of the covariance matrix $\left|\sum_l\right|$ will be,

$$\left|\sum_l\right| = \mathbf{S}_l^{\mathbf{d}}$$

Hence, by substituting this determinant we can get the MDL formulation for kernel k-means algorithm as below:

$$\mathbf{KMDL}(\mathbf{k}) = -\sum_{l=1}^{k} \mathbf{n}_l \log\left(\frac{\mathbf{n}_l^2}{\mathbf{S}_l^{\mathbf{d}}}\right) + \mathbf{k}\left(\mathbf{d}^2 + 3\mathbf{d} + 2\right)\log(\mathbf{N})/2$$