Computers & Geosciences 86 (2016) 64–74

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Case study

Fast multiple-point simulation using a data-driven path and an efficient gradient-based search

Mohammad J. Abdollahifard

Department of Electrical Engineering, Tafersh University, Tafresh, Iran

ARTICLE INFO

Article history:
Received 24 May 2015
Received in revised form
14 September 2015
Accepted 15 October 2015
Available online 17 October 2015

Keywords:
Geostatistical modeling
Texture synthesis
MPS simulation

ABSTRACT

Multiple-point geostatistics has recently attracted significant attention for modeling different environmental variables. These methods employ the patterns of a training image (TI) to complete a simulation grid (SG), resulting in realizations with good spatial continuity and structural properties. Most existing multiple-point statistics (MPS) methods scan the SG in a random or raster order. In this paper, a new method is presented with a data-driven scanning path giving high priority to pixels with high gradient magnitude. As a result, the image edges are synthesized first, resulting in better connectivity preservation. Although MPS methods usually produce promising results compared to traditional variogram-based modeling, their further development is somehow limited by their excessive computational burden. An efficient search space reduction method, consistent with the proposed ordering scheme, is also presented in this paper. Experiments on different geological fields show results comparable to the state-of-the-art with a significant improvement in CPU time.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Multiple-point statistics (MPS) simulation is a spatial modeling technique which has recently attracted significant attention for characterization of different spatial variables. It relies on training images (TIs) for modeling the spatial variability of environmental variables (Guardiano and Srivastava, 1993).

The modeling of subsurface behavior is usually a difficult problem, due to the presence of complicated structures formed by sedimentological and erosional processes (Huysmans and Dassargues, 2009). Traditional variogram-based modeling approaches are not so efficient in reproduction of realistically complex geological structures (Journel and Zhang, 2006; Schlüter and Vogel, 2011). Object-based modeling methods are useful in simulating complex structures, but they lack flexibility in data conditioning (Michael et al., 2010). MPS simulation methods are capable of handling complex structures and conditioning constraints simultaneously. However, they suffer from some disadvantages including their heavy computational burden, the difficulty of selecting a representative and adequate training image specifically when enough information is not available for such a decision (Pyrcz et al., 2008), and the difficulty in TI parametrization (Suzuki and Caers, 2008).

MPS simulation proceeds by sampling from the conditional probability distribution function (cpdf) at different SG nodes conditioned to hard data and previously simulated data. This is done by extracting a data-event from the SG and searching the data-base of TI patterns to find a similar pattern and pasting the data from the found pattern into the SG. While pixel-based methods fill only one pixel in each step, patch-based methods fill one patch at a time, resulting in faster simulation (Arpat and Caers, 2007).

To further increase the simulation speed, some researchers suggested to cluster the pattern data-base into a limited number of clusters and compare the data-event only with the cluster representatives (Zhang et al., 2006; Honarkhah and Caers, 2010; Abdollahifard and Faez, 2013). Such methods usually require intensive pre-computations.

Instead of searching a data-base of patterns, Mariethoz et al. (2010) suggested to work directly with the TI. To reduce the excessive computational burden of this approach, Rezaee et al. (2013) suggested pasting a bunch of nodes in each step, and Abdollahifard and Faez (2013a) suggested using an approximate gradient-descent template matching method.

Recently it has been realized that MPS simulation algorithms are similar in many ways to techniques developed for texture synthesis in computer graphics (Mariethoz and Lefebvre, 2014). Tahmasebi et al. (2012) followed the line of Efros and Leung (1999) by adopting a raster scanning path for simulation. Following the image quilting (IQ) idea of Efros and Freeman (2001), Mahmud et al. (2014) suggested to find a minimum error boundary cut between subsequent patches resulting in seamless realizations with very good spatial continuity. Markov mesh models are also

E-mail address: mj.abdollahi@tafreshu.ac.ir

http://dx.doi.org/10.1016/j.cageo.2015.10.010
0098-3004/© 2015 Elsevier Ltd. All rights reserved.

employed for geostatistical modeling (Daly, 2005; Kjønsberg and Kolbjørnsen, 2008; Stien and Kolbjørnsen, 2011).

While in most MPS simulation methods the SG is scanned in a random or raster order, the effect of attentive path selection has been tested in the computer graphics community resulting in remarkable improvements in pattern connectivity (Criminisi et al., 2004). Inspired by this work, in this paper a Fast Prioritized SIMulation (FPSIM) method is proposed which assigns priorities to SG nodes and selects the node with the highest priority as a new node on the path.

By assigning high priority to pixels having high gradient magnitude, the image edges are synthesized under minimal constraints resulting in improved connectivity patterns. In order to achieve better conditioning we suggest to include the location of hard conditioning data in the computation of priorities as well. It should be noted that since the proposed algorithm is a patch-based method, it synthesizes the low-gradient pixels around the high-gradient point located at the center of the patch. As a result, the method is capable of preserving the image proportions by using large enough patches.

Informed simulation paths based on the information obtained from observed data were previously exploited in MPS simulation (Liu and Journel, 2004; Eskandaridalvand and Srinivasan, 2010). Furthermore, Renard et al. (2011) proposed a method to condition stochastic simulations of lithofacies to connectivity information available before starting the simulation process. In this paper no connectivity information is assumed to be available ahead of time. Instead, the algorithm attempts to achieve better conditioning and mimic the connectivity patterns of the TI by attentive path selection and continuous update of the priority function. It should be noted that the proposed prioritization is applicable to many existing patch-based simulation methods.

Furthermore, in this paper a new gradient based search algorithm is also proposed which reduces the search space up to hundreds of times. By considering the image gradient as an important factor in determining the priority, most of selected data-events have high gradient values in their central pixel. Given the gradient vector in the central pixel, the search space can be confined from the whole TI to templates with comparable gradient in their center.

This paper is organized as follows. In Section 2 the details of the proposed method are presented along with reasons on its importance and usefulness. In Section 3 the algorithm is tested and analyzed on a number of test cases. Finally we conclude in Section 4.

2. Methodology

In MPS simulation methods, an incomplete simulation grid is completed using patterns of a training image (Fig. 1). The simulation proceeds by scanning the SG nodes in a specific order. For each node, a data-event is extracted around the node, then the pattern data-base is searched to find a match, and finally a piece of data is transferred from the found pattern to the SG. The contributions of this paper are twofold. First, it is shown that the order of scanning the nodes has significant effect on simulation quality, and a prioritization method is presented for path selection (Section 2.1). Second, an efficient method for confining the search space and reducing the computational complexity is proposed (Section 2.2). It should be noted that the proposed search method does not provide remarkable performance in general template matching problems. However, as will be discussed later it works very well for patches selected through the proposed prioritization. Therefore, the two advances proposed in this paper are by no means independent.

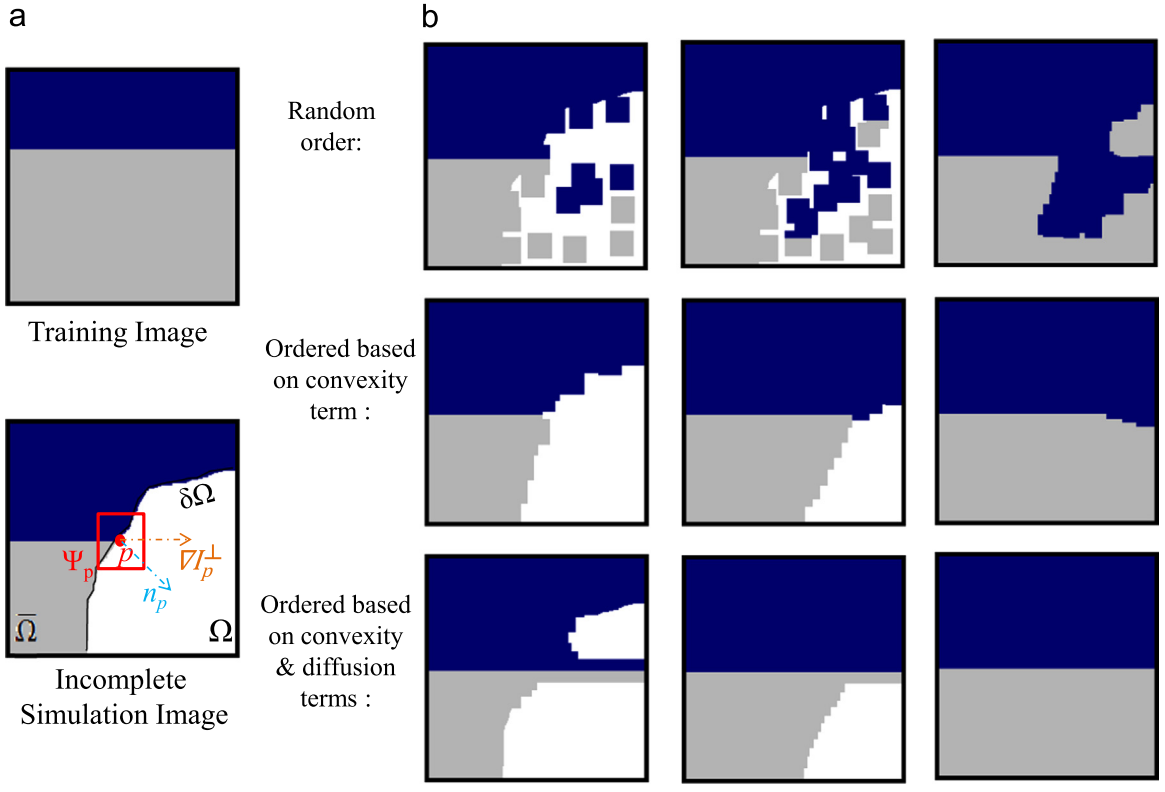

Fig. 1. (a) A simple TI along with an incomplete SG. (b) The completion progress using three different ordering scenarios. First row: random order, second row: ordered based on priorities defined using only the convexity term, and third row: ordered based on priorities defined using Eq. (3).

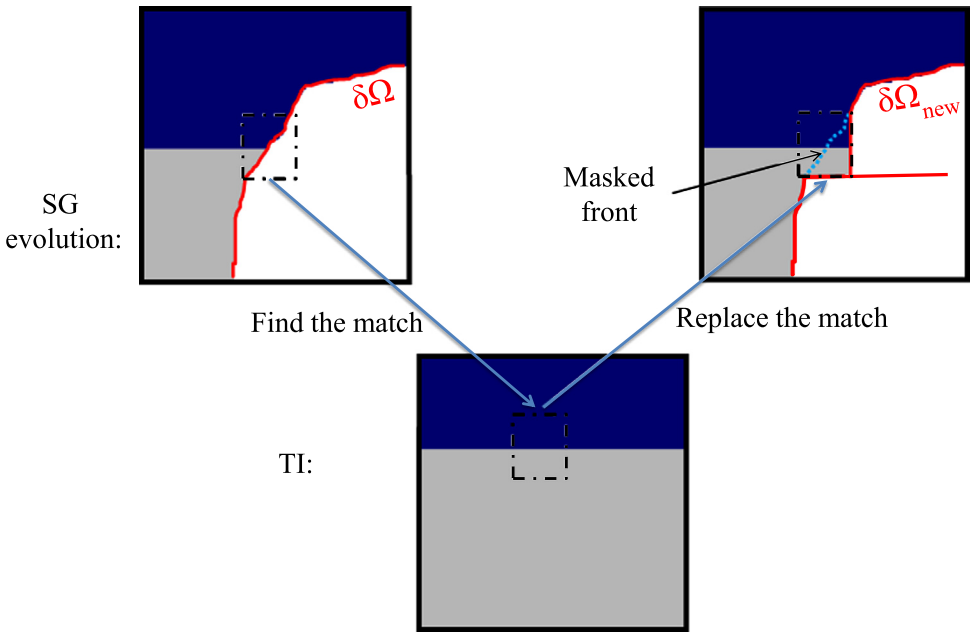

Fig. 2. Our completion algorithm: $P(p)$ is computed for all points on the fill front and the point with highest priority is selected as a new node on the path. Then a data-event is extracted around the node and the TI is searched to find a match. Finally the match is replaced in the SG and the front is updated accordingly.

2.1. Filling order

The data-event extracted around each node has a limited extent, ignoring far pixels. Therefore, synthesis of values inconsistent with far pixels seems inevitable. The first row of Fig. 1b shows the process of completion of an incomplete image using the simple TI of Fig. 1a with a random scanning order. For data-events which contains no data the match is selected randomly from the TI. Such randomly selected patterns are likely to be inconsistent with far pixels. As the simulation proceeds, propagation of initial long-range inconsistencies leads to short range inconsistencies which are not manageable anymore due to the limited patterns (i.e., behaviors) available in the TI.

Our algorithm is intended to reduce the chance for such inconsistencies by modifying the scanning path. Let us assume that at the nth iteration of the simulation algorithm the SG can be partitioned into two continuous regions: the filled region $\bar{\Omega}$, and the empty (or target) region Ω. The contour between Ω and $\bar{\Omega}$ is denoted as $\delta\Omega$. In order to prevent inconsistent synthesis, the new node p is confined to be selected on the fill front $\delta\Omega$ ($p \in \delta\Omega$). The data-event around p is denoted by Ψ_p. To encourage convex growth of $\bar{\Omega}$, a convexity term, $C(p)$, is defined proportional to the number of known neighbors (including hard conditions or previously synthesized values):

$$C(p) = \frac{|\Psi_p \cap \bar{\Omega}|}{|\Psi_p|}, \tag{1}$$

where $|\cdot|$ denotes the number of elements. This term is adopted from Criminisi et al. (2004) with slight modifications to become applicable to MPS simulation. By prioritizing the points on the fill front based on the convexity term, the filling will proceed as depicted in the second row of Fig. 1b. The new path improves the simulation output significantly, specifically in textureless areas. Since the filling proceeds uniformly along the fill front, the blue region reaches the horizontal image edge sooner than the gray region. This causes an undesirable deviation in the image edge (Fig. 1b, middle-right).

The continuity and structure of the synthesized image around the edges is very important, specifically as far as flow and transport processes are concerned. In order to achieve better connectivity, points with high gradient values are given high priorities. Let ∇I_p denote the gradient vector of the SG at point p, and n_p denote a unit vector orthogonal to the fill front at point p (see Fig. 1a). The edges perpendicular to the fill front have larger impact and the edges tangent to the front have no impact on the target region. Therefore, a diffusion term is defined as follows:

$$D(p) = \nabla I_p^{\perp} . n_p, \tag{2}$$

where $\perp$ denotes the orthogonal operator. This term is exactly borrowed from image inpainting application of Criminisi et al. (2004).

Combining convexity and diffusion terms, the priority is defined as follows:

$$P(p) = C(p) T_{[0.1,1]}\{D(p)\}. \tag{3}$$

$T_{[\xi,1]}$ is a linear transformation which maps its argument to the interval $[\xi, 1]$.[1] Note that particularly in the case of categorical variables the gradient magnitude is likely to be absolutely zero in many points. Confining the minimum value of $D(p)$ to 0.1 prevents the diffusion term from nullifying the convexity term in such points. In contrary to ordinary MPS methods which select the new nodes randomly, in this paper $P(p)$ is computed for all points on the fill front and the point with highest priority is selected as a new node on the path. Then a data-event is extracted around the node and the TI is searched to find a match. Finally the match is replaced in the SG and the front is updated accordingly (see Fig. 2). As depicted in the third row of Fig. 1b, defining the priorities using Eq. (3) results in diffusion of image edges under minimal constraints.

2.2. Efficient search

For most MPS simulation algorithms, the template matching problem is the most time-consuming part. In this paper, an

[1] Consider the variable z defined in the interval $[z_{min}, z_{max}]$. Such variable can easily be mapped to the arbitrary interval of $[w_{min}, w_{max}]$ as follows:
$w = T_{[w_{min},w_{max}]}(z) = \frac{z - z_{min}}{z_{max} - z_{min}}(w_{max} - w_{min}) + w_{min}$.

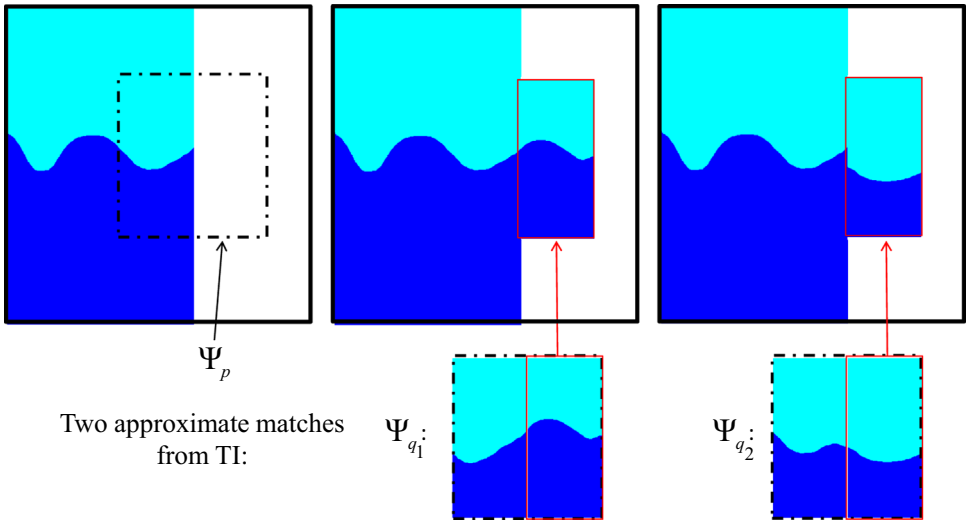

Fig. 3. Our search algorithm preserves pattern connectivity by gradient-based screening. Ψ_{q_1} and Ψ_{q_2} are two approximate matches for Ψ_p extracted from the TI. Although $d\{\Psi_p, \Psi_{q_2}\} < d\{\Psi_p, \Psi_{q_1}\}$, Ψ_{q_1} is preferable because of better connectivity preservation. Our algorithm rejects Ψ_{q_2} in candidate selection phase and prefers Ψ_{q_1}.

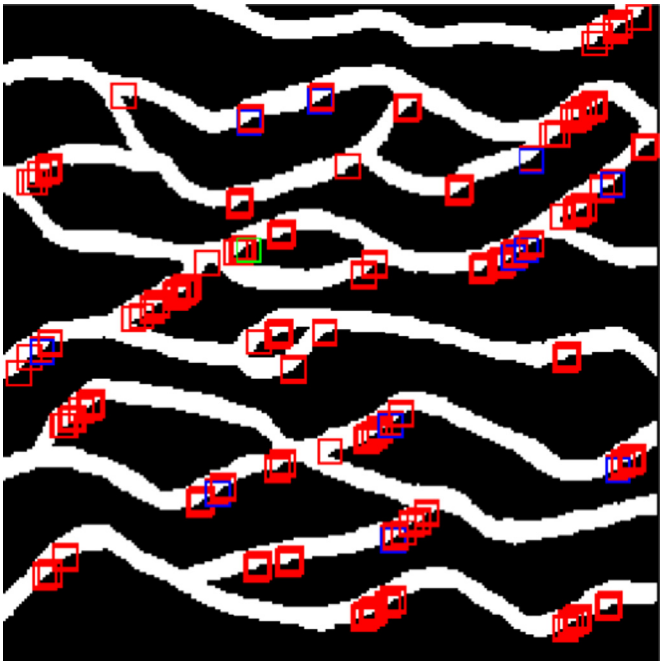

Fig. 4. A channelized image used to test our search algorithm. 233 matches are found as candidate set (and surrounded by rectangles) for the data-event extracted from the patch in the green rectangle. See the text for more details. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

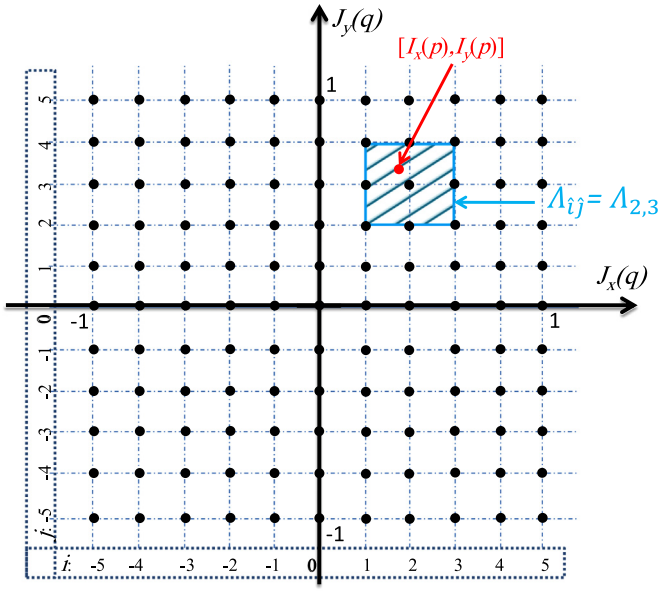

Fig. 5. The space of possible gradient vectors in the gradient plane. The candidate set for point p with gradient vector $[I_x(p), I_y(p)]$ – depicted as a red dot – is determined by finding $[\hat{i}, \hat{j}]$ using Eq. (7), and finding $L\{\hat{i}, \hat{j}\} = L\{2, 3\}$ using Eq. (6). $L\{2, 3\}$ contains TI points whose gradients lie within the shaded region $\Lambda_{2,3}$. Here k is set to 5.

efficient search algorithm is proposed taking the nature of the prioritization algorithm into account. The priority function (3) encourages the points with high gradients to be selected on the path. When a match is replaced around such a node, its neighbors are masked and removed from the front (Fig. 2). Consequently, most of low-gradient points on the front will never get the chance to be selected on the scanning path. Therefore, nearly all selected data-events have high gradient magnitude at their center, representing a strong edge. A typical strong edge is not limited to a single point but it continues in a long spatial range. As a result the behavior of the image in the neighborhood is mainly determined by the strength and direction of the edge. In this paper, the gradient vector of the central pixel is employed as an effective way to reduce the search space.

Consider an incomplete data-event Ψ_p around a point p with a strong gradient $\nabla I(p) = (\partial I(p)/\partial x, \partial I(p)/\partial y) = (I_x(p), I_y(p))$. In order to find a match for Ψ_p, we first compare its gradient vector with all pixels in the TI to find a candidate set Q:

$$Q = \left\{ q: \left(\left|J_x(q) - I_x(p)\right| < t_d\right) \& \left(\left|J_y(q) - I_y(p)\right| < t_d\right) \right\}, \tag{4}$$

where J is the TI and t_d is a threshold. Then the match is found by computing the normalized Euclidean distance between Ψ_p and Ψ_q for all $q \in Q$:

$$d\{\Psi_p, \Psi_q\} = \frac{\sum_{u \in \Psi_p} \left(\Psi_p(u) - \Psi_q(u)\right)^2}{|\Psi_p|} \tag{5}$$

where u is a two-dimensional location vector and $\Psi_p(u)$ denotes the value of Ψ_p at location u. Note that $|\Psi_p|$ denotes the number of known pixels in Ψ_p and the above summation is evaluated only

over such pixels. As depicted in Fig. 3, this approach not only reduces the computational cost compared to exhaustive search methods, but also results in improvements in pattern connectivity.

In order to demonstrate the effectiveness of this search strategy, let us consider the 251×251 channelized image of Fig. 4. The image shows a reservoir with soil properties classified into sand (white) and shale (black) (Strebelle, 2002). The 10×10 patch delimited by the green rectangle is extracted from the image and its right half is removed to form a hypothetical incomplete data-event. Then the candidate set Q is computed using Eq. (4) by setting $t_d = 6\%$. The candidate set obtained contains 233 points which are surrounded by 10×10 colored (green, blue and red) rectangles in the figure. After full comparison it has been realized that, among candidate patches, 11 patches, distinguished by blue rectangles, are exactly equal to the initial data-event. In this test our algorithm performs 22 times faster than a full search. Note that the efficiency of this algorithm increases by increasing the patch size. For example if the same test is done by a 20×20 patch the gain would be around 84.

Considering that the search space is reduced 270 times (from 251×251 to 233), one may expect to achieve a speedup of 270. However, much time is needed for computation of the candidate set. Further improvements can be achieved by pre-computing the candidate sets via clustering the TI pixels based on their gradients. Assume that partial derivatives are normalized so that their values lie in the interval $[-1,1]$. In other words, $\nabla J(q)$ belongs to the rectangular region $[-1, 1]^2$ for all q in the TI (this region is depicted in the gradient plane in Fig. 5). The subregion Λ_{ij} designated by

$$\begin{cases} (i-1)/k \le J_x \le (i+1)/k \\ (j-1)/k \le J_y \le (j+1)/k \end{cases}$$

is a square-shaped subregion centered at $c_{ij} = [i/k, j/k]$ whose horizontal and vertical positions can be controlled by i and j respectively ($i, j \in [-k, \ldots, k]$, see Fig. 5 where $\Lambda_{2,3}$ is shaded). List of the positions of the TI pixels whose gradient vectors lie within this region is denoted by $L\{i, j\}$

$$L\{i,j\} = \{q \,|\, \nabla J(q) \in \Lambda_{i,j}\}. \tag{6}$$

For n dimensional variables, $(2k+1)^n$ lists are formed. It should be noted that these lists are overlapping so that each pixel belongs to four lists (Fig. 5).

In Fig. 5, c_{ij}s are depicted with black dots. In the simulation stage, at first the gradient vector of the selected point p, i.e. $\nabla I(p)$, is compared with c_{ij}s:

Table 1
Unconditional simulation algorithm.

1. select p and q ($\in R^2$) randomly,
2. $\bar{I}(p) \leftarrow \Psi_q^I$, $\bar{I}_x(p) \leftarrow \Psi_q^{J_x}$, $\bar{I}_y(p) \leftarrow \Psi_q^{J_y}$,
3. initialize $\delta\Omega$ by borders of the inserted patch,
4. **while** there exist any unfilled pixel in SG **do**
5. compute $P(p)$ for all $p \in \delta\Omega$ (Eq. (3)),
6. $\hat{p} \leftarrow \text{argmax}_p(P(p))$,
7. find $[\hat{i}, \hat{j}]$ for $p = \hat{p}$ (Eq. (7)),
8. $Q \leftarrow L\{\hat{i}, \hat{j}\}$,
9. compute $d\{\Psi_{\hat{p}}^I, \Psi_q^I\}$ for all $q \in Q$ (Eq. (5)),
10. find N best matches: $\Psi_{q_1}, \ldots, \Psi_{q_N}$,
11. $\hat{n} \leftarrow$ produce a random integer from 1 to N,
12. $\bar{I}(\hat{p}) \leftarrow \Psi_{q_{\hat{n}}}^I$, $\bar{I}_x(\hat{p}) \leftarrow \Psi_{q_{\hat{n}}}^{J_x}$, $\bar{I}_y(\hat{p}) \leftarrow \Psi_{q_{\hat{n}}}^{J_y}$,
13. update $\delta\Omega$,
14. **end while**

$$[\hat{i}, \hat{j}] = \text{argmin}_{i,j} \left\{ d_{man}(\nabla I(p), c_{ij}) \right\}, \tag{7}$$

where $d_{man}(.,.)$ denotes the Manhattan distance. Then, the points belonging to the list $Q = L\{\hat{i}, \hat{j}\}$ are considered as the candidate set.

2.3. Unconditional simulation

The proposed prioritization method is applicable to many existing patch-based simulation algorithms. However, in this section a specific MPS simulation algorithm is presented based on the proposed prioritization and search methods described in previous sections. The simulation algorithm is composed of two main stages namely the preprocessing and simulation stages. In the preprocessing stage, first the partial derivatives of the TI are computed as follows:

$$J_x = J_* F, \quad J_y = J_* F^T, \tag{8}$$

where '$*$' denotes convolution and F is a vertical high-pass filter. In this paper, F is defined as follows:

$$F = 1/3 \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}. \tag{9}$$

After computing partial derivatives, the lists $L\{i, j\}$ are formed using Eq. (6). Note that the lists are formed by processing only the gradient vectors using a very simple clustering scheme. Therefore the processing time required for this phase is several times less than the preprocessing time required for clustering-based methods like FILTERSIM (Zhang et al., 2006).

After the preprocessing stage, the simulation algorithm is accomplished as summarized in Table 1. Ψ_p^X denotes a patch extracted from image X around the point p, and $\bar{X}(p)$ is a portion of the image X related to such a patch. For each data-event, the candidate set is determined using the lists formed in the preprocessing stage (Eq. (7)). Then the patches in the candidate set are fully compared with the data-event to find the N most similar patches. Among them, one patch is selected randomly and inserted in the SG. Note that the previously synthesized values are not overwritten and the insertion takes place only in unknown locations (see Fig. 3).

In this algorithm, the gradient vector is not computed in the SG; instead it is transferred from TI just like patch values. This ensures the presence of at least one point in the list corresponding to the TI point from which the current node is synthesized.

The generalization of the algorithms for 3D grids is straightforward. Using three cubic filters, the gradient vectors are computed for all TI voxels in three perpendicular directions (x, y and z). Then, the gradient vectors are indexed in $(2k+1)^3$ lists. In the simulation stage, the priorities are computed for voxels on the fill front (note that for 3D grids the fill front is a surface). Then the candidate set is determined using the gradient vector and the final patch is selected in the candidate set just like the 2D case. Note that although ∇I_p has infinite number of perpendicular vectors $\nabla I_p^{\perp}$ in 3D space, each one of them can be equivalently used in Eq. (2).

2.4. Conditional simulation

MPS simulation algorithms are usually applied to fields with a limited number of informed nodes called hard conditioning data. Since conditioning data are usually obtained by direct measurement, they have to be honored in the simulation process. Two different approaches were proposed in the literature to meet this requirement namely patch selection method and template

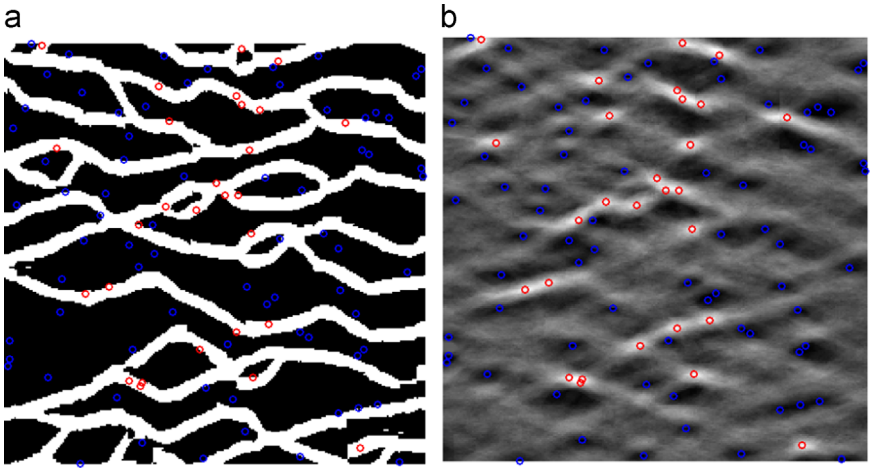

Fig. 6. Conditional simulation using the TI of Fig. 4 and 100 hard conditioning data depicted as red and blue circles for channel and non-channel samples. Patch size: 35×35, $k=15$, $N=3$. (a) One of the realizations and (b) E-type of 100 realizations. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

splitting method (Tahmasebi et al., 2012; Mahmud et al., 2014). The first approach is adopted in this paper. In conditional simulation the following equation is used instead of Eq. (5):

$$d = \alpha d_h + (1 - \alpha) d_s, \tag{10}$$

where d_h and d_s are distances between data-event and TI pattern evaluated using Eq. (5) on hard data and simulated data regions respectively. α is the weight factor which determines the relative importance of the hard data with respect to the simulated data (α is set to 0.9 in our tests).

Furthermore, the priority function (3) is modified giving more priority to points with more hard neighbors. Consequently, such points will see less synthetic data around themselves allowing more concentration on hard data. Suppose that hard conditioning data are available at a set of points $\{p_i^h\}_{i=1:M}$. Then, $Conf(p)$ is defined as follows:

$$Conf(p) = \sum_{i=1}^{M} \exp\left(- \frac{(p - p_i^h)^T (p - p_i^h)}{2\eta^2} \right), \tag{11}$$

where η is considered as one-third the width of the patches. It should be noted that near the hard conditions more confident predictions can be made and therefore this term is named as the confidence term. Since $Conf(p)$ is independent of the simulation progress, it is evaluated one time for all SG points in the preprocessing stage. In conditional simulation the priority of Eq. (3) is modified as follows using the confidence term:

$$P(p) = C(p) T_{[0.1,1]}\{D(p)\} T_{[0.3,1]}\{Conf(p)\}. \tag{12}$$

Furthermore, unlike unconditional simulation which selects the first node randomly (line 1 of Table 1), in conditional simulation the first node is selected at global maximum of $Conf(p)$.

Like other patch-based algorithms, our algorithm simulates faster using larger templates. However, handling a large number of hard data in a large template is difficult and may reduce the conditional data satisfaction degree. Tahmasebi et al. (2012) suggested to use template splitting to handle this problem which increases computational complexity and CPU time. Because of higher priority given to points with large number of hard neighbors, our proposed algorithm achieves better conditioning using patch selection method described above. In this paper the data-event size is considered constant during simulation. However, the algorithm can easily be modified to achieve better conditioning by allowing data-events with variable size. The data-event size can be defined in inverse relation with $Conf(p)$. Note that, unlike clustering methods, our proposed preprocessing does not impose any constraint on the patch size.

3. Simulation results

In this section the proposed algorithm is tested on different training images. Unless otherwise specified, in this section the algorithm parameters are set as follows: patch size$=35 \times 35$, $k=15$, $N=3$. Furthermore, in all tests the SG has the same size as the original TI.

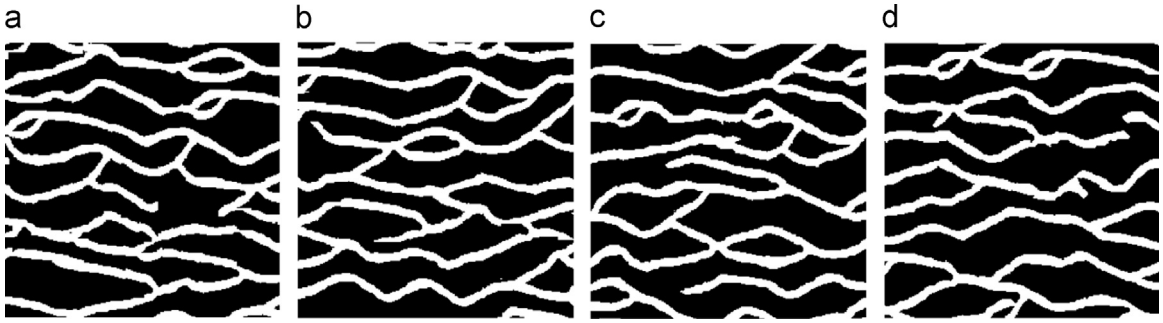

Fig. 7. Unconditional simulation using the TI of Fig. 4. Patch size: 35×35, SG size: 251×251. (a) and (b) are obtained using FPSIM in 0.81 and 0.78 s respectively. (c) and (d) are obtained using IQ in 4.93 and 4.88 s respectively.

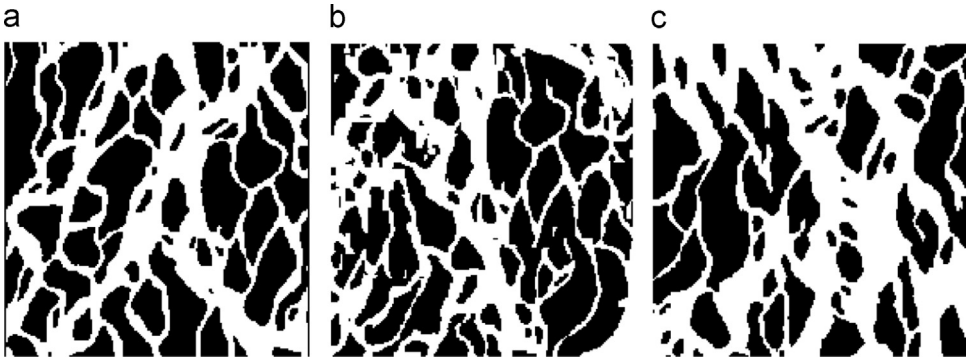

Fig. 8. (a) A 243 × 243 binary image obtained based on a satellite image of Ganges delta (Bangladesh), with channels in white and alluvial bars in black. (b) and (c) two unconditional realizations obtained using FPSIM in 0.651 s and IQ in 3.57 s respectively. Patch size: 35 × 35, SG size: 243 × 243.

3.1. Conditional simulation

In order to generate conditioning data with consistent histogram and variogram, first an unconditional realization is produced and then a number of randomly located samples are extracted from the realization. Using this strategy, 100 samples are generated based on the training image of Fig. 4. Then the conditional simulation algorithm is employed to generate 100 different conditional realizations. Fig. 6a shows one such realization. The produced realization not only satisfies the conditioning data well, but also preserves the connectivity of the channels.

The E-type image calculated as the average of 100 different realizations is also depicted in Fig. 6b. From the E-type image it can be verified that all conditioning data are honored in all realizations. Furthermore, the E-type image shows obvious channels where the channel samples are well aligned.

3.2. 2D unconditional simulation

Image quilting (IQ) method as one of the most efficient patch-based MPS simulation methods was recently proposed by Mahmud et al. (2014). This algorithm scans the SG in a raster order and preserves the pattern connectivity very well by finding the minimum error boundary cut between subsequent patches. Furthermore, the algorithm is very fast in simulating images. The tests have shown that IQ simulates up to hundreds times faster than direct sampling (Mahmud et al., 2014).

In this section the results of the proposed algorithm will be presented and compared with IQ. It should be noted that the patch cutting feature of IQ results in seamless realizations with extremely good connectivities. Although the patch cutting methods are applicable to our algorithm with the potential of improving pattern connectivities, the tests show that our algorithm produces acceptable realizations even without using such methods. As suggested by Mahmud et al. (2014), the IQ overlap size is set to 1/3 of patch size in all tests in this paper. Furthermore, N is set to 10 for IQ in this section. All simulations are carried out in MATLAB on a laptop computer with a 2.6 GHz Processor.

Based on the training images of Figs. 4, 8a, 9a, 10a and 11a, a number of realizations are produced using FPSIM and IQ and depicted in the following five figures i.e. Figs. 7, 8, 9, 10 and 11 (more examples can be found in the supplementary materials). Fig. 8a shows a binary image obtained based on a satellite image of the Ganges delta (Bangladesh), with soil properties classified to channels (white) and alluvial bars (black). Fig. 9a shows a continuous image representing a stone wall. The complex and structured image of Fig. 10a is a satellite image captured from Lena delta in Russia. Fig. 11a shows another complex image representing mud cracks. All TIs are obtained from the website of the book Mariethoz and Caers (2014).

The time required for each realization is reported in the figure caption indicating that the proposed method is faster than IQ by a factor ranging from 5 to 11. The reason lies in the different template matching approaches employed. While IQ searches the whole TI to find a match for a given data-event, the proposed method searches only a candidate set. It seems that both methods perform well in terms of pattern reproduction and connectivity preservation.

3.3. Quantitative evaluation

It is difficult to evaluate the pattern reproduction capability of MPS methods visually based on a limited number of realizations. Quantitative indicators can be employed to summarize the characteristics of a statistically representative number of realizations. Tan et al. (2014) have recently proposed a quantitative measure for comparing geostatistical simulation methods. The measure is computed as the ratio of relational between-realization variability

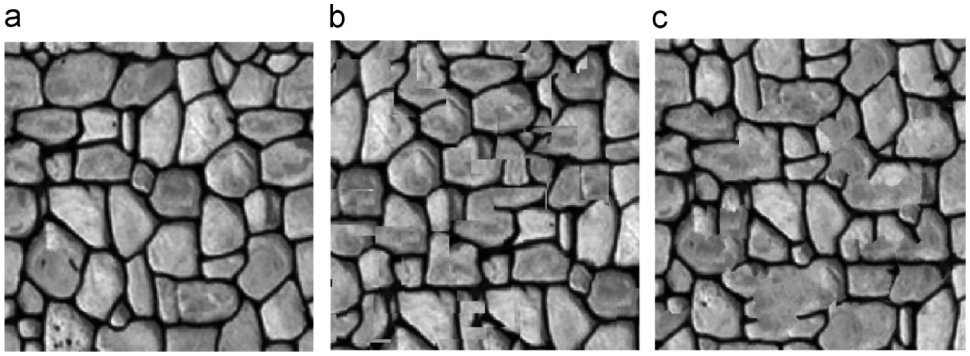

Fig. 9. (a) A 200 × 200 continuous training image. (b) and (c) two unconditional realizations obtained using FPSIM in 0.301 s and IQ in 2.158 s respectively. Patch size: 35 × 35, SG size: 200 × 200.

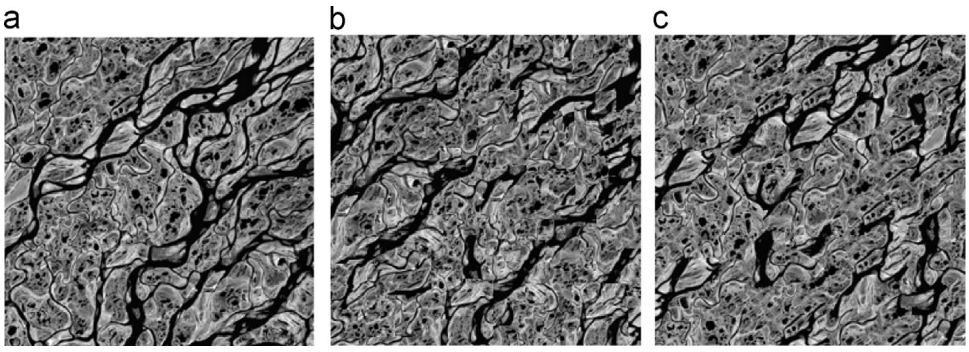

Fig. 10. (a) A 250 × 250 continuous satellite image of Lena delta, Russia used as TI. (b) and (c) two unconditional realizations obtained using FPSIM in 0.508 s and IQ in 5.182 s respectively. Patch size: 35 × 35, SG size: 250 × 250.

to relational within-realization variability:

$$r_{A,B} = \frac{r_{A,B}^{between}}{r_{A,B}^{within}}. \tag{13}$$

Relational between-realization variability of the algorithm A with respect to the reference algorithm B (i.e. $r_{A,B}^{between}$) summarizes the variability present between different realizations. Large values for $r_{A,B}^{between}$ indicate that the algorithm A produces more diverse realizations with respect to B, better representing the spatial uncertainty. On the other hand, $r_{A,B}^{within}$ summarizes the variability present between the realizations and the training image. Small values for $r_{A,B}^{between}$ indicate that the realizations produced using algorithm A adhere well to the provided model (TI) comparing with B. Note that $r_{A,B}$ is equal to 1 if $A=B$ and $r_{A,B} > 1$ indicates that the algorithm A outperforms the algorithm B.

In order to compare the realizations either with each other or with the TI, their multiple-point histograms (MPHs) or clustering-based histograms of patterns (CHPs) are compared using Jensen–Shannon divergence in different resolutions (Tan et al., 2014). For binary images, MPH is employed with the patch size of 4 × 4 in 10 different resolutions. For continuous images CHP is used with the patch size of 20 × 20 in 3 different resolutions.

The experiments show that FPSIM produces acceptable realizations using different TIs by setting $N=3$. However, since IQ is sensitive to N to some extent, three different values are considered for its N namely 3, 5 and 10. For each scenario, namely FPSIM ($N=3$), IQ ($N=3$), IQ ($N=5$) and IQ ($N=10$), 50 different realizations are generated based on each TI and the quantitative evaluation results are summarized in Table 2.

In most cases FPSIM outperforms IQ. The results suggest that the proposed algorithm is well suited for complicated structures where the image contains many pixels with high gradients (see for example Figs. 10 and 11). As depicted in Fig. 7 and quantified in Table 2, IQ outperforms FPSIM in the case of channelized TI which is a simple binary image. It should be noted that for small N, the verbatim copy problem degrades the IQ results significantly.

Table 2
Quantitative comparison between FPSIM and IQ for different TIs. Each cell contains $r_{A,B}$ where A is FPSIM ($N=3$) and B is IQ ($N=3$), IQ ($N=5$) or IQ ($N=10$).

Reference algorithm	Fig. 4	Fig. 8a	Fig. 9a	Fig. 10a	Fig. 11a
IQ ($N=3$)	0.90	0.97	7.62	13.03	3.57
IQ ($N=5$)	0.79	1.12	3.35	3.77	1.52
IQ ($N=10$)	0.75	1.13	1.18	1.48	3.93

Table 3
Comparison of FPMPS and IQ in terms of average CPU time and average number of patch comparisons (N_C). The average for 20 similar simulations is reported in each row. Patch size = 35 × 35, $k=15$, $N=3$.

TI	No. of SG nodes	CPU time (s)		G_t	N_C (× 10^5)		G_n
		FPSIM	IQ		FPSIM	IQ	
Fig. 4	63 001	0.785	4.951	6.3	0.918	56.977	62
Fig. 8a	59 049	0.657	3.56	5.4	0.700	52.854	76
Fig. 9a	40 000	0.302	2.164	7.2	0.192	22.320	116
Fig. 10a	62 500	0.510	5.041	9.9	0.420	56.454	133
Fig. 11a	154 224	1.709	18.624	10.9	1.500	268.122	178

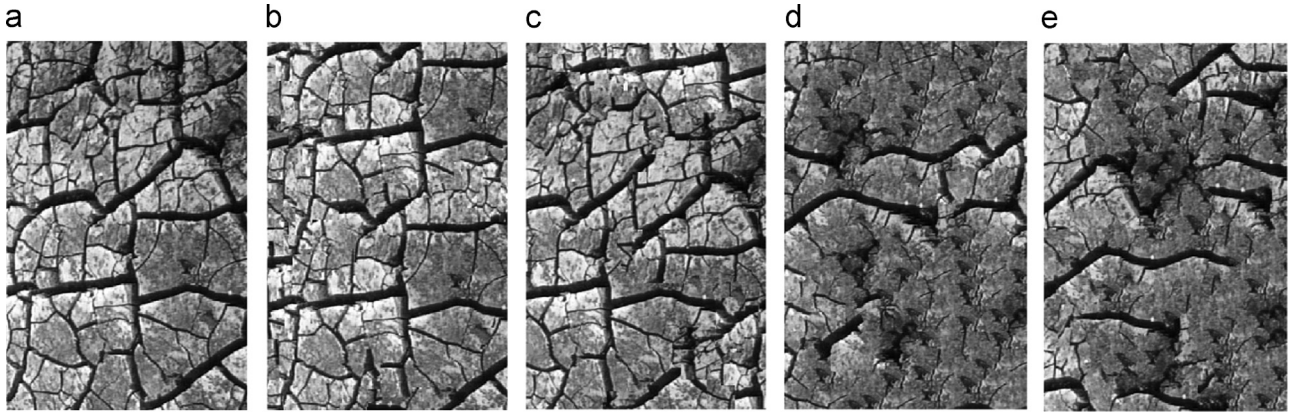

Fig. 11. (a) A 440 × 288 continuous TI depicting mud cracks. (b) and (c) two FPSIM realizations obtained in 1.75 and 1.69 s respectively. (d) and (e) two IQ realizations obtained in 18.3 and 18.4 s respectively. Patch size: 35 × 35, SG size: 440 × 288.

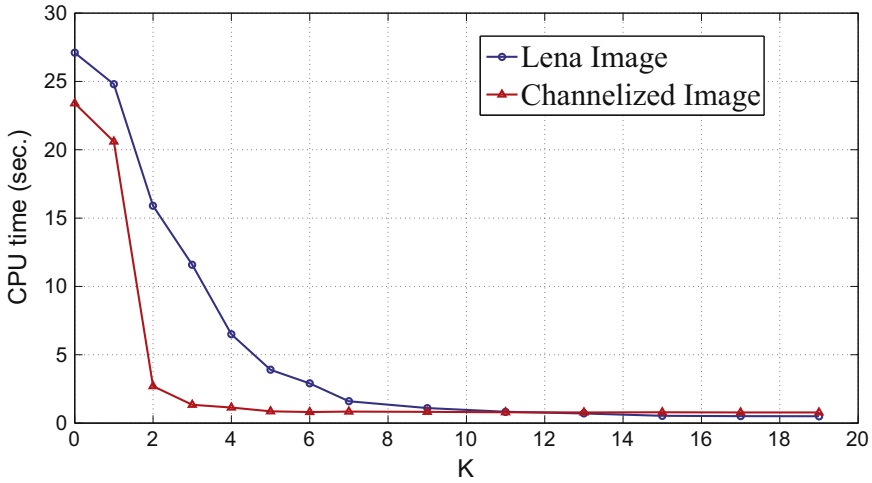

Fig. 12. CPU time of FPSIM for simulation of 250×250 images using TIs of Figs. 4 and 10a for different values of k, patch size$=35 \times 35$, $N=5$.

Increasing N for reducing the verbatim copy problem degrades the pattern connectivity. FPSIM is less prone to verbatim copy and presents acceptable results even for $N=3$.

3.4. Algorithm performance

As indicated before the proposed algorithm performs several times faster than IQ. In this section the average CPU time for several similar simulations is reported in Table 3.

Note that in the IQ method, the image is scanned in a raster order. Therefore, the overlap of each data-event with previously synthesized data has a pre-specified shape (Γ shape) and size. This allows to use MATLAB built-in functions (like filt2) to find the match for each data-event. In the FPSIM algorithm however, the overlap region has a different shape for each data-event. This makes it impossible to use built-in functions. Since MATLAB built-

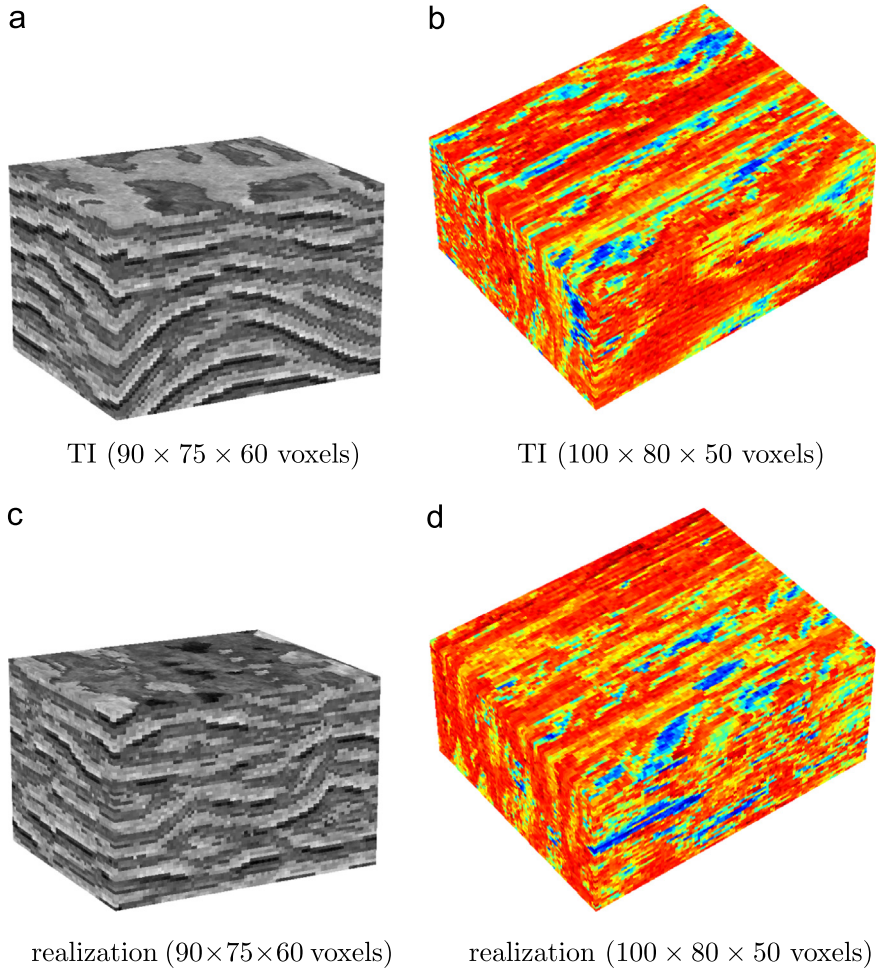

Fig. 13. Unconditional simulation of 3D grids, patch size: $30 \times 20 \times 20$, $k=10$, $N=3$. See the text for more information. (a) TI ($90 \times 75 \times 60$ voxels), (b) TI ($100 \times 80 \times 50$ voxels), (c) realization ($90 \times 75 \times 60$ voxels), and (d) realization ($100 \times 80 \times 50$ voxels).

in functions are remarkably faster than their counterparts written by users, it seems that the processing time is not a good representative for computational complexity in MATLAB.

As a fairer criterion, the average number of patch comparisons required for each simulation (shown by N_C) is also reported in Table 3. It is worth emphasizing that in all MPS simulation algorithms most of the computations are dedicated to template matching and the additional computations are negligible. IQ compares each data-event with all of the TI patterns but FPSIM selects a candidate set via gradient vectors and compares the data-event only with the candidate set. As a result the computational complexity of FPSIM is several times less than IQ as depicted in Table 3.

G_t is the ratio of IQ CPU time to the FPSIM CPU time. The number of comparisons ratio is also denoted by G_n $(=N_C(IQ)/N_C(FPSIM))$. While G_t varies between 5.4 and 10.9 in our tests, G_n varies in the interval [62,178], indicating that the proposed algorithm has the potential of being implemented several times faster in an optimized programming environment.

Another important issue to be considered is the remarkable variation of G_t and G_n for different TIs. In binary images a majority of pixels have a gradient of zero and the other pixels have high gradient magnitudes. There is not any pixel with intermediate gradient magnitude in such images. As a result, most of the gradient cells (see Fig. 5) are empty and the gradient vectors are concentrated in a limited number of bins. The non-uniform distribution of the gradient vectors in the gradient plane, leads to an increase in the number of required comparisons.

In structured images however, the gradient vectors are distributed more uniformly, leading to cells with lower number of members. Consequently the number of comparisons will be lower in more structured images. Thus, the proposed method has an interesting property that not only produces more realistic results for structured TIs, but also handles them much faster.

As another way to assess the effectiveness of the proposed search algorithm, CPU times of FPSIM are reported for different values of k in Fig. 12. The case with $k=0$ is equivalent to exhaustive search. By increasing k from 0 to 19 the CPU time decreases by the factors of 30 and 54.2 for TIs of Figs. 4 and 10a respectively. Due to the limited number of gradient vectors present in the binary image of Fig. 4, increasing k beyond 4 has approximately no effect on CPU time. On the other hand, more diverse gradient vectors of Fig. 10a cause a smoother change in CPU time for increasing values of k.

3.5. 3D unconditional simulation

Since the number of grid points increases dramatically in real 3D fields, the computational efficiency becomes more important in such cases. The proposed algorithm could be employed to handle 3D grids in a limited time. The algorithm efficiency in simulating 3D fields is evaluated in this section on training images of Figs. 13a and b. The first TI shows meandering channels (Mariethoz and Caers, 2014) and the second one is a portion of 3D flume model of Paola et al. (2001). The parameters settings are defined as follows in both tests: patch size: $30\times20\times20$, $k=10$, $N=3$. Unconditional realizations are depicted in Figs. 13c and d. The results seem acceptable in terms of pattern reproduction and continuity preservation.

The CPU times for these realizations are 11.3 s (for Fig. 13c) and 9.8 s (for Fig. 13d). Similar tests with $k=0$ (using exhaustive search) take 1275 s and 1183 s respectively, indicating that the proposed search algorithm accelerates the simulation process more than 100 times. These experiments prove the applicability of the FPSIM algorithm on real 3D fields with hundreds of thousands of points.

4. Conclusion

In this paper a novel MPS simulation algorithm is proposed which works based on two important factors: a new way to determine the SG scanning path (based on an inpainting algorithm taken from computer graphics literature), and an efficient search algorithm proposed herein. The method proposed by Criminisi et al. (2004) is modified to handle hard conditioning data by giving more priority to points with more hard neighbors and computing a weighted distance giving higher weights to hard data. Furthermore the original method is generalized to simulate 3D fields.

The proposed method is tested on different TIs ranging from binary to continuous. Giving high priorities to points with high gradient magnitudes results in proper continuity preservation. Furthermore, the efficient search algorithm introduced in this paper reduces the CPU time by a factor of the order of 10, compared with IQ (Mahmud et al., 2014). Sedimentological and erosional processes usually cause complex subsurface structures. FPSIM shows special efficiency in simulating structured fields where not only produces high quality results, but also achieves higher time gains. The experiments suggest that it is possible to achieve more time efficiency by implementing the algorithm in an optimized programming environment.

Appendix A. Supplementary data

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.cageo.2015.10.010.

References

Abdollahifard, M.J., Faez, K., 2013a. Fast direct sampling for multiple-point stochastic simulation. Arab. J. Geosci., 1–13.

Abdollahifard, M.J., Faez, K., 2013. Stochastic simulation of patterns using Bayesian pattern modeling. Comput. Geosci. 17 (1), 99–116.

Arpat, G.B., Caers, J., 2007. Conditional simulation with patterns. Math. Geol. 39 (2), 177–203.

Criminisi, A., Pérez, P., Toyama, K., 2004. Region filling and object removal by exemplar-based image inpainting. IEEE Trans. Image Process. 13 (9), 1200–1212.

Daly, C., 2005. Higher order models using entropy, Markov random fields and sequential simulation. In: Geostatistics Banff 2004. Springer, pp. 215–224.

Efros, A.A., Freeman, W.T., 2001. Image quilting for texture synthesis and transfer. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. ACM, pp. 341–346.

Efros, A.A., Leung, T.K., 1999. Texture synthesis by non-parametric sampling. In: Proceedings of the 7th IEEE International Conference on Computer Vision, 1999, vol. 2. IEEE, pp. 1033–1038.

Eskandaridalvand, K., Srinivasan, S., 2010. Reservoir modelling of complex geological systems—a multiple-point perspective. J. Can. Pet. Technol. 49 (08), 59–69.

Guardiano, F.B., Srivastava, R.M., 1993. Multivariate geostatistics: beyond bivariate moments. In: Geostatistics Troia92. Springer, pp. 133–144.

Honarkhah, M., Caers, J., 2010. Stochastic simulation of patterns using distance-based pattern modeling. Math. Geosci. 42 (5), 487–517.

Huysmans, M., Dassargues, A., 2009. Application of multiple-point geostatistics on modelling groundwater flow and transport in a cross-bedded aquifer (Belgium). Hydrogeol. J. 17 (8), 1901–1911.

Journel, A., Zhang, T., 2006. The necessity of a multiple-point prior model. Math. Geol. 38 (5), 591–610.

Kjønsberg, H., Kolbjørnsen, O., 2008. Markov mesh simulations with data conditioning through indicator kriging. Proceedings of Geostats 8.

Liu, Y., Journel, A., 2004. Improving sequential simulation with a structured path guided by information content. Math. Geol. 36 (8), 945–964.

Mahmud, K., Mariethoz, G., Caers, J., Tahmasebi, P., Baker, A., 2014. Simulation of earth textures by conditional image quilting. Water Resour. Res. 50 (4), 3088–3107.

Mariethoz, G., Caers, J., 2014. Multiple-Point Geostatistics: Stochastic Modeling with Training Images. Wiley-Blackwell.

Mariethoz, G., Lefebvre, S., 2014. Bridges between multiple-point geostatistics and texture synthesis: review and guidelines for future research. Comput. Geosci. 66, 66–80.

Mariethoz, G., Renard, P., Straubhaar, J., 2010. The direct sampling method to perform multiple-point geostatistical simulations. Water Resour. Res. 46 (11).

Michael, H., Li, 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 (5).
Paola, C., Mullin, J., Ellis, C., Mohrig, D., Swenson, J., Parker, G., Hickson, T., Heller, P., Pratson, L., Syvitski, J., Sheets, B., Strong, N., 2001. Strong experimental stratigraphy. GSA Today 11 (7), 4–9.
Pyrcz, M.J., Boisvert, J.B., Deutsch, C.V., 2008. A library of training images for fluvial and deepwater reservoirs and associated code. Comput. Geosci. 34 (5), 542–560.
Renard, P., Straubhaar, J., Caers, J., Mariethoz, G., 2011. Conditioning facies simulations with connectivity data. Math. Geosci. 43 (8), 879–903.
Rezaee, H., Mariethoz, G., Koneshloo, M., Asghari, O., 2013. Multiple-point geostatistical simulation using the bunch-pasting direct sampling method. Comput. Geosci., 293–308.
Schlüter, S., Vogel, H.-J., 2011. On the reconstruction of structural and functional properties in random heterogeneous media. Adv. Water Resour. 34 (2), 314–325.
Stien, M., Kolbjørnsen, O., 2011. Facies modeling using a Markov mesh model specification. Math. Geosci. 43 (6), 611–624.
Strebelle, S., 2002. Conditional simulation of complex geological structures using multiple-point statistics. Math. Geol. 34 (1), 1–21.
Suzuki, S., Caers, J., 2008. A distance-based prior model parameterization for constraining solutions of spatial inverse problems. Math. Geosci. 40 (4), 445–469.
Tahmasebi, P., Hezarkhani, A., Sahimi, M., 2012. Multiple-point geostatistical modeling based on the cross-correlation functions. Comput. Geosci. 16 (3), 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.
Zhang, T., Switzer, P., Journel, A., 2006. Filter-based classification of training image patterns for spatial simulation. Math. Geol. 38 (1), 63–80.