Published in Stochastic Environmental Research and Risk Assessment 32, 1375-1396, 2018
which should be used for any reference to this work 

# Random partitioning and adaptive filters for multiple-point stochastic simulation

**Mansoureh Sharifzadehlari[1,2] · Nader Fathianpour[3] · Philippe Renard[2] · Rassoul Amirfattahi[1]**

**Abstract** Multiple-point geostatistical simulation is used to simulate the spatial structures of geological phenomena. In contrast to conventional two-point variogram based geostatistical methods, the multiple-point approach is capable of simulating complex spatial patterns, shapes, and structures normally observed in geological media. A commonly used pattern based multiple-point geostatistical simulation algorithms is called FILTERSIM. In the conventional FILTERSIM algorithm, the patterns identified in training images are transformed into filter score space using fixed filters that are neither dependent on the training images nor on the characteristics of the patterns extracted from them. In this paper, we introduce two new methods, one for geostatistical simulation and another for conditioning the results. At first, new filters are designed using principal component analysis in such a way to include most structural information specific to the governing training images resulting in the selection of closer patterns in the filter score space. We then propose to combine adaptive filters with an overlap strategy along a raster path and an efficient conditioning method to develop an algorithm for reservoir simulation with high accuracy and continuity. We also combine image quilting with this algorithm to improve connectivity a lot. The proposed method, which we call random partitioning with adaptive filters simulation method, can be used both for continuous and discrete variables. The results of the proposed method show a significant improvement in recovering the expected shapes and structural continuity in the final simulated realizations as compared to those of conventional FILTERSIM algorithm and the algorithm is more than ten times faster than FILTERSIM because of using raster path and using small overlap specially when we use image quilting.

**Keywords** Multiple-point · Principal component analysis · Geostatistics · FILTERSIM

✉ Mansoureh Sharifzadehlari
m.sharifzadeh@ec.iut.ac.ir

[1] Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran

[2] Center of Hydrogeology and Geothermics, University of Neuchâtel, Rue Emile Argand 11, 2000 Neuchâtel, Switzerland

[3] Department of Mining Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran

## 1 Introduction

Multiple-point statistics (MPS) simulation is one of the most recent reservoir modeling techniques. It had an important impact because of its ability to reproduce complex geological patterns that cannot be modeled by two-point statistics moments (i.e., variograms). MPS has been used in many applications like reconstruction of porous media (Zhang et al. 2016), stochastic reconstruction of spatial data (Zhang et al. 2017a) or geostatistical simulation of a petroleum reservoir (Carvalho et al. 2016).

From an historical perspective, the original MPS algorithm proposed by Guardiano and Srivastava (1993) has been impractical because the algorithm needed to scan the whole training image every time a grid node had to be simulated and consequently, it was extremely CPU demanding. SNESIM (Strebelle 2002) was the first efficient algorithm. This technique is now used in many fields of application such as hydrogeology (Renard 2007; Michael et al. 2010), oil industry (Strebelle et al. 2002; Aitokhuehi and Durlofsky 2005), or inverse modeling

(Caers and Hoffman 2006; Mariethoz et al. 2010; Alcolea et al. 2009). But SNESIM may require large amounts of RAM and can be computationally demanding for the simulation of large 3D reservoirs with many facies. To solve these issues, Strebelle and Cavelius (2013) propose a technique to optimize the template size. In any case, SNESIM is not designed for continuous variables and for the co-simulation of multiple variables (Strebelle and Cavelius 2013). One of the main research avenue which has been investigated to overcome the limitation of SNESIM, was the simulation of patterns instead of individual pixels. SIMPAT was one of the first pattern-based MPS method, introduced as an alternative approach for multiple-point conditional probability (Arpat and Caers 2007). In this method, training patterns are classified into clusters based on a distance function. Storing pattern prototype requires much memory. Also, SIMPAT has difficulties for conditioning with hard data (Strebelle and Cavelius 2013). To overcome those difficulties, Honarkhah and Caers (2010) presented an efficient solution using multi-dimensional scaling (MDS) and kernel clustering to represent and classify the patterns efficiently. In the same spirit, Chatterjee and Mohanty (2015) use Principal Component Analysis and kernel clustering to classify the patterns and accelerate the search. While these methods appear to improve the connectivity of the patterns and reduce the CPU requirement, some significant discontinuities in the patterns can still be identified (Tahmasebi et al. 2012; Chatterjee and Mohanty 2015). Another pattern-based approach, FILTERSIM, introduces filters to summarize high dimensional patterns into a filter score space (Zhang et al. 2006). Then, it classifies the patterns into a limited number of classes. The filters are used to reduce the dimension of the problem and help to speed up the simulation. This method can work with categorical and continues variables. FILTERSIM uses six predefined filters, which are not specific to the Training Image. FILTERSIM has been used for different applications such as digital elevation data fusion (Tang et al. 2015) or fullbore image reconstruction (Zhang et al. 2017a, b).

The aim of this paper is to propose several improvements to the initial FILTERSIM technique. The new technique is named Random Partitioning with Adaptive Filters SIMulation method (RPAFSIM). It follows the basic algorithm of FILTERSIM and can deal with both categorical variable, such as rock type, and continuous variable, such as porosity or permeability.

In contrast to FILTERSIM, RPAFSIM uses by default some filters which are designed specifically for any given Training Image using Principal Component Analysis (PCA). This idea was already envisioned by Zhang (2006) in his PhD and mentioned by Wu et al. (2008) but this was not tested and implemented by default to derive the filters. We note that PCA was also used in previous applications related to MPS simulations. For example, Sarma et al. (2008) use kernel principal component analysis (KPCA), which is a nonlinear generalization of PCA (Scholkopf et al. 1998; Scholkopf and Smola 2002) is used to propose a differentiable parameterization of non-Gaussian random fields (characterized by multipoint geostatistics). But the method is used only to solve an inverse problem using a gradient-based algorithm and not to simulate stochastic realization set al.

In addition, RPAFSIM uses a combination of raster path and random partitioning. The raster path was used in the past (Mattoccia et al. 2008; Tahmasebi et al. 2012; Gardet et al. 2016) and generally resulted in simulations showing a good continuity of the patterns. The random partitioning is a new feature that was never applied to our knowledge. It allows to better explore the space of uncertainty as compared to the straightforward use of the raster path.

Finally, the use of raster path is known to create difficulties and artefacts when producing conditional simulations. To overcome a part of these issues, RPAFSIM includes a new, very simple, and efficient hard data conditioning method. The technique is based on a pre-processing step. In addition to the previous improvements, an optional intermediate step using image quilting can be applied. Image Quilting (IQ) was developed by Efros and Freeman (2001) to synthesize textures. IQ uses reduces the overlap error by removing vertical and horizontal artifacts (Mahmud et al. 2014). Combining RPAFSIM with IQ brings several important advantages together: working on an efficient low dimensional space and at the same time using a raster path which accelerates significantly the simulation while IQ improves connectivity.

The paper is organized as follows. Section 2 summarizes the background information related to the FILTERSIM and Image Quilting algorithms. Section 3 provides a detailed description of RPAFSIM and the new conditional algorithm. Section 4 presents some results and a discussion. Finally, Sect. 5 is the conclusion.

## 2 Background information

In this section, we summarize the main features of the previously published algorithms which constitute the base of RPAFSIM.

### 2.1 FILTERSIM

The core algorithm of RAFSIM is based on FILTERSIM (Zhang et al. 2006). In this algorithm, a search template T, having a square shape, is used to extract all the patterns from the Training Image (Zhang et al. 2006; Wu et al.

2008). Usually six fixed filters are defined, they are matrices having the same size as the search template T and they contain a set of weights associated to any location of the search template. The application of a filter on a pattern consists in multiplying all the pixels in the pattern by the weight corresponding to that position and adding all those values. These results in a weighted value called score value. By applying the six filters, the data dimension of the patterns is reduced to the number of filters (6 for a 2D training image and 9 for a 3D training image). Then, the filter score space is partitioned using the fast cross-partitioning method and similar patterns are grouped together. The cross-partition algorithm is fast but the patterns are classified according to single score similarity, not similarity according to all scores together (Wu 2007). Once the classes are defined, each pattern group (or class) is represented by a pattern prototype, which is the point-wise average of all training patterns falling into that class. For the simulation, a random path is used. In each point of the grid, a template of size T, is used to extract the conditioning data event. The prototype closest to that data event, based on a distance function, is found. A pattern is randomly selected from the prototype and pasted in the simulation grid. The inner part of the pasted pattern is frozen as hard data and will not be visited any more during the simulation. For hard data conditioning, FILTERSIM uses some weights associated to each template node.

### 2.2 Image quilting

Image Quilting is a technique that was proposed in the field of texture synthesis by Efros and Freeman (2001). It has been applied and extended to the framework of MPS by Mahmud et al. (2014). The main idea of Image Quilting (IQ) is to minimize the boundary cut error when to patterns are superposed. In IQ, the error is computed in following manner. With two overlapping blocks $B_1$ and $B_2$ sharing an overlapping region, an error surface is defined as:

$$e = \left(B_1^{ov} - B_2^{ov}\right)^2 \tag{1}$$

To find the minimum error cut path, one has first to compute the map of cumulative minimum error E for all paths as:

$$E_{i,j} = e_{i,j} + \min\left(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1}\right) \tag{2}$$

Then, the minimum value of the last row in E corresponds to the end of the minimum error path. Starting from that pixel, one can track back the minimum value for each row and identify iteratively the minimum error cut path.

$$\min \begin{cases} E_{i-1,j-1}, E_{i-1,j} & \textit{for the last pixel in a row} \\ E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1} & \textit{for all pixels between first and last} \\ E_{i-1,j}, E_{i-1,j+1} & \textit{for the first pixel in a row} \end{cases} \tag{3}$$

Note that in the case of a categorical variable, several paths can be found to have the same minimal error with this method. In this case, one path can be randomly selected or one can solve the problem like Mahmud et al. (2014) by adding a uniform random noise of small magnitude to $e$. The IQ algorithm is used in horizontal or vertical direction according to the orientation of the overlap region.

## 3 The RPAFSIM algorithm

This section describes the RPAFSIM algorithm. It includes four main steps: the selection of the template size, the design of specific filters using Principal Component Analysis, the random partitioning, and the conditioning to hard data.

### 3.1 Template size selection

In all MPS methods, the size of the search template strongly affects the simulated realizations. In a pattern-based approach, the template size should be small to allow flexibility during the simulation and obtain meaningful statistics but it should also be large enough to represent the actual features occurring in a training image and to allow a fast simulation. The ideal size of the template depends on the training image itself. Some images require larger template than others because they contain more complex structures. To select the template size, Honarkhah and Caers (2010) use the mean entropy of the pattern distribution. This requires computing the histogram of the patterns. Here, we propose to replace the mean entropy by the mean variance, which is slightly simpler to compute. The training image is scanned with a template of size $\omega$, all patterns are extracted as shown in Fig. 1 and the mean variance $MV(\omega)$ is calculated.

$$MV(\omega) = \frac{1}{n_{T^\omega}} \sum_{k=1}^{n_{T^\omega}} variance\left(pat_{T^\omega}^k\right) \tag{4}$$

where $n_{T^\omega}$ represents the number of patterns that can be extracted with the particular template size $\omega \times \omega$ and $pat_{T^\omega}^k$ represents the kth pattern with this template size. The operation is repeated for a broad range of $\omega$.

Like the mean entropy, and as discussed by Honarkhah and Caers (2010), the mean variance increases rapidly in the beginning and flattens when increasing the template size does not allow to capture significantly more structures. Therefore, the elbow of that plot is selected as the point that characterizes the optimal template size.

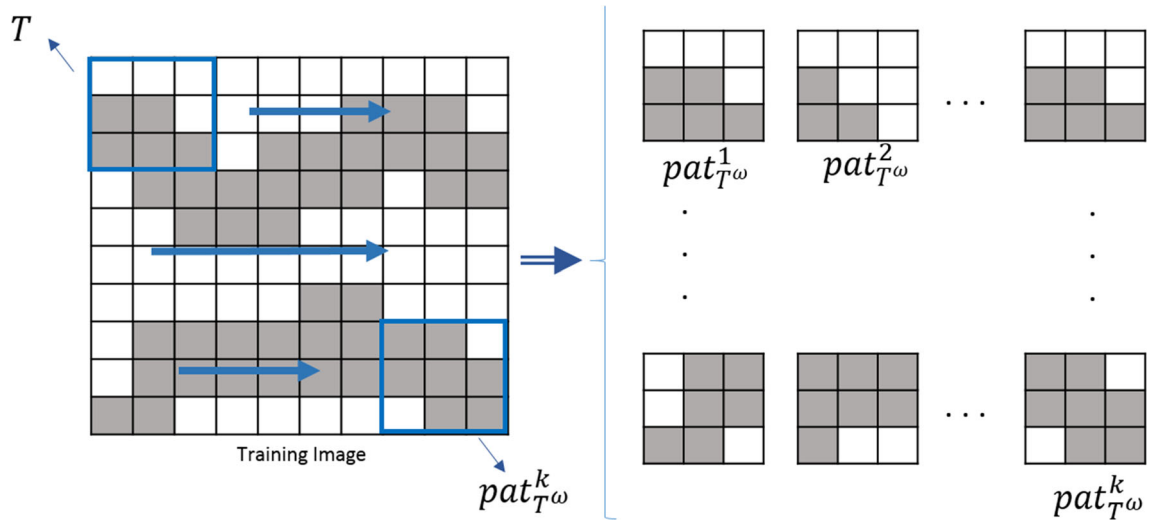



**Fig. 1** Illustration of the extraction of all the patterns $pat^k_{T^\omega}$ of a specific size (here $\omega = 3$) from the training image

Variance calculation is easier and quicker than entropy and can be considered as an alternative method. Figure 2b, c show mean entropy and mean variance for the training image in Fig. 2a. As both methods should be implemented only once for each training image, the speed for running algorithm is not a critical issue. In all algorithms in this paper, the variance method is used to define the template size.

### 3.2 Adaptive filters design

In FILTERSIM, Zhang (2006) uses six fixed and predefined filters to reduce data dimension. In this paper, we use principal component analysis (*PCA*) to design adaptive filters, which are specific to any training image. It allows us to characterize the N-dimensional data in a lower-dimensional space and identify in an unsupervised manner the proper features from the Training Image (Duda et al. 2001). PCA uses the eigenvectors corresponding to the largest eigenvalues of the covariance matrix to select a set of basis functions such that the original signal can be represented by a linear combination of these bases (Fukunaga 2013).

PCA is used in many applications (Jolliffe 1986; Chatterjee and Mohanty 2015) for its simplicity and computational ease. Its advantage is that it can map the data in a new space where the PCs are normal and therefore independent of each other. Since the PCs are extracted by eigenvalue decomposition, the first few principal components can capture the maximum data variability. Therefore, preserving only the first few PCs and eliminating the rest of the PCs can significantly reduce the dimension of the pattern database.

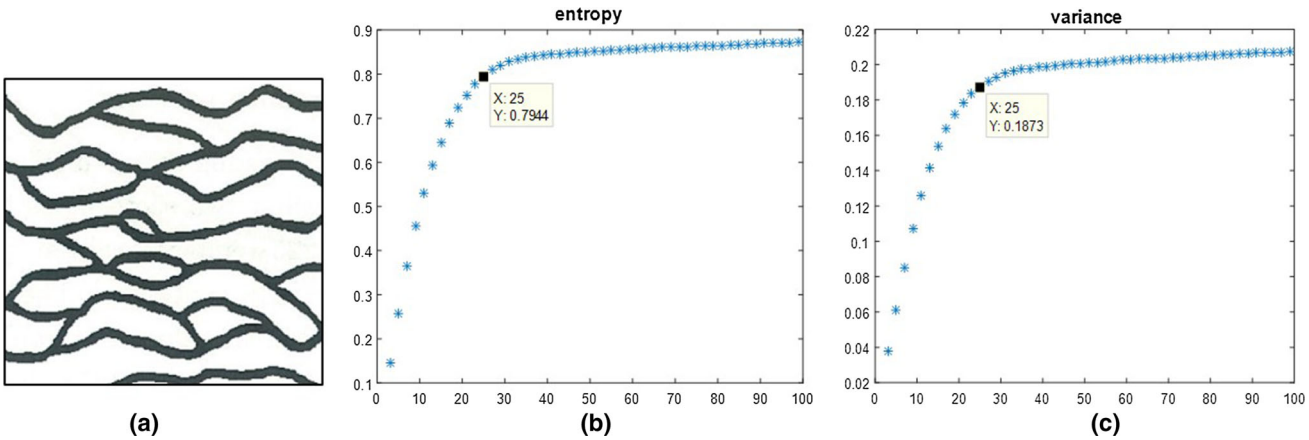



**Fig. 2** Finding the optimal template size. **a** A training image, **b** mean entropy calculation for all template size where *horizontal axis* is template size and *vertical axis* is mean entropy, **c** mean variance calculation for all template size where *horizontal axis* is template size and *vertical axis* is mean variance

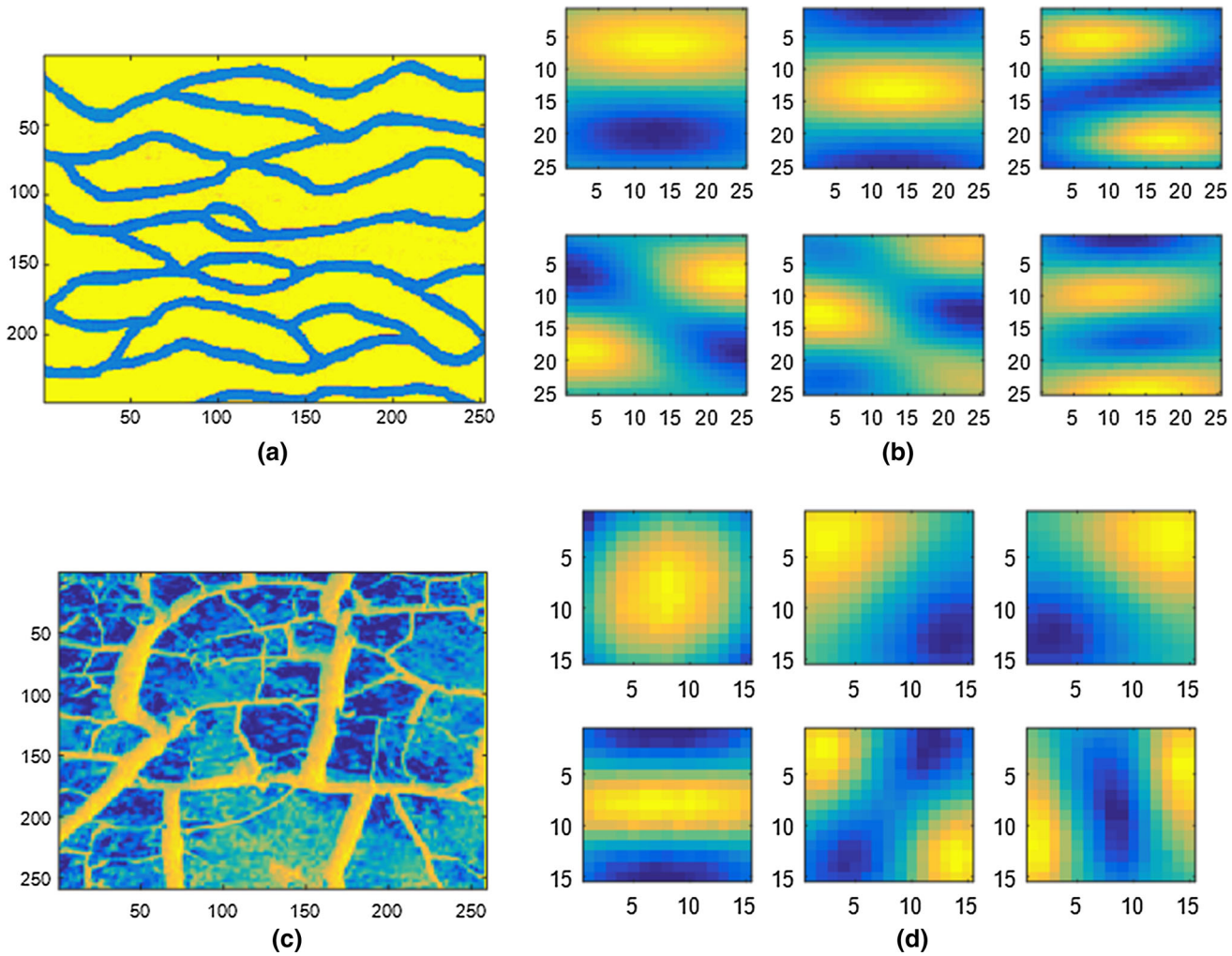



**Fig. 3** Examples of **a** a categorical TI and **b** the corresponding 6 adaptive filters; **c** a continuous TI and **d** the 6 adaptive filters related to it. The six filters are placed in decreasing order from the *top left* to the *bottom right*. The first one (*top left*) corresponds to the first principal component explaining the maximum of the variance. The second is on its *right*, and so on

The basic principle of PCA as applied here for classification of patterns in a training image is the following; let $X$ be the training set of the $N$ patterns extracted from the training image: $X = \{x^{(n)} \in \mathbb{R}^d\}_{n=1}^{N}$. $x^{(n)}$ is a vector containing the values of the $d$ pixels corresponding to the nth pattern. $X$ is then a $d \times N$ matrix with each row being a feature (i.e. the location of a pixel) and each column being a sample (a pattern). The $d \times d$ covariance matrix $C$ is calculated as:

$$C = \frac{1}{N-1}\sum_{n=1}^{N}\left(x^{(n)} - \bar{x}\right)\left(x^{(n)} - \bar{x}\right)^T = \frac{1}{N-1}(X - \bar{x})(X - \bar{x})^T \tag{5}$$

where $\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x^{(n)}$. Then, the eigenvalues $(\lambda_i, i = 1, \ldots, d)$ and eigenvectors $(\vec{e}_i, i = 1, \ldots, d)$ of the covariance matrix are calculated. Each couple of eigenvalues and eigenvector is defined by the following relationship:

$$C\vec{e}_i = \lambda_i \vec{e}_i (\lambda_i, i = 1, \ldots, d) \tag{6}$$

where $\vec{e}_i$ is the $i$ th principal component (PC)

$$\vec{e}_i = \begin{pmatrix} e_{1,i} \\ \vdots \\ e_{d,i} \end{pmatrix} \tag{7}$$

Equation (6) can be written in matrix form as

$$CE = E\Lambda \tag{8}$$

where

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_d \end{pmatrix}, \; E = \left(\vec{e}_1 \vec{e}_2 \ldots \vec{e}_d\right). \tag{9}$$

Note that the eigenvalues in $\Lambda$ are sorted in a descending order. Since the values of $\vec{e}_i$ $(i = 1, \ldots, d)$ are orthonormal, $E^{-1} = E^T$. Therefore, we have

$$C = E\Lambda E^T \tag{10}$$

Finally, the Principal Component (PC) coefficient matrix U is calculated as

$$U = E^T X = \begin{pmatrix} u_{1,1} & u_{1,2} & \ldots & u_{1,N} \\ u_{2,1} & u_{2,2} & \ldots & u_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{d,1} & u_{d,2} & \ldots & u_{d,N} \end{pmatrix} = \begin{pmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vdots \\ \vec{u}_d \end{pmatrix} \tag{11}$$

where $\vec{u}_i = \left(u_{i,1}, u_{i,2}, \ldots, u_{i,N}\right)$ are the $i$th PC coefficients.

In the next step, the first $K$ eigenvectors related to the first $K$ largest eigenvalues are selected and used as the filters. $K$ depends on the complexity of the training images. For comparing our results with FILTERSIM method, $K = 6$ is used in this paper. Figure 3 shows a categorical and a continuous training image and the corresponding adaptive filters. One can see in this figure that even if some filters are similar for the two training images (the second of training image (a) and the 4th of the training image (c) for example) they are in general different and therefore adapted to each specific training image.

Once the filters are selected, they are applied to all patterns to compute filter scores. This process, illustrated in Fig. 4, is the same as in FILTERSIM (Zhang et al. 2006) but with adaptive filters. Using that procedure, any pattern is represented by a vector in a $K$ dimensional space.

### 3.3 Pattern classification

The k-means algorithm is used to classify all patterns in filter score space. K-means classify patterns according to all scores together in contrast to the cross partition algorithm, used in FILTERSIM, witch classifies the patterns according to single score similarity.

After classification, the average of all the patterns belonging to each class is calculated and saved as class prototypes. The number of classes (C) is chosen as a function of the complexity of the training image, usually in the order of a thousand and adapted by the user by trial and error. Note that the choice of proper pattern classes is very important for the next stages of the algorithm, inaccuracy at that stage will induce artefacts in the simulation and should therefore be avoided. To tackle the problem, the advantage of using adaptive filters is revealed when analyzing the results of the classification. Figure 5a shows some patterns belonging to one category after classification when adaptive filters are used. Figure 5b shows the average of all the patterns belonging to that class. It is clear that all patterns are very similar. However, with default filters as used in in the standard FILTERSIM method, the pattern classification can show more variability within a class. As an example, Fig. 6 shows a category with eleven patterns and Fig. 6b is the average of the corresponding patterns.

### 3.4 Random partitioning and simulation strategy

To describe the simulation algorithm, let us first introduce some notations. In the following, the algorithm is explained mainly in 2D to facilitate its illustration. The method works similarly in 3D.

The training image TI contain either a categorical or continuous variable that we will denote $Z$. $G$ is a regular $2D$ grid. The aim is to generate an ensemble of realizations

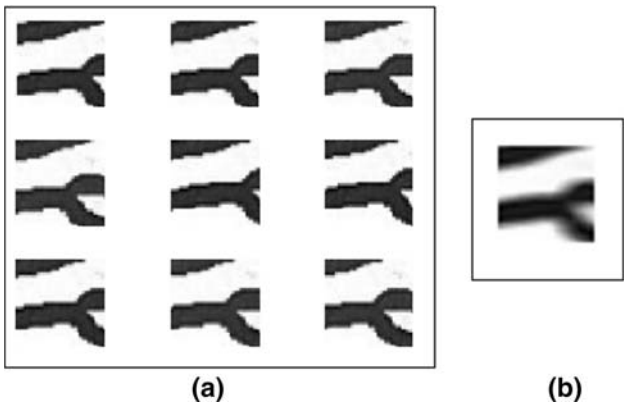



**Fig. 5** **a** Patterns extracted from the TI shown in Fig. 2a belonging to one category after classification with adaptive filters; **b** average of the patterns in this category

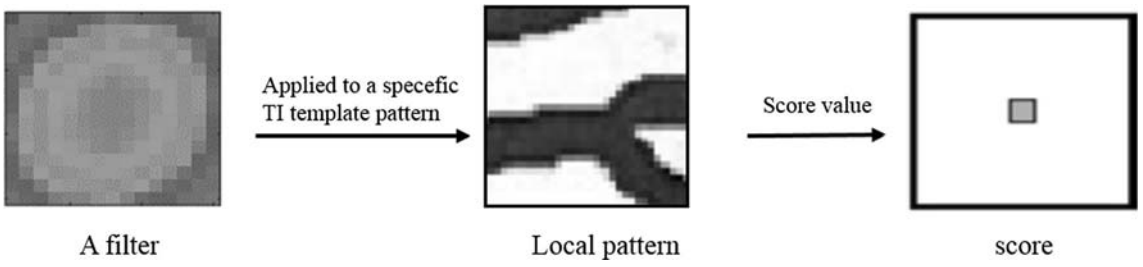



**Fig. 4** Procedure for obtaining the filter score for a pattern

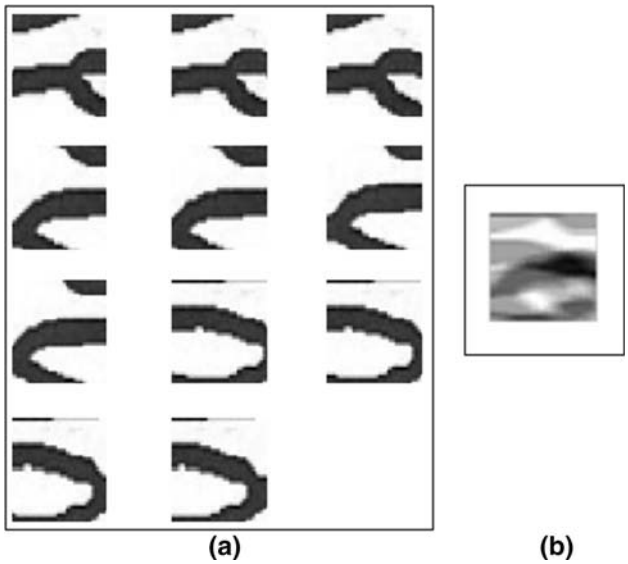



**Fig. 6** **a** Patterns belonging to one category after classification with standard filters used in the FILTERSIM method; **b** average of the patterns in this category

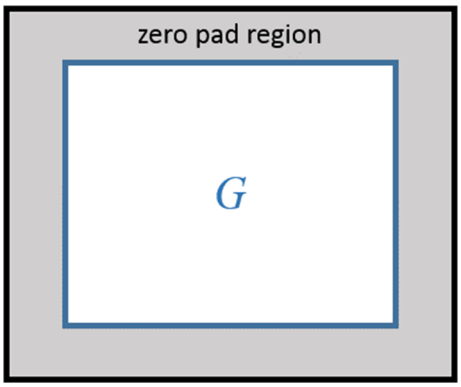



**Fig. 7** DE: zero padding of G

of the variable $Z$ on $G$ such that the structure of the patterns on the TI and on $G$ are similar. The location of a grid node (or a pixel) of $G$ is denoted $\vec{u} = (x, y) \in G$.

$DE$ represents a grid that will be used internally for the simulation. In the basic algorithm, $DE$ will be larger than $G$. It includes a so called *zero-pad region* which extends laterally the grid (Fig. 7) and contains initially some zero values in all those pixels. The size $\varepsilon$ of the extension is half the template size ($\varepsilon = \omega/2$). *Zero padding* means extending $G$ to obtain $DE$. This operation is needed to ensure a good continuity of the patterns during the simulation when we will select patterns on the edge of $G$. When applying random partitioning as explained below, we will decompose $DE$ into subregions.

$de_T\left(\vec{u}\right)$ represents a data-event, it is a vector containing the values of the variable Z that have been previously simulated or that come from conditioning and that fall within a template $T$ centered around $\vec{u}$. $OS$ represents the overlap region between $de_T\left(\vec{u}\right)$ and the previously simulated pattern.

The simulation algorithm proceeds as follows. First, zero padding is applied on $G$ to get $DE$. Then each conditioning data is assigned to the closest grid node in $E$. After this initialization step, the algorithm selects a random location $r(x, y)$, according to an isotropic two-dimensional Gaussian probability distribution (Fig. 8a) in $DE$. The main idea is to partition randomly $DE$ into four non-overlapping regions $\{DE1, DE2, DE3, DE4\}$, as shown in Fig. 8b and then simulate each region separately. For each new realization, the partitioning is different. This allows generating simulations covering a wider space of uncertainty as we will show later in the paper.

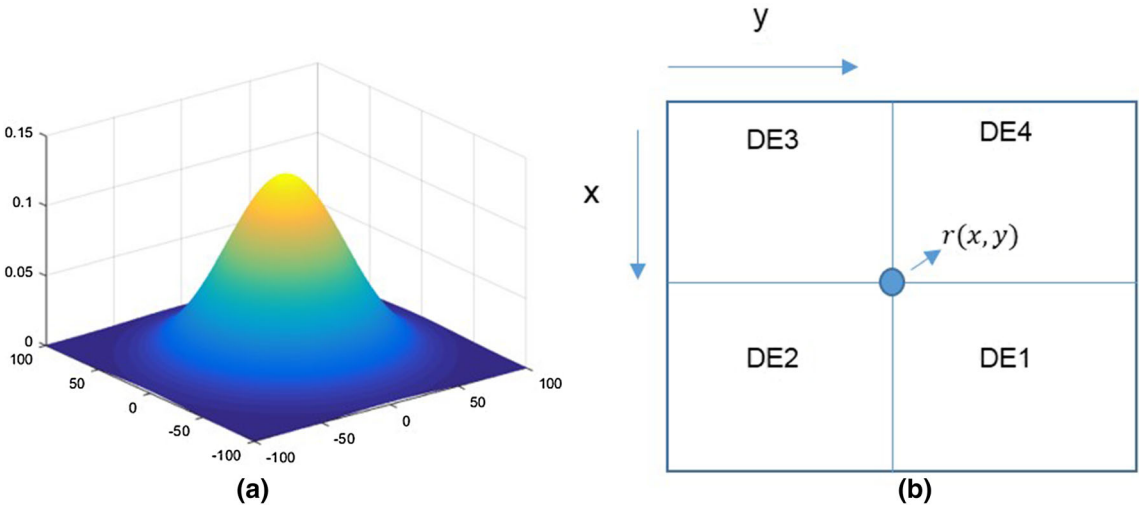



**Fig. 8** **a** Gaussian distribution to select first random data and **b** r(x,y) in De and partition DE to 4 non overlap regions {DE1, DE2, DE3, DE4}

The simulations of each region use a raster path. Experiments with random path showed that the resulting simulations often display losses of continuity in the patterns and take longer computational time. This is due to possible inadequate prototype selection in regions with a few known pixels, resulting in lost continuity, and possible random selection of a region with a few unknown pixels making the prototype useless and time demanding.

Then, for each four regions:

1. Start from $r(x,y)$, and place the template T so that $r(x,y)$ is the central node of T and extract $\mathrm{de_T}\left(\vec{u}\right)$ (Fig. 9).
2. If all pixels in $\mathrm{de_T}(\vec{u})$ are unknown, a random pattern is selected from the data base of patterns, inserted in the place of $\mathrm{de_T}(\vec{u})$ and go to stage 6; else go to 3.
3. In order to select the prototype closest to $\mathrm{de_T}(\vec{u})$, a distance function is used:

$$DT^{c}\left(\vec{u}\right) = \sum_{s=0}^{n_T-1}\left|\mathrm{de_T}\left(\vec{u}+\mathrm{s}\right) - \mathrm{prot^c}\left(\vec{u}+\mathrm{s}\right)\right| \qquad (12)$$

   where $c = 1,\ldots,C$, and $\mathrm{prot^c}$ is c'th prototype. This calculation is performed only for *OS* in $\mathrm{de_T}(\vec{u})$ and thus, the rest of $\mathrm{de_T}(\vec{u})$ is ignored so, the calculation is very fast. Equation (12) can be used for conditional simulation too, but the conditioning procedure will be described in detail later in this paper.
4. When the closest prototype to $\mathrm{de_T}(\vec{u})$ is determined, a *sampled pattern* (*SP*) is randomly picked from the selected prototype class.
5. SP is pasted on $\mathrm{de_T}(\vec{u})$ locations and assembled with the pre-existing conditioning data. In this stage, the user can select one among two possible options. Option one: like in the original FILTERSIM, the inner part of SP will be considered as conditioning data for the next steps. The size of the inner part is determined according to the template size, which is generally $2/3 \times \omega$. Option two: paste all the points of SP in $\mathrm{de_T}(\vec{u})$ as done by Chatterjee and Mohanty (2015) but in this case the results show a lot of disconnectivity and artifacts. To avoid these problems and to ensure the best continuity between the existing conditioning data (previously simulated) and the newly pasted template, we apply the quilting technique that is described in Sect. 2.2. The principle is to find the best boundary (or cut) between the previously pasted data and the newly selected candidate. In all this paper, by default we use the first option as it is the most similar to FILTERSIM. We indicate specifically when we select and test the second option.
6. If the current position is not located along the last raw of $G$, translate the template $T$ by a distance $\varepsilon$ (half template size) along the $y$ direction (see Fig. 9 for the directions); else if the current location is not along the last column, translate T by $\varepsilon$ in the $x$ direction; in both cases select next $\mathrm{de_T}(\vec{u})$; else go to 8.
7. Repeat 2–6 to simulate the region completely.
8. If the algorithm is not in region 4, go to the next region and repeat 1–7 else go to 9.
9. Remove zero pad region from *DE*.

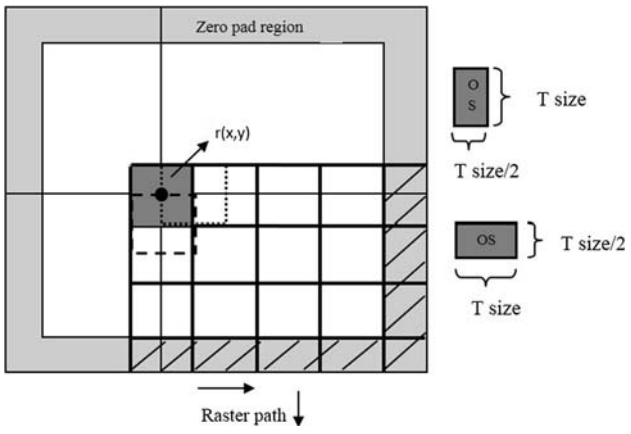



**Fig. 9** Raster path and simulation procedure of region 1

The simulation procedure for region 1 is illustrated in Fig. 9. The dashed regions will not be simulated and it would be removed at the last stage. For each region, the overlapping regions *OS* can be different as shown in Fig. 10.

For 3D simulations, the procedure is identical. A random point is selected in the 3D simulation grid using a 3D gaussian distribution in order to split the domain in eight 3D parts. Each part is simulated separately. All the other steps are identical.

### 3.5 Data conditioning

Conditioning the simulations with punctual data is always a difficult step for patch based MPS methods especially when using a raster path. Here, we propose a new simple technique for conditioning which improves significantly the quality of the results. The procedure consists in one pre-processing step that is applied before launching the general algorithm described in Sect. 3.5.1, and one post-processing step which is used to correct some remaining artefacts only when the optional quilting algorithm is applied.

#### *3.5.1 Pre-processing step*

The pre-processing step is illustrated for a binary case in Fig. 11. First, all the conditioning hard data are assigned to

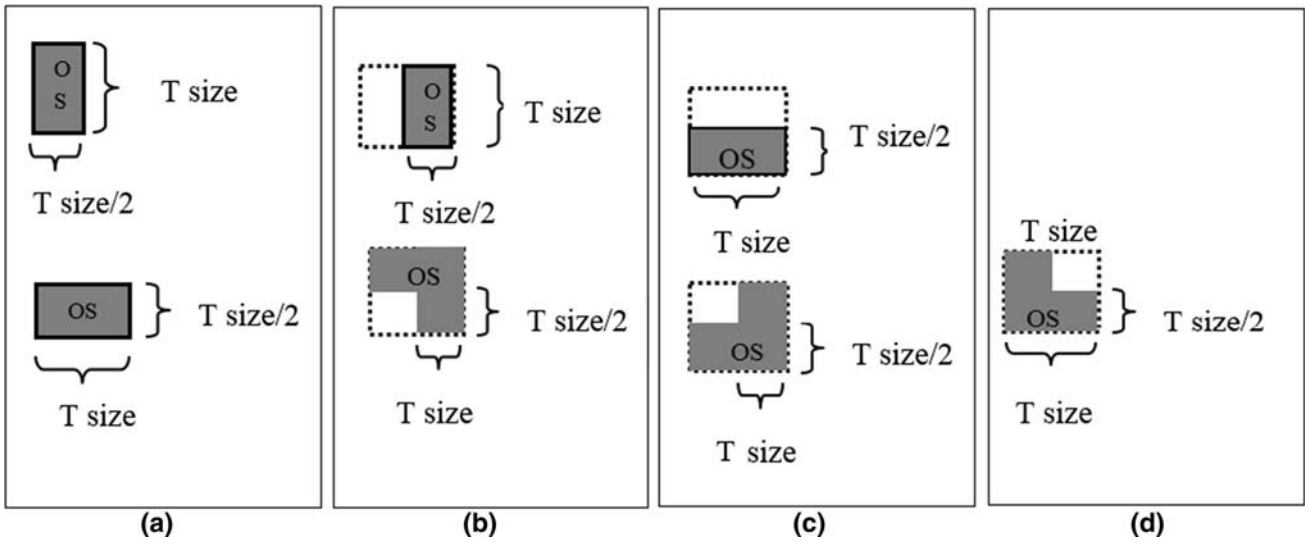



**Fig. 10** **a–d** overlapping regions OS for simulating regions DE1–DE4 respectively

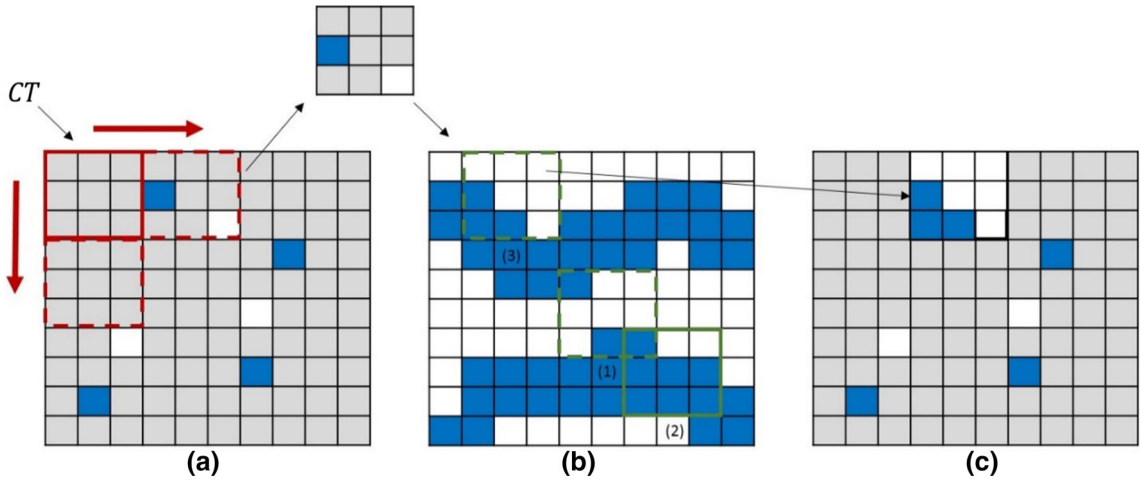



**Fig. 11** Conditioning stage **a** searching *G* with *CT* to find hard data, **b** searching training image to find a best match and **c** pasting the best match in *G*

the closest grid node in *G*. Then, the simulation grid *G* is scanned with a conditional template *CT* of appropriate size to find the hard data. The searching path is a raster path without overlap and it starts from the top left pixel of *G*. The size of *CT* depends on training image and is determined by trial and error.

As soon as a hard data (*HD*) is located within the template (*CT*), the corresponding data event in *CT* is extracted. We then search in a random manner within the TI to find a best match for this data event in *CT* as it is shown in Fig. 11. The use of a random path at this stage allows to increase the variability of the results. The number of hard data in CT is usually moderate and therefore finding a good match in the *TI*, especially for a categorical *TI* with little categories, is rather fast.

For continuous or categorical TI with more than four categories, we first search only for one of the hard data, and then randomly test all patterns near that point to find the best match. This trick, speeds up the process effectively. In these TIs, if we do not find an exact match, we take the best match according to the similarity measure.

As soon as a match is found for CT, the whole pattern in CT is pasted in *G*. This process is continued until the complete simulation grid *G* is treated. At the end of this procedure, all the cells around the hard data will be pre-simulated with patterns compatible with the hard data and the TI. Once this is ready, the simulation procedure described in Sect. 3.4 is applied.

### *3.5.2 Post-processing step*

By construction, and because of the procedure explained above, the conditioning data are generally well honored when applying the proposed algorithm. However, if we

apply the quilting option, the computation of the optimal cut does not treat in a specific manner the conditioning data, and it may happen that some hard data are not honored. In this case, and once the simulation is completed, we apply a final post-processing step to check if all the hard data are properly assigned in all the realizations and to correct the most obvious errors. To quantify the problem, we count the number of mismatches (MM) and calculate the mean percent of HD which are in their right position (PHR) as:

$$PHR = \frac{\sum_{i=1}^{N} (\#\ of\ MM)_i}{N \times H} \times 100 \tag{13}$$

where $(\#\ of\ MM)_i$ is number of mismatch in *i*th simulation and N and H are the numbers of simulations and hard data respectively.

It is remarkable that in almost all the simulations, some mismatch *HD* are located at the edge of a region and it is possible to correct the mismatch point by replacing the value by the neighborhood color. To identify and correct

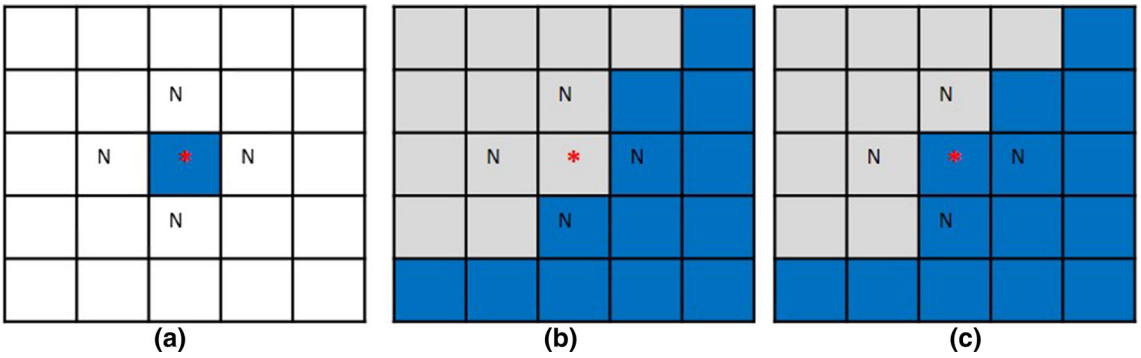



**Fig. 12** **a** A hard data in a location, **b** mismatch simulation of the HD and **c** finding the right HD *color* in one of the 4 neighbors are shown with N

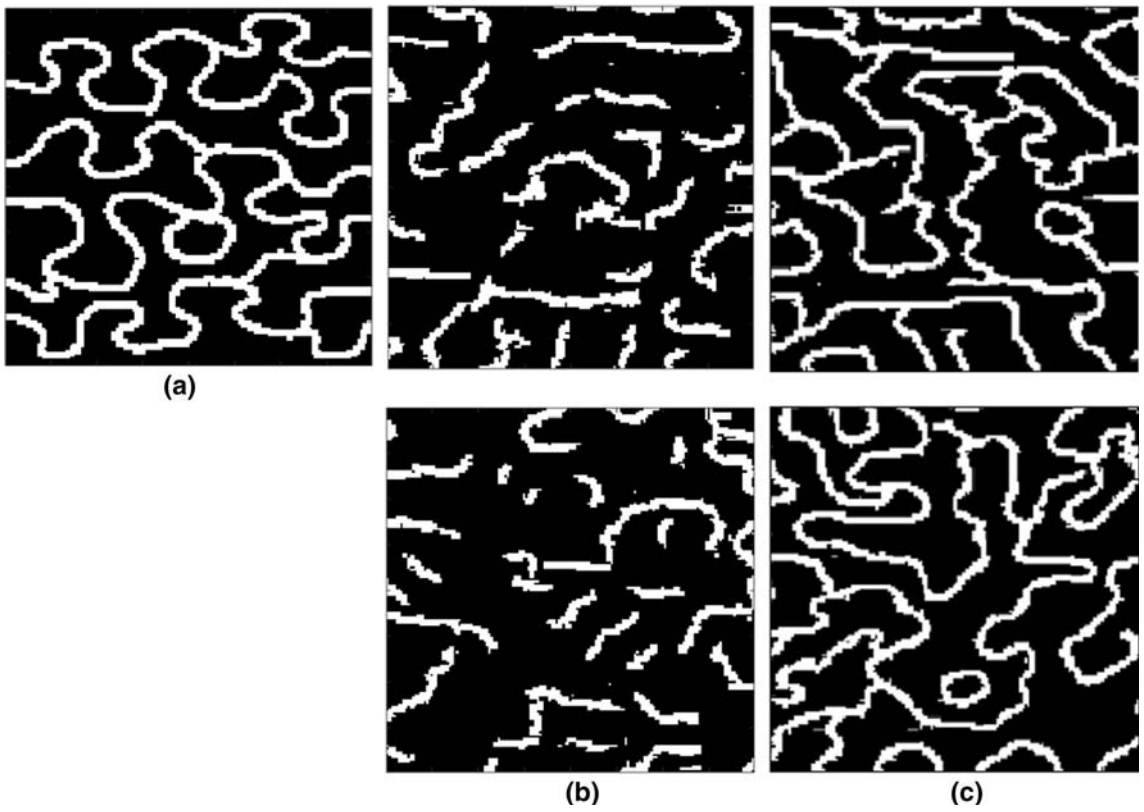



**Fig. 13** Comparison of the FILTERSIM method, with the RPA, **a** quasi-stationary meandering training image, **b** two realization with the FILTERSIM, **c** two realization with the RPAFSIM method

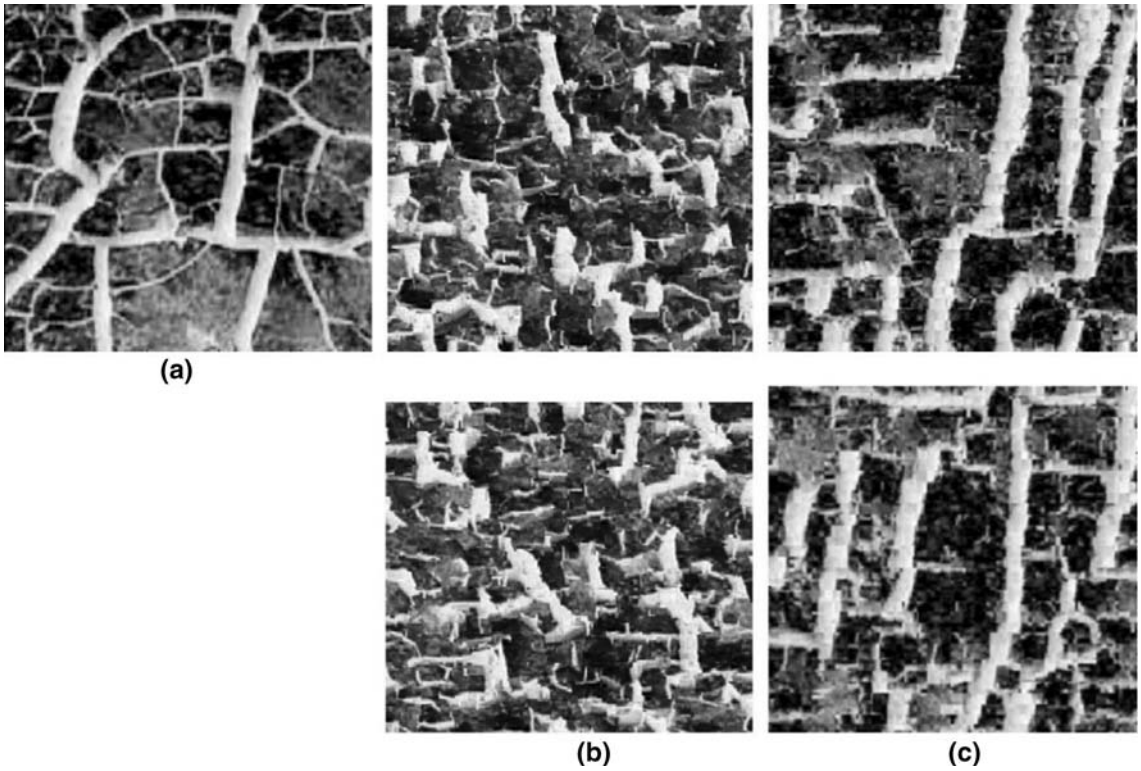



**Fig. 14** Comparison of the FILTERSIM method, with the RPAFSIM for a continuous training image, **a** continuous training image, **b** two realization with the FILTERSIM method, **c** two realization with the RPAFSIM method

**Fig. 15** 3D training image

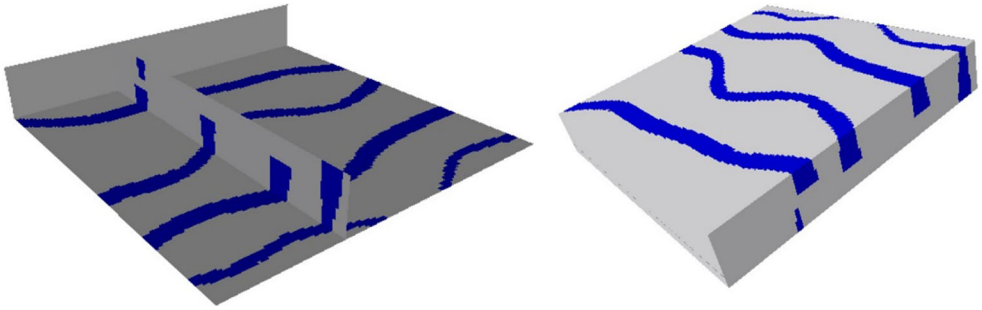
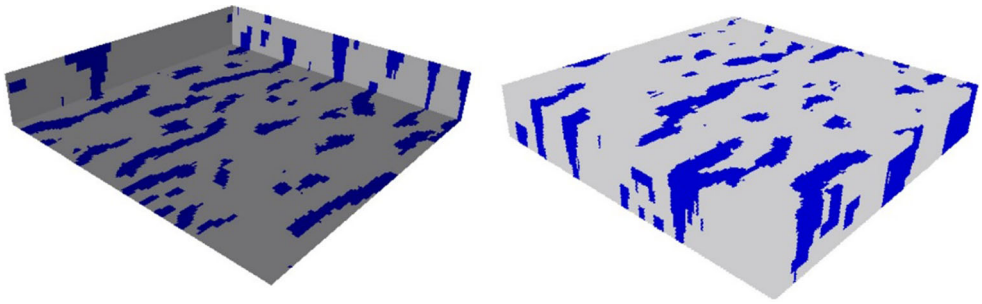


**Fig. 16** A realization of FILTERSIM algorithm for training image of Fig. 15



those situations, we check the 4 neighbors in up, down, right and left directions ($N_d(d = up, down, leftandright)$). For the categorical realizations, if we find the right category (corresponding to the HD) in one of these 4 points, we change the value at the mismatch point to the proper value. The procedure is shown in Fig. 12 for a binary *TI*. In this figure, the location of the mismatch *HD* is shown with a red star and the four neighbors are shown with N.

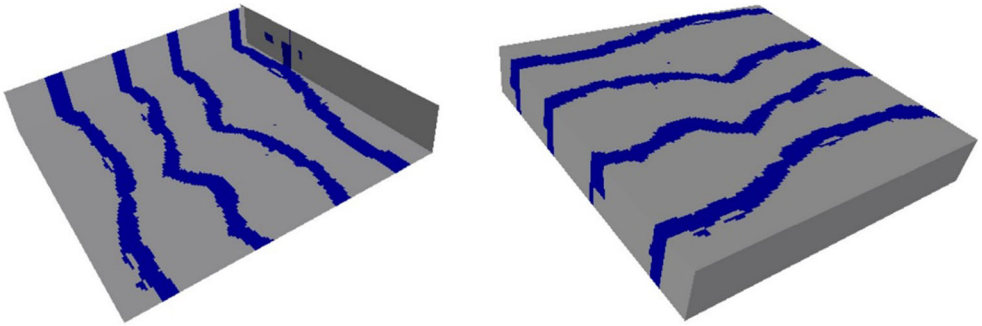


**Fig. 17** A realization of RPAFSIM algorithm for training image of Fig. 15

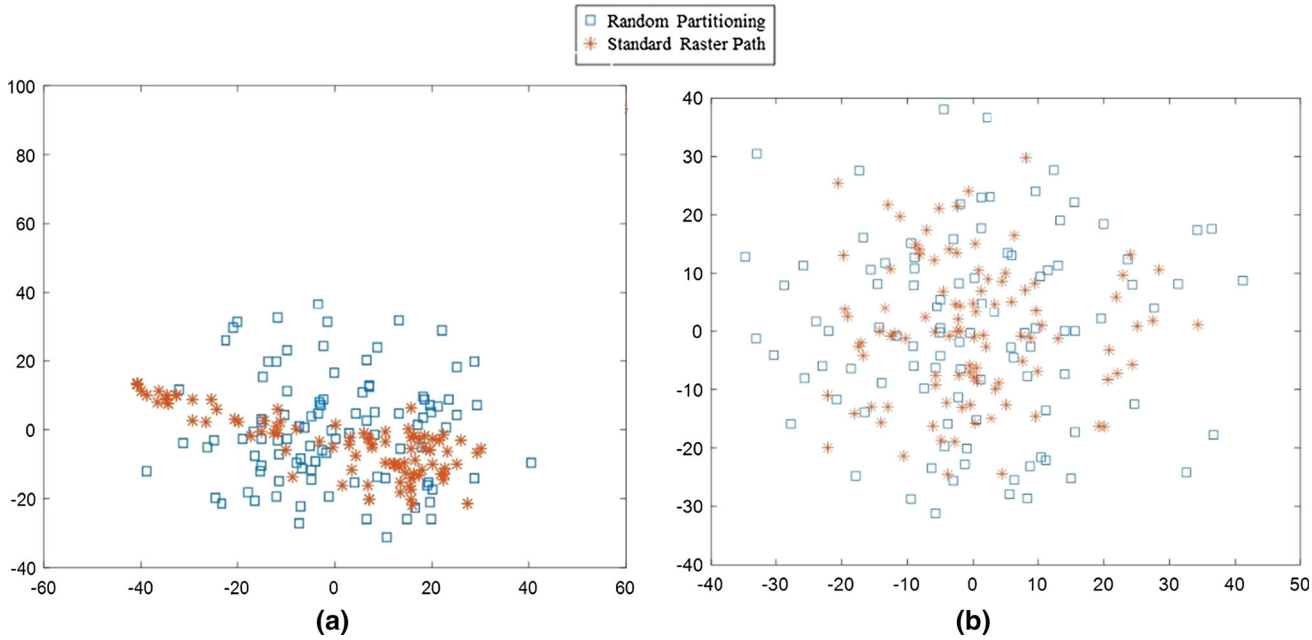



**Fig. 18** MDS of 100 realizations for proposed method with random partitioning and a method with raster path but without random partitioning using **a** Euclidian distance and **b** Modified Hausdorff distance

In the case of a continuous variable, we define a threshold ($TH$) to calculate the mismatch $MM$. PHR can be calculated with Eq. (13), but MM is obtained from:

$$MM = \sum_{i=1}^{H} u[|HD - SP| - TH] \tag{14}$$

where H is the number of $HD$, $SP$ is the simulated value at the location where the hard data should be equal to HD, the vertical bars represent the absolute value and $u[.]$ is the unit function defined as:

$$u(a) = \begin{cases} 1 & if\ a > 0 \\ 0 & if\ a < 0 \end{cases} \tag{15}$$

If $|M - N_d| < TH$, with M the value at the mismatch point and $N_d(d = up, down, leftandright)$ the values in the neighbor locations, change the mismatch point to the corresponding value in the neighborhood. $TH$ can be determined for each TI and chosen by trial and error.

## 4 Results and discussion

In this section, we illustrate the results obtained with the proposed RPAFSIM method, and compare them with FILTERSIM simulations, because of the similarity between the two algorithms, using visual comparison and quantitative metrics.

### 4.1 Unconditional simulation

Hereafter, several different TIs corresponding to different structures are used for comparison.

#### *4.1.1 Examples of simulations results*

*Binary training image* Fig. 13 shows a 159 × 159 quasi-stationary meandering TI. This TI has been used in previous test cases because many algorithms fail to reproduce the complex features—the meanders—often the results are

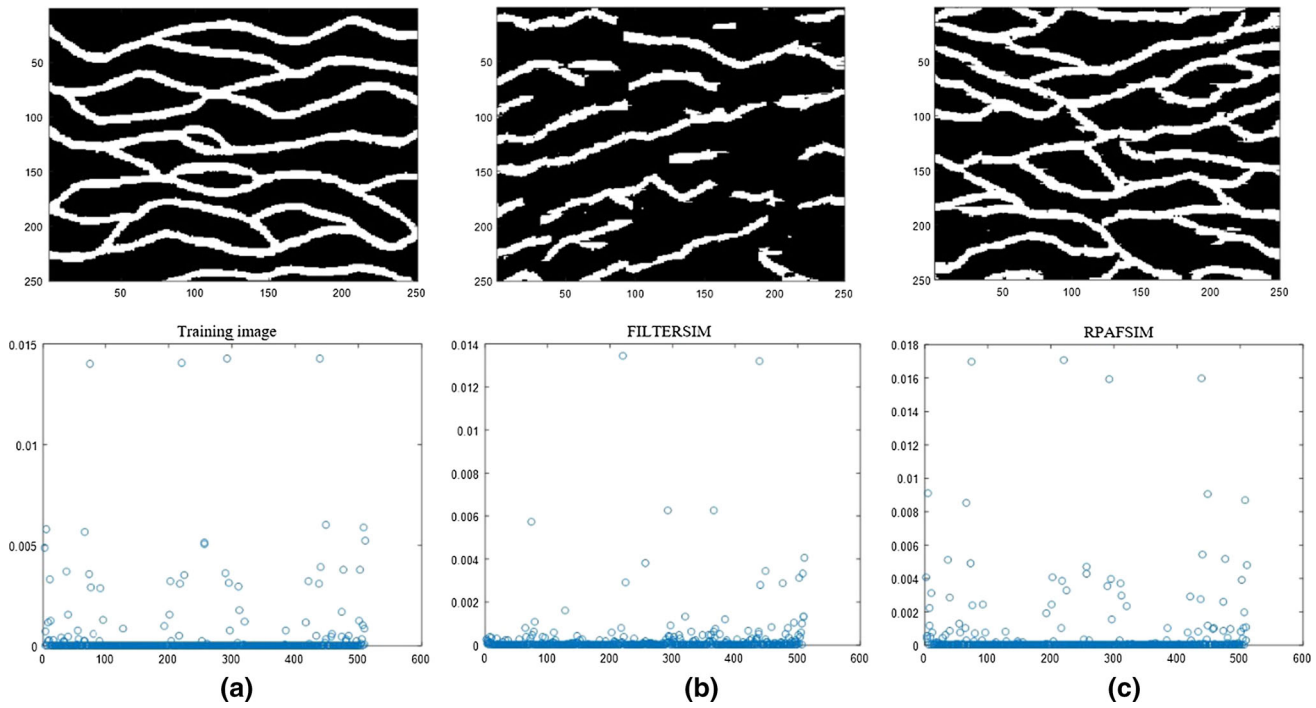



**Fig. 19** A realization of FILTERSIM and RPAFSIM algorithm and multiple point histogram of **a** training image, **b** FILTERSIM and **c** RPAFSIM method with template size 3 × 3

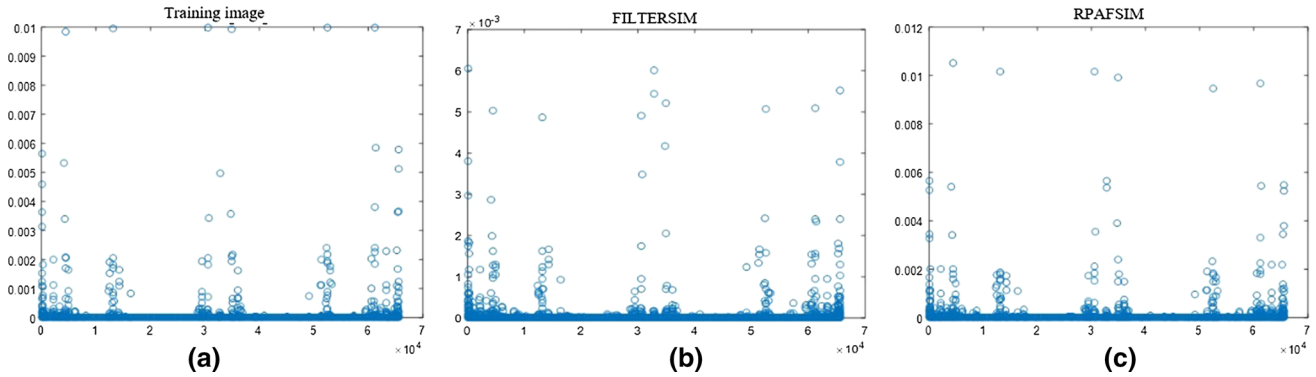



**Fig. 20** Multiple point histogram of **a** training image, **b** FILTERSIM and **c** RPAFSIM method with template size 4 × 4

**Table 1** Mean of multiple point histogram absolute error for the two methods

| The method | Multiple point histogram error for template size 3 × 3 | Multiple point histogram error for template size 4 × 4 | Multiple point histogram error for template size 5 × 5 |
|---|---|---|---|
| RPAFSIM | 4000 ± 1000 | 5300 | $4.5 \times 10^6$ |
| FILTERSIM | 9800 ± 1400 | 20,600 | $12.9 \times 10^6$ |

The mean was computed for 100 realizations with the small template size (3 × 3). For the other template sizes, it has been computed only on one realization

**Table 2** Jenson–Shannon divergence between the multiple point histogram for FILTERSIM and RPAFSIM with different template size for computing Histogram

| Methods | Divergence JS for MP histogram of block size 3 × 3 | Divergence JS for MP histogram of block size 4 × 4 | Divergence JS for MP histogram of block size 5 × 5 |
|---|---|---|---|
| RPAFSIM | 0.024 | 0.044 | 0.062 |
| FILTERSIM | 0.128 | 0.152 | 0.347 |

$D_{JS}$ was computed for 100 realizations with the small template size (3 × 3). For the other template sizes, it has been computed only on one realization

disappointing as shown by Honarkhah and Caers (2010). Two independent realizations generated with RPAFSIM and FILTERSIM with a template T of size 15 × 15 pixels and inter template size 9 × 9 and number of class (C) is 1000 for both methods, are shown in Fig. 13b, c respectively. Visually those results are much closer from the TI than those obtained with FILTERSIM. In particular, the continuity of the channels is better preserved with the proposed method. Note that here first option of part 5 in algorithm procedure is selected and connectivity are good.

*Continuous training image* A continuous TI of size 159 × 159 [obtained from Zhang (2006)], representing a soil with desiccation cracks (thick and narrow) is used for analysis. Figure 14 shows two realizations with the FILTERSIM method and the RPAFSIM method with template size of 15 × 15, inter template size 11 × 11 and $C = 200$. In this specific continuous TI, it is more difficult to decide which realization is better. However, the visual inspection of Fig. 14 gives a slight advantage to RPAFSIM in terms of reproducing the thick structures.

*A 3D training image* A 3D simulation result for 3D training image of Fig. 15 with FILTERSIM and RPAFSIM algorithm is shown in Figs. 16 and 17 respectively. In these results, Template size is 11 × 11 × 11, C = 200 and inter template size = 5 × 5 × 5. Performance of RPAFSIM is better than FILTERSIM for 3D simulation.

### 4.1.2 Effect of random partitioning

We now illustrate the impact of using random partitioning with a raster path within the algorithm. The main advantage of the proposed approach is that it adds some degree of randomness within the method. The consequence is that the algorithms produces realizations covering a wider space of uncertainty. To illustrate that feature, we generated 100 realizations with the proposed method using either random partitioning or the standard raster path without random partitioning. All the other parameters are kept identical, we use the TI of Fig. 2a with template size 25 × 25. We then compare the realizations using the Euclidian distance and Modified Hausdorff distance between all pairs of realizations and represent the simulations in a low dimensional space using multidimensional scaling (Cox and Cox 1994). Figure 18a, b shows the corresponding plots respectively. Each point represents a realization. One can see on these figures that the simulations generated by RPAFSIM with random partitioning (blues squares) are more dispersed than those using the standard raster path (orange stars on the graph) specially it is clearer when Euclidian distance is used.

### 4.1.3 Multiple point histogram

In this section, multiple point histogram (MPH) (Honarkhah and Caers 2010) for FILTERSIM and RPAFSIM are compared. Figure 19 shows a training image and a realization obtained with both FILTERSIM and RPAFSIM methods. The MPH are computed for all of them with block size of 3 × 3 and 4 × 4 as shown in Figs. 19 and 20. The absolute error between histogram of each realization and training image, as a measure of quality, are calculated using Eq. 16. In this equation, $C_k$ is the count of the patterns with configuration index of k and d is number of all patterns with a special block size. Calculation results of MPH for block size of 3 × 3, 4 × 4 and 5 × 5 are shown in Table 1. As it is shown there, MPH of RPAFSIM is better than FILTERSIM.

$$Error = \sum_{k=1}^{d} abs(C_k^{TI} - C_k^{realization}) \tag{16}$$

We also evaluated the difference between the histograms using the Jenson–Shannon (JS) divergence (Endres and Schindelin 2003). JS divergence is based on the

**Table 3** results of $r^{between}$, $r^{within}$ and $r^{total}$ for proposed method with and without random partitioning

| | $r^{between}$ | $r^{within}$ | $r^{total}$ |
|---|---|---|---|
| Proposed method using random partitioning | 0.2189 | 0.1565 | |
| Proposed method *without* using random partitioning | 0.1730 | 0.1651 | |
| Total | 1.2653 | 0.9479 | 1.3348 |

**Table 4** results of $r^{between}$, $r^{within}$ and $r^{total}$ for proposed method and FILTERSIM

| | $r^{between}$ | $r^{within}$ | $r^{total}$ |
|---|---|---|---|
| Proposed method using random partitioning | 0.2189 | 0.1565 | |
| FILTERSIM | 0.1296 | 0.2301 | |
| Total | 1.6890 | 0.6801 | 2.4835 |

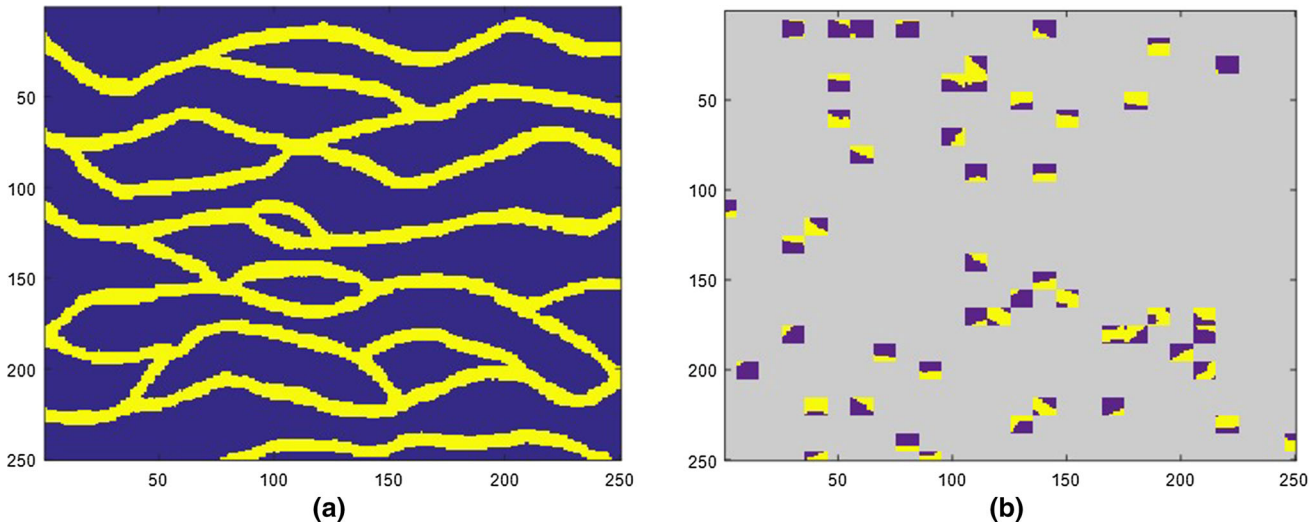



**Fig. 21** **a** Training image, **b** result of conditioning stage for 50 hard data

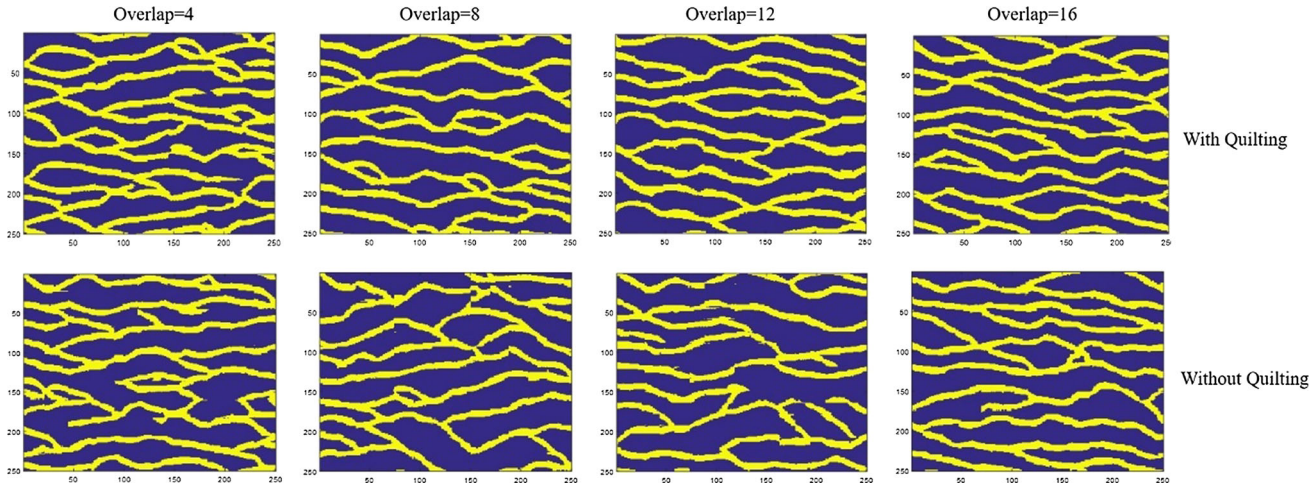



**Fig. 22** Results with varying overlap in two states of using and not using quilting

Kullback–Leibler divergence but it is symmetric and it always provides a finite value.

The Kullback–Leibler divergence ($D_{KL}$) is defined as:

$$D_{KL}(p\text{II}q) = \sum_i p_i log \frac{p_i}{q_i} \tag{17}$$

where $p_i$ represents the empirical probability of finding the pattern $i$ within the simulation results and $q_i$ is the empirical probability of obtaining the same pattern $i$ but within the training image. If two probability distributions are close, $D_{KL}$ tends toward zero, if they are very different $D_{KL}$ tends toward very large values.

The Jenson–Shannon divergence $D_{JS}$ is then defined as:

$$D_{JS}(p\text{II}q) = \frac{1}{2} D_{KL}(p\text{II}m) + \frac{1}{2} D_{KL}(q\text{II}m) \tag{18}$$

where $m = \frac{1}{2}(p+q)$. The results for the two methods are shown in Table 2. $D_{JS}$ for RPAFSIM is always much smaller than the one for FILTERSIM. It confirms the results obtained with the mean error between histograms

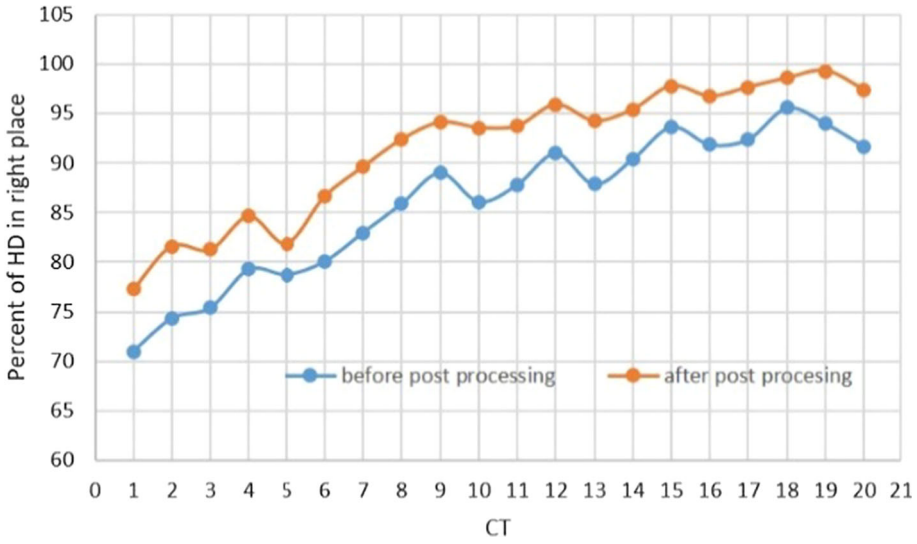



**Fig. 23** Mean percent of HD properly simulated before and after post processing for 50 realizations

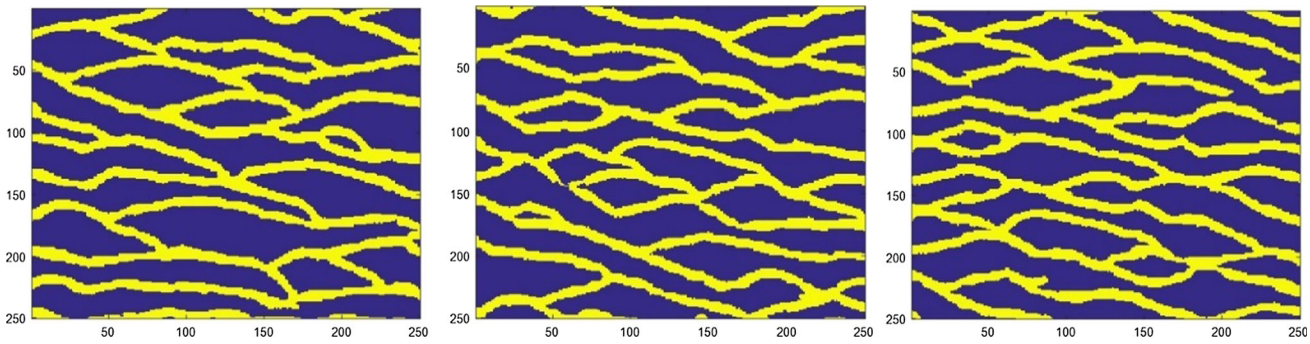


**Fig. 24** Three realization of training image Fig. 21a with proposed method and 50 hard data

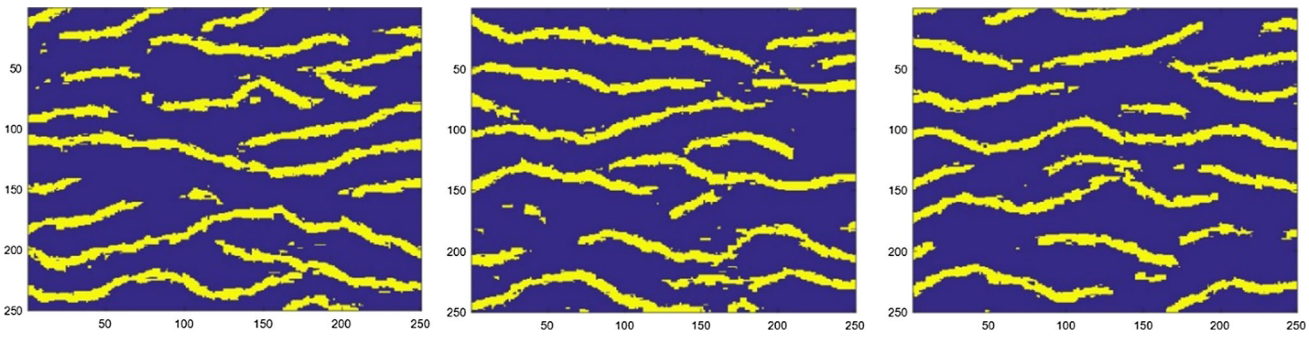


**Fig. 25** Three realization of training image Fig. 21a with FILTERSIM and 50 hard data

and it means that proposed method better reproduces the pattern statistics from to training image.

We also compare proposed method with FILTERSIM based on pattern reproduction and spatial variability (Tan et al. 2014). The idea is to compute a measure which is the ratio of relational between-realization variability to relational within-realization variability:

$$r_{k,m} = \frac{r_{k,m}^{between}}{r_{k,m}^{within}} \tag{19}$$

$r_{k,m}^{between}$ quantifies between-realization variability of the algorithm k related to the algorithm m and it presents variability between different realizations. Large value for $r_{k,m}^{between}$ shows that the algorithm k produces realizations covering a wide space of uncertainty (abdollahifard 2016).

$r_{k,m}^{within}$ quantifies the variability between the realizations and training image. Small values for $r_{k,m}^{within}$ indicate that the realizations produced using algorithm k are more similar to

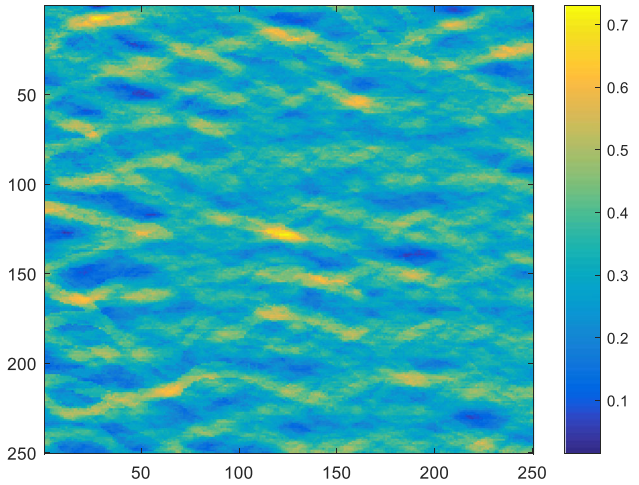



**Fig. 26** E-type of 200 realization of proposed method with 50 hard data

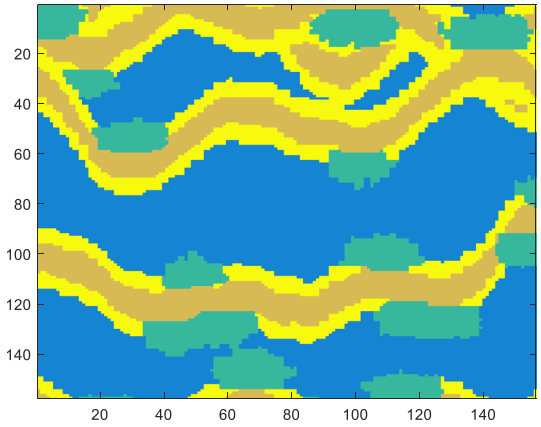



**Fig. 28** Training Image with four categories

the training image comparing to m. $r_{k,m} = 1$ shows that both algorithm perform the same and $r_{k,m} > 1$ shows that algorithm k perform better than m.

In order to compute $r_{k,m}^{within}$ and $r_{k,m}^{between}$, multiple-point histograms (MPHs) or clustering based histogram of patterns (CHPs) of realizations and training image are compared using Jenson–Shannon divergence in different resolutions. Here MPH is employed with the patch size of $4 \times 4$ for training image of Fig. 19a with template size $25 \times 25$ and overlap $= 10$. Table 3 shows results for the RPAFSIM using either random partitioning or the standard raster path without random partitioning and Table 4 Shows the results for comparing RPAFSIM and FILTERSIM.

As it is shown in Table 3 and Table 4, $r^{total}$ for both case is larger than one and it shows that proposed method with partitioning method is better than the method without using it and FILTERSIM according to this measure.

### 4.2 Conditional simulation with proposed methods

In this section, we illustrate the results obtained with the proposed conditioning method for three kinds of training image and compare the quality of the results obtained with the different options.

*Training image with two categories* The first test uses a TI corresponding to a binary reservoir which is a mixture of shale and sand (Strebelle 2002; Zhang et al. 2006) as shown in Fig. 21a. For the conditioning stage, a value of CT of $9 \times 9$ is selected by trial and error. The result of the pre-processing of the conditioning stage for 50 hard data for this training image is shown in Fig. 21b.

According to the mean variance curve of Fig. 2, the optimal template size for this TI is $25 \times 25$ but because of

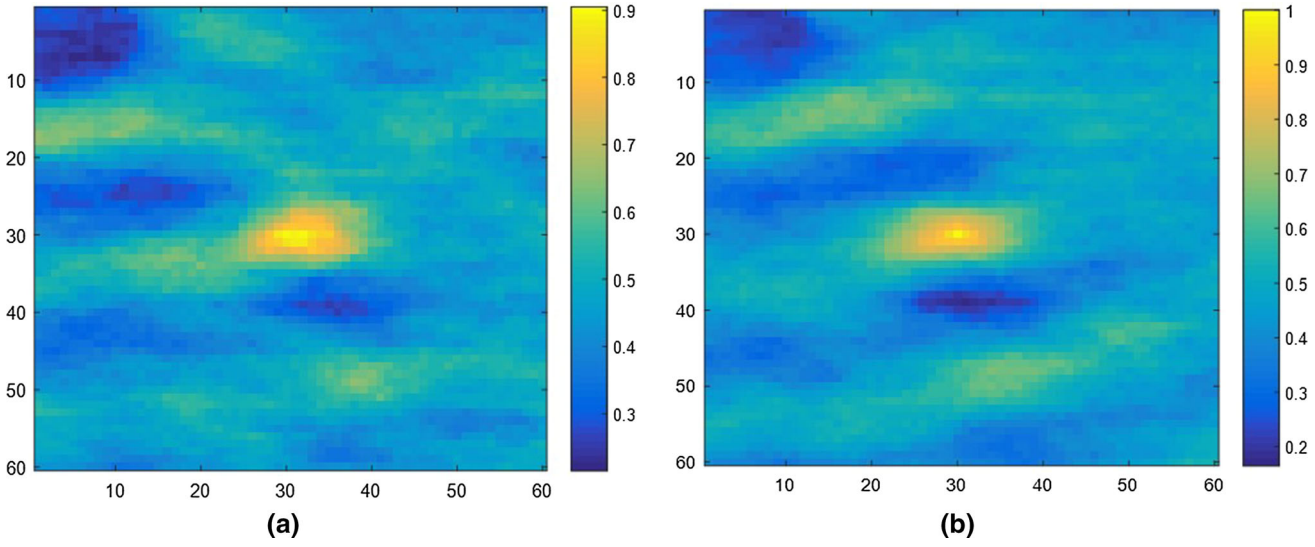



**Fig. 27** **a** E-type of 200 realization for one hard data with proposed method and **b** E-type of rejection for 200 realization

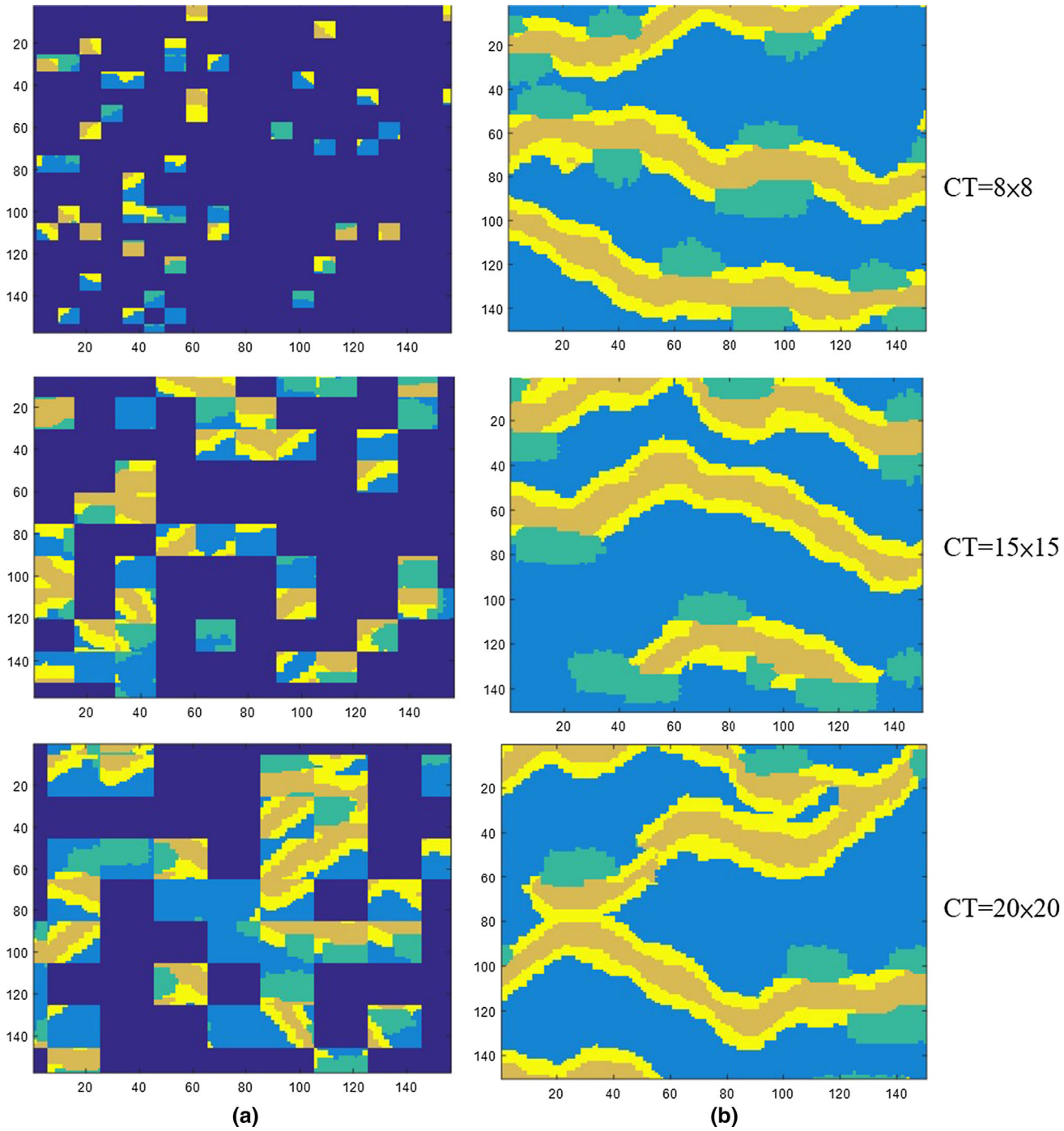



**Fig. 29** **a** Result of conditioning stage with 50 hard data with CT of size 8 × 8, 15 × 15 and 20 × 20 and **b** simulation results with template size 30 × 30

using quilting here we can use larger template without any worry about disconnectivity. We select a template size 40 × 40 by trial and error to speed up the algorithm.

To select the overlap region for the simulation stage, we ran the algorithm with different amount of overlap with using quilting and without using it. The simulation time increases as the overlap region gets larger, but as it is shown in Fig. 22, the results are improved with increase overlap. For this training image, an overlap region between eight and twelve is found to be efficient. Both the quality of the results and the computing time are good in this range. However, in all the tested situation the results using quilting have more connectivity and better quality than without using it.

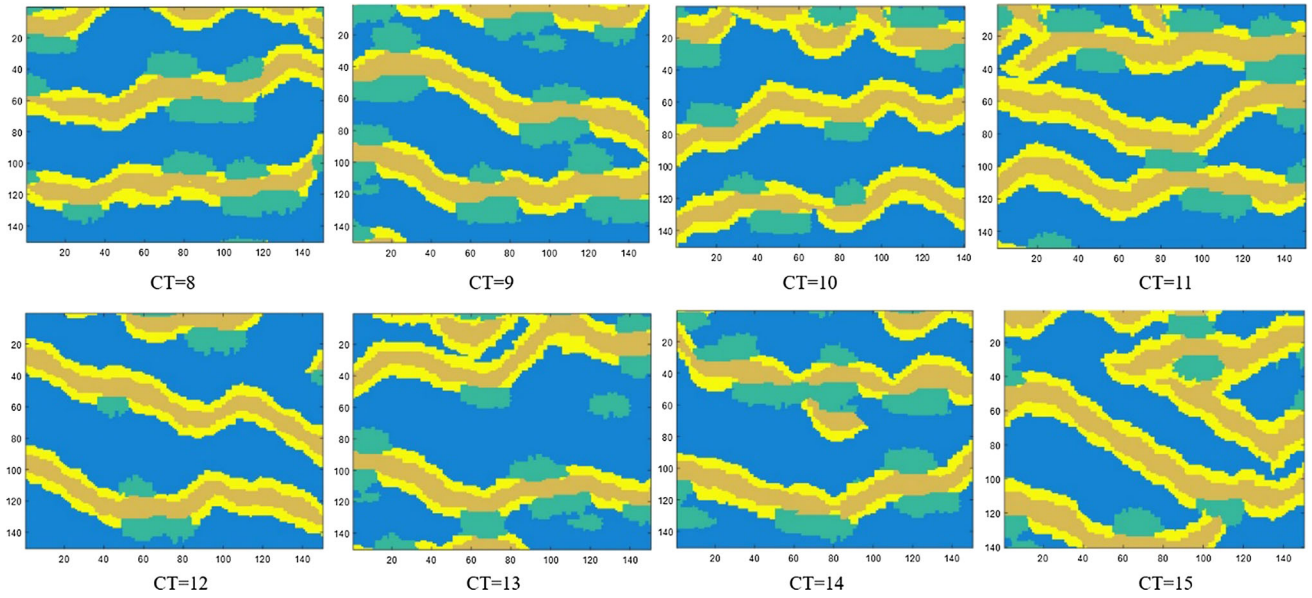



**Fig. 30** Simulation results for various *CT*

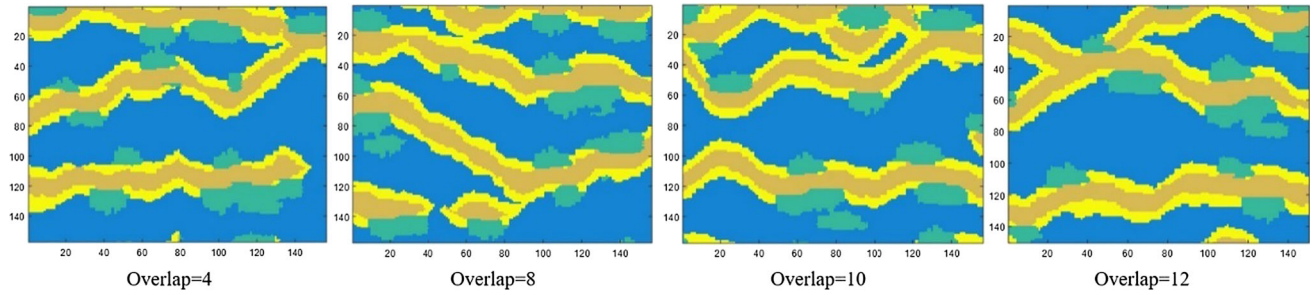



**Fig. 31** Simulation results with varying overlap

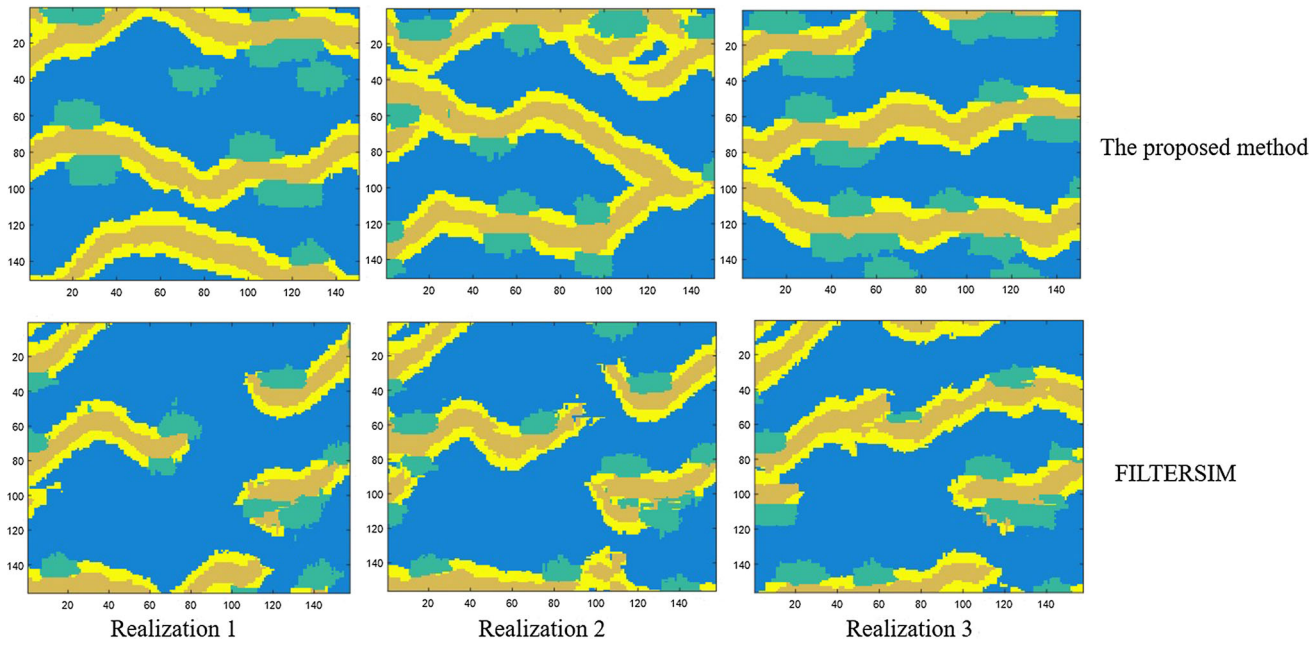



**Fig. 32** Four realizations for TI of Fig. 28 with proposed method and FILTERSIM, with the same parameters

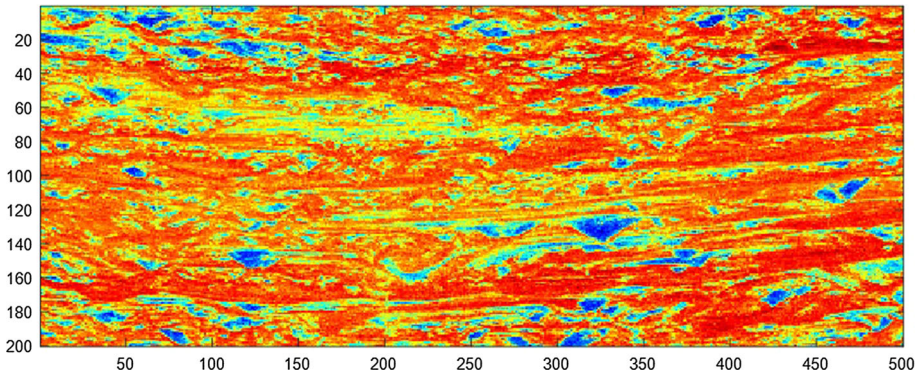



**Fig. 33** A continuous training image of size 200 × 500

To test the performance of the conditioning process with quilting, we calculate *PHR* using Eq. (13) for all realizations after the simulation stage. The mean percentage for 50 realizations and 50 hard data for different CT is shown in Fig. 23. As shown in the figure, for $CT > 7$ more than 85% of all HD are reproduced properly. After performing the post processing step the percentage is increased by roughly 10% in all situations.

Three realizations for the training image of Fig. 21a with $T = 40 \times 40$, $CT = 9 \times 9$, $OS = 8$ pixels, and 50 hard data, are shown in Fig. 24 and three realizations of FILTERSIM obtained with the SGeMS software for the same training image and the same parameters (except template size which should be odd in FILTERSIM and we selected $39 \times 39$) are shown in Fig. 25. These figures show the better connectivity of the proposed method realizations in comparison with FILTERSIM.

The E-type of 200 realizations of the proposed method with 50 hard data with $T = 40 \times 40$, $CT = 9 \times 9$ and overlap $= 8$ pixels are shown in Fig. 26. We do not observe any artefacts.

Figure 27a shows the E-type for 200 realizations with $T = 40 \times 40$, overlap $= 8$ and $CT = 9 \times 9$ for the training image displayed in Fig. 21a for only one hard data in central location. Figure 27b shows the reference probability map that one should get with proper conditioning technique (rejection sampling). There is a probability of 1 of finding a channel at the central location, and this probability progressively diminishes laterally. These figures show that the proposed method do not bias the results when conditioning to hard data.

*4-facies categorical training image* Here, the performance of the algorithm is tested with a TI that has more facies than the previous one. Figure 28 shows the training image that we used. It includes four facies and has a size of $157 \times 156$ pixels (Wu 2007). To determine an optimal size for *CT*, we generated simulations on a grid of size $150 \times 150$ with 50 hard data with various *CT*. When *CT* decreases, the simulations tend to unconditional simulation and if it increases, the connectivity of realization decreases. For example, a result of conditioning stage and final simulation results for $CT = 8 \times 8$, $CT = 15 \times 15$ and $CT = 20 \times 20$ are shown in Fig. 29a, b. In these figures, number of HD is 50, overlap region is 12 and T size is $30 \times 30$. As it is clear in this Figure, with increasing *CT* connectivity decreases.

Simulation results with template size $30 \times 30$, for 50 hard data, and for various CT are shown in Fig. 30. The figure shows that $CT < 14$ have good results.

To find the suitable amount for overlap region, as it is shown in Fig. 31, the algorithm is applied with various amount of overlap. In all these tests, T size is $30 \times 30$, CT size is $12 \times 12$ and there are 50 hard data. The figure shows that $overlap \geq 10$ pixels is required for this training image.

Figure 32 shows three realizations of the proposed method and FILTERSIM algorithm (obtained with SGeMS) with template size of $30 \times 30$ ($29 \times 29$ for FILTERSIM), CT size is $12 \times 12$, overlap 12 pixels and 50 hard data. The connectivity of the simulations obtained with the proposed method is much better than those obtained with FILTERSIM.

*Continuous training image* Fig. 33 showsa continuous variable training image of size $200 \times 500$ taken from Mahmud et al. (2014). It represents internal geological structures obtained by sedimentary processes within a flume experiment.

For this training image, the best overlap for the simulation stage is eight pixels, which is obtained by trial and error.

Like in the previous training image, we apply the algorithm with various CT to find the best CT size for this training image. With an overlap of 8 pixels, the best CT for this training image is $8 \times 8$. Three realizations of the proposed method and FILTERSIM with $CT = 8 \times 8$, overlap $= 6$ pixels, 20 hard data and template size of $16 \times 16$ ($15 \times 15$ for FILTERSIM) are shown in Fig. 34. Again, the results for FILTERSIM are obtained with the SGeMS software.

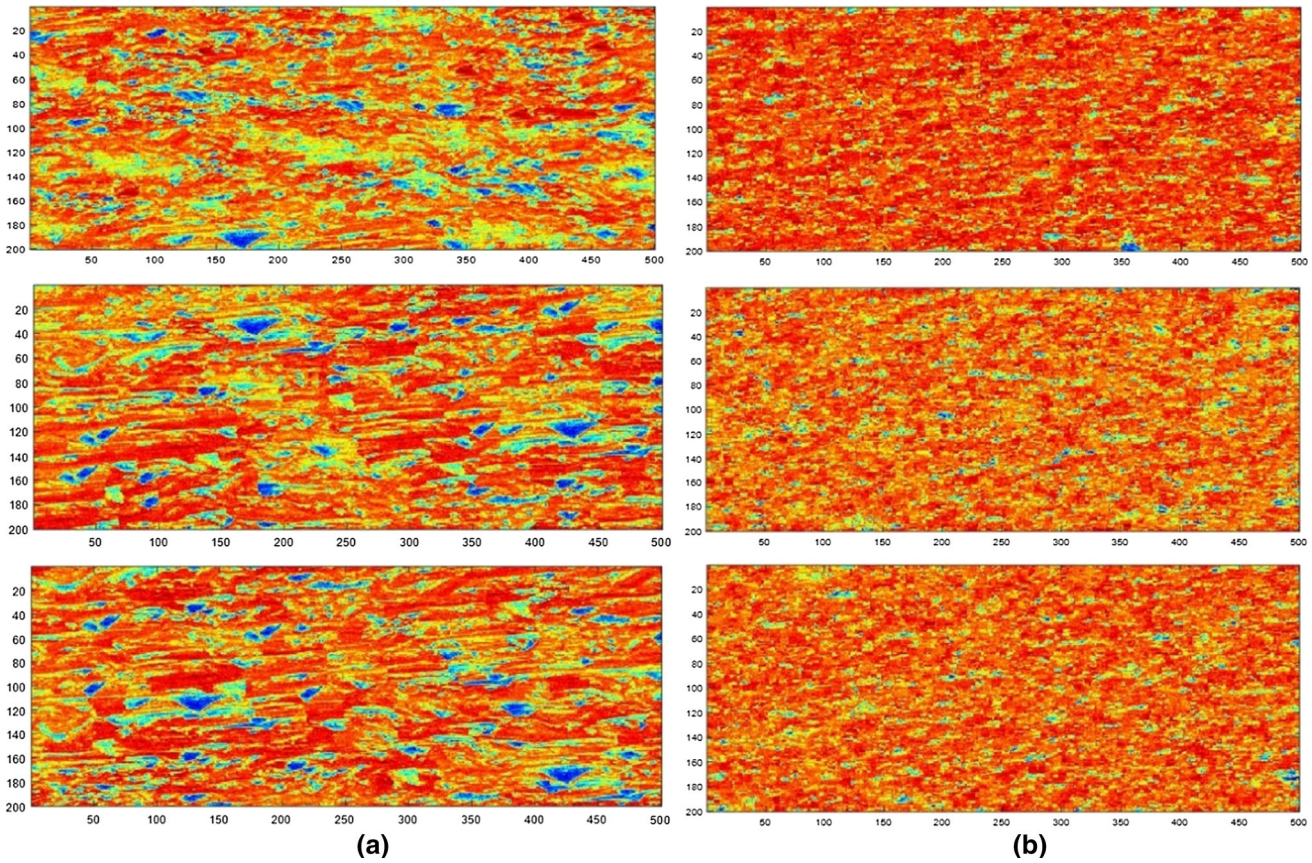



**Fig. 34** **a** Three realizations of proposed method and **b** three realization of FILTERSIM algorithm

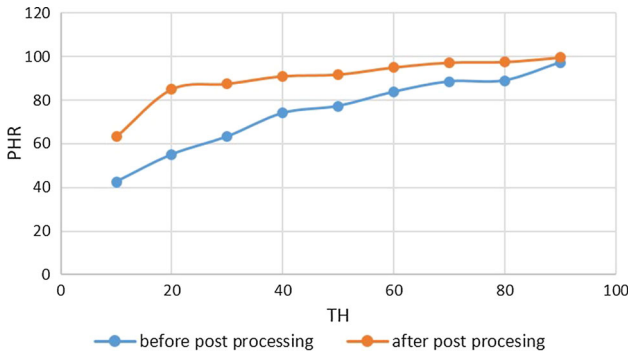



**Fig. 35** Percent of hard data in the right place before and after post processing

To calculate mean percent of hard data properly reproduced, we study the impact of varying *TH* and calculate *PHR* before and after post processing for 50 realizations (Template size of $14 \times 14$, CT $= 8 \times 8$, overlap $= 6$ pixels and 50 hard data). Figure 35 shows the results. *PHR* in all *TH* is better after post processing.

## 5 Conclusion

In this paper, a new algorithm named RPAFSIM, for MPS simulation with random partitioning and adaptive filters is proposed. The proposed method includes: a method for the selection of the template size; the design of specific filters adapted to the training image using PCA, the number of filters is also selected as a function of the complexity of the training image and can be increased for complex training images; the use of random partitioning allowing a better coverage of the uncertainty space; and finally, a simple conditioning method.

The results show that the proposed method has significant advantages over the standard implementation of FILTERSIM. The connectivity of the channels is much better preserved. Generally, the statistics of the pattern are also reproduced more accurately.

In terms of numerical efficiency, RPAFSIM uses raster path and overlap region which makes it fast because the overlaps are smaller than the data event, and therefore, the algorithm is less CPU demanding.

Because RPAFSIM selects a random point at start and continues to simulation using raster path, it represents an algorithm that is able to generate a wider diversity of realizations. In other words, in most of the previous pattern-based algorithms that use random paths, the grid $G$ is completed gradually, and during the simulation, some of the patterns are deleted from the competition cycle. Thus, the algorithm enables the data events to be more "informed." However, in RPAFSIM, the comparison or similarity criteria with the training image is based only on the overlap region. Therefore, the patterns always have a chance to be a part of the realization. This feature leads the algorithm to be less sensitive to the initial state of the system and the preceding patterns and enables it to deal with most of the training images. Using quilting specially with the proposed conditioning method improves connectivity significantly and at the same time accelerates the algorithm because of using larger template size and smaller overlap.

## References


Abdollahifard MJ (2016) Fast multiple-point simulation using a data-driven path and an efficient gradient-based search. Comput Geosci 86:64–74

Aitokhuehi I, Durlofsky LJ (2005) Optimizing the performance of smart wells in complex reservoirs using continuously updated geological models. J Pet Sci Eng 48:254–264

Alcolea A, Renard P, Mariethoz G, Bertone F (2009) Reducing the impact of a desalination plant using stochastic modeling and optimization techniques. J Hydrol 365:275–288

Arpat GB, Caers J (2007) Conditional simulation with patterns. Math Geosci 39(2):177–203

Caers J, Hoffman T (2006) The probability perturbation method: a new look at Bayesian inverse modeling. Math Geol 38(1):81–100

Carvalho PRM, Costa JFCL, Rasera LG, Varella LES (2016) Geostatistical facies simulation with geometric patterns of a petroleum reservoir. Stoch Environ Res Risk Assess. doi:10.1007/s00477-016-1243-5

Chatterjee S, Mohanty MM (2015) Automatic cluster selection using gap statistics for pattern-based multi-point geostatistical simulation. Arab J Geosci 9(8):7691–7704

Cox TF, Cox MA (1994) Multidimensional scaling. Chapman & Hall, London

Duda R, Hart P, Stork D (2001) Pattern classification. Wiley, Hoboken

Efros AA, Freeman WT (2001) Image quilting for texture synthesis and transfer. Paper presented at the ACM SIGGRAPH conference on computer graphics, Los Angeles

Endres DM, Schindelin JE (2003) A new metric for probability distributions. IEEE Trans Inf Theo 49(7):1858–1860

Fukunaga K (2013) Introduction to statistical pattern recognition. Acad Press, Cambridge

Gardet C, Ravalec M, Gloaguen E (2016) Pattern-based conditional simulation with a raster: a few techniques to make it more efficient. Stoch Environ Res Risk Assess 30(2):429–446

Guardiano FB, Srivastava RM (1993) Multivariate geostatistics: beyond bivariate moments. In: Soares AO (ed) Proceeding of geostatistics Troia 1992. Springer, Netherlands, pp 133–144

Honarkhah M, Caers J (2010) Stochastic simulation of patterns using distance-based pattern modeling. Math Geosci 42:487–517

Jolliffe I (1986) Principal component analysis. Springer, New York

Mahmud K, Mariethoz G, Caers J, Tahmasebi P, Baker A (2014) Simulation of earth textures by conditional image quilting. Water Resour Res. doi:10.1002/2013WR015069

Mariethoz G, Renard P, Caers J (2010) Bayesian inverse problem and optimization with iterative spatial resampling. Water Resour Res 46:17. doi:10.1029/2010WR009274

Mattoccia S, Tombari F, Di Stefano L (2008) Reliable rejection of mismatching candidates for efficient ZNCC template matching. In: Image processing. ICIP 2008. 15th IEEE international, pp 849–852

Michael H, Boucher A, Sun T, Caers J, Gorelick S (2010) Combining geologic-process models and geostatistics for conditional simulation of 3-D subsurface heterogeneity. Water Resour Res 46:W05527

Renard P (2007) Stochastic hydrogeology: What professionals really need? Ground Water 45(5):531–541

Sarma P, Durlofsky LJ, Aziz K (2008) Kernel principal component analysis for efficient, differentiable parameterization of multipoint geostatistics. Math Geosci 40(1):3–32

Scholkopf B, Smola AJ (2002) Learning with Kernels. MIT Press, Cambridge, p 664

Scholkopf B, Smola AJ, Muller KR (1998) Nonlinear component analysis as a Kernel eigenvalue problem. Neur Comput 10:1299–1319

Strebelle S (2002) Conditional simulation of complex geological structures using multiple-point statistics. Math Geol 34(1):1–21

Strebelle S, Cavelius C (2013) Solving speed and memory issues in multiple-point statistics simulation program SNESIM. Math Geosci 46:171–186

Strebelle S, Payrazyan K, Caers J (2002) Modeling of a deep water turbidite reservoir conditional to seismic data using multiple-point geostatistics. In: SPE annual technical conference and exhibition, number SPE 77425. Society of Petroleum Engineers

Tahmasebi P, Hezarkhani A, Sahimi M (2012) Multiple-point geostatistical modeling based on the cross-correlation functions. Comput Geosci 16:779–797

Tan X, Tahmasebi P, Caers J (2014) Comparing training-image based algorithms using an analysis of distance. Math Geosci 46(2):149–169

Tang Y, Zhang J, Jing L, Li H (2015) Digital elevation data fusion using multiple-point geostatistical simulation. IEEE J Sel Top Appl Earth Obs Remote Sens 8(10):4922–4934

Wu J (2007) 4D seismic and multiple-point pattern data integration using geostatistics. Ph.D. thesis dissertation, Stanford University

Wu J, Boucher A, Zhang T (2008) A SGeMS code for pattern simulation of continuous and categorical variables: FILTERSIM. Comput Geosci 34:1863–1876

Zhang T (2006) Filter-based training pattern classification for spatial pattern simulation. Ph.D. Dissertation. Stanford University. Stanford CA, pp 137

Zhang T, Switzer P, Journel AG (2006) Filter-base classification of training image patterns for spatial simulation. Math Geol 38(1):63–80

Zhang T, Du Y, Huang T, Yang J, Lu F, Li X (2016) Reconstruction of porous media using ISOMAP-based MPS. Stoch Environ Res Risk Assess 30(1):395–412

Zhang T, Du Y, Li B, Zhang A (2017a) Stochastic reconstruction of spatial data using LLE and MPS. Stoch Environ Res Risk Assess 31(1):243–256

Zhang T, Gelman A, Laronga R (2017b) Structure and texture-based fullbore image reconstruction. Math Geosci 49:195–215