Stochastic Simulation of Patterns Using Distance-Based Pattern Modeling

Mehrdad Honarkhah & Jef Caers

Stanford Center for Reservoir Forecasting
Stanford University

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, such as snesim, 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, such as simpat and filtersim, 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, the sequential simulation will be carried on 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 study an entirely different approach will be taken towards pattern analysis and generation.

In the developed methodology, 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 are more flexible than the standard (fixed and linear) filters in filtersim and can be tailored to the application at hand. Afterwards, by significantly reducing the dimensionality of the problem, an improved pattern classification algorithm is obtained using kernel space mapping. The results are compared with previous methods. An improved pattern continuity and data-conditioning capability is observed in the generated realizations for both continuous and categorical variables. The proposed methodology is much less sensitive to the user-provided parameters, and at the same time reduces computational time significantly.

1 Introduction

Snesim was the first truly practical algorithm proposed for the application of Multiple-Point Geostatistics (MPS). In this method, the conditional probability is sampled from the training image by looking for replicates of the data event. 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, which is inherent to any stochastic simulation tool based on probability theory. Additionally, any type of trend, such as vertical or areal proportion changes need to be explicitly enforced.

Simpat and filtersim as alternative pattern-based algorithms were proposed by [1, 11]. The sequential simulation method of simpat/filtersim 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. In these algorithms, patterns are extracted from the training image and directly pasted into the simulation grid. If that pattern is unique or contains a non-stationary element of the training image, then the uniqueness or non-stationarity of that pattern will be presented in the simulation grid. No morphing or filtering is applied to the patterns, and they are extracted from the training image without modification.

Randomness/variability is introduced in filtersim through randomly sampling a pattern from a class of patterns, however it should be kept in mind that these patterns originate from the same training image which has a limited/finite pattern database. Hence, any additional randomness is due to an insufficient pattern reproduction.

As a proposition, one can argue that the latest MPS algorithms, simpat or filtersim, can be improved in many ways. The starting point for improvement would be an approach which improves on the limitations inherent to the use of spatial filters in filtersim; a unified method that does not depend on any filter specification nor does it have to deal with the sometimes overly deterministic nature of simpat. The recognition capability of the linear filters used in filtersim strongly depends on the patterns themselves. Some patterns may contain a specific features that different filters would be incapable of discriminating between them, and even if the patterns are different, they will be classified in one cluster. One solution is switching into applying more filters on each pattern, and as a consequence, increasing the dimensionality of the problem. While nowadays this approach is feasible with increasingly fast speed of computers, but a more robust technique, which does not depend on filter definitions and the number of filters, and also at the same time, can help to overcome other limitations mentioned before, needs to be developed.

2 Distance-based Method

One of the new and promising tools in reservoir modeling for improvement are the distance-based methods. While these techniques have been used so far to model variability between realizations [26], the same techniques are used here at the pattern level. The idea behind this distance approach is that by applying a metric, called distance function, to any pair of patterns, a measure of their similarity is obtained. In other words, one tries to find 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 the patterns in the database, intuitively speaking, one can gain a tremendous insight into the underlying structure of the patterns. A technique, called multi-dimensional scaling, can reveal these hidden structures by only having the pair-wise distances is needed.

Multidimensional scaling (MDS) encompasses a collection of methods which allow gaining insight in the underlying structure and 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; and therefore placing these patterns as points in a multidimensional space, where the dissimilarity between the patterns is related to the distances of the corresponding points in the multidimensional space. The inter-point distances in MDS space are related to the similarities obtained by a distance function. MDS method can also reveal the inner dimension of the patterns that can meaningfully describe them. This multidimensional representation is evidently useful as a basis for building a mathematical model for categorization, which will be later used for pattern classification. Before formulating the algorithms in the following sections, some required notations will be provided for a binary sand/shale case.

2.1 Training image

Define $ti(\mathbf{u})$ as a value of the training image 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 follow:

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

Also define $ti_{\mathbf{T}}(\mathbf{u})$ as a specific multiple-point vector of $ti(\mathbf{u})$ values within a template $\mathbf{T}$ centered at node $\mathbf{u}$, i.e., $ti_{\mathbf{T}}(\mathbf{u})$ is the vector:

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

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

2.2 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 $ti_{\mathrm{T}}(\mathbf{u})$ vectors in a database. Each such $ti_{\mathrm{T}}(\mathbf{u})$ is called a pattern of the training image and the database is called the pattern database and is denoted by $patdb_{\mathbf{T}}$. 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 $\mathbf{u} \in \mathbb{G}_{ti}$ of a pattern; only the content of the pattern is stored. Hence, k^{th} pattern can be denoted like this:

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

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

2.3 Dissimilarity distance Matrix

Having a pattern database $patdb_{\mathbf{T}}$ 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.

2.4 Dissimilarity distance function

The goal of the first part of this study is to classify the patterns in different groups, called clusters. In other words, the patterns that are similar to each other should be grouped into one cluster. To this end, a measure of similarity between the patterns is needed. This measure, which is a function that calculates the dissimilarity between a pair of patterns, is called the *dissimilarity distance function*. This concept was first introduced by [1, 12]. 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 $\mathbf{pat}_{\mathbf{T}}^{m}(\mathbf{u})$, $\mathbf{pat}_{\mathbf{T}}^{n}(\mathbf{u})$ be two patterns from the pattern database $patdb_{\mathbf{T}}$. The Euclidean distance $d_{\mathrm{E}}(\mathbf{x}, \mathbf{y})$ is given by:

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

As can be seen in formula above, the Euclidean distance is only a summation of the pixel-wise differences between two patterns. However, this distance function may not provide much discriminative power for complex patterns. Another distance function called *Proximity Transform distance function* has been used as a dissimilarity distance function in this study.

3 Multi-Dimensional Scaling

Multi-dimensional scaling (MDS) is aimed to represent high dimensional data in a low dimensional space with preservation of the similarities between data points. This reduction in dimensionality is crucial for analyzing and revealing the genuine structure hidden in the data. Mathematically speaking, multidimensional scaling (MDS), translates a dissimilarity matrix into a configuration of points in n-D Euclidean space $\mathbb{R}^d$ [3]. 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 [8].

An example application of classical MDS is represented in Figure 1. For illustration purposes, a 2D Euclidean space, $\mathbb{R}^2$ was chosen for mapping 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 observed in Figure 1.

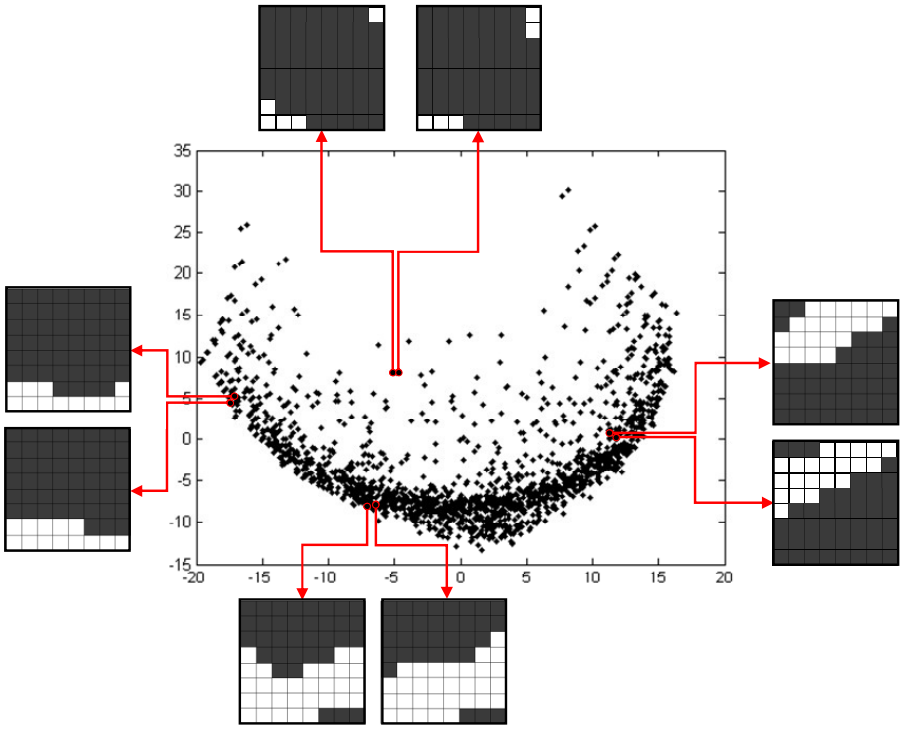


Figure 1: Multidimensional scaling sample result using an 8×8 pattern template. Any two patterns that are similar map as two close points in Euclidean space

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 pattern, 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. However, 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. On the other hand, considering 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 for large databases of patterns, a new metric MDS method with a low computational complexity is developed, and explained hereafter.

3.1 *SEQ*-MDS Algorithm

The first idea for improving the MDS speed is the anchor point MDS method [13], where only a portion of data is used to reconstruct the layout for intermediate steps. The data are grouped into clusters, so that the distances between points in different clusters are less meaningful than the distances between points in the same cluster. In the anchor point method, some points in the same cluster are chosen as anchors and others are considered as floaters. Distance information of anchors is used to construct the coarse structure of layout, and the floaters are used to update the fine structure. However, the intermediate steps for calculating MDS do not need to employ all entries of the dissimilarity matrix. 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). Please refer to [32] for the details and experimental studies of the algorithm.

3.2 Scaling Dimensions

More often than not, the intrinsic dimensionality of the data is much lower; i.e., even though the data are lying in a high-dimensional space, only a few dimensions are actually important for the analysis. 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 mapping can be inevitably seen as a dimensionality reduction technique which can further speed up future mathematical computations. An important issue in MDS mapping is the choice on the number of dimensions for the scaling solution (e.g. 2 dimensions as seen in Figure 1). A configuration with a high number of dimensions achieves very small errors, but it can later on result in computationally expensive mathematical analysis. On the other hand, a solution with too few dimensions might not reveal enough of the structures in the data. One needs to find an automatic procedure that can compromise between these two extremes and decide on the number of coordinates to be retained.

The previous methodology used to select the dimensionality through the correlation between the inter-point distances in Euclidean space and their corresponding dissimilarity distance values. The correlation coefficient was plotted against the dimensions. 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. One of the shortcoming in this method was the computational burden associated with it. For each dimension, a distance matrix corresponding to the MDS space points needed to be constructed and be compared with the original distance matrix. The more dimensions one retains, the more these two distance matrices would be correlated. This procedure needed to be done for each dimension. Calculating the inner-point distances for all dimensions has a computational complexity of $O(n_T N^3)$. Therefore, another approximate technique needs to be developed.

One proposed solution is to use Maximum Likelihood Estimation (MLE) on the dimensionality of the MDS space. The method was first introduced by [15]. A modified version was adapted to our analysis. Generally speaking, an appropriate choice for the dimension is

made by considering the magnitude of the singular values, which provides a measure of how much variation is being captured in each dimension. The analysis has been simplified by using principal component analysis (PCA) on the patterns first. This can be done extremely fast. Next, according to the elbow of the scree plot, the dimensionality is found by using the maximum of a log-likelihood function. The assumption behind this technique is that the singular values, 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 in order to point out the differences with the original method proposed by [15].

let $d_1 \geq d_2 \geq \cdots \geq d_{n_T} > 0$ be the ordered eigenvalues obtained using PCA. For every fixed q, we define two samples of $\varphi_1 = \{d_1, d_2, \ldots, d_q\}$ and $\varphi_2 = \{d_{q+1}, d_{q+2}, \ldots, d_{n_T}\}$. Observing a gap in the scree plot corresponds to the assumption that these are two populations each with their own distribution function. Letting the log-likelihood function to be a gaussian distribution, one would obtain:

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

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.

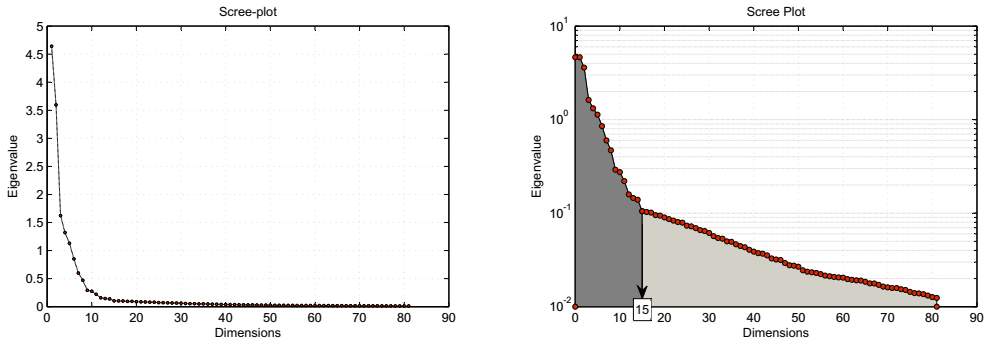
$$\hat{q} = \underset{k=2,3,\ldots,n_T-1}{\arg\max} \; l_q(k)$$

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

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

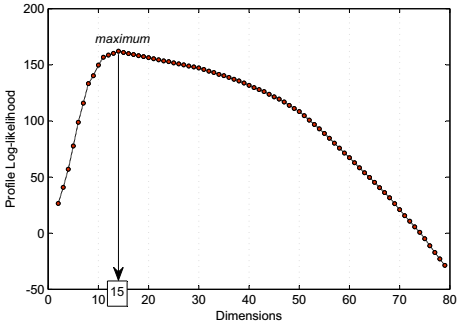
where, μ_1, μ_2 are the means and σ_1, σ_2 are the variances of φ_1, φ_2 respectively.

In order to test the dimensionality selection, the same training image with the same pattern template as before was chosen. 2209 patterns were used for analysis. The scree plot, obtained from the eigenvalues of the covariance matrix is shown in Figure 2(a). In order to find the elbow of this scree plot, the eigenvalues are shown in logarithmic scale in Figure 2(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(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 method merely formalizes and automates what can be easily accomplished by human eye and provides the best dimensions for MDS space.



(a) Scree-plot

(b) Semilog Scree-plot



(c) Profile Likelihood

Figure 2: MDS dimensionality selection. (a) The scree plot, (b) the elbow of the scree plot is easily observed in the semilog scale, (c) the profile log-likelihood of the scree plot showing 15 as the maximum.

4 Pattern Classification

One of the goals of this novel distance-based approach is to come up with a technique that can help us remove the spatial filters used in filtersim and also to have an avenue for new pattern reproduction concepts. Instead of some pre-defined filters, one requires a universal classifier. 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. Considering that, the patterns can now be easily classified. To this end, a clustering method to organize the points into different clusters needs to be employed.

Clustering techniques 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 3. The input patterns, represented by points are shown in Figure 3(a), and the desired clusters are shown in Figure 3(b). As can be observed, points belonging to the same cluster are given the same label. This approach eliminates the need for any filters and probably results in the best classification within any database of patterns.

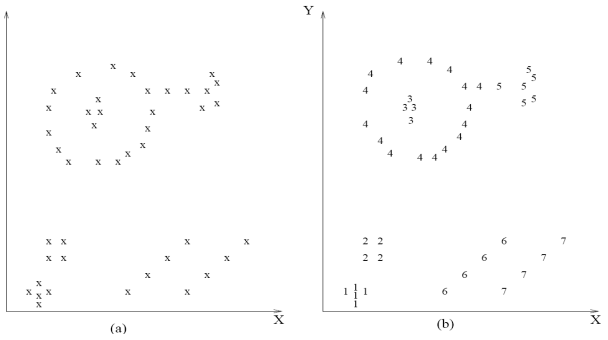


Figure 3: Clustering illustration

Before introducing the clustering methods, some further notations will be provided for clarity.

4.1 Notation

Suppose that all patterns have been mapped into a d-dimensional MDS space, $\mathbb{R}^d$, as points. Hence, we can now show each pattern by a location (points) $\mathbf{x}_k$ in a d-dimensional row vector as follow:

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

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

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\mathrm{N}1} & x_{\mathrm{N}2} & \cdots & x_{\mathrm{N}d} \end{bmatrix}$$

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 $patdb_{\mathbf{T}}$.

4.2 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 $\mathcal{L} = \{l_1, \ldots, l_{\mathrm{N}}\}$m where $l_i \in \{1, \ldots, k\}$ and 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:

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

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:

$$\mathbf{v}_i = \frac{1}{\mathrm{N}_R} \sum_{p=1}^{\mathrm{N}_R} \mathbf{x}_p \qquad , \mathbf{x}_p \in l_i$$

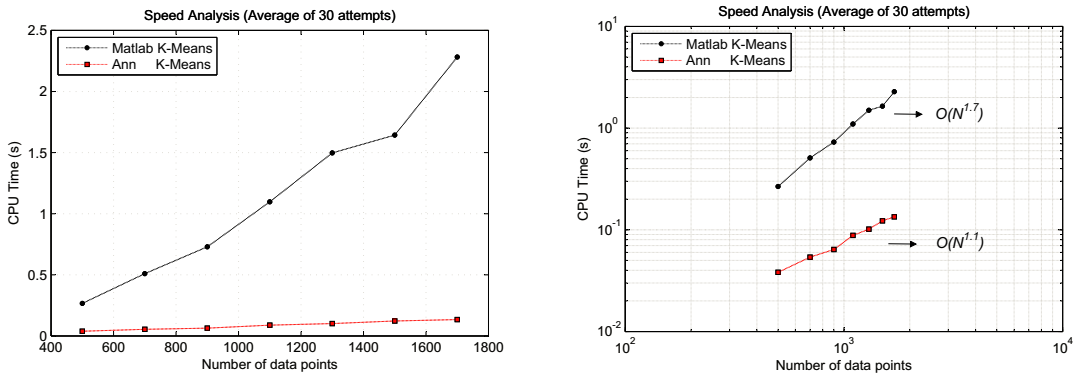
where N_i is the number of patterns in $\mathcal{A}_i$. It should be noted that the reason behind choosing a Euclidean norm in the minimization phase of the algorithm is simply because of the specific metric distance used in multidimensional scaling. In mapping the dissimilarity distances between patterns to MDS space, 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.

4.2.1 H-Ann K-means

The huge amount of data collected and stored in databases increases the need for effective analysis methods to use the information contained implicitly there. The k-means method has been shown to be effective in producing good clustering results for many practical applications. However, the k-means algorithm requires time proportional to the product of number of patterns and number of clusters per iteration. This computationally may be expensive especially for large databases;i.e. in a large 3D training image. 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 by keeping track of centroids of the subsets, and issuing a large number of nearest-neighbor queries. A kd-tree is a data structure for storing a finite set of points from a finite-dimensional space [16]. Kd-trees can be used to reduce the number of nearest-neighbor queries in k-means by using the fact that their nodes can represent a large number of points. Originally, kd-trees were used to accelerate nearest-neighbor queries. We could, therefore, use them in the k-means inner loop transparently. For this method to work, the centers need to be stored in the kd-tree. This will bring huge improvements over the standard k-mean algorithm. There has been many attempts on using kd-trees in kmeans clustering [17, 18]. The details of these algorithms will not be explained here. However, a speed comparison is made with Matlab k-means algorithm. The results are shown in Figure 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 in this context of MPS is favorable 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 will bring speed improvements only in the case of low-dimensional data. By using kd-trees, one tries to finds approximate nearest neighbors in each iteration (ANN K-means). In order to further speed up this important part of the MPS simulation, we have proposed a hierarchical ANN k-means (H-Ann K-means). It has a very simple concept. Since the computational complexity of the k-means algorithm depends on a good initial centroids, and also, on the number of data points, the hierarchical k-means will try to find



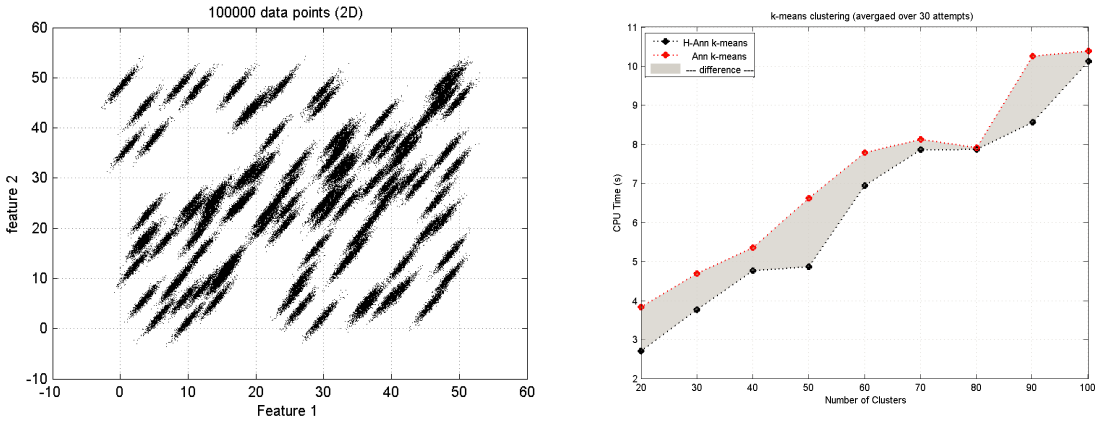
(a) Speed Comparison

(b) Log-Log scale

Figure 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.

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 become more evident on larger datasets.

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 lower hierarchical k-means implementation. Since this can also take some CPU time, a random sampling of points is assumed to be acceptable. 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 on the dataset is chosen for the first Ann k-means. The final centroids, obtained from the reduced dataset, are given as the initial centroids for the whole dataset. In order to compare two methods, a dataset of 100000 points are generated in $\mathbb{R}^2$ with different observable cluster distributions as shown in Figure 5(a). The comparison of Ann k-means with H-Ann k-means is shown in Figure 5(b). It is noticeable that H-Ann k-means is more efficient with some number of clusters. This strongly depends on the configuration of the points and also on the true number of clusters.



(a) Data points

(b) Speed comparison

Figure 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.

5 Kernel Methods

Considering the non-linear variation of locations of patterns when they are represented in a cartesian space with MDS and the inability of linear methodologies presented before to handle them, kernel methods should be employed. In this section, the definition of kernel transformation and feature space is introduced first. Some clustering algorithms in feature space are also explained. Next, the computational complexity related to clustering methods is analyzed and some improvements are provided. Finally, a methodology for automatically selecting the number of clusters is explained with some examples.

5.1 Definition

Considering the amount of available patterns in the pattern database of a training image, simple, fast and efficient clustering algorithm, such as K-means method, offer a solution for classification. However, 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 [5]. 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 [10]. In other words, by mapping the data points to that higher dimensional space F, the nonlinear relationship among the information provided by the data can be captured. A sample representation of this process is shown in Figure 6 [9] 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 hence, a stronger and more accurate pattern classification will result. The formulation to achieve this non-linear transformation

will be introduced hereafter.

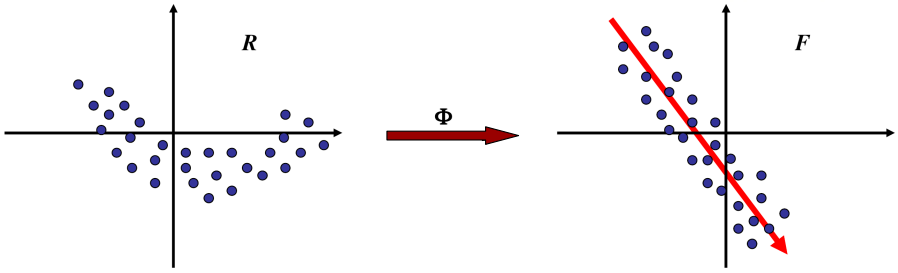


Figure 6: Kernel transformation, and the principal component in the linear kernel space

Generally, by this transformation, the similarity between the data needs to be preserved in the feature space. One particularly simple yet surprisingly useful notion of this similarity - the one that is 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 their dot product in some high-dimensional feature space 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 F using $\boldsymbol{\varphi} : \mathbb{R}^d \rightarrow F, \quad \mathbf{x} \mapsto \Phi(\mathbf{x})$, 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 nonlinear function $k(\cdot)$ in the input space, i.e. by means of a kernel trick.

$$k(\mathbf{x}_k, \mathbf{x}_l) = \langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_l) \rangle$$

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 $\mathbb{R}$(input space) through a kernel function. Hence, $k(\mathbf{x}_k, \mathbf{x}_l)$ 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. 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* [6] as an alternative to K-Means.

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

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

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 7.

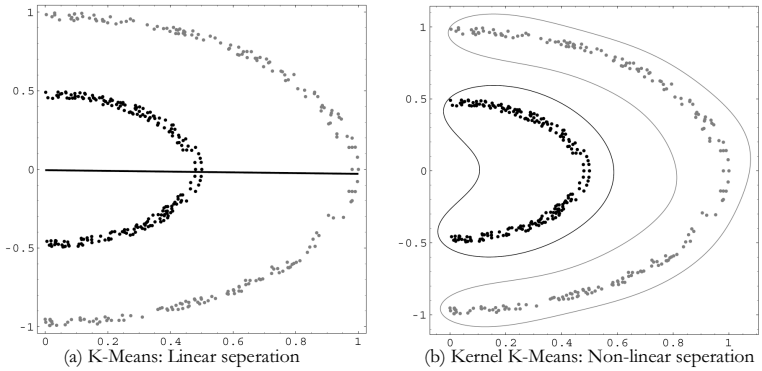


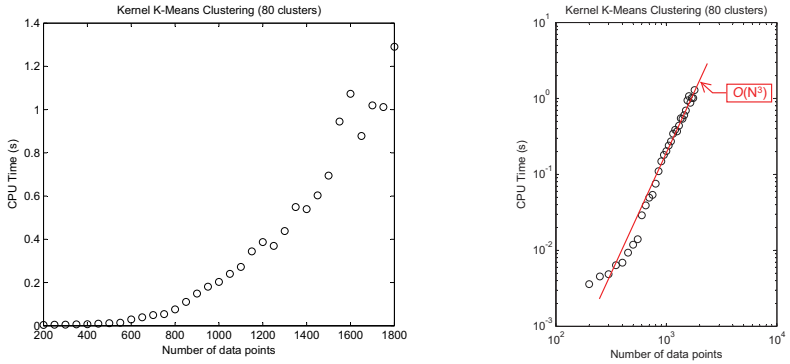
Figure 7: (a) K-Means on dataset. The solid line indicates the linear separation line determined by K-Means. (b) Kernel K-Means on dataset. The region delimited by black line identifies the first cluster which has been properly separated from the second data cluster, identified by the gray line.

5.2 Computational Complexity

Kernel functions 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 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 applications of kernel k-means algorithm are confined to small datasets to its expensive computation and storage cost. 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 for 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 an improved and efficient kernel k-means algorithm is provided in Figure 8. As observed, the kernel k-means algorithm has a computational complexity of $O(N^3)$. This is one of the most prohibitive parts of the entire methodology.

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



(a) Computational Speed (b) Log-Log scale

Figure 8: (a) kernel k-means speed analysis, (b) kernel k-means speed shown in log-log scale, showing the computational complexity of the algorithm, $O(N^3)$.

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 because of a 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 make it susceptible to being trapped in local optimum. This is more pronounced because now the initial centroids depend on the previous linear clustering results. The speed improvement of this method is shown in Figure 9. The larger the database of the patterns or the larger the training image, the larger is the improvement gained using the proposed methodology.

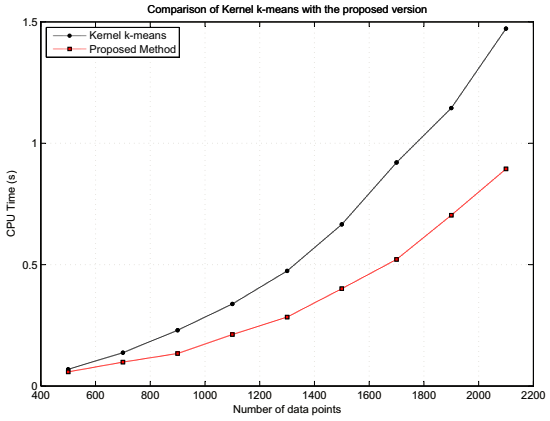


Figure 9: Comparison of kernel k-means with the proposed methodology of using k-means as an initializer for kernel k-means centroids.

5.3 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 kernel k-means method, a procedure that can account for the feature space transformation should be applied (since we do not know the points themselves but their dot products). In the previous paper [19], the optimal number of clusters was selected on the basis of 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 was computationally expensive and every number of cluster needed to be tested explicitly. A new automatic method is implemented which is much faster than the previous technique and would provide a straightforward routine for computing the optimal number of clusters. The simplicity is obtained by taking into account of the fact that kernel analysis of the patterns has provided a database of all the pairwise distances between the points. This, in fact, can be used in order to find the optimal number of clusters. By analyzing the inherent structures within the distance matrix, which is already available from kernel analysis, one is able to find the best number of clusters. The details of the algorithm with an experimental proof is provided in [32].

6 Illustration of Previous Concepts

The concepts that were introduced in the previous sections, such as multi-dimensional scaling and mapping into kernel space for clustering, will be analyzed in this section. First, the training image used for the illustrations is introduced. This training image will be used in most of the upcoming MPS experiments. Next, each step of the algorithm is explained with examples, and the classification capability of the proposed method is compared with filtersim method.

6.1 Training Image

The training image that was used to illustrate the previous concepts has binary sand/shale channel structures. Figure 10 shows this training image. The workflow of the pattern classification part of this study is outlined in Algorithm 1.

Here, a 100×100 training image is used, with a template size of 8×8. 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 kernel space are shown in Figure 11. The classification is then applied in kernel space and a set of 25 clusters were obtained. The cluster prototypes are shown in Figure 12. 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 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 64-dimensional pattern, a correlation coefficient of 0.99996 is obtained.

The benefit of MDS space to feature space transformation is illustrated in Figure 11. 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

Algorithm 1 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 $\overset{\text{MDS}}{\mapsto}$ points in $\mathbb{R}^d$
10: **else**
11: Map the (patterns in that set + overlapping points with previous set) $\overset{\text{MDS}}{\mapsto}$ 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

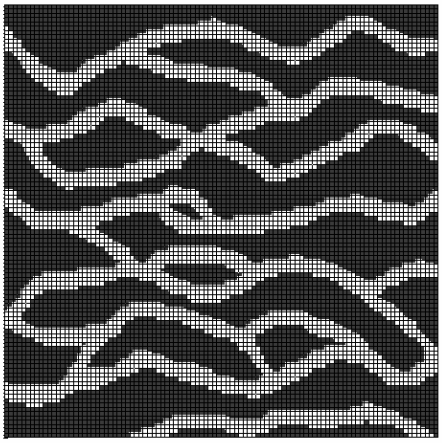


Figure 10: Training Image used in this study having 100×100 dimensions with 8×8 template

principal components in that space.

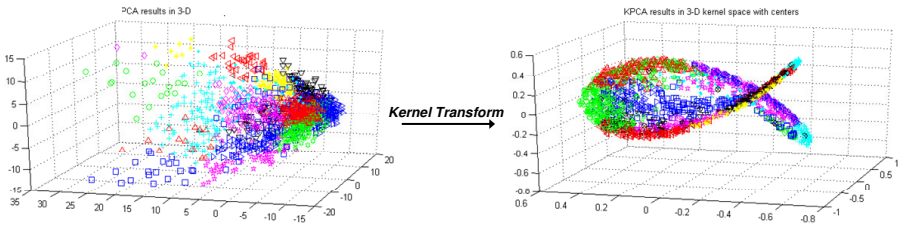


Figure 11: patterns represented by points in the Euclidean space (MDS space) with their final classified results shown in different colors (left figure), and feature space representation of the same points having more linear behavior in a higher-dimensional space, but shown in 3 dimensions (right figure)

Finally, the prototypes of the resulting clusters are shown in Figure 12. 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 12 is considered sharp, while cluster 20 is not. According to these qualitative visual concepts, a quantitative sharpness index which can help 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 zero and for the pixels close to 1 or zero, 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 is as follow:

$$\text{Sharpness Index} = SI = \frac{1}{k}\sum_{i=1}^{k} |2 \times \mathbf{v}_i - 1| \ \in [0,1]$$

where $\mathbf{v}_i = \frac{1}{n_{l_i}} \sum_{k \in l_i} \mathbf{pat}_{\mathbf{T}}^{k}(\cdot)$ and k = number of clusters. 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 $SI \approx 0.5$. 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 accurately result in a small sharpness index, indicating poor classification. The following formula is therefore proposed:

$$\text{Weighted Sharpness Index} = WSI = \frac{1}{k}\sum_{i=1}^{k} \left(\frac{n_{l_i}}{n_{pat_{\mathbf{T}}}}\right) |2 \times \mathbf{v}_i - 1|$$

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 [1]. 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.

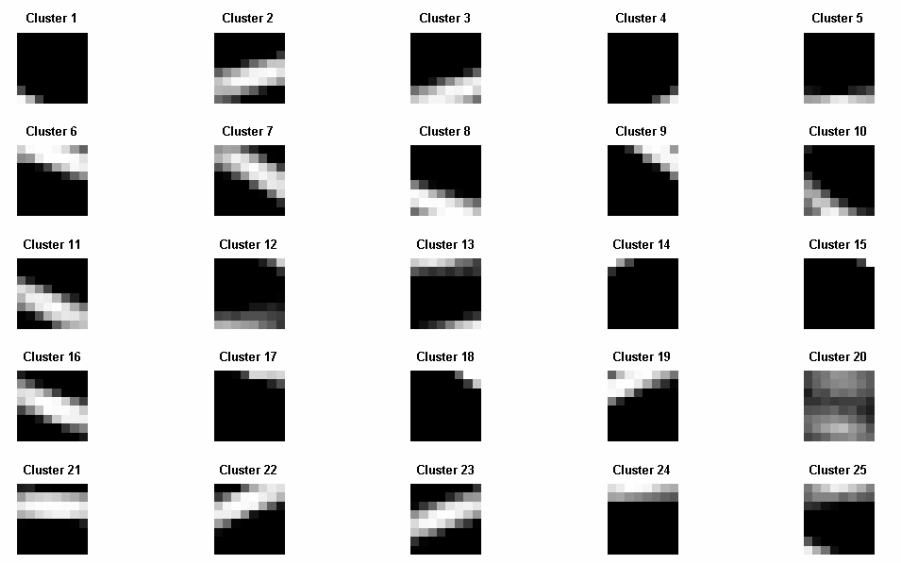


Figure 12: Cluster prototypes for all 25 clusters, more sharpness means better clustering

6.2 Comparison with Filter-Based Classification

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 the proposed 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 13. 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. In this case study, 20 clusters is chosen for the final results. In Filtersim, for the sake of comparison, the k-means algorithm is also used for the classification method.

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 14 and Figure 15 show the prototypes of each cluster obtained by FilterSim and the proposed method. The sharpness index of each prototype is indicated below them. Visual perception indicates that a better classification is obtained within

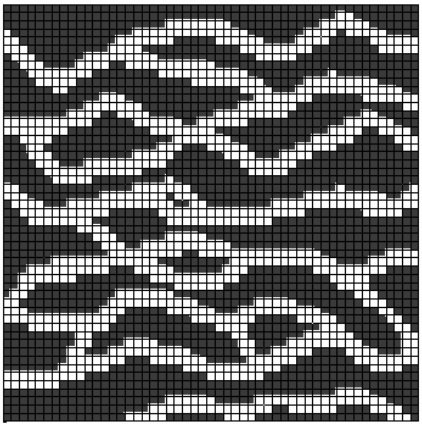


Figure 13: 51×51 Training Image used for FilterSim comparison with 9×9 template

this 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 this method in comparison with the method employed in FilterSim.

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

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

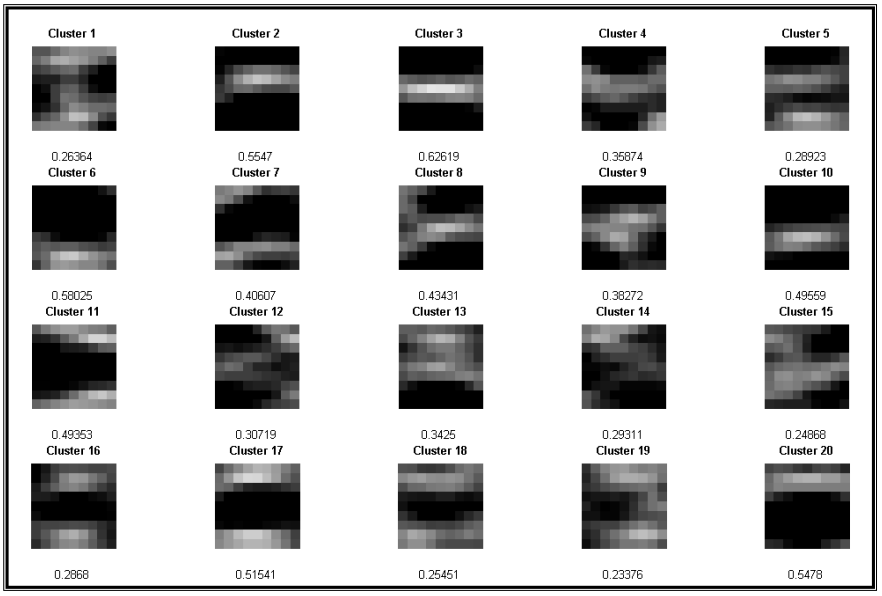


Figure 14: FilterSim classification results; each sub-figure shows the prototype of each cluster

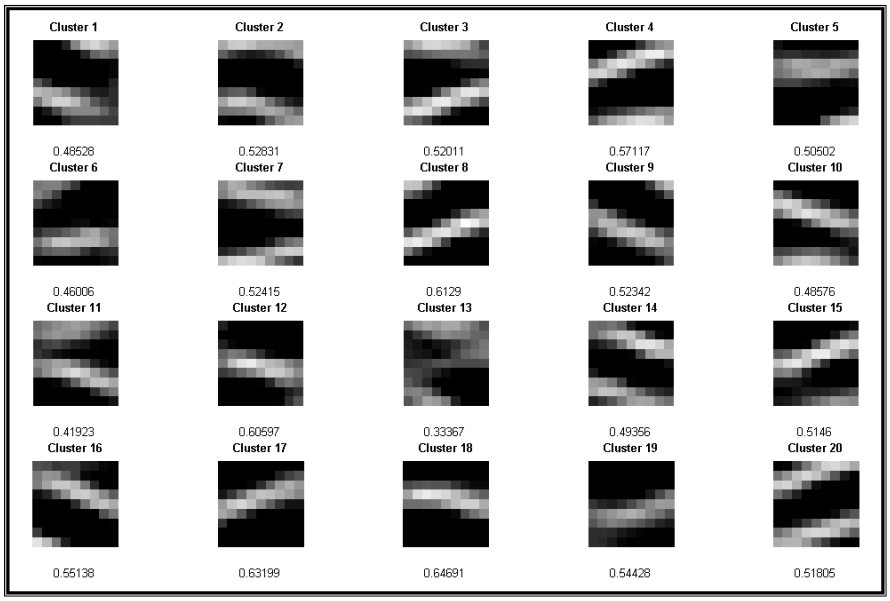


Figure 15: Better classification results of our algorithm (improved sharpness of the results in comparison to FilterSim shown above).

7 Simulation

Having introduced various new techniques for pattern analysis and classification, we now compare these techniques with filtersim by means of distances in a simulation framework. Filtersim method is chosen for comparison because of the similarities of the proposed approach to filtersim. Simpat, on the other hand, does not perform any classification on the patterns. During the course of simulation, simpat finds the most similar pattern to the data event by ana exhaustive search in the pattern database. The classification approach, used in the pattern-based MPS method of filtersim, provides tremendous computational improvements over simpat. A program has been coded to follow the same procedures as in filtersim. However, in here, patterns will be mapped to MDS space, and will then be classified in kernel space. The rest of the methodology is almost similar to filtersim, in which, the data event is compared with cluster prototypes and the most similar one will be selected to obtain the pattern that is going to be pasted on the simulation grid. This procedure continues until all the nodes on the simulation grid have been visited. First, in this section, a new automatic methodology for selecting the template size will be provided. Next, the the MDS-related procedure for filling up the unknown nodes of an incomplete pattern during the course of the simulation will be outlines. And finally, some comparisons between the proposed algorithm and filtersim will be made using different training images and geological scenarios.

7.1 Template Size Selection

In order to generate realizations in any geostatistical simulation, one needs to know some statistical measures related to the reservoir model under study. For example, in sequential Gaussian simulation, a mean and a variogram alone are sufficient to generate realizations. Variogram, which is a two-point statistical measure, provides a range such that two sample points with that distance apart would be uncorrelated. Similarly, in multiple-point geostatistics, one acquires the statistics from a conceptual training image. However in two-point statistics, the variogram range is implicitly taken into account by the formulation. There is no need for the range to be provided explicitly. But in multiple-point facies modeling, the relevant statistics should be clearly defined by means of a template size. In probabilistic approaches, such as snesim, it corresponds to the search neighborhood for the calculation of probabilities, and in pattern-based approaches, such as simpat and filtersim, it corresponds to the size of the patterns that are used for generating realizations. So far, selection of the true template size, associated with the training image, has been made with trial-and error by analyzing the reproduction of patterns and large-scale structures in the simulated realizations. In the next section, an algorithm that automatically selects the optimal template size will be provided. And afterwards, its application on different training images will be explored.

7.1.1 Algorithm

The template size depends 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 concerning the actual features (i.e. the actual features are insufficient 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 it should be chosen as large as necessary to represent the actual features.

Previously, the application of two-point entropy to analyze the training image and obtain a 2D entropy map of points in an arbitrary template has been applied in [5]. 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 is not provided and a subjective visual analysis of the map is used for this selection. In this paper, the same general concept of entropy will be used in order to find an optimal template size. However, a much simpler single-point entropy is considered instead of the two-point entropy.

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 [23], 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 $n_x n_y n_z$ can be computed as follow:

$$H = \sum_{i=1}^{K} p_i \, \log\left(p_i\right)$$

where K = number of possible outcome 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 exception, 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). Therefore, Shannon's definition of entropy, when applied to patterns, can determine the minimum information required in order to reliably represent the pattern as encoded 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 would 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 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. According to these concepts, the algorithm for finding the optimal template size is straightforward.

Automatic template selection starts by scanning through the training image with different template sizes. A bound should be 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: $\mathcal{T} = \{ (3 \times 3), (5 \times 5), \ldots, (n' \times n') \}$, 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. The mean entropy of all the patterns having a specific template size would be calculated and plotted with respect to the template size. As will be seen later, the mean

entropy would increase dramatically in the beginning, and will start to flatten out as an ideal template size is reached. Hence, one needs to picks the elbow of that plot as the point representing the optimal template size. According to the previous described algorithm, the profile log-likelihood of the mean entropy will be calculated and the maximum would be chosen as the elbow. However, in order to obtain a more robust value which would not depend on the maximum number of template size 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. and therefore the curvature would increase from a negative value and more significantly flatten out near zero in the later stage. For clarity, the approach is outlined in Algorithm 2. It should be noted that this approach is not CPU demanding in comparison with the effort that has to be put for manually finding the ideal template size. Entropy calculations are really fast due to the single-point entropy measure used throughout the algorithm. In the next section, some examples will be provided on the effectiveness of the proposed methodology.

Algorithm 2 Automatic Template Selection

Require: Stationary training image {TI $(N_x \times N_y)$}
1: $n' \leftarrow 0.4 \times \max(N_x, N_y)$
2: Set the bound for template sizes: $\mathcal{T} = \{ (3 \times 3), (5 \times 5), \ldots, (n' \times n') \}$
3: **for** Template size $(s_i \times s_i) = \mathcal{T}(1)$ to $\mathcal{T}(\text{end})$ **do**
4: Window $W \leftarrow W(s_i \times s_i)$
5: Construct pattern database $\{Patdb_W :$ 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: $P \times P$

7.1.2 Experiments

In order to effectively asses the applicability of this algorithm, we start by a very simple training image such that the optimal template size would be apparent. The training image is constructed by filling it with 6×6 squares, everywhere. Before analyzing the training image, it should be mentioned that the true optimal template size should be 5×5, because the stationary features of the training image are not the squares themselves, but the two perpendicular lines of each square. Therefore, the stationary feature that is repeated is actually of size 5×5. The training image is shown in Figure 16. The mean entropy curve is also plotted. It can be seen that there is a large jump at template size 5×5 and afterwards becomes almost flat. The profile log-likelihood used for automatic selection of the elbow is

also shown to demonstrate the exceptional match with our visual inspection.

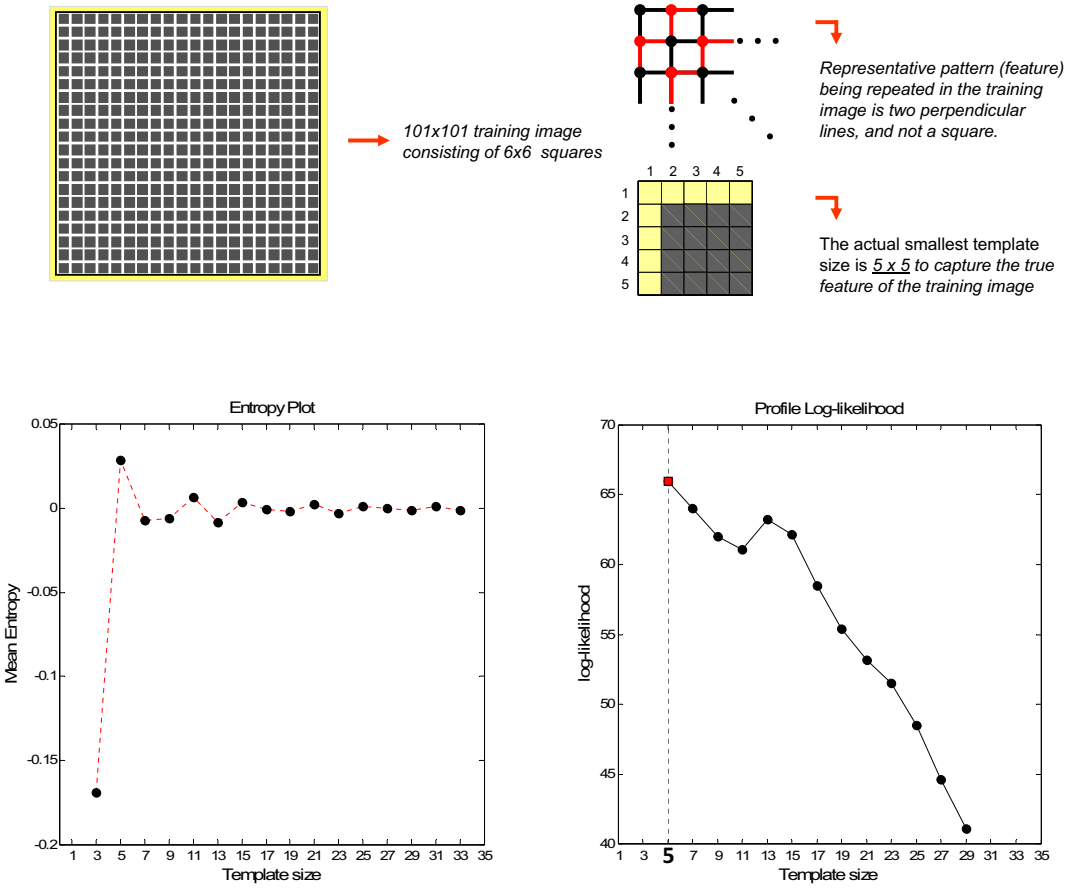


Figure 16: The application of automatic template selection on a square-filled training image, showing mean entropy curve (lower left) and profile log-likelihood of that curve (lower right), indicating the best template to be of size 5×5.

The next experimental training image is more complex. The binary training image is generated using the TIgenerator utility in S-GeMS. 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. There is no single feature that could be extracted and assumed to be the optimal feature. One has to account for the interactions which creates other shapes in the MPS simulation. For that matter, the best template size that can generate the most visually-appealing realizations will be chosen by a set of MPS simulations using different template sizes. Here, the inner patch is always set equal to the template size to increase the reliability of the assessment. 6 different templates with sizes (11×11), (13×13), ..., (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 . The selection of the best realization out of these 6 is a subjective task. Hence, to simplify the 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 17. It can be seen that the realization obtained by templates of the sizes of (17×17), (19×19), (21×21) contain a clean circle like 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 18. According to the figure, the best template size, as expected, is obtained to be 17×17 (although 19×19 can also have equal effects on the simulations).

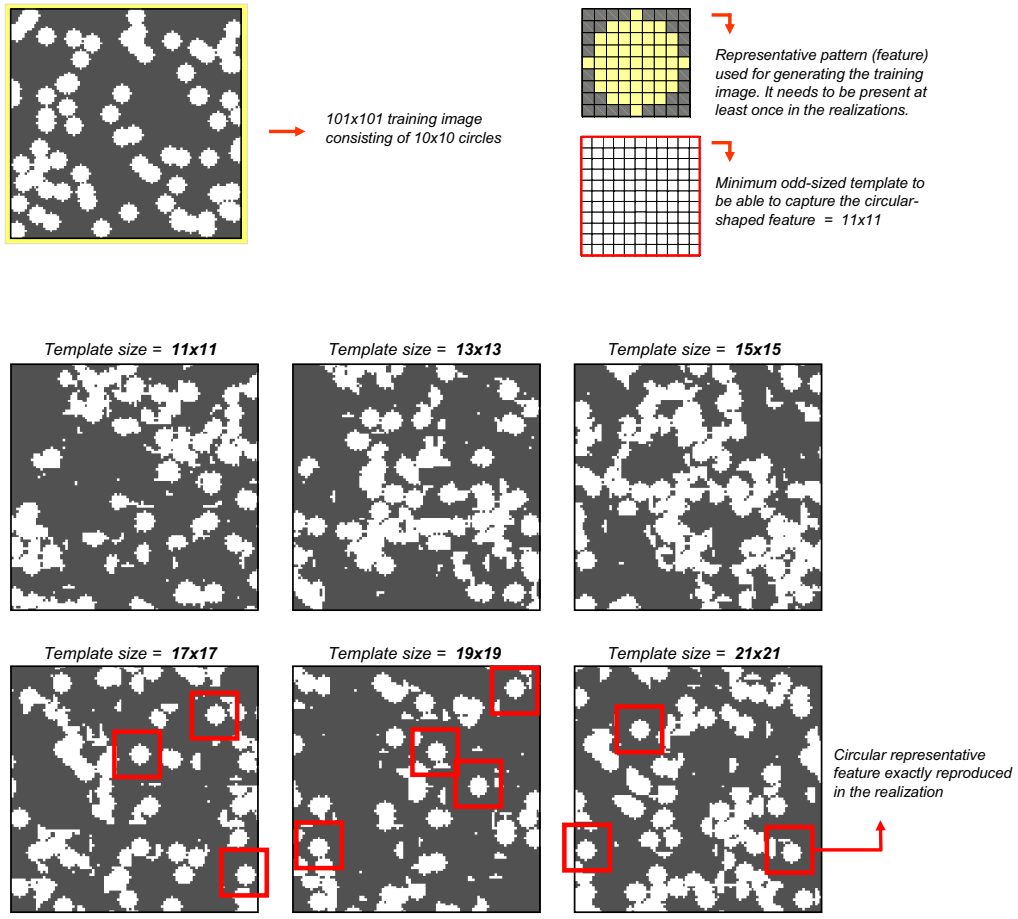


Figure 17: The training image and the original feature used in its construction are shown on top. 6 different realization using different template sizes are shown in the middle and lower parts of the figure.

A more complex 2D training image consisting of channels (sand/shale) is also tested with this methodology. This training image has no special feature related to it. Therefore, human interpretation is a necessity. A sensitivity analysis has already been made on filtersim parameters in [24]. In this paper, the exact same training image is used for the sensitivity analysis of the realizations to the template dimension. The templates that were tested are (7×7), (11×11), (15×15). The best realization that resulted in a good channel connectivity and large-scale pattern reproduction has been generated by a template size of 11×11. Applying our algorithm on this training image yields 13×13 for the best template

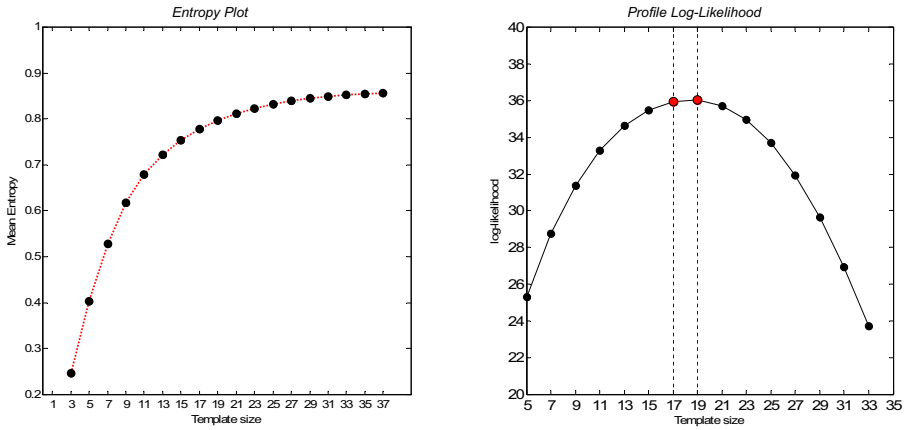


Figure 18: The application of automatic template selection on a 2D training image consisting of circles. Mean entropy curve (left) and profile log-likelihood of that curve (right) indicate the best template to be of size 17×17.

size as shown in Figure 19. This template dimension was not tested in the mentioned paper, but 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.

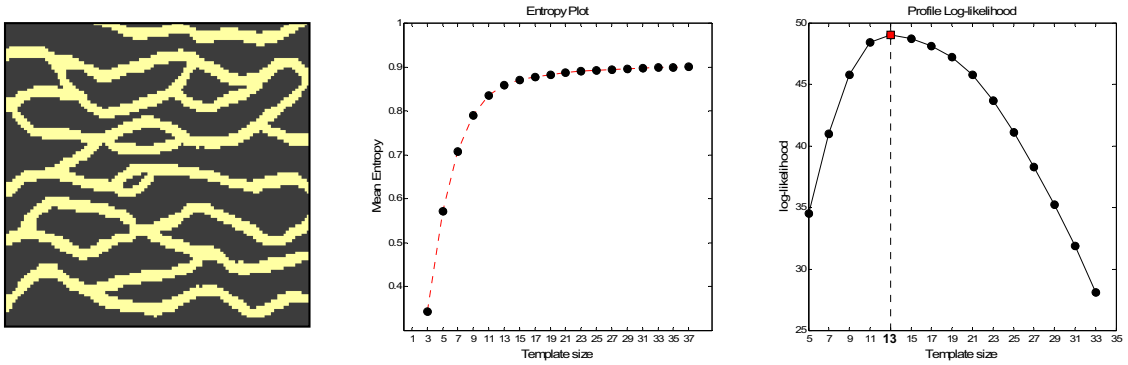


Figure 19: The application of automatic template selection on a 2D channelized training image. The figure shows the training image (left), the mean entropy curve (middle) and the profile log-likelihood (right). The best template dimension of 13×13 has been obtained.

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 clear optimal template size because no template can adequately find a specific scale for the features in the training image but the training image itself. Therefore, one expects that the proposed method would provide us with an entropy curve, that increases non-linearly, does not reveal any 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 20. The entropy of both of the 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 show 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 should significantly decrease to zero when the template size grows larger than the optimal size. However, in contrary to the stationary case, the non-stationary training image's variance does not drop to zero. This indicates the existence of non-stationary regions over the tested template sizes.

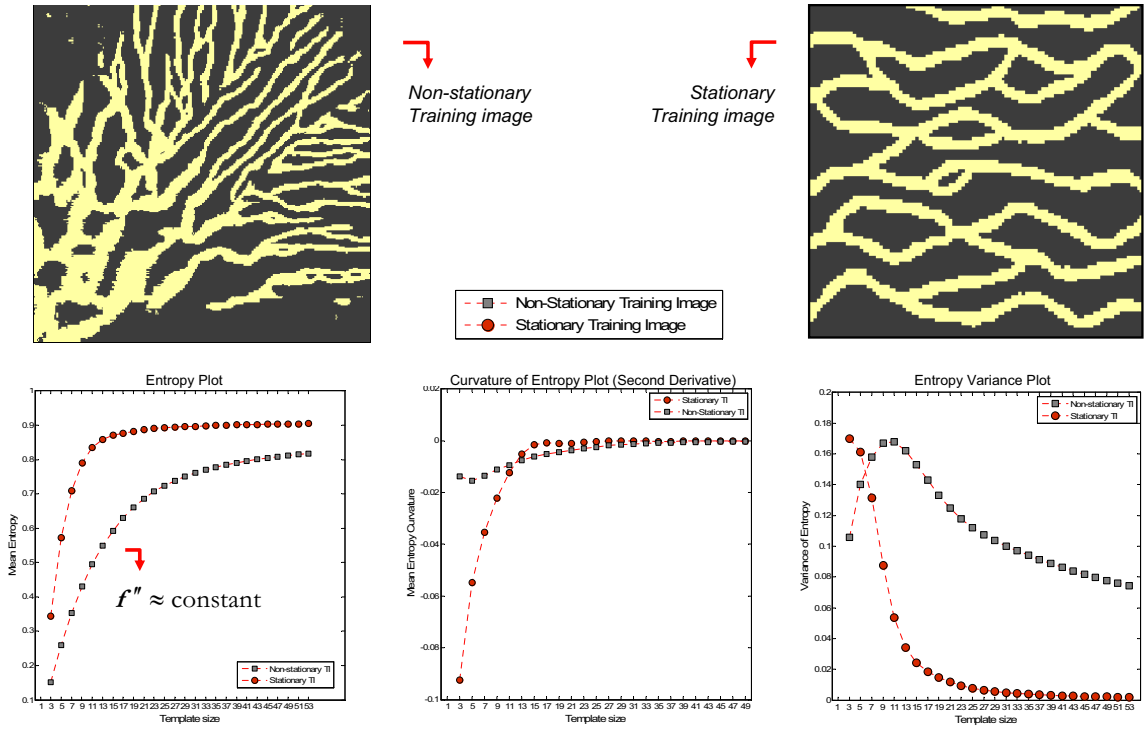


Figure 20: The application of automatic template selection on a non-stationary training image and the comparison with a stationary training image. The plots represent the mean entropy curve (left) and the second derivative of entropy curve (middle) and the variance of the entropy values (right) for both stationary and non-stationary cases. The plots demonstrate the non-existence of an appropriate template size for non-stationary training image.

7.2 Incomplete Pattern

In an unconditional sequential simulation, a data event $\mathbf{dev_T(u)}$ is defined as the set 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.

Therefore, an essential aspect for improvement in the new proposed methodology is a procedure to handle these incomplete data events. Incomplete, in simulation terms, refers to a situation that not all of the nodes within a chosen pattern template have been visited; and their values are still unknown. This situation is encountered when some nodes in a pattern

are already determined as being, in case of binary sand/shale, zero or one but the rest of the nodes are yet unresolved. Therefore, a solution to mapping an incomplete pattern to MDS space needs to be found. This will ensure the applicability of dimensionality reduction for the rest of the simulation in the same manner as the pattern clustering section. One simple idea for the binary case is to assign a value of 0.5 for the unknown nodes right before mapping them into the MDS space. Value of 0.5 has a null characteristic in term of dissimilarity distance search in the Euclidean framework, because the absolute difference of 0.5 with either one or zero is the same. For the continuous case however, one can opt to calculating the distances between the known nodes since it will provide the same mapping but in a different scale. This limitation is due to the fact that 0.5 is not a spatial average of the pattern values. Unknown nodes be assigned with a continuous value which can not be averaged between the patterns and will produce a biased distance calculations towards the mean of the spatial structure.

During the course of MPS 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 each of 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 plus the additional incomplete pattern. Although MDS technique has been tremendously improved in terms of computation, mapping an incomplete pattern for every single node in a large 3D simulation grid makes this approach inappropriate. One simple solution for this problem is to use the same idea of *SEQ*-MDS technique, which was introduced before, to map the new incomplete pattern. This is done by randomly selecting a number of patterns (greater than the dimensionality of MDS space) from the pattern database and combining them with the incomplete pattern into one 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 cumbersome. In order to reduce the number of distance calculations in each loop of the algorithm, a pre-defined set of patterns need to be found prior to simulation so that the distances between each of them can be stored beforehand. 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. Therefore, the pre-selected patterns need to effectively span the lower-dimensional space. To resolve this issue, a fast k-Medoid algorithm is applied on all of the points in MDS space in order to find a sample representative set of patterns for future mappings. Algorithm 3 summarizes the procedure on handling the incomplete data events in MDS mapping.

In summary, after mapping all the patterns to the MDS space, a set of representative points will be selected by a k-medoid algorithm, and any future mapping of data events will be carried out by *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 are the ones 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

Algorithm 3 Incomplete Pattern Mapping

Require: Pattern database $Patdb_{\mathbf{T}}$

1: Map the patterns $\in Patdb_{\mathbf{T}} \overset{\text{MDS}}{\mapsto}$ points in $\mathbb{R}^d$
2: A subset of patterns $\mathcal{S} : \{s_1, \ldots, s_{2d}\} \leftarrowtail$ using k-Medoid algorithm in MDS space $\mathbb{R}^d$
3: Store $\Delta_{\mathcal{S}}^{2d \times 2d}$, dissimilarity distance Matrix, of set $\mathcal{S}$
4: **for all** $\mathbf{dev_T}(\mathbf{u})$) : Incomplete data events during the course of simulation **do**
5: Unknown nodes $\leftarrow 0.5$
6: $\mathcal{S}' \leftarrow \mathcal{S} \cup \mathbf{dev_T}(\mathbf{u})$
7: Construct $\Delta_{\mathcal{S}}^{'}$ by using $\Delta_{\mathcal{S}}$ and $\mathcal{S}'$
8: *SEQ*-MDS mapping of $\Delta_{\mathcal{S}}^{'}$
9: obtain $\mathbf{dev_T}^{MDS}(\mathbf{u})$
10: **Do** closest pattern search in the low-dimensional MDS space
11: Paste it on the simulation grid
12: **end for**

(sand/shale), shown before, has been used. 2209 patterns of 9×9 were mapped to a 12-dimensional MDS space, and then, k-Medoid algorithm is applied to find 60 representative subset of 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 (i.e. using the whole dataset for mapping) is obtained. Therefore, a random pattern is first selected from the database and is mapped to the MDS space using a the proposed methodology and its reconstruction error is examined. The results are shown in Figure 21. 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 true coordinates of the pattern and the approximated coordinates, respectively. 3 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 (smaller index) have greater significance on the results, due to the larger influence of the sorted eigenvalues relating to them as shown in Figure 21.

Another experiment is conducted to assess the handling of incomplete patterns. Some nodes in the middle region of a pattern are assumed unknown, and assigned a value of 0.5. The applicability of the method is assessed by mapping the incomplete patterns to MDS space with the proposed approach. Next, the search for the closest point is carried out in the low-dimensional MDS space. In order to evaluate the accuracy of the mapping, the closest pattern corresponding to the closest point is selected as the approximate outcome of the mapping. Figure 22 illustrates two examples having different unknown node regions. It is observed that even in an incomplete case, one can reasonably reconstruct the missing nodes. It should be kept in mind that the approximate mapping does not affect the simulation capability of the MPS method. Because the MPS algorithm, first, tries to find the closest prototype to the data event, and hence even with a small error, the cluster that the data event belongs to can be correctly retrieved.

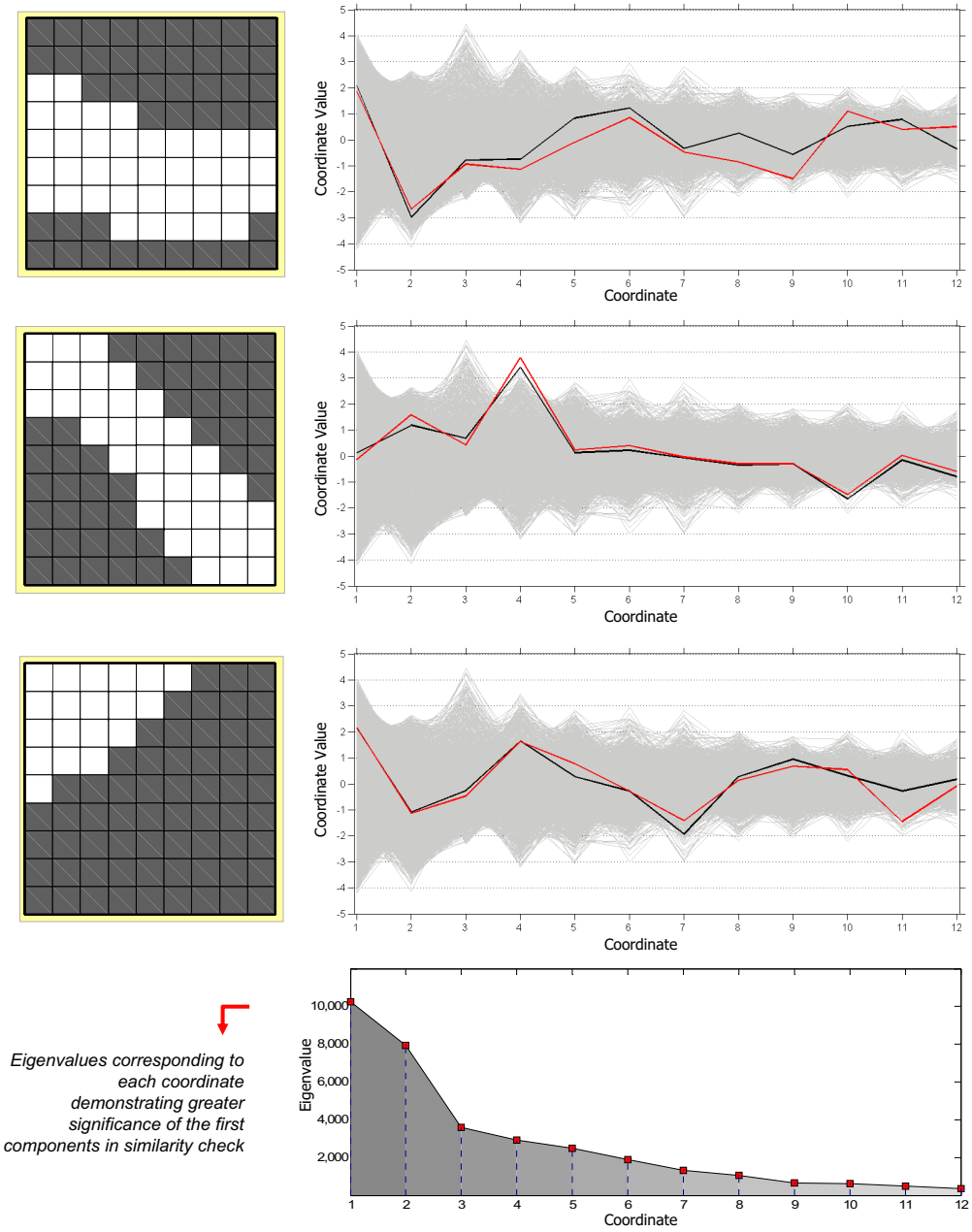


Figure 21: The application of new pattern mapping (data event) to MDS space using a subset of patterns by the proposed *SEQ*-MDS method. The patterns from database is shown on left, and the 12-dimensional MDS coordinates are shown on right. Gray lines represent the entire database, black line represents the true coordinates and red line represents the approximated coordinate.

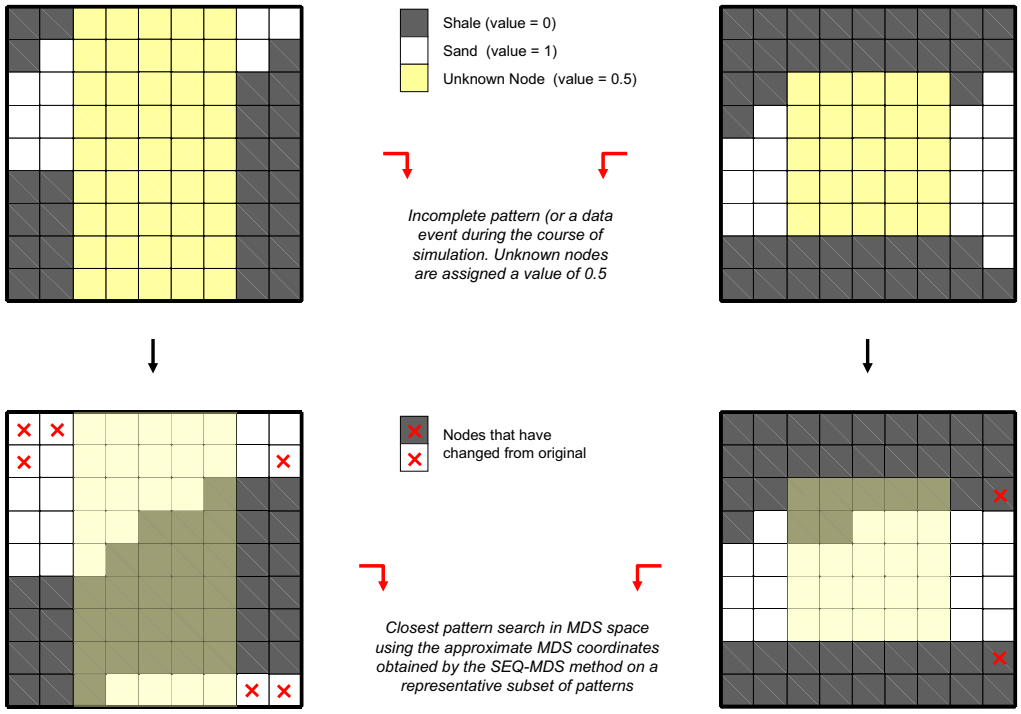


Figure 22: The application of incomplete pattern mapping to MDS space using a subset of patterns. Unknown nodes are represented by yellow, and the closest pattern to the obtained MDS point is shown.

7.3 Simulation Comparison

In this section a comparison of the proposed method will be made with the filtersim method. The filtersim results are obtained using the S-GeMS software. S-GeMS is the Stanford geostatistical modeling software that implements several geostatistical algorithms including multiple-point geostatistics paradigm as explained in [28]. Hereafter, several different training images corresponding to different subsurface structures will be used for comparison.

Simple binary 2D training image: In order to better capture the differences, a simple training image is used. This training images is similar to a puzzle board, consisting of distinct rectangles only. The size of each rectangle is 6×6. The training image is shown in Figure 23. According to the previous algorithm on automatic selection of template size, we conducted the simulation with inner patch $= 5 \times 5$ and template size $= 9 \times 9$. Some unconditional simulations were done and the most visually appealing simulated realization between 3 different attempts were chosen for illustration. In this analysis, the number of clusters were initially chosen to be $k = 100$, due to the small number of distinct patterns in this simple training image. It can be seen in Figure 23 that filtersim can not reproduce the true patterns inherent in the training image as good as the proposed method. There are many discontinuous lines in Filtersim results in comparison with our results. This stems from the poor pattern selection methodology in Filtersim. However, by increasing the

template size to 13×13 a better pattern reproduction is observed. A Filtersim realization and its comparison with the proposed method are also shown for this template size in Figure 23. Besides a small region on top left corner of Filtersim realization, the rest of the patterns follow the conceptual training image. However, the proposed methodology, in this case, produces a perfect realization.

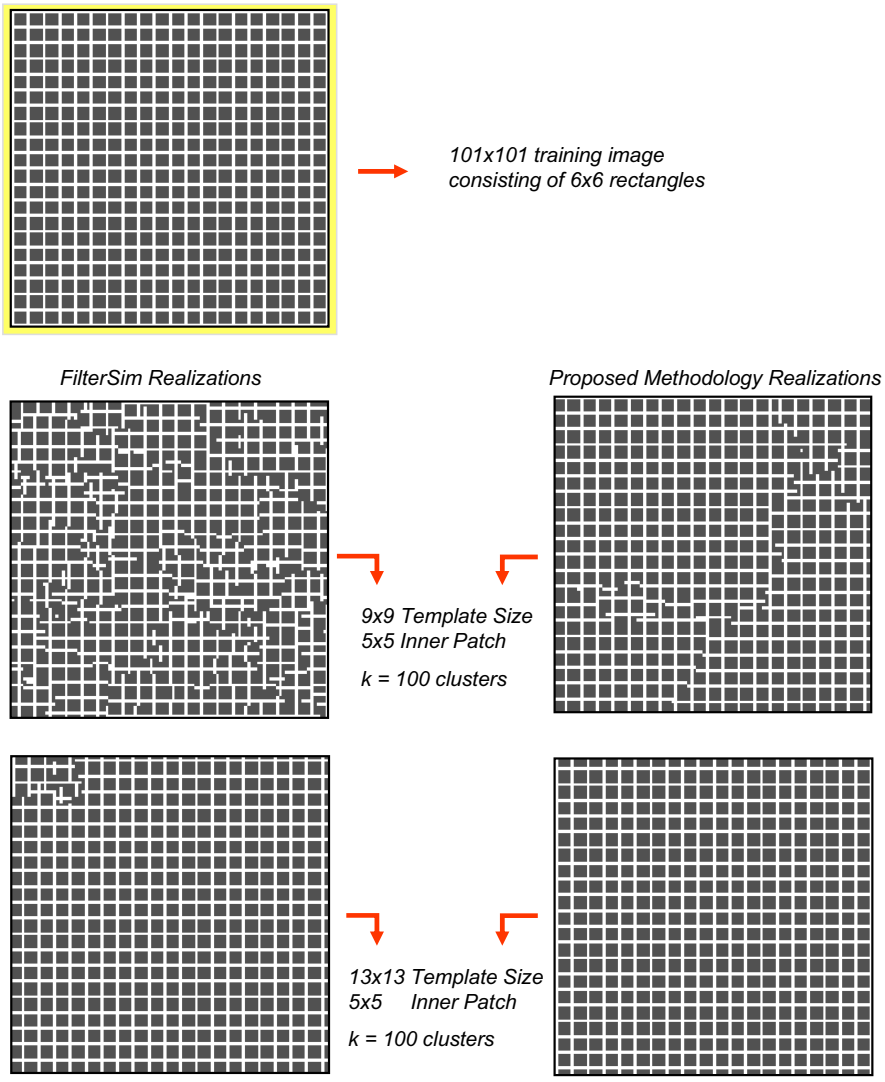


Figure 23: Comparison of Filtersim with the proposed methodology for the training image shown above. Different template sizes of 9×9 and 13 $times$13 are illustrated for comparison.

complex binary 2D training image: Next, a more geologically realistic training image, the same training image shown in Figure 10, is used for comparison. This training image represents a channelized depositional system. The parameters of the simulation in both

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

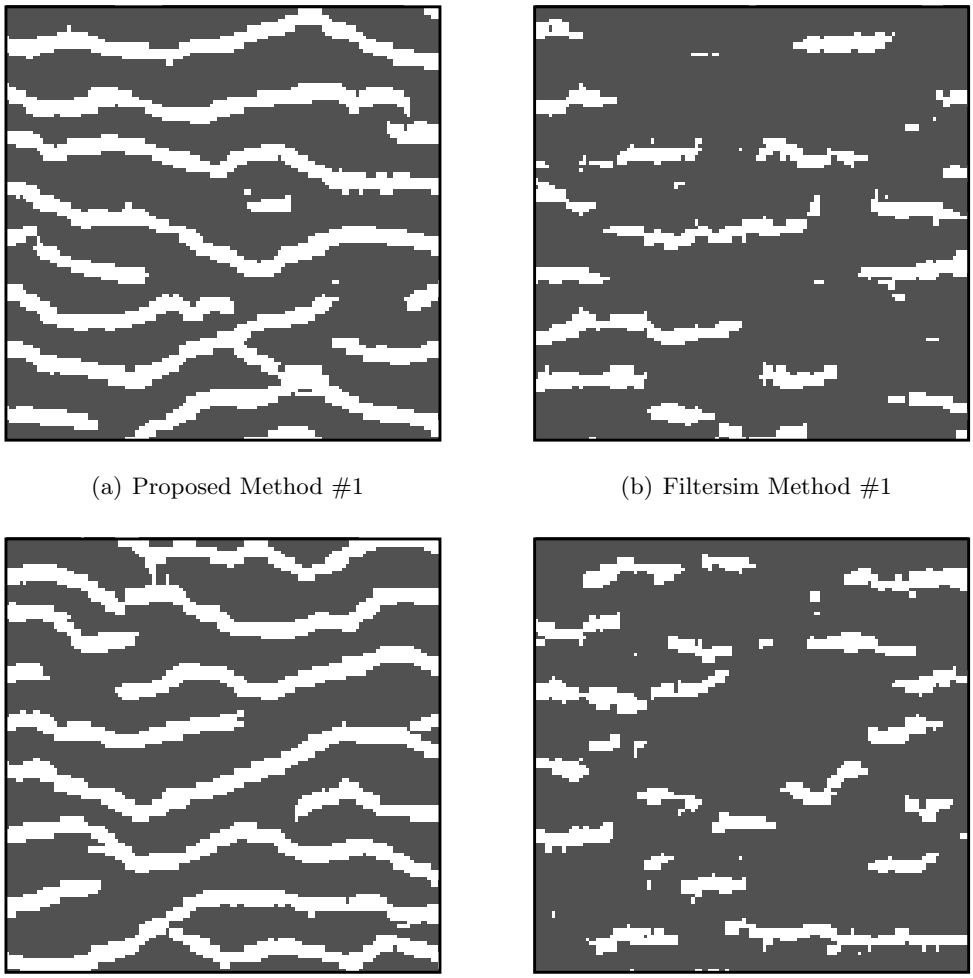
Table 2: Simulation parameters

filtersim and the new proposed methodology are provided in Table 2.

Two different unconditional realizations, generated by the presented method and the filtersim, are illustrated in Figure 24. It can be clearly observed that in the case of filtersim, by using 100 clusters, the complex spatial patterns provided through the training image are not reproduced. There are several discontinuities. Also, the large-scale spatial features are not captured to any extent. On the other hand, the proposed method can clearly enhance the pattern reproductivity.

It is noteworthy to mention that this impressive pattern reproduction in our method does not render filtersim as incompetent. One of the key parameter affecting filtersim simulated realizations is the number of clusters. 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 new proposed methodology should be observed. However, considering the total number of pattern in the pattern database, this large number of clusters results in the filtersim algorithm to perform similar to simpat method. In simpat, each data event is compared with all the pattern in the database in order to find the most similar one. Choosing a large value for the number of clusters, such as 1600, will result in nearly all 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. By removing this redundancy in filtersim algorithm, a faster and a computationally less expensive algorithm can be obtained.

The effect of changing the number of clusters in filtersim is tested and a comparison is made with our method. Intuitively, by increasing the number of clusters, the realization will look much closer to the conceptual training image. However, due to the poor classification capability, filtersim can not reach the effectiveness of our proposed method even with 100 clusters. The results are illustrated in Figure 25. The improvement brought by our method is evident in the provided realizations.



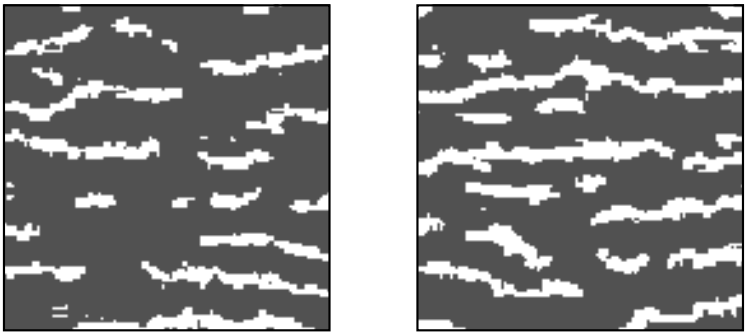
(a) Proposed Method #1

(b) Filtersim Method #1

(c) Proposed Method #2

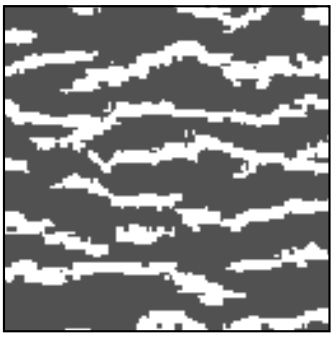
(d) Filtersim Method #2

Figure 24: 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.

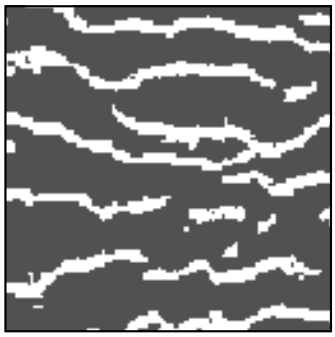


(a) Filtersim with $k = 200$

(b) Filtersim with $k = 400$



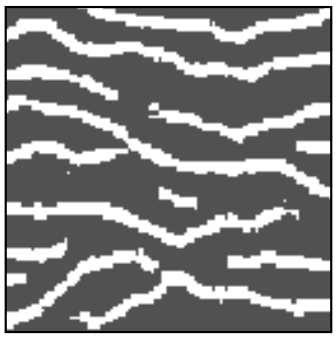
(c) Filtersim with $k = 800$



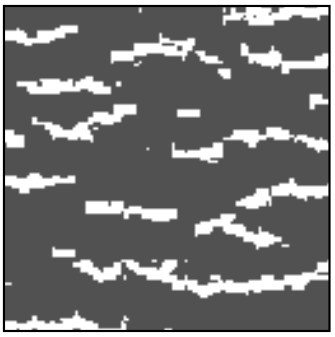
(d) Filtersim with $k = 1600$

Figure 25: Filtersim resulting realization with increasing number of clusters, k, in k-means clustering.

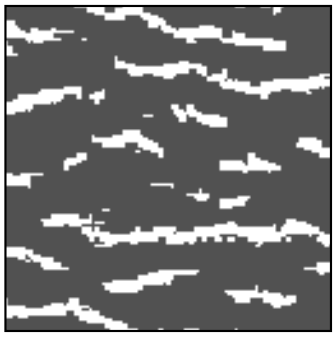
One other aspect that strongly affects the simulated realizations is the search template and inner patch sizes. The choice of the template size depends on the complexity and the scale of the patterns to be reproduced. Generally speaking, by increasing the size of the template size and inner patch, realizations will look 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 bigger template size of 13×13 and a bigger inner patch of 7×7 is chosen for simulation. The results are shown in Figure 26. Noticeably, the increase in the search template did not produce much better realization for cluster numbers of 100 and even 400. On the other hand, the proposed method was robust and could reproduce reasonably-appealing realizations with the training image. This demonstrates the robustness of the algorithm and its low sensitivity to the parameters required as input to the algorithm.



(a) proposed method, $k = 100$



(b) Filtersim, $k = 100$



(c) Filtersim, $k = 400$

Figure 26: 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.

quasi-stationary binary 2D training image: A more challenging training image that represent meandering channels is chosen in this analysis. According to [31], the meander train is assumed to be the result of the stochastic fluctuations of the direction of flow due to the random presence of direction-changing obstacles in the river path. The training image characterizing this behavior exhibits quasi-stationary, hence difficult, channel structures. The application of the proposed methodology is tested on this training image and is compared with filtersim results.

Figure 27 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. For the proposed methodology, 1000 is chosen as the number of clusters, and 2000 for filtersim method. This advantage is provided for filtersim to reduce the approximation effects of random pattern selection from each cluster, as the number of patterns inside a cluster will be significantly reduced. Three different realizations, using both methods, is shown in Figure 27. Clearly, filtersim results are less structured than the realizations from 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. The idea of generating new patterns will eventually improve on this matter.

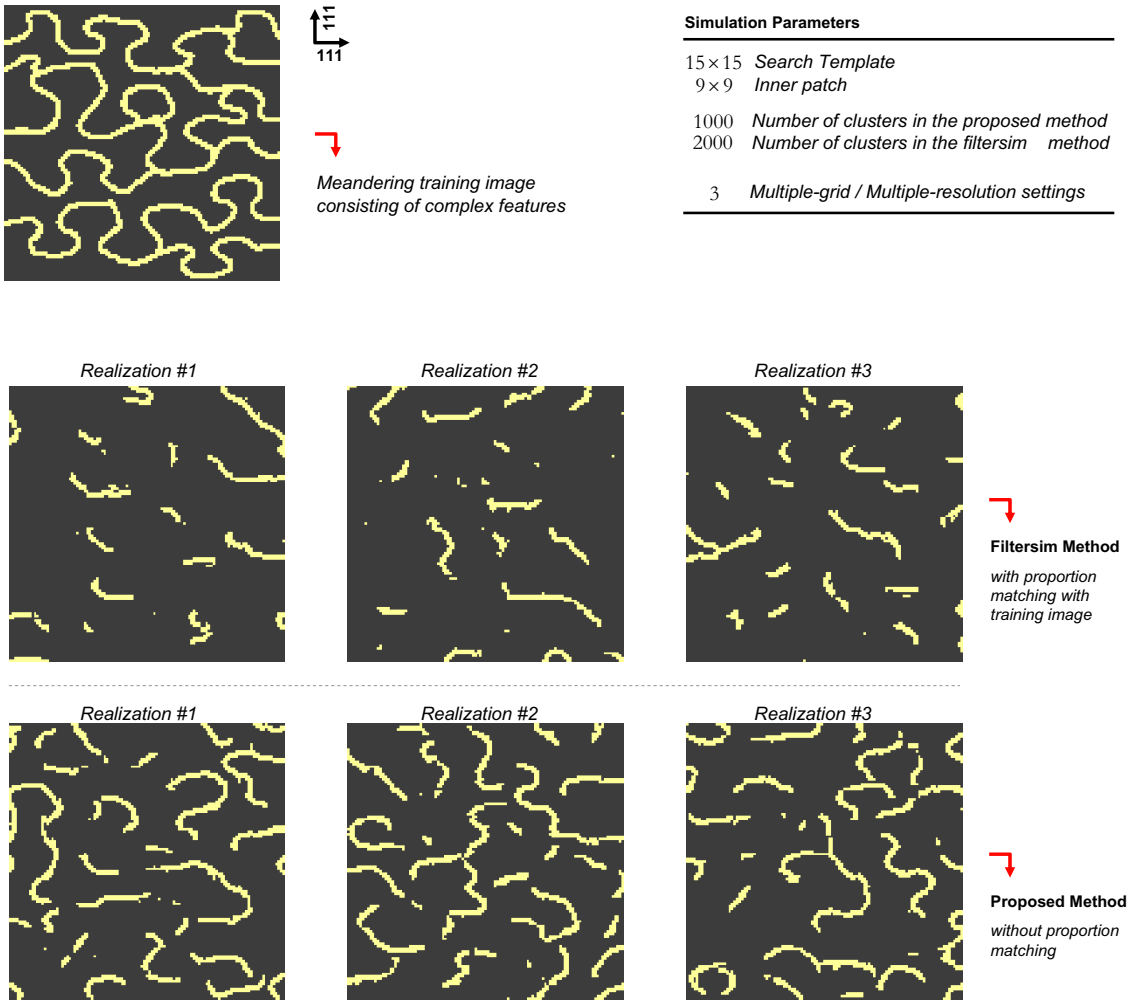


Figure 27: Comparison of Filtersim with the proposed methodology for a quasi-stationary meandering training image. Three realizations are shown for filtersim (top) and the proposed method (down).

4-facies categorical 2D training image: Next, some unconditional simulations were performed on a categorical training image with 4 different 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 [29]. The 3 best results between different realizations are shown in Figure 28. Here, the template size of 11×11 and inner patch of 7×7 was chosen according to the pattern homogeneity scale. Similar to previous results, it can be observed that the proposed method can better honor the provided conceptual geological model. More distortion exists in the filtersim realizations. This originates from the incapacity of filtersim

classification to regroup training patterns with extremum values within a small number of classes is even more important at the coarse scale (first grid being simulated), because of the large scale template. On the other hand, in the proposed method, a better coarse scale simulation result will considerably improve the final pattern reproductivity.

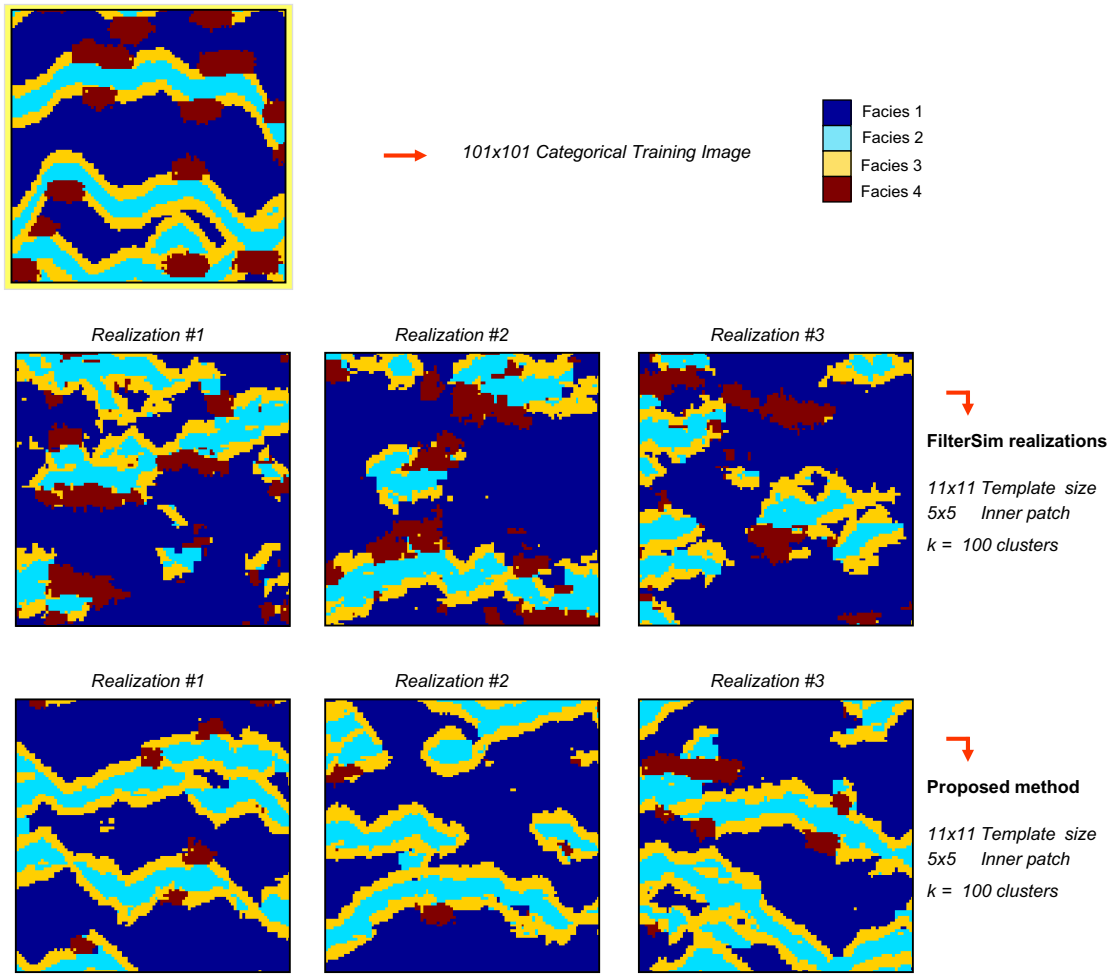


Figure 28: Comparison of Filtersim with the proposed methodology for a categorical training image. Template size of 11×11 and an inner patch of 7 $times7$ are used for simulation. 3 realizations are shown for filtersim (top) and the proposed method (down).

It should be mentioned that the filtersim realizations have been forced to match the training image proportions, whereas no histogram matching has been performed on our realizations. This proportion matching tries to produce realizations that look similar to the training image in one-point statistical sense. In order to see how filtersim would behave without this external influence on the simulated realization, 3 different unconditional simulation were generated using filtersim. The result are shown in Figure **??** A poor pattern and histogram reproduction in the realizations is evident. This validates the dramatic improvement of our proposed method to filtersim, since these are the realizations are generated

with the same parameters as in our method.

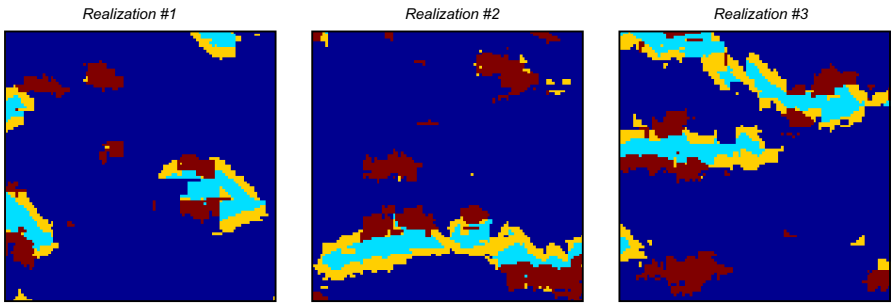


Figure 29: Unconditional filtersim results using the previous categorical training image but without any histogram matching. Template size of 11×11 and an inner patch of 7×7 are used for simulation. These are the realizations that should be compared with the ones generated using the proposed method.

Continuous 2D training image: A continuous training image, consisting of thick and narrow cracks as its extreme features, is used for this analysis(obtained feom [11]). The training image is of size 159×159. For this continuous multiple-point simulation, 200 clusters are used for k-means algorithm due to more variability in pattern database. The dissimilarity function used in the continuous case is the manhattan distance function, since the distance proximity transform is inappropriate for a continuous pattern and one needs to clearly take care of the any edges in the patterns. 3 different realizations are generated with filtersim and the proposed method. The results are shown in Figure 30. In this specific continuous case it is difficult to distinguish between a good or bad pattern reproduction. High values that are close to one or the edges of thick cracks alter ones perception about the goodness of the reproduced patterns. However other structures/patterns, which are not apparent to the human eye, should also be taken into account. Statistical measures, such as histogram or variogram can be chosen for comparison. But in this case the one-point and two-point statistical measures match closely to the training image. Thus, we comply with the visual inspection of the realizations. It can be inferred from Figure 30 that the proposed method can produce slightly better realizations than filtersim. In the proposed approach, we have removed some of the patterns in the first and second multiple-grids due to the limitation on matrix sizes in Matlab environment. A C++ implementation of the code, which would have no memory limitation as in Matlab, can improve on the small scale pattern reproduction and general performance of the algorithm.

binary 3D training image: The application of a 3D training image on both methods is tested next. One unconditional realization is generated using filtersim and the proposed method. 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 3 multiple-grids setting is chosen for the simulations. The resulting realizations are shown in Figure 31. The figures demonstrate that the proposed methodology can adequately model

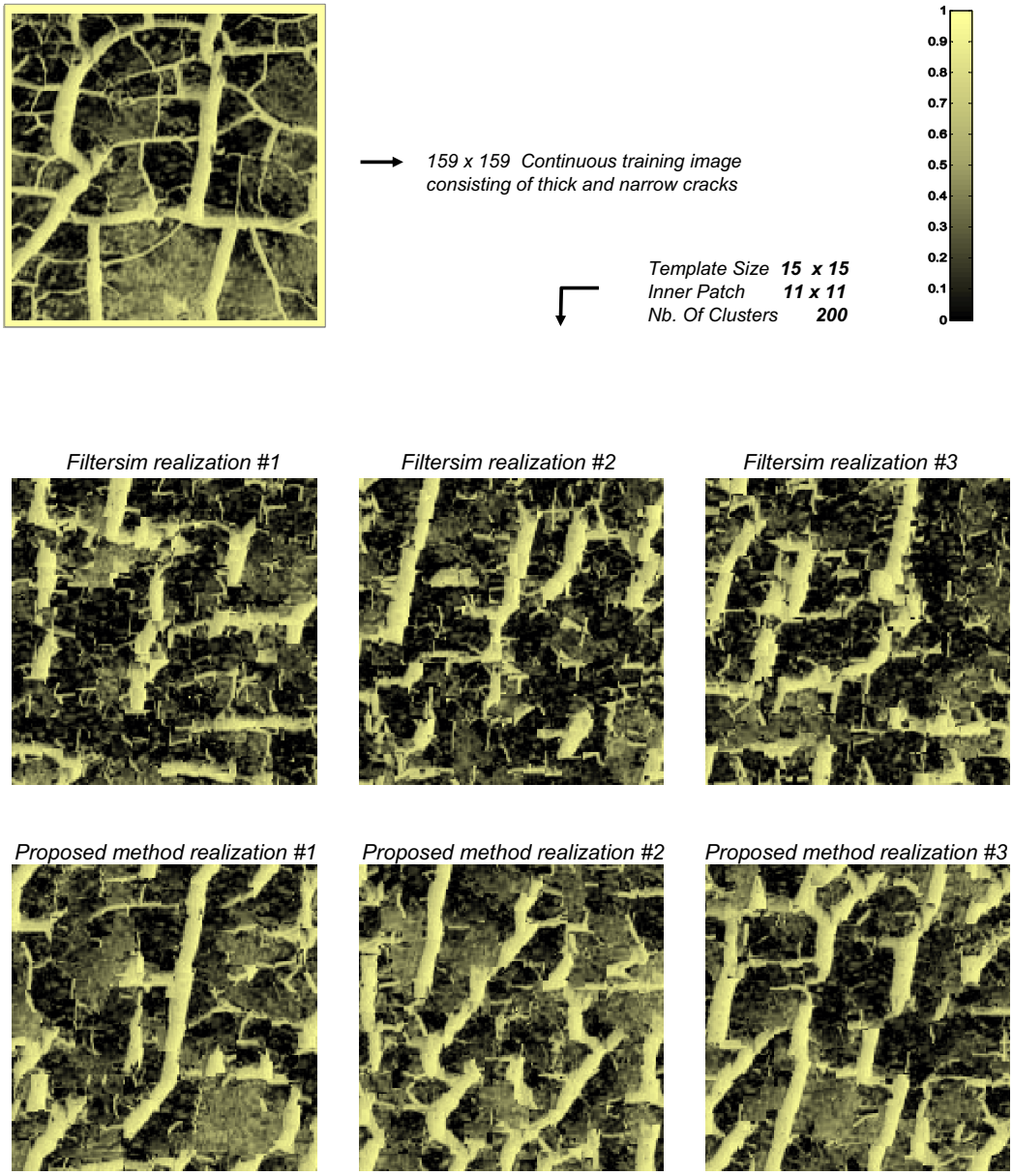


Figure 30: Continuous training image (159×159) shown on top is chosen for comparison. Three different filtersim realizations (middle) and three different realization using the proposed method (bottom) are shown.

complex, 3D geological scenarios. On the other hand, filtersim realization display poor pattern reproduction. Given the result of Figure 31, one might argue that, it may be possible to obtain better realizations using a larger template size in filtersim to adequately capture the desired large-scale patterns, but it is assumed appropriate for the comparison. Figure 32 shows three random horizontal slices of the training image and the realizations for better visual analysis. It is evident that 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. This has a great impact on the speed of the simulation (i.e., pattern similarity search during the simulation) especially in the case of the final multiple-grid. Although the code is implemented in a Matlab environment, faster executions are observed in comparison to the filtersim implementation which is in C++. However, due to the size restrictions on the matrices in Matlab, half of the patterns are randomly selected for the analysis, and this can somehow reduce the overall strength of the algorithm.

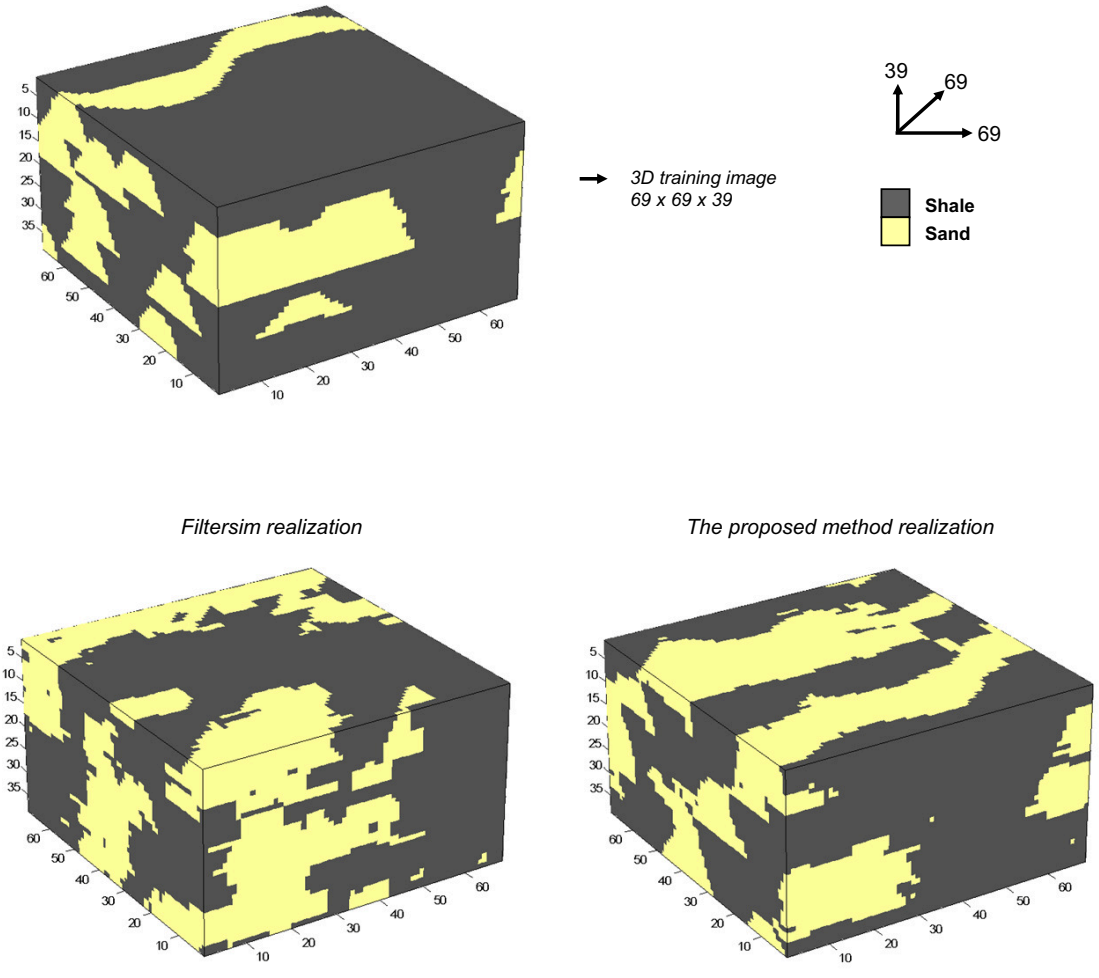


Figure 31: 3D unconditional simulation on a $69 \times 69 \times 39$ training image shown in top. Filtersim realization (on left) and the proposed method's realization (on right) are illustrated for comparison.

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. Two-point statistics often cannot fully characterize

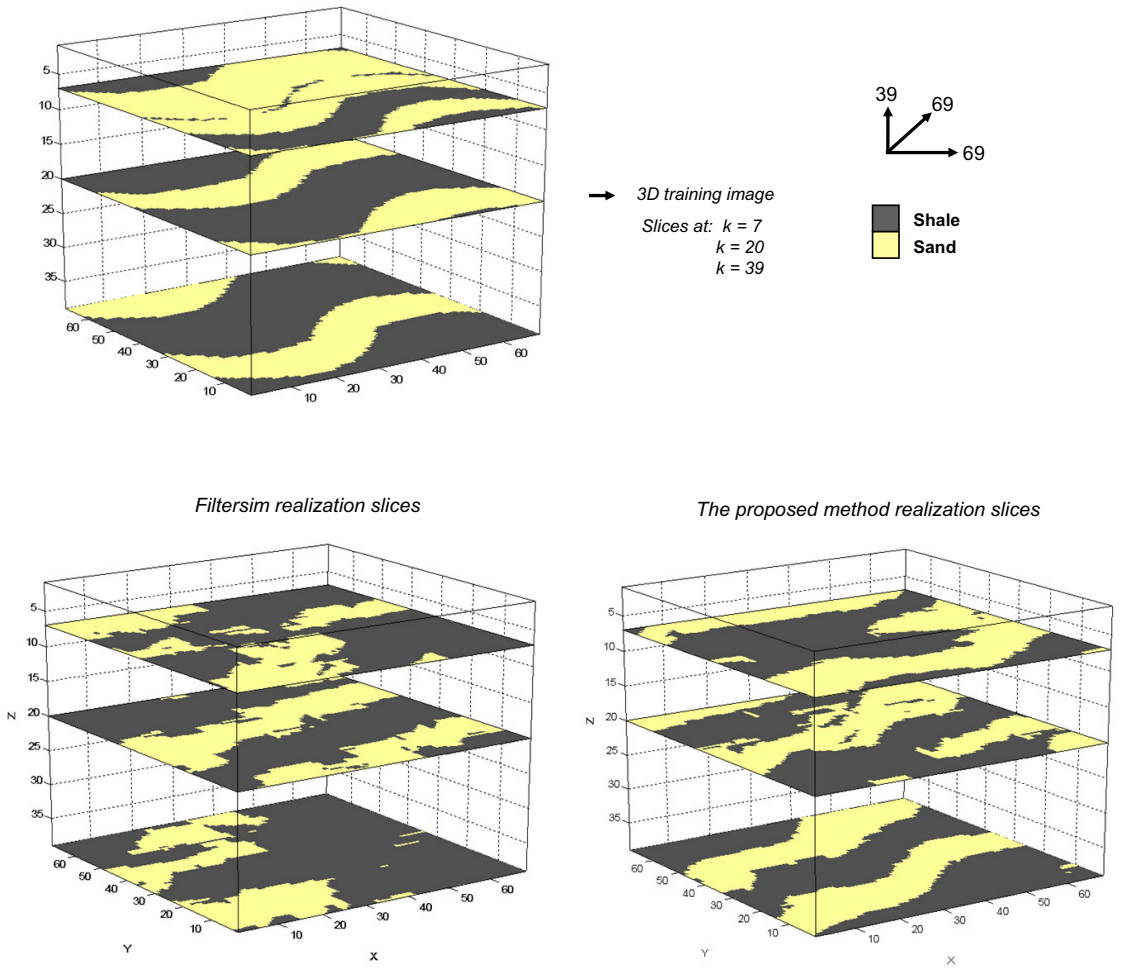


Figure 32: 3 slices of the training image (on top) and two realizations, one from filtersim (on left) and the other from the proposed method (on right), are shown on horizontal planes at locations $7, 20$ and 39.

geological arrangements containing curvilinear features and other complex shapes. Therefore, a common multiple-point statistic, called multiple-point histogram, is chosen for the analysis [21]. A multiple-point template of the size $3 \times 3 \times 2$ is chosen. Every single binary configuration of this template is searched in the 3D realization and their frequency is obtained. This multiple-point template produces $\sum_{i=1}^{18} \left(2^{i-1}\right) = 262143$ different configurations. These multiple-point statistics also contain all lower-order statistics as well. By indexing each configuration, one can plot their frequency with respect to their configuration. The multiple-point histogram (MPH) plots for the 3D training image (reference), one filtersim realization and one realization of the proposed methodology are shown in Figure 33. There are some differences between the realizations MPH and the training image MPH. This can be due to the (1) bad template size selection for simulation, (2) and the variation between the original template size used for the simulation and the one used for the his-

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

Table 3: MPH errors

togram evaluation. Be that as it may, one can calculate the absolute error between these histograms as a measure of the goodness of the realizations. The error is obtained according to the norm-1 difference as follow:

$$error = \sum_{k=1}^{d} |C_k^{TI} - C_k^{realization}|$$

where $C_K = \{$count of the patterns with configuration index of $k\}$, $d =$ number of configurations $= \sum_{i=1}^{n_x n_y n_z} \left(2^{i-1}\right)$, where n_x, n_y, n_z are the multiple-point template dimensions. The resulting errors are shown in Table 3. It can be seen that the proposed method has a better multiple-point histogram reproduction than filtersim.

One may argue that the absolute error between the histograms is not a good measure as the proportions of sand and shale can significantly change the one-point statistics of a histogram. In order to obtain a better measure, we calculated the JensenShannon (JS) divergence. 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 $q = \{q_i\}$ [22],

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

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

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

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 true 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. The resulting divergences are shown in Table 4. It can be easily seen that the realization generated with the proposed method provides a smaller D_{JS} than the one from filtersim. This proves the better multiple-point statistical outcome in the proposed methodology.

7.4 Data Conditioning

One of the main advantages of multiple-point statistical methods over object-based simulations is their capability in conditioning the realization to dense hard and soft data. They

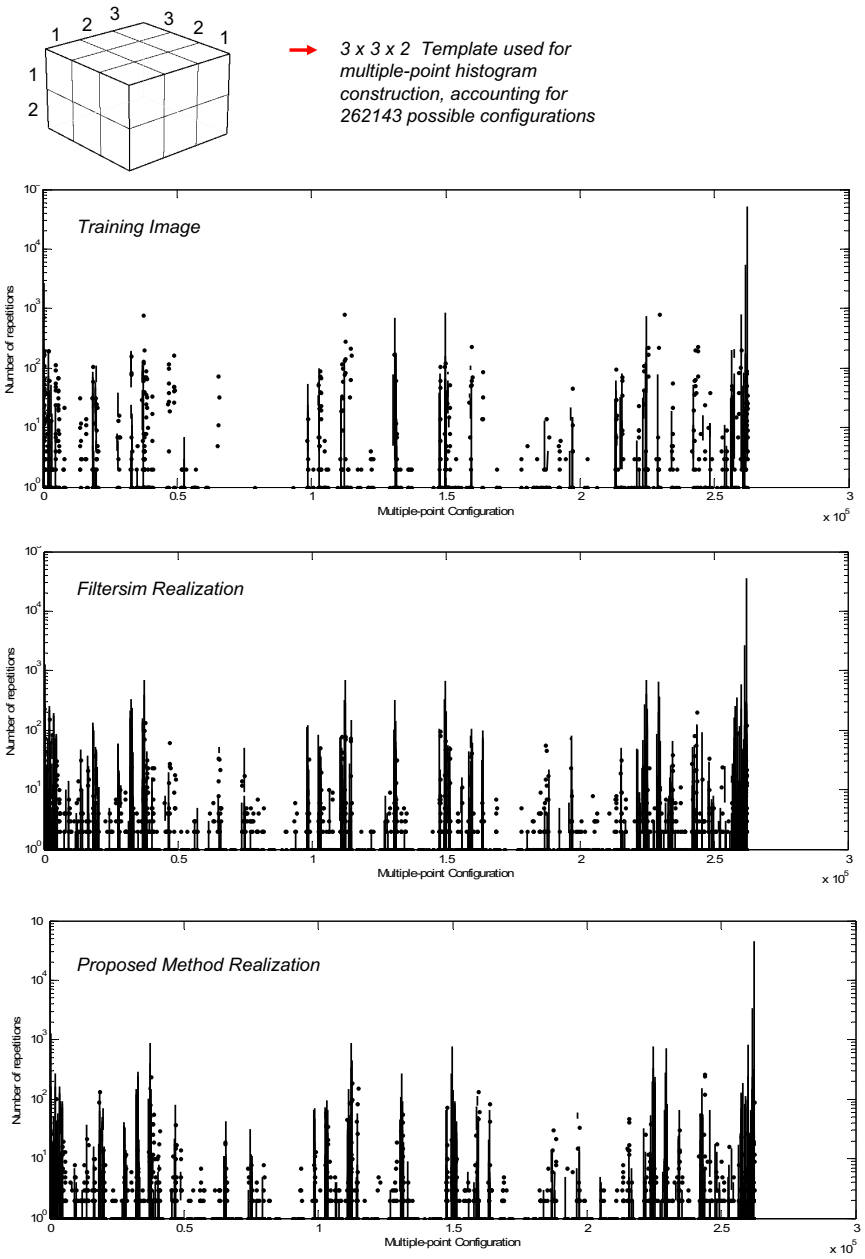


Figure 33: Multiple-point histogram (MPH) of the models, considering the multiple-point template of size $3 \times 3 \times 2$. x-axis is labeled according to the index related to the multiple-point configurations, and y-axis represents the number of observed instances of each configuration.

provide auxiliary information that, if integrated within the model, will reduce the uncertainty in the generated realizations. To perform a full comparison of the filtersim method with the proposed methodology, hard data conditioning will be examined in this section.

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

$D_{JS}\left(\text{hist}_{TI} \parallel \text{hist}_{Realization}\right)$	
Filtersim Method	The proposed Method
0.0705	0.0513

Table 4: JensenShannon divergence between the multiple-point histogram of the training image with respect to each realizations.

node locations in the similarity measurement. For example, hard data nodes are given 0.5, and frozen nodes 0.3, and the rest of the nodes 0.2 for the weight. 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. There is no limitation in our algorithm to provide weights in the dissimilarity functions. It should be mentioned that weighting different node locations is performed just for comparison purposes. Alternative approaches are currently being considered.

First, the channelized training image (sand/shale) is chosen for testing purposes. In the first test, 5 hard data are chosen for analysis. The E-types are used to check the quality of conditioning. If the algorithm conditions properly, an E-type should give less uncertainty near the hard data points. Therefore, 100 realizations are generated using the proposed method and filtersim and sgsim (with a proper variogram) and their E-types are calculated. The results are shown in Figure 34. It is seen that all three methods can condition to hard data. The E-type of the proposed method shows the channel structure around the hard data locations. Also, Figure 34 demonstrates the expected spatial structures as one moves away from 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 we see a stronger tendency towards shale facies. However, as will be demonstrated later, the structural patterns can not be fully captured in the filtersim E-type. In order to verify this, the E-type of 100 sgsim realizations is also shown. The E-types from sgsim accounts only for two-point statistics, yet, provides similar results as filtersim. This demonstrates the better data conditioning obtained by the proposed method.

Next, 100 hard conditioning data are used for the comparison. The results are shown in Figure 35. Recall that filtersim was used without proportion control resulting in more shale facies present in the realization, 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.

Another experiment is performed with two neighboring hard data. In this analysis, a sand and a shale facies are attributed 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 E-type is calculated for three MPS methods: filtersim, snesim and the proposed methodology. The parameters for snesim are given in such a way that they can most closely emulate the pattern template as used in the pattern-based methods. The search template geometry in snesim is $(9, 9, 9)$, indicating a square search template as in filtersim, and the number of nodes in the search neighborhood is 25, indicating an inner patch of 5×5. The rest of

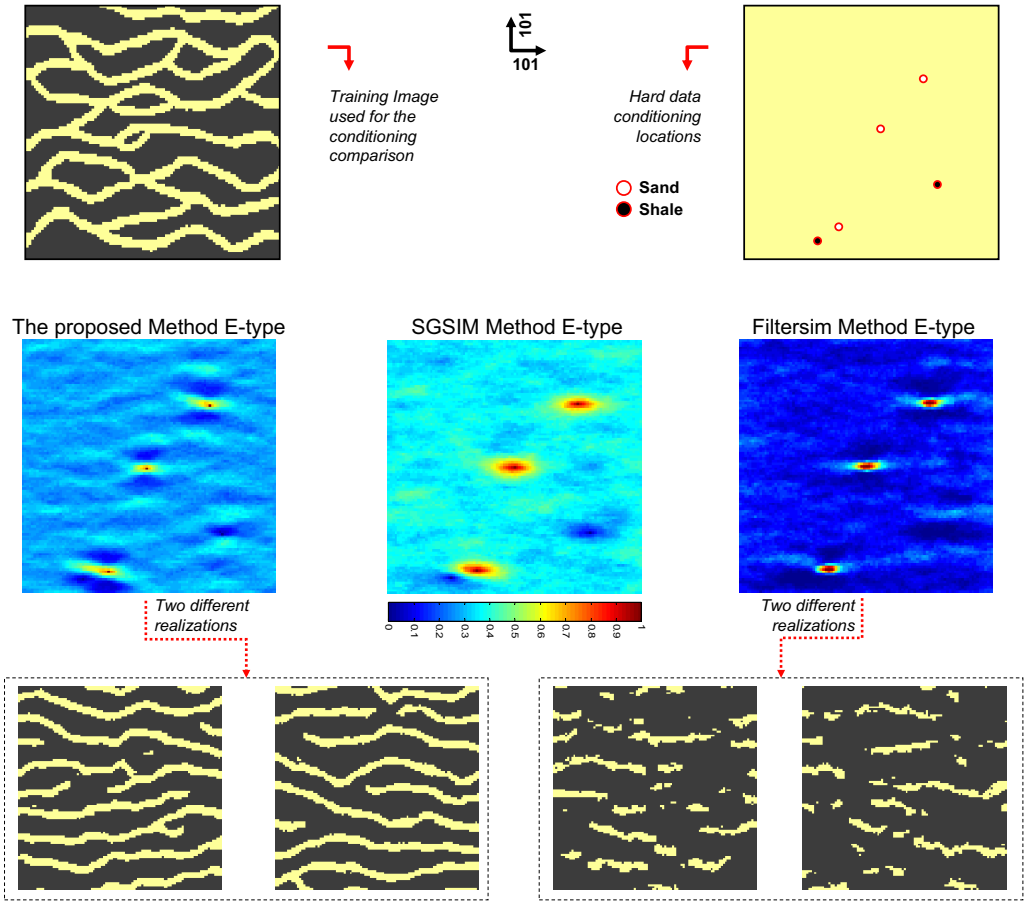


Figure 34: The hard data conditioning locations are shown in top. The E-type average of 100 realization generated using the proposed method (left), the sgsim method (middle) and filtersim method (right) are shown. The multiple-point structures are better captured in the proposed methodology as seen in the E-type averages. Two different MPS realizations of each method is also shown on the bottom.

the parameters remain untouched. It is worthwhile mentioning that every algorithm has its own parameters that needs tuning for better results, but for the sake of comparison we have assumed similar situations in every MPS simulation. The results are shown in Figure 36.

It is observed that all considered MPS methods exhibit the expected sand and shale structure that was provided with hard data. However, the results obtained by filtersim and snesim demonstrate the poor pattern reproductivity in terms of the certainty of the facies as one moves away from the hard data locations. It means that, in general, a human expert by looking at the provided conceptual training image structures can deduce that a channel with a more lateral continuity is passing through sand facies hard data and shale facies is existing above it. Also, knowing the average width of a channel, the human expert can infer a shale structure underneath the channel. Neither of the algorithms, snesim or filtersim, were able to accurately capture the multiple-point information that was provided

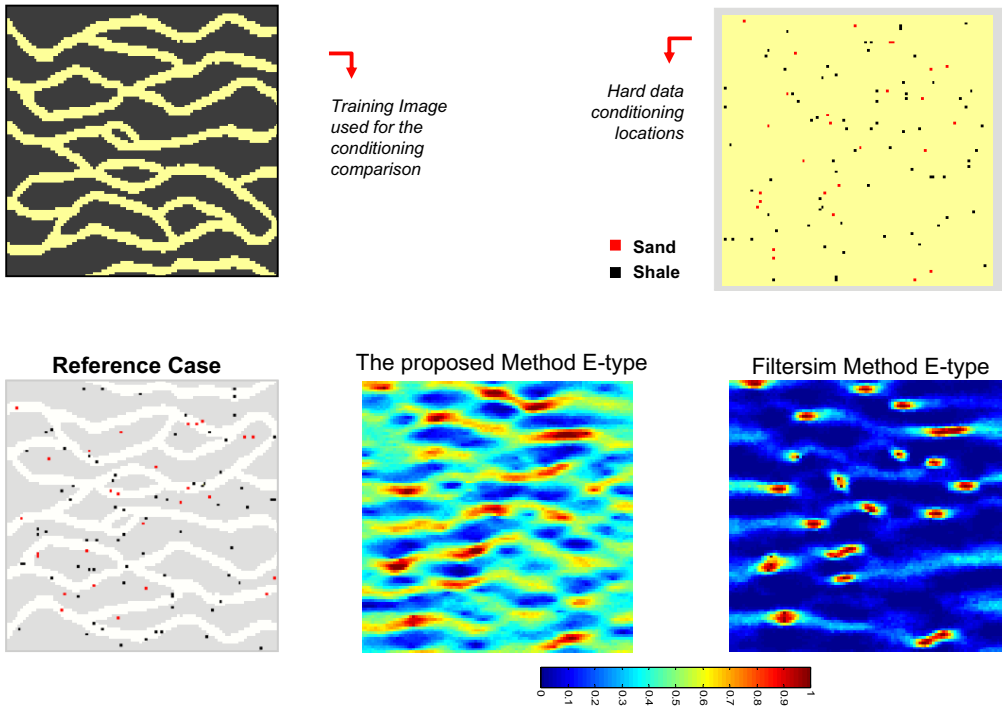


Figure 35: The hard data conditioning locations are shown in top. The E-type average of 100 realizations generated using the proposed method (middle-bottom) and filtersim method (right-bottom) are shown. The reference case is also shown on left.

through the training image. On the other hand, the E-Type average realization obtained by the proposed methodology demonstrates the same channel structures as anticipated by a human expert. In order to confirm this, the inherent structure presented through the training image should be revealed. Therefore, a window of size 35×35 is used to scan over the training image and store all the patterns that honor the same hard data configuration at their central node. This analysis provides us with the large-scale pattern structures that honor the conditioning data. Averaging the stored patterns would approximately yield the hidden structure that should be revealed through the E-Type of several realizations. The resulting pattern is illustrated in Figure 37 with the zoomed-in version of the central region of the E-types for each algorithm. It is perceived that the training image structures resemble the ones seen in the central part of the E-Type obtained by the proposed methodology. Although the number of patterns within the specified window size of 35×35 is limited, the multiple-point information has to approach the average, since the patterns/probabilities are being borrowed from the training image.

In conclusion, the conditioning in the proposed method is as good as other algorithms and at the same time produces the essential features of the training image. This has the advantage of generating better geologically realistic realizations and reservoir models.

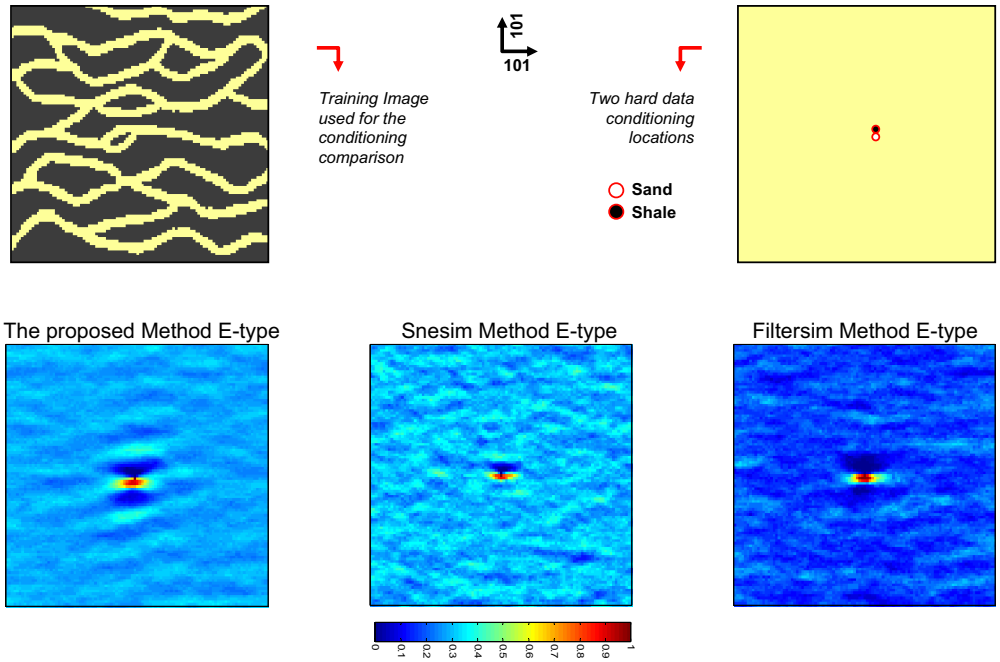


Figure 36: The neighboring hard data conditioning locations are shown in top. The E-type average of 100 realizations generated using the proposed method (left), snesim method (middle) and filtersim method (right) are shown in the bottom.

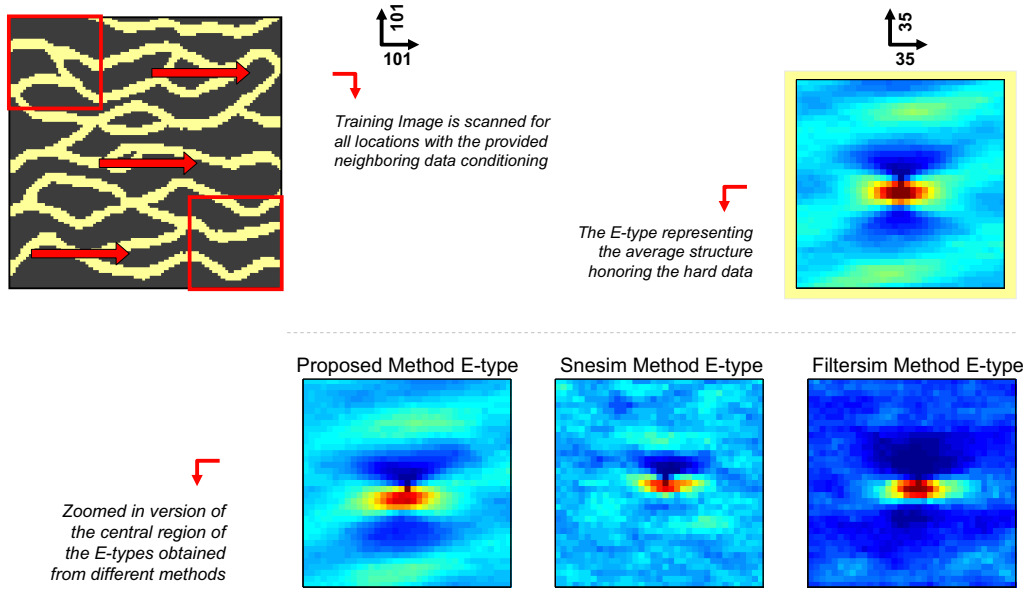


Figure 37: The average of all 35×35 patterns in the training image that honor the neighboring hard data at their central nodes is shown on top-right corner. The zoomed-in version of the E-type of different methods is shown for comparison on the bottom.

7.5 Variability between Realizations

In the previous sections, the proposed methodology was compared to filtersim using different training images. 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. 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's realizations may be seen as rather limited in comparison with filtersim. This may affect the uncertainty analysis that a reservoir engineer makes during the 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 $\mathcal{L} : \{l_1, l_2, \ldots, l_N\}$, 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 should give zero for 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 would account for multiple-point information. Therefore, it should be a function of the multiple-point histogram of the two realizations. In other words, $f(l_i, l_j) = g\left(\mathrm{MPH}(l_i), \mathrm{MPH}(l_j)\right)$, where f represents the distance function and g represents another distance function defined over two probabilities distributions. In our analysis, we use Jensen-Shannon divergence (JSD) as a distance function between the 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 multiple-point information histogram at any scale. In a binary case, a square template of size $t \times t$ will results 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 environment. 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 38. 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 not suitable 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 parallel coordinate representation of the data with the first 20 dimensions is shown in Figure 39(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.

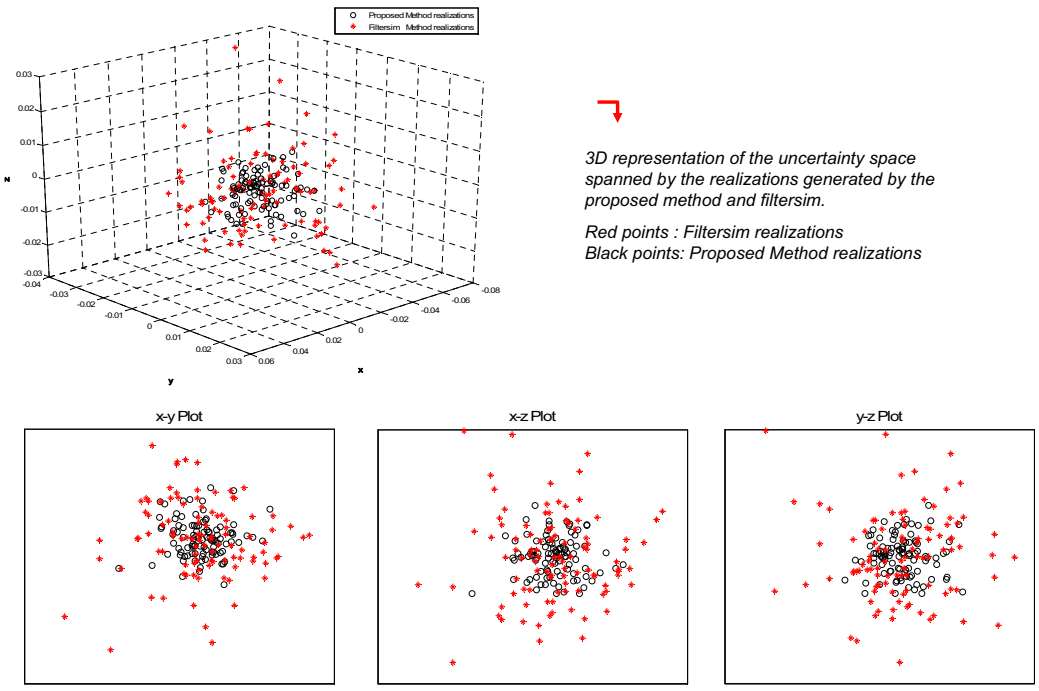
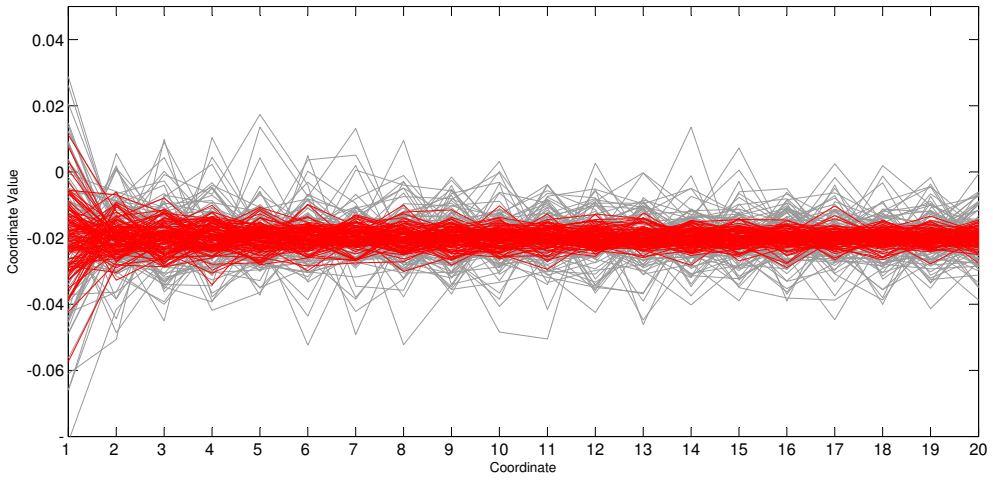


Figure 38: Comparison of the variability between the 100 realizations generated by filtersim (red) and the proposed method (black) by the first 3 principal coordinates of the space of uncertainty. 2 dimensional views of the data are also plotted for clarity.

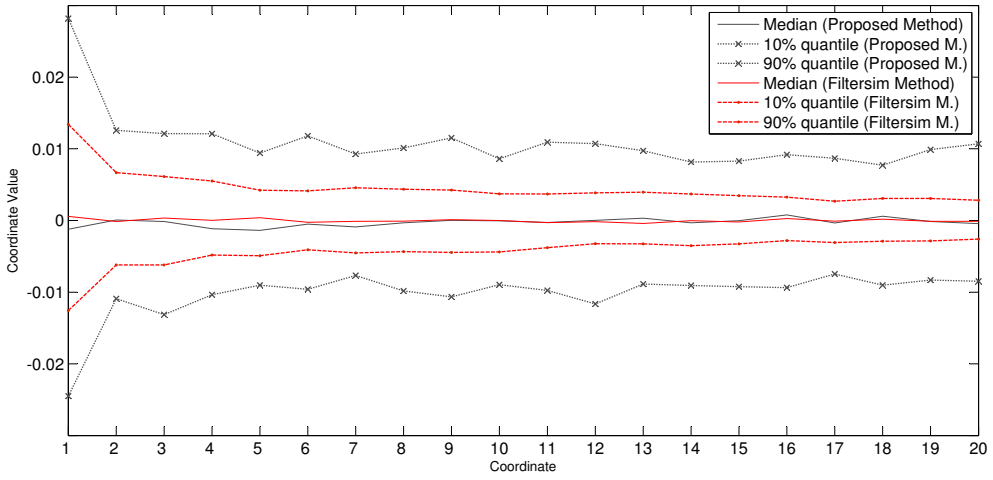
larger coordinate values clearly demonstrate the larger variability in the filtersim results. Figure 39(b) shows the 10% and 90% quantiles of the data. The variability range is distinctly revealed on this plot.

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, uncertainty is mostly related to the decision that a geostatistician or a geologist makes about the training image itself. Such uncertainty originates from the knowledge or understanding of the geologist about the reservoir structure under study (or a lack thereof). The uncertainty behind the geological scenario can be taken into account by using a set of training images instead of one.

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 proposed method 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 generated realizations is of less relevance . As stated before, the better the patterns are reproduced, the smaller the variability is. This is because of the limited set of patterns existing in the training image. On the other hand, a method that does not entirely capture the features within a training image results in some patterns in the simulated realization



(a) Full MDS variability



(b) Quantile MDS limits

Figure 39: 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.

that are not in accordance with the provided conceptual geological scenario. Hence, such increase in uncertainty/variability is merely an artifact that is not controlled or motivated by any geology. It can be understood that the larger uncertainty in filtersim is caused by a set of 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 postulation is 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 40. 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 has got no control on the implicit training images used to generate these random patterns. Therefore, one of the main purposes in using distance based methods and kernel mapping is to acquire the capability for generating new patterns that are constrained to the provided geological scenario in the training image (ongoing research).

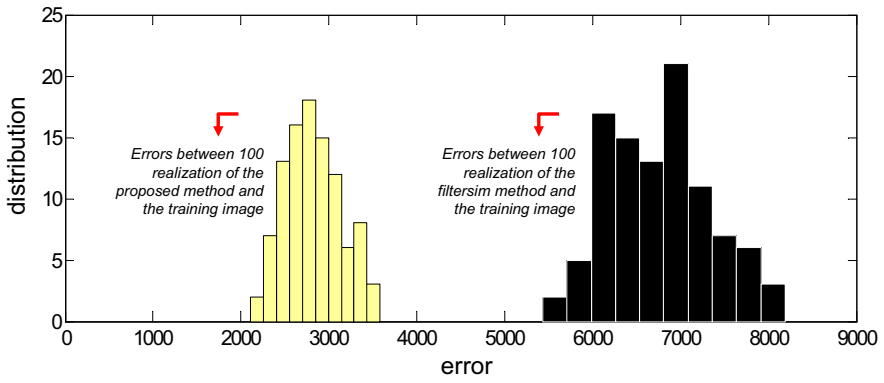


Figure 40: The error between the realizations generated by the proposed method (yellow) and filtersim method (black) with respect to the training image in terms of pattern reproductivity. The error is defined by $error = JSD(\mathrm{MPH}_{realization}, \mathrm{MPH}_{Training\,Image})$.

7.6 Multiple-Resolution Simulation

The idea of multiple-grid simulation was first proposed by [25] in order to reduce the CPU and memory demand of sequential simulations, and at the same time, help is the reproduction of long range correlations. According to [1], the multiple-grid view of a grid $\mathbf{G}$ is defined by a set of cascading coarse grids $\mathbf{G}^g$ and sparse templates $\mathbf{T}^g$ instead of a single fine grid and one large dense template where $g = 0, \ldots, n_g - 1$ and n_g is the total number of multiple-grids for grid $\mathbf{G}$.

There are some shortcomings inherent to 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 and will result in complexity/randomness in the generated realization at that multiple-grid simulation. More complexity suggests less structure. One solution to this problem was 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 settings. For example, a particular multiple grid $\mathbf{G}^g$ might not include the location containing the conditioning data. Moving 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 does not change significantly during the subsequent multiple-grid simulations. Besides, the search for the dual template that honors the hard data is of limited scope and situation where there is no exact match can easily arise. A new method, called multiple-resolution simulation is proposed that can reduce the existing limitations.

In order to effectively reproduce the large-scale and fine-scale pattern structures of a training image, the simulation has to be performed at different resolutions. First, a coarse resolution realization will be obtained and the simulation continues towards the finest resolution. The multiple-resolution views of the training image needs to be constructed. Therefore, the training image will be resized using bicubic interpolation. In a categorical case, it also needs to be thresholded to obtain a categorical coarse-resolution training image. In order to find a suitable threshold, Otsu's method will be employed [30]. This algorithm assumes that the training image to be thresholded contains two classes of nodes (e.g. foreground and background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal. The simulation proceeds with this low-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, and as before, the training image will also be resized to the resolution in that stage. Simulation continues until all multi-resolution levels are finished. Data conditioning will now consist of applying the multi-resolution concept on the data themselves. It is noteworthy that there is no need for a dual template in this case because the all the nodes in the simulation grid will be always filled. Hence, the multi-resolution concept brings consistency and robustness in MPS simulations. The multi-resolution approach for the unconditional case is clearly outlined in Algorithm 4.

The multi-resolution concept is examined by the 2D sand/shale training image. Three multi-resolution settings are chosen for the analysis. The training image (TI) and the two other multi-resolutions (TI^2, TI^3) are illustrated in Figure 41. 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.

Some simulations using the multi-resolution concept and multiple-grid concept are pro-

Algorithm 4 Unconditional Multi-Resolution Simulation

Require: n^r = number of multi-resolutions
Require: TI = Training Image
Require: $\mathbf{T}$ = search Template
for $r = n^r$ to 1 **do**
  $TI^r \leftarrow$ resize TI by a factor $\frac{1}{r}$ using bicubic interpolation
  $level \leftarrow$ find the threshold of TI^r using Otsu's method
  Threshold TI^r by $level$ towards a binary values of 0/1.
  if $r == n^r$ **then**
    Initialize realization $\mathbf{re}^r$ with the same size as TI^r
  else
    realization $\mathbf{re}^r \leftarrow$ resize $\mathbf{re}^{r+1}$ by a factor of $\frac{r+1}{r}$
    $level \leftarrow$ find the threshold of $\mathbf{re}^r$ using Otsu's method
    Threshold $\mathbf{re}^r$ by $level$ towards a binary values of 0/1.
  end if
  using template $\mathbf{T}$ and training image TI^r, simulate the realization $\mathbf{re}^r$
end for

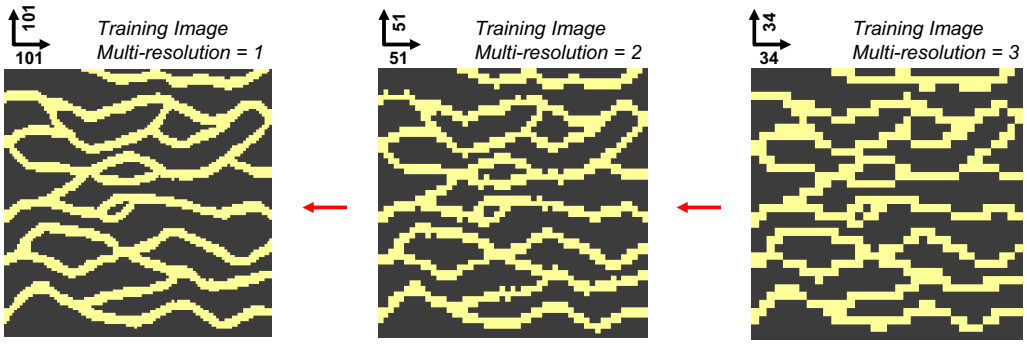


Figure 41: Multi-resolution training images used for each stage of the coarse to fine-scale simulations.

vided in Figure 42 for comparison. It is difficult to perfectly evaluate the structure/pattern reproduction in different realizations, but undoubtedly multi-resolution concept has comparatively enhanced the large-scale continuity and pattern reproductivity. On the other hand, the structures with less repetitiveness than the others are skillfully captured and reproduced in the multi-resolution approach. 200 realizations are generated using multiple-grid and multiple-resolution approaches and their multiple-point histogram differences with the training image are obtained using a template of 4×4. The meandering 2D training image as shown in Figure 43(a) is used in this study. The resulting error distributions are plotted in Figure 43(b) for both multiple-grid (red) and multiple-resolution (black) simulations (using the proposed methodology outlined in this paper). However, the distribution of the errors in filtersim realizations, with multiple-grid approach, is also shown in 43(b) in yellow. As expected, the multiple-resolution approach better reproduces the underlying patterns of the training image within the specified multiple-point template. One can also verify the improvements, in terms of pattern reproduction, brought by the proposed

methodology. These promising results need to be supported by using different training images in the future. More examples are needed to test the effectiveness of multiple-resolution approach on data conditioning.

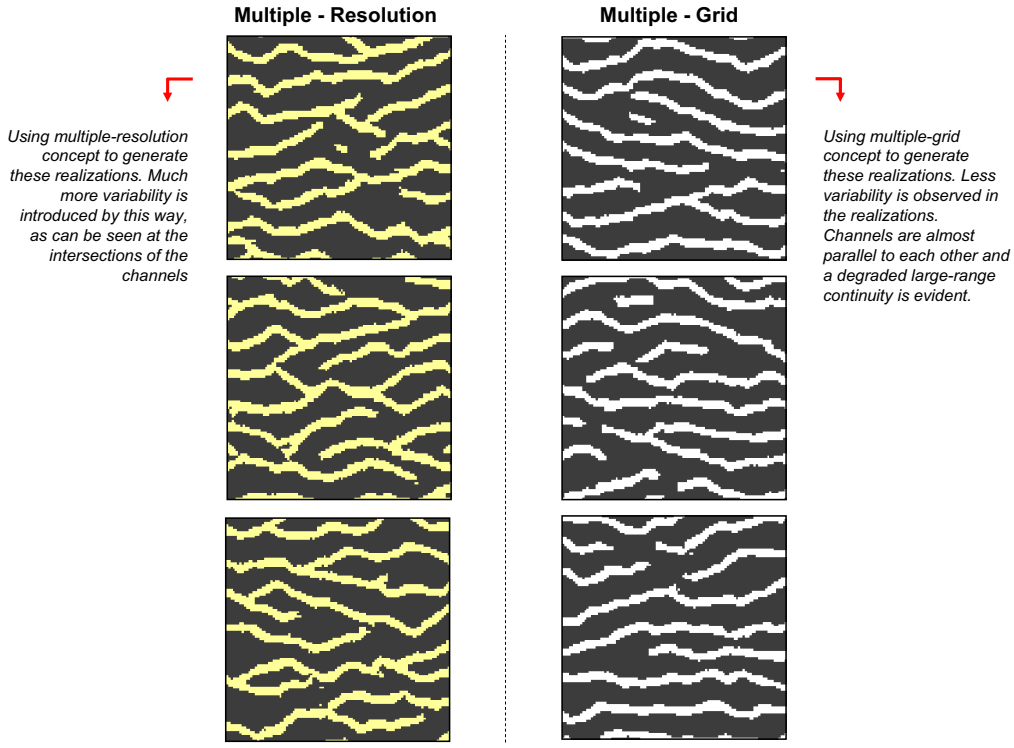
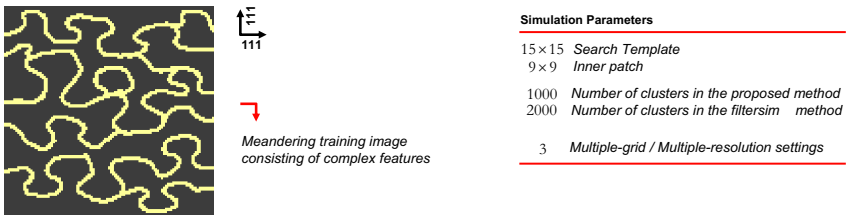
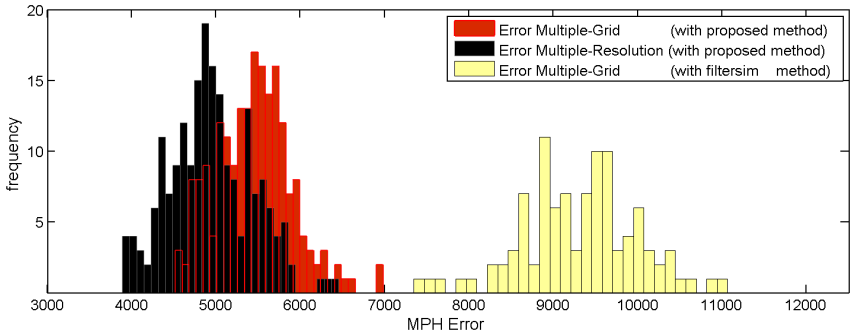


Figure 42: Comparison of multiple-resolution simulation with multiple-grid simulation over 3 generated realizations.



(a) Training Image



(b) Error distribution

Figure 43: Multiple-pint histogram error distribution between 200 realization 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.

8 Conclusion

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 when applied to sequential pattern simulation. However, their advantages over pixel-based methods are not well explained and understood. 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.

So far, a new multiple-point geostatistical algorithm using distance-based method and kernel mapping has been developed. This technique broadens the capabilities of a multiple-point simulation process. Generating new patterns, reducing the dimensionality of the patterns, providing a universal classifier are some examples introduced within this technique. Specifically, the better pattern classification capability of the proposed method is one of the main advantages provided by distance-based methods. The classification improvement was demonstrated with some examples. Also, a series of unconditional simulations were performed on different training images and the improvement in pattern reproduction and

large-scale continuity was observed in both categorical and continuous cases. Hard data conditioning was also examined by obtaining an E-type of 100 realizations. The enhanced data conditioning established a promising future for this method. An algorithm on automatic template size selection was also introduced and exhaustively tested for its accuracy. This combined with a multiple-resolution simulation algorithm provides a robust and user-free methodology that captures the essential features of any conceptual training image. In conclusion, the proposed method provides a global improvement over all previous MPS algorithms such as snesim, simpat and mainly filtersim.

References

[1] Arpat, G. B., *SIMPAT: stochastic simulation with patterns*. Stanford Center for Reservoir Forecasting, 17th SCRF Meeting, 2004.

[2] Bentley J. L., *Multidimensional binary search trees used for associative searching*. Communications of the ACM, 18:509-517, 1975.

[3] Borg I., Groenen P.J.F., *Modern Multidimensional Scaling*. New York, Springer-Verlag, 1997.

[4] Calinski, R.B., Harabasz, J., *A dendrite method for cluster analysis*. Communications in Statistics, 3, 1-27, 1974.

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

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

[7] Milligan, G.W., Cooper, M.C., *An examination of procedures for determining the number of clusters in a data set*. Psychometrika, 50, 159-179, 1985.

[8] Russ J.C., *Image Processing Handbook*. CRC Press, Boca Raton, Florida, 2nd edition, 1995.

[9] Scholkopf B. and Smola A. J., *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

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

[11] Shawe-Taylor J. and Cristianini N., *Filter-based training pattern classification for spatial pattern simulation*. PhD thesis, Stanford University, Stanford, CA, 2006.

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

[13] Buja A, Swayne DF, Littman M, Dean N, Hofmann H, *XGvis: Interactive data visualization with multidimensional scaling*. Journal of Computational and Graphical Statistics, 2001.

[14] Lehoucq, R.B. and D.C. Sorensen, *Deflation Techniques for an Implicitly Re-Started Arnoldi Iteration*. SIAM J. Matrix Analysis and Applications, Vol. 17, pp. 789-821, 1996.

[15] Zhu, M. and Ghodsi, *Automatic Dimensionality Selection from the Scree Plot via the Use of Profile Likelihood*. Computational Statistics and Data Analysis, 2006.

[16] Bentley,, Jon Louis, *Multidimensional divide-and-conquer*. Commun. ACM, Vol. 23, number 4, 1980.

[17] Alsabti, K., Ranka, S. and Singh, V., *An efficient k-means clustering algorithm*. IPPS/SPDP Workshop on High Performance Data Mining, EEE Computer Society Press, 1998.

[18] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman R. and Wu, A. Y., *The analysis of a simple k-means clustering algorithm*. ACM Symposium on Computational Geometry, pp. 100109, 2000.

[19] Honarkhah, M., Caers, J., *Classifying existing and generating new training image patterns in kernel space*. SCRF affiliate meetong, 2008.

[20] Girolami, M., *Mercer Kernel-Based Clustering in Feature Space*. IEEE Transactions on Neural Networks, Vol. 13, No. 3, 2002.

[21] Deutsch, C.V., Gringarten, E., *Accounting for multiple-point continuity in geostatistical modeling*. 6th International Geostatistics Congress, Geostatistics Association of Southern Africa, 2000.

[22] Cover TM., Thomas JA., *Elements of information theory*. Wiley, New York, 1991.

[23] Shannon, C.E., *A Mathematical Theory of Communication*. Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, 1948.

[24] Dujardin B., Wu J., Journel A., *Sensitivity analysis on filtersim*. Stanford Center For Reservoir Forecasting, SCRF affiliate meeting, 2006.

[25] Tran T., *Improving variogram reproduction on dense simulation grids*. Computers and Geosciences, 20(7):11611168, 1994.

[26] Scheidt, C., Caers, J., *Representing spatial uncertainty using distances and kernels* . Mathematical Geosciences, 2008.

[27] Lyster, S., *Simulation of Geologic Phenomena Using Multiple-Point Statistics in a Gibbs Sampler Algorithm*. University of Alberta, PHD Thesis, 2009.

[28] Remy, N., Boucher A., Wu J., *Applied Geostatistics with SGeMS: A User's Guide*. Cambridge University Press (New York) , 2008.

[29] Wu J., *4D Seismic and Multiple-Point Pattern Data Integration Using Geostatistics*. Stanford University, PHD Thesis, 2007.

[30] Otsu N., *A threshold selection method from gray-level histograms*. IEEE Trans. Sys., Man., Cyber. 9: 6266. doi:10.1109/TSMC.1979.4310076, 1979.

[31] Scheidegger, Adrien E., *Morphotectonics*. Berlin, New York: Springer. p. 113., ISBN 3540200177, 2004.

[32] Honarkhah, M., Caers, J., *Stochastic Simulation of Patterns using Distance-Based Pattern Modeling*. SCRF affiliate meetong, 2009.