Computers & Geosciences 107 (2017) 49–60

Contents lists available at ScienceDirect

Computers & Geosciences

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

Research paper

LSHSIM: A Locality Sensitive Hashing based method for multiple-point geostatistics

Pedro Moura[*], Eduardo Laber, Hélio Lopes, Daniel Mesejo, Lucas Pavanelli, João Jardim, Francisco Thiesen, Gabriel Pujol

Departamento de Informática, PUC-RIO, Marques de São Vicente 225, Gávea, Rio de Janeiro, RJ 22453-900, Brazil

A R T I C L E I N F O

Keywords:
Multiple-point geostatistics
Pattern modeling
Training image
Locality Sensitive Hashing
Run-Length Encoding

A B S T R A C T

Reservoir modeling is a very important task that permits the representation of a geological region of interest, so as to generate a considerable number of possible scenarios. Since its inception, many methodologies have been proposed and, in the last two decades, multiple-point geostatistics (MPS) has been the dominant one. This methodology is strongly based on the concept of training image (TI) and the use of its characteristics, which are called patterns. In this paper, we propose a new MPS method that combines the application of a technique called Locality Sensitive Hashing (LSH), which permits to accelerate the search for patterns similar to a target one, with a Run-Length Encoding (RLE) compression technique that speeds up the calculation of the Hamming similarity. Experiments with both categorical and continuous images show that LSHSIM is computationally efficient and produce good quality realizations. In particular, for categorical data, the results suggest that LSHSIM is faster than MS-CCSIM, one of the state-of-the-art methods.

1. Introduction

In the last few decades, multiple-point geostatistics (MPS) became very popular. It provides a variety of techniques to model and generate scenarios of reservoirs for a given geological region. In contrast to traditional parametric techniques based on variogram, which make use of two points statistics, MPS is non-parametric and based on higher-order statistics to describe complex structures. It generates less artificial simulations, having more realistic geological characteristics (Mariethoz and Caers, 2014).

The source of these statistics and a fundamental concept in this area is the *training image* (TI), which typically represents a specific geological region of interest. The TI could be a hand draw made by a specialist, such as a geologist, or obtained by the application of another type of technique, such as a Boolean model realization (Caers, 2011).

The aim is to generate simulations following the geometry of facies associations seen in the TI while honoring specific constraints related to reservoir data (Chilès and Delfiner, 2012). In fact, these constraints correspond to real measurements (a.k.a. hard data) made in the regions of interest using some suitable equipment. When the hard data are not used/available, the process is called unconditional; otherwise, it is called conditional and every realization must honor these data.

The first methods in literature were based on simulating each pixel (or node) of the realization, while more recent ones follow an approach that is called patch-based, because it's highly focused in the extraction and reproduction of a contiguous group of pixels from the TI.

In the geostatistical field, Arpat and Caers (2007) were the first to propose working with patterns. Generally, in a pattern-based MPS approach, a realization (scenario) is built through the execution of a loop where the two following steps are performed several times: (i) a location of a certain size/shape of the realization under construction is selected; (ii) this location is replaced with a similar pattern of the same size/shape from the TI. The locations selected from the realization are known in the MPS literature as *data events*. The similarity between patterns and data events is defined according to some similarity/distance measure (e.g. Euclidean distance) (Arpat and Caers, 2007).

Fig. 1 illustrates this process by considering a TI containing black and white facies. From left to right, it shows a realization, a data event defined at a given location and its comparison with patterns of the TI. One of these patterns is chosen and pasted at that location. Note that blue values in realization correspond to regions that are not yet filled.

In typical applications where MPS is employed, a large number of realizations have to be generated, so that the time taken to perform each one should be as low as possible. This is a very sensitive problem and a central point when performing a geostatistical study. Therefore, this motivates the study and application of new algorithmic techniques to speed up the process.

In this paper, we introduce LSHSIM, a new method that generates

[*] Corresponding author.
E-mail address: pedrosouzamoura@gmail.com (P. Moura).

http://dx.doi.org/10.1016/j.cageo.2017.06.013
Received 23 November 2016; Received in revised form 6 June 2017; Accepted 14 June 2017
Available online 24 June 2017
0098-3004/ © 2017 Elsevier Ltd. All rights reserved.

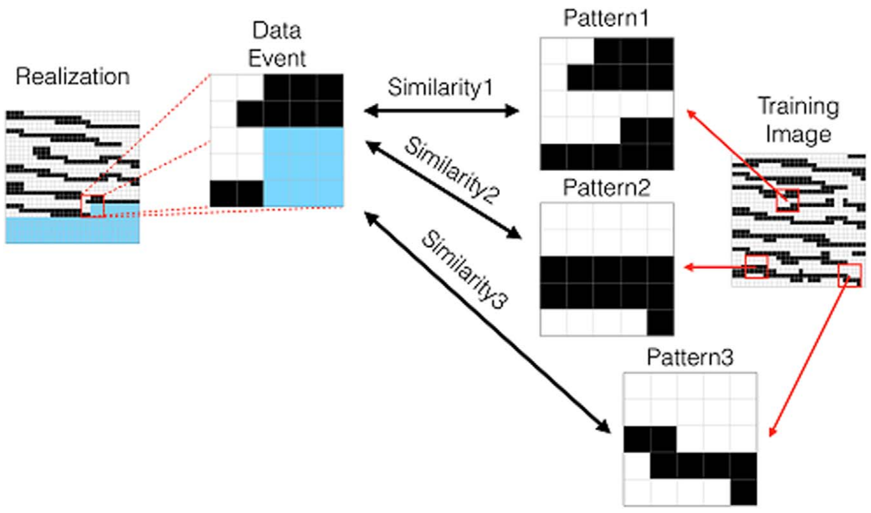


Fig. 1. General structure of a pattern-based approach.

realizations faster than MS-CCSIM (Tahmasebi et al., 2014), which is currently the state-of-the-art method. The key innovations introduced by LSHSIM are the application of the *Locality Sensitive Hashing* (LSH) technique to filter patterns similar to a given data event and the use of a compression technique, based on *Run-Length Encoding* (RLE), to speed up the calculation of similarity between patterns when the image is categorical. Our experimental study suggests that our method produces realizations with similar quality in almost one order of magnitude faster than MS-CCSIM. In addition, LSHSIM also guarantees a good variability among the generated realizations.

Our paper is organized as follows: in Section 2, we briefly present previous related works, following their evolution since the inception of MPS, and discuss in more details the MS-CCSIM method. In Section 3 we give some background that is necessary for the understanding of our proposal. In Section 4, we introduce LSHSIM, describing each of its components. In its turn, in Section 5, we discuss our computational experiments, regarding computational time, realization's quality and variability. Finally, in Section 6 we present our conclusions.

2. Related work

The first MPS methods followed a pixel by pixel motivation (Guardiano and Srivastava, 1993). The adoption of pattern-based approaches lowered the computational time and improved the quality of realizations. However, they introduced a new difficulty, the high dimension of the patterns. SIMPAT (Arpat and Caers, 2007) indexed in a list all possible patterns in a TI, so as to cope with this issue. In its turn, the FILTERSIM method (Zhang et al., 2006) proposed a clusterization based on image features.

DISPAT (Honarkhah and Caers, 2010), which can be seen as an extension of SIMPAT and FILTERISM, is another important method in the development of MPS algorithms.

In the CCSIM method, Tahmasebi et al. (2012) proposed the use of the cross-correlation distance (convolution) in association with a raster path simulation. In this kind of simulation, patterns that have an *overlap area* similar to a given data event are pasted in realization. As an example, for the data event shown in Fig. 1, its non-blue values correspond to an overlap area of size 2. They also claimed that the adopted distance captures better the similarity between patterns and its calculation is performed in the spatial domain, i.e., applying a naive convolution directly from the formula. In this way, they were able to generate better simulations than previous methods. Concerning the conditioning, the method performs sequential subdivisions in the template size, so as to find a pattern honoring the hard data.

The work of Gardet et al. (2016) also applies a K-Means technique to cluster patterns and thus accelerate its search. It also proposes the use of a wavelet decomposition to reduce the time required to compute distances, defining a similarity measure over the decomposed patterns. They compared their method with CCSIM and reported a wider variability, but a worse pattern reproduction.

Recently, Abdollahifard (2016) proposed the FPSIM method which explores two points: (i) a new path strategy that prioritizes data-events placed in the contour between the filled and empty regions of a realization; (ii) a search scheme that is based on the gradient vector of the central pixel of data-events. This search first compares this gradient vector with the gradient of each TI's pixel, in order to obtain a set of candidate patterns, and then performs a search in this set using the Euclidean distance. The authors claim to reduce the search space up to hundreds of times.

The search phase of LSHSIM resembles that of FPSIM in the sense that it first filters patterns that are likely to be similar to a given data event and then it looks for a good candidate in the filtered set. Besides, the reduction on the search space can be controlled by a parameter α. As an example, for the experiments with 2D categorical TI's, presented in Section 5, we use $\alpha = 0.5\%$, which reduces the original space of patterns by a factor of at least 200 and, hence, is comparable to the reduction of hundreds of times reported in Abdollahifard (2016). In Section 1 of the Supplementary Material, we discuss the relation of our methods with those presented in the recent papers by Yang et al. (2016) and Abdollahifard and Nasiri (2017).

2.1. Review of the MS-CCSIM algorithm

The MS-CCSIM (Tahmasebi et al., 2014) is an extension of the CCSIM method that introduces two new ideas that accelerate the search for a pattern and the convolution's calculation: (i) the use of a multi-scale approach, in which the TI is represented in increasingly different resolutions and so the search space of a query is reduced; and (ii) the calculation of the cross-correlation function in the frequency domain using the fast Fourier transform (FFT) (Cooley and Tukey, 1965).

In addition to that, the MS-CCSIM adopts a raster path, which brings some problems when dealing with hard data. For this reason, the method employs the idea of a co-template, such as proposed by Parra and Ortiz (2011). It is a way of “looking ahead”, trying to verify if there is some hard data lying ahead of the path. It selects then training patterns whose co-patterns satisfy these constraints.

Another important issue brought by this method was the approach to the patchiness problem, which typically brings discontinuities to generated realizations. Aiming to deal with this question, it applies the technique of minimum error boundary cut, originally proposed in the Image Quilting method by Efros and Freeman (2001), which was tailored for the texture synthesis area. However, this approach has some limitations and this fact was later discussed and addressed by Tahmasebi and Sahimi (2016a), who applied a graph network formulation to this problem. Tahmasebi and Sahimi (2016b) describe some of the advantages and disadvantages of raster path algorithms, as well as other strategies for dealing with hard data, other than co-template. Mahmud et al. (2014) also worked on this issue, proposing an extension of the Image Quilting method to conditioning and to 3D images, while having other similar characteristics to the CCSIM method.

3. Background

In this section we discuss some concepts that are required to understand our work. We first discuss how to address the problem of finding similar patterns using LSH and then we discuss the Hamming similarity and how to calculate it over patterns compressed with RLE. Those acquainted with the LSH scheme may skip the Section 3.1.

3.1. Locality Sensitive Hashing

As mentioned in Section 2, one of the challenges to implement the pattern-based approach is the high dimensionality of data. To address this issue, we propose the application of the so called *Locality Sensitive Hashing* (LSH).

In order to explain the technique we first recall that a hash table is a data structure that implements an associative array: given an object x, a hash function $\boldsymbol{h}(\cdot)$ is used to determine the position in the structure/array where we can find information about x (see Cormen et al., 2009).

The LSH was first proposed by Indyk and Motwani (1998) and Gionis et al. (1999). Given a set of elements S and a set of buckets B, a family $\mathcal{H}$ of functions $h: S \to B$, together with a distribution probability $\mathcal{D}$ over the functions in $\mathcal{H}$, is a LSH for a similarity measure $s(\cdot, \cdot)$ if, for any $x, y \in S$, we have

$$Pr[h(x) = h(y)] = s(x, y),$$

where the probability is taken according to the distribution $\mathcal{D}$. This way similar elements have large probability to be assigned to the same bucket while non-similar ones have a small probability.

One of the main applications of LSH is as a tool to address the *Approximate Nearest Neighbor* (ANN) problem (Gionis et al., 1999). This problem admits the following formulation:

Input. A set of points S, a query point q and a value $\epsilon > 0$.

Output. A point $p \in S$ such that $s(q, p) \geq (1 - \epsilon)s(q, S)$, where $s(q, S)$ is the similarity of q to its most similar point in S.

The ANN problem naturally arises in the context of pattern based simulation since a key operation in this kind of simulation consists of finding patterns that are (very) similar to a given data event.

To address the ANN problem, via the LSH approach, we have two phases:

- *Preprocessing Phase*. In this phase K hash functions are randomly selected from $\mathcal{H}$ using the probability distribution $\mathcal{D}$. Let $h_1, h_2 \ldots h_K$ be the chosen functions and $\boldsymbol{h} = h_1 h_2 \ldots h_K$ be the function obtained by the concatenation of these functions. Then, $\boldsymbol{h}$ is used to build a hash table that maps each $x \in S$ into a bucket $\boldsymbol{h}(x) \in B$. This procedure is repeated L times so that we end up with L hash tables, each of them storing all the elements in S.
- *Search Phase*. Given a point q, we find its bucket/position in each one of the L hash tables using the hash function $\boldsymbol{h}$. Let C_q be the set of points that are mapped to the same bucket of q in at least one of the L hash tables. Then, we can either return an arbitrarily chosen point in C_q or return the most similar element to q among those in C_q. The latter possibility increases the chance of returning patterns that are more similar to q but it is more expensive in terms of computational time. Another possibility in the search phase is to return the most similar point after inspecting some fraction of the points in C_q. This way we trade-off between the quality of the returned point and the computational time.

By choosing the values of K and L properly it is possible to guarantee a high probability of returning a point that is among the most similar to the query q with respect to the similarity $s(\cdot, \cdot)$.

3.2. LSH for Hamming and Euclidean distance

A natural way to measure similarity among categorical data is through the Hamming similarity (Hamming, 1950). For two vectors $p = (p_1, \ldots, p_n)$ and $q = (q_1, \ldots, q_n)$, the Hamming similarity is defined as the ratio between the number of coordinates in which p and q match and n (e.g. if p = (a,b,b) and q = (a,c,b), then Hamming(p, q) = 2/3).

Let S be a set of points in a n-dimensional space. In addition, for i = 1,...,n, let h_i: $S \mapsto R$ be a function that maps each $x \in S$ into x_i, which is the i-th coordinate of x. A well known result in the theory of LSH states that the family $\mathcal{H} = \{h_1, \ldots, h_n\}$, together with a uniform distribution $\mathcal{D}$ over $\mathcal{H}$, is a LSH scheme for the Hamming similarity. This scheme is used by LSHSIM for categorical images, so as to filter patterns that are similar to a given data event.

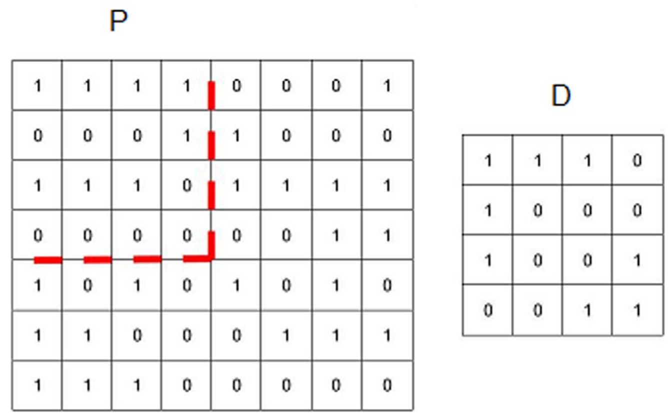
On the other hand, for continuous data, we employ the Euclidean distance, which has been used in pattern-based methods since the work of Arpat and Caers (2007). In the LSH scheme for the Euclidean distance, each hash function h in the family $\mathcal{H}$ is associated with a random line in the n-dimensional space. Given a constant a, this line is divided into segments of length a, which correspond to the buckets. Each $x \in S$ is then projected onto the line and hashed to the bucket concerning the segment in which it lies.

3.3. Computing similarities over compressed patterns

One important characteristic of categorical TI's is that they contain a small number of facies. This means that its pixels assume few distinct values, which contributes to their compressibility. This scenario motivates the application of compression techniques that have a special structure for speeding up the calculation of similarities between patterns. In Laber et al. (2016), an in-depth study was carried on and it was concluded that for a certain range of template sizes, that are reasonable to use in practice, the calculation of the cross-correlation between training images and templates compressed with the RLE technique is even faster than a FFT based convolution. Here, we show how to extend this technique to efficiently compute the Hamming similarity between data events and patterns.

The RLE is a simple technique used for compressing sequences that have many repetitions among consecutive symbols. As an example, the Fig. 2 exhibits a small TI, with facies 0 and 1, and a pattern (or block) P of size 4 × 4 delimited by the red dashed line. If we scan P row by row continuously, its RLE is $\{(5, 1), (3, 0), (3, 1), (5, 0)\}$, where the first value in each pair denotes the number of repetitions and the second one denotes the facie. When compressing P, we followed a horizontal continuous scan of the block. However, depending on the image's characteristics, other types of scanning orders would result in a better compression of its blocks. This issue was explored in Laber et al. (2016).

By storing the RLE representation of the patterns of a TI, we can calculate the Hamming similarity between a data event D and each pattern in a given list $(P_1, \ldots, P_m)$ in time proportional to $|D| + \sum_{i=1}^{m} p_i^R$, where $|D|$ is the size of D and p_i^R is the size of the RLE representation for P_i. For that, it is enough to preprocess D to obtain a 3-dimensional structure *Sum* where $Sum[f, i, j]$ stores the number of times facie f occurs between the i-th and the j-th position of D. Then, the Hamming similarity between a data event D and a pattern P, with RLE representation $\{(c_1, v_1), \ldots, (c_k, v_k)\}$, is given by



P

1	1	1	1	0	0	0	1
0	0	0	1	1	0	0	0
1	1	1	0	1	1	1	1
0	0	0	0	0	0	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	1	1	1
1	1	1	0	0	0	0	0

D

1	1	1	0
1	0	0	0
1	0	0	1
0	0	1	1

Fig. 2. An example TI and a possible pattern. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

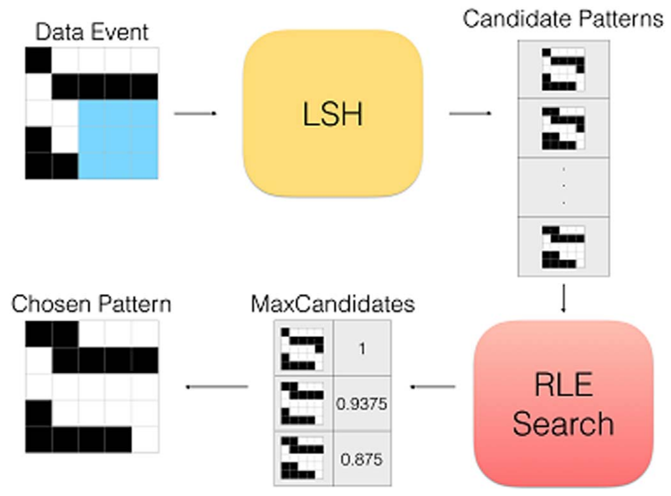


Fig. 3. General search procedure of LSHSIM.

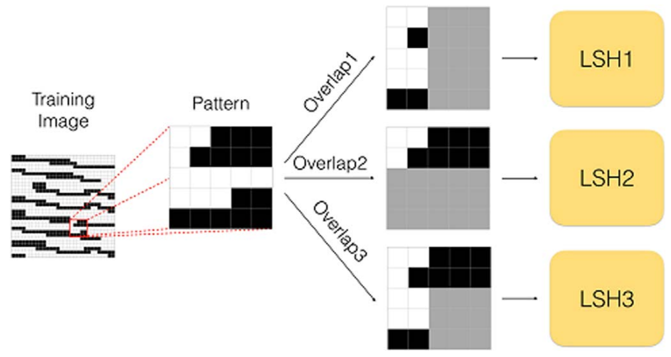


Fig. 4. Preprocessing phase of LSHSIM.

$$HammingSimilarity(P,\, D) = \sum_{i=1}^{k} Sum\left[v_i,\, \sum_{j=1}^{i-1} c_j,\, \left(\sum_{j=1}^{i} c_j\right) - 1\right].$$

As an example, for the data event D and the pattern P of Fig. 2, we have

$$\begin{aligned} HammingSimilarity(P,\, D) \;=\;& Sum[1,\, 0,\, 4] + Sum[0,\, 5,\, 7] + Sum[1,\, 8,\, 10] \\ &+ Sum[0,\, 11,\, 15] = 8 \end{aligned}$$

4. LSHSIM

Two points are often considered by MPS methods available in the literature: (i) the choice of the similarity measure and (ii) how to efficiently find a pattern in the TI that is (very) similar to a given data event. To address (i), we use the Hamming similarity for categorical images and the Euclidean distance for continuous images. With regard to the second point, we propose the application of the LSH scheme to filter patterns that are likely to be similar to a given data event, followed by an exhaustive search. This search is used to find the most similar patterns among the filtered ones and is based on the RLE similarity calculation when the TI is categorical. Our techniques can be adapted to work together with different types of simulation paths as random or raster paths. Here, we explain how they are used with raster paths.

The pseudocode of our method for categorical TI's is presented in Algorithm 4.1. Further, we explain how to modify it for handling continuous TI's. In line 2, the set of hash tables for the LSH scheme is built. The details of how LSHSIM applies this scheme are given in Sections 4.1 and 3.2. In line 3 each pattern of the TI is compressed using the RLE method described in Section 3.3. We observe that these two lines, that are computationally expensive, just need to be executed once in the usual case where multiple realizations are generated.

In line 4, a raster path is defined based on the template size, size$_T$, and the overlap size, size$_{OL}$. In this step, our method chooses a random corner of the realization as a starting point, as well as a random direction (between horizontal or vertical), to generate the path. For each location `u` defined along the path, the corresponding data event dataEvent$_u$ is extracted from the realization `R` and then the search phase of the LSH scheme is executed (lines 6 and 7) so that the set `cand`, which is supposed to contain patterns similar to dataEvent$_u$, is obtained. Note that the size of this set `cand` is limited to at most the value of α times the number of TI patterns, where α is a positive value smaller than 1. If this set is not empty, the Hamming similarity is calculated between the data event and each of the filtered patterns using the RLE approach (lines 8–9). Otherwise, the same approach is applied over the compressed TI, considering only a fraction α of all the patterns of the TI (lines 10–12), thus reducing the search space. In both cases, the subset `bestCand`, containing the `MaxCandidates` most similar candidates, is obtained. Finally, in lines 13 and 14, a random pattern from this set is chosen and pasted in realization at location `u`. Fig. 3 provides an overview of our method.

For continuous data, the Algorithm 4.1 requires some small changes: line 3 is not executed, because the RLE method does not apply for this case. Lines 8 and 9 perform a non-compressed search, calculating the Euclidean distance between the data event and each filtered pattern. Lines 10 - 12 perform a non-compressed search in the original TI, considering only a fraction α of all its patterns.

Algorithm 4.1. Pseudocode for LSHSIM.

```
     Result: Realization R
1    LSHSIM (ti, size_T, size_OL, maxCandidates, K, L, α)
2      PreprocessLSH(ti, size_T, size_OL, K, L)
3      compressedTI ← PreprocessRLE(ti, size_T, size_OL)
4      path ← generateRasterPath(size_T, size_OL)
5      for each  location u ∈ path do
6            dataEvent_u ← R(u)
7            cand ← applyLSH(dataEvent_u, K, L, α)
8            if  cand ≠∅
9      bestCand ← exhaustiveSearchCandidatesSet(dataEvent_u,
     cand, maxCandidates)
10           else
11     bestCand ← exhaustiveSearchTrainingImage(dataEvent_u,
     compressedTI, α, maxCandidates)
12     end  if
13     chosenPat ← drawRandom(bestCand)
14     R(u) ← chosenPat
15     end  for
16   return  R
```

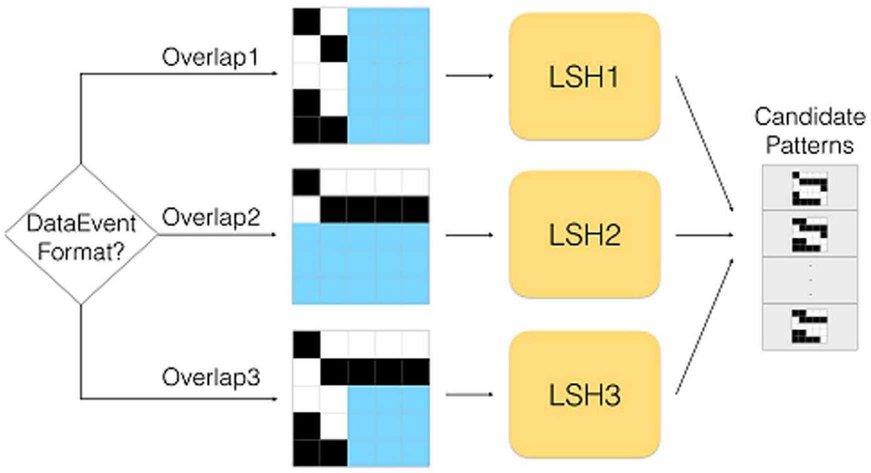


Fig. 5. Search phase of LSHSIM.

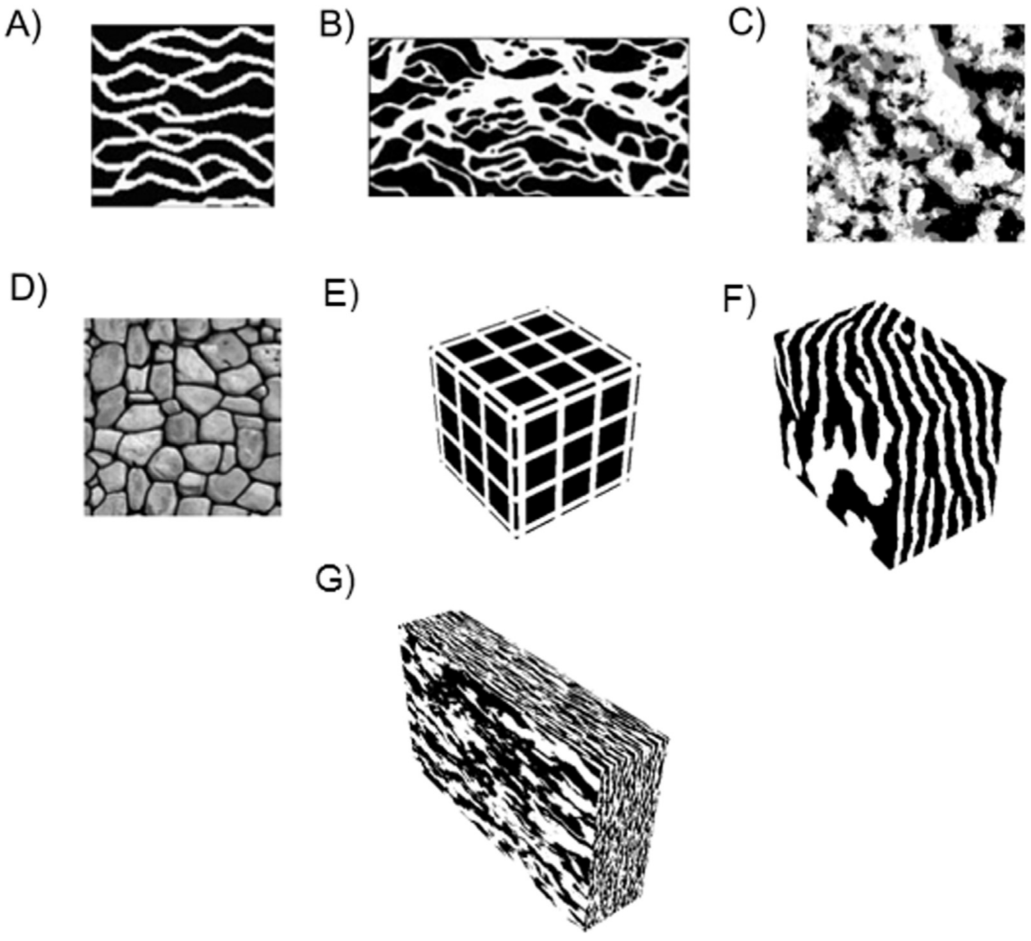


Fig. 6. Training images adopted in our experiments: available in TrainingImagesLibrary (2016).

Table 1
Main features of the images used for the experimental study.

TI	Image size	Dimensions	Type
(A) Strebelle	250 × 250	2D	Binary
(B) Bangladesh	768 × 243	2D	Binary
(C) C_Wlticat	400 × 400	2D	Ternary
(D) Stonewall	200 × 200	2D	Continuous
(E) Checker	50 × 50 × 50	3D	Binary
(F) Fold_Categorical	180 ×150 × 120	3D	Binary
(G) Maules_Creek	340 × 200 × 80	3D	Binary

4.1. Filtering patterns via LSH

In the preprocessing phase (line 2 of Algorithm 4.1), LSHSIM builds 3 sets of LSH tables as explained in Section 3.1. Each set corresponds to one of the 3 possible types of overlap regions described in Tahmasebi et al. (2012). Thus, for each pattern of a given size in the TI, the method extracts three regions and inserts each of them in the corresponding set of LSH tables. Fig. 4 illustrates this phase using a TI with two facies (black and white), template size of 5 × 5 and overlap

Table 2
Average realization time in milliseconds for 2D categorical images.

Image	Real. size	Temp. size	Overlap	LSHSIM	MS-CCSIM	Ratio
Strebelle	256 × 256	16 × 16	4	**11.85**	106.62	9.00
Strebelle	256 × 256	32 × 32	4	**3.82**	29.40	7.70
Strebelle	256 × 256	32 × 32	8	**4.91**	36.34	7.40
Strebelle	400 × 400	16 × 16	4	**29.64**	262.31	8.85
Strebelle	400 × 400	32 × 32	4	**9.59**	71.68	7.47
Strebelle	400 × 400	32 × 32	8	**12.79**	93.60	7.32
Bangladesh	256 × 256	16 × 16	4	**32.29**	272.92	8.45
Bangladesh	256 × 256	32 × 32	4	**11.46**	56.86	4.96
Bangladesh	256 × 256	32 × 32	8	**14.11**	70.90	5.02
Bangladesh	400 × 400	16 × 16	4	**78.78**	671.19	8.52
Bangladesh	400 × 400	32 × 32	4	**28.47**	139.93	4.91
Bangladesh	400 × 400	32 × 32	8	**35.88**	183.92	5.13
C_Wlticat	256 × 256	16 × 16	4	**40.17**	254.90	6.35
C_Wlticat	256 × 256	32 × 32	4	**23.08**	50.46	2.19
C_Wlticat	256 × 256	32 × 32	8	**26.36**	65.98	2.50
C_Wlticat	400 × 400	16 × 16	4	**100.38**	597.56	5.95
C_Wlticat	400 × 400	32 × 32	4	**55.53**	130.02	2.34
C_Wlticat	400 × 400	32 × 32	8	**58.81**	169.96	2.89

Table 3
Preprocessing time in milliseconds for 2D categorical images.

Image	Real. size	Temp. size	Overlap	Preprocessing time
Strebelle	256 × 256	16 × 16	4	418.08
Strebelle	256 × 256	32 × 32	4	510.90
Strebelle	256 × 256	32 × 32	8	519.48
Strebelle	400 × 400	16 × 16	4	426.66
Strebelle	400 × 400	32 × 32	4	509.34
Strebelle	400 × 400	32 × 32	8	520.26
Bangladesh	256 × 256	16 × 16	4	1460.95
Bangladesh	256 × 256	32 × 32	4	1948.45
Bangladesh	256 × 256	32 × 32	8	1878.25
Bangladesh	400 × 400	16 × 16	4	1469.53
Bangladesh	400 × 400	32 × 32	4	1909.45
Bangladesh	400 × 400	32 × 32	8	1853.29
C_Wlticat	256 × 256	16 × 16	4	1835.35
C_Wlticat	256 × 256	32 × 32	4	2666.06
C_Wlticat	256 × 256	32 × 32	8	2737.04
C_Wlticat	400 × 400	16 × 16	4	1830.67
C_Wlticat	400 × 400	32 × 32	4	2650.46
C_Wlticat	400 × 400	32 × 32	8	2695.7

Table 4
Preprocessing and realization times in milliseconds for 2D continuous image.

Image	Real. size	Temp. size	Overlap	Preproc. Time	Real. time
Stonewall	256 × 256	16 × 16	4	4034.97	101.40
Stonewall	256 × 256	32 × 32	4	6442.84	31.98
Stonewall	256 × 256	32 × 32	8	9906.84	67.08
Stonewall	400 × 400	16 × 16	4	4197.21	278.46
Stonewall	400 × 400	32 × 32	4	6364.06	79.56
Stonewall	400 × 400	32 × 32	8	9773.46	180.18

size of 2. The second image, from left to right, is an arbitrarily chosen pattern, say *P*, from the TI. On its right side, we have three images, in each of them the non-gray values represent a possible overlap area of *P*. These areas are inserted in the corresponding LSH table.

In the search phase (line 7 of Algorithm 4.1), it first verifies the type of overlap of the data event dataEvent_u and then search in the corresponding set of LSH tables, such as illustrated in Fig. 5. In order to speed up the search, the size of the returned set, `cand`, is limited by a fraction α of all possible patterns of the TI.

It can be proved that the probability of including a pattern *P* in the set `cand` (line 7) is given by

$$1 - (1 - Sim(P, \text{dataEvent}_u)^K)^L$$

Thus, *K* and *L* shall be defined in order to guarantee that patterns similar (non-similar) to dataEvent_u have a large (small) probability of being included in `cand`. As an example, by setting $K = 10$ and $L = 30$, the probability of including a pattern with similarity 0.8 is 95% while the probability of including a pattern with similarity 0.5 is less than 3%.

The extension of LSHSIM to 3D TI's makes use of 7 sets of LSH tables, each set corresponding to one of the 7 possible types of overlap regions. Apart from that, the preprocessing and search phases proceed analogously as described above.

4.2. Conditioning

We adapted LSHSIM so as to consider conditioning data. In this sense, we introduced an additional filter when searching for a given data event. After applying the LSH scheme and obtaining the set of candidate patterns, we filter this set to those patterns which honor all the hard data associated with the data event. Finally, we perform a RLE based search in this reduced set of candidates, looking for the most similar ones to the data event in the overlap region. In case this reduced set of candidate patterns is empty, we perform a RLE search in the training image.

It shall be noted that, in order to avoid low quality realizations, the α parameter should be increased with respect to unconditional simulations, since we now have this additional filter that restricts the set of candidate patterns to those which honor the hard data.

The experiments performed showed that LSHSIM is able to achieve good quality realizations while honoring conditioning points. The computational times are higher than those for unconditional realizations due to the increased value of α used. These experiments are described in details in Section 2 of the Supplementary material.

5. Experimental study

In our experiments, we considered a set of four 2D and three 3D TI's available in (TrainingImagesLibrary, 2016), so as to evaluate our proposed solution. The images are presented in Fig. 6, while their main features are described in Table 1. The image (A) is the well known TI proposed by Strebelle (2002), while the image (C) is a ternary and less compressible one.

The Stonewall image (D) was selected to validate our method with continuous data. Lastly, the images (E), (F) and (G) are categorical 3D TI's used to validate our method with 3D models.

All experiments were executed under the following settings of hardware and software: Intel Core i7-3960X CPU @ 3.30 GHz running Windows 7 64 bits, with 32 GB of memory. All codes of our method

Table 5
Preprocessing and realization times in seconds for 3D images.

Image	Real. size	Temp. size	Overlap	Preproc. time	Real. time
Checker	256 × 256 × 256	10 × 10 × 10	2	1.88	3.88
Checker	256 × 256 × 256	12 × 12 × 12	4	1.84	5.67
Checker	256 × 256 × 256	16 × 16 × 16	4	1.53	2.69
Checker	400 × 400 × 400	10 × 10 × 10	2	1.89	15.37
Checker	400 × 400 × 400	12 × 12 × 12	4	1.85	22.43
Checker	400 × 400 × 400	16 × 16 × 16	4	1.52	10.24
Fold_Categorical	256 × 256 × 256	10 × 10 × 10	2	76.82	165.75
Fold_Categorical	256 × 256 × 256	12 × 12 × 12	4	81.18	221.04
Fold_Categorical	256 × 256 × 256	16 × 16 × 16	4	91.99	102.85
Fold_Categorical	400 × 400 × 400	10 × 10 × 10	2	74.20	651.95
Fold_Categorical	400 × 400 × 400	12 × 12 × 12	4	71.09	804.84
Fold_Categorical	400 × 400 × 400	16 × 16 × 16	4	84.47	394.16
Maules_Creek	256 × 256 × 256	10 × 10 × 10	2	128.76	379.24
Maules_Creek	256 × 256 × 256	12 × 12 × 12	4	143.82	467.81
Maules_Creek	256 × 256 × 256	16 × 16 × 16	4	188.30	242.48
Maules_Creek	400 × 400 × 400	10 × 10 × 10	2	145.97	1588.56
Maules_Creek	400 × 400 × 400	12 × 12 × 12	4	153.37	1937.34
Maules_Creek	400 × 400 × 400	16 × 16 × 16	4	172.87	877.82

were implemented in C++. Regarding the MS-CCSIM method, we adopted the following strategy: we used the MATLAB code available in (MS-CCSIM, 2016) to generate realizations and we also implemented a version in C++, employing the OpenCV library (OPENCV, 2016), in order to compare its computational time with LSHSIM's time. Note that this library is an optimized code belonging to the computer vision area, having very efficient implementations for some of the techniques required to implement MS-CCSIM as the fast Fourier transform (FFT) and multi-scale algorithms.

For parameterization of the MS-CCSIM method, both in MATLAB and C++ implementations, we set the number of scales to 3, which is the highest in its MATLAB code. Regarding LSHSIM, when dealing with categorical TI's, we defined L and K, the LSH parameters, to 30 and 10, respectively. On the other hand, for continuous TI's, we set L and K to 30 and 8, respectively. For both methods, we also set `MaxCandidates` to 10, while varying template and overlap sizes according to the experiment being made.

To determine a suitable value of the α parameter, that is to say, the one that achieves a good balance between computational time and realization's quality, we performed several experiments for different configurations of template and overlap sizes. We end up with α equals to 0.5% for categorical 2D TI's, 1% for categorical 3D TI's and 5% for the continuous 2D TI.

Both MATLAB and C++ implementations of MS-CCSIM apply the minimum-error boundary cut approach to the patchiness problem. For the sake of a fair comparison, we also employ this method in our implementation of LSHSIM.

5.1. CPU performance

In this subsection, we evaluate the performance of our method regarding the computational time for generating realizations. More specifically, for each TI under consideration, we generated 20 realizations for different configurations of template and overlap sizes.

We then measured the time taken for performing each realization and calculated its average. For 2D categorical TI's, we compare the performance of LSHSIM with our implementation of the MS-CCSIM in C++. Table 2 shows these times in milliseconds, where the best one for each configuration is in bold.

For binary images, LSHSIM was able to outperform MS-CCSIM by a factor of approximately 7 on average. The difference was bigger for the Strebelle image, the most compressible one, for which our method was 8 times faster. Regarding the ternary one, our method was 3.70 times faster than MS-CCSIM on average, ranging from 2.19 to 6.35 times. This difference is explained by the fact that this image is less compressible and hence each exhaustive search, which uses the RLE similarity calculation, takes longer. We shall note that we are not taking into account the preprocessing time in this specific evaluation.

Table 3 exhibits, for the same configurations, the preprocessing time required for building the LSH data structure and applying the RLE compression to the training image. This preprocessing time is on average equivalent to the time of 48 realizations, which yields a non-negligible overhead for applications that only require the generation of a few realizations. For applications that involve a large number of simulations the preprocessing time of LSHSIM becomes almost irrelevant. The results of the experiments discussed so far, with MS-CCSIM and LSHSIM, suggest that the latter outperforms the former for applications where more than a dozen of realizations have to be generated. Moreover, the larger the number of realizations the larger is the advantage towards LSHSIM.

With regard to continuous data, Table 4 exhibits, for the same configurations as above, the preprocessing and realization times in milliseconds using LSHSIM for the Stonewall TI. It can be noted that our method was able to obtain satisfactory realization times for this continuous TI.

Finally, Table 5 gives the preprocessing and realization times in seconds obtained by applying LSHSIM to the 3D TI's for some selected configurations. These times are three to four orders of magnitude larger than the ones exhibited in Table 2, for 2D images. However, this is not surprising since the sizes of the 3D realizations are about two to three orders of magnitude larger than the one for 2D images. We shall remark that, for these 3D images, the preprocessing time of our method is generally much smaller than the time taken for performing a single realization.

5.2. Realization's quality

We now analyse LSHSIM concerning simulation's quality. For this purpose, we compare LSHSIM's simulations with MS-CCSIM's for the configurations defined in the last section. In addition, we also show realizations of MS-CCSIM using only 1 scale, since it improves its quality, although at the cost of increasing the computational time by a factor of approximately 10 with respect to the times presented in the previous section.

Fig. 7 (A), (B) and (C) shows two realizations generated with LSHSIM, MS-CCSIM with 3 scales and MS-CCSIM with 1 scale, respectively, for the Strebelle TI. Each realization has 256 × 256 pixels, template size of 32 × 32 and overlap of 4. Moreover, Fig. 8 (A), (B) and

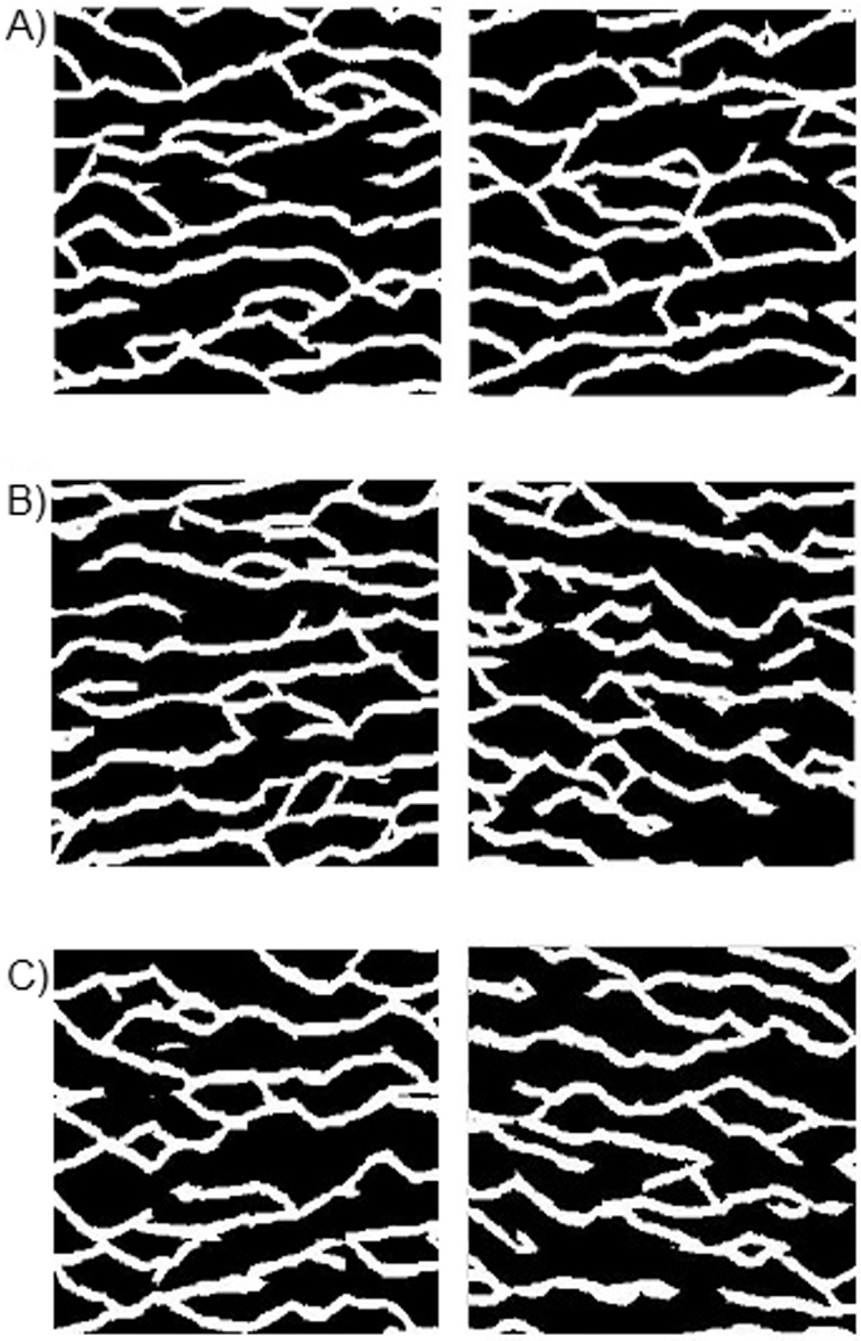


Fig. 7. Unconditional realizations for the TI of Fig. 6 (A): using LSHSIM, using MS-CCSIM with 3 scales (B) and using MS-CCSIM with 1 scale (C).

(C) shows two realizations for the Bangladesh TI using LSHSIM, MS-CCSIM with 3 scales and MS-CCSIM with 1 scale, respectively. Both have 256 × 256 pixels, and they were generated using template size of 32 × 32 and overlap size of 4. Note that these images are resized to better fit the paper.

For these two binary TI's, Strebelle and Bangladesh, we notice that LSHSIM generated realizations with good quality, in the sense that it reproduced well the spatial continuity of the TI's. Both LSHSIM and MS-CCSIM generated realizations containing low level of patchiness, since they employ the minimum-error boundary cut approach.

Fig. 9 (A), (B) and (C) presents two realizations generated with LSHSIM, MS-CCSIM with 3 scales and MS-CCSIM with 1 scale, respectively, for the C_Wlticat image, setting the realization size to 400 × 400 pixels, template size to 16 × 16 and overlap to 4. Again, LSHSIM was able to achieve a good quality, that is to say, representing well the image's characteristics.

LSHSIM was also able to produce realizations with good quality for continuous data. In this sense, Fig. 10 (B) and (C) shows two realizations generated with LSHSIM for the Stonewall TI (A). Each of them has 256 × 256 pixels, template size of 16 × 16 and overlap of 4. One can notice that LSHSIM was able to express well the TI's spatial continuity.

Finally, LSHSIM was also successful for 3D images. Fig. 11 presents realizations for the Checker TI (A), for the Fold_Categorical TI (B) and for the Maules_Creek TI (C), setting the realization size to 256 × 256 × 256, template size to 16 × 16 × 16 and overlap size to 4. Analogously, Fig. 12 exhibits realizations for the same TI's with 400 × 400 × 400 pixels, template size of 16 × 16 × 16 and overlap of 4.

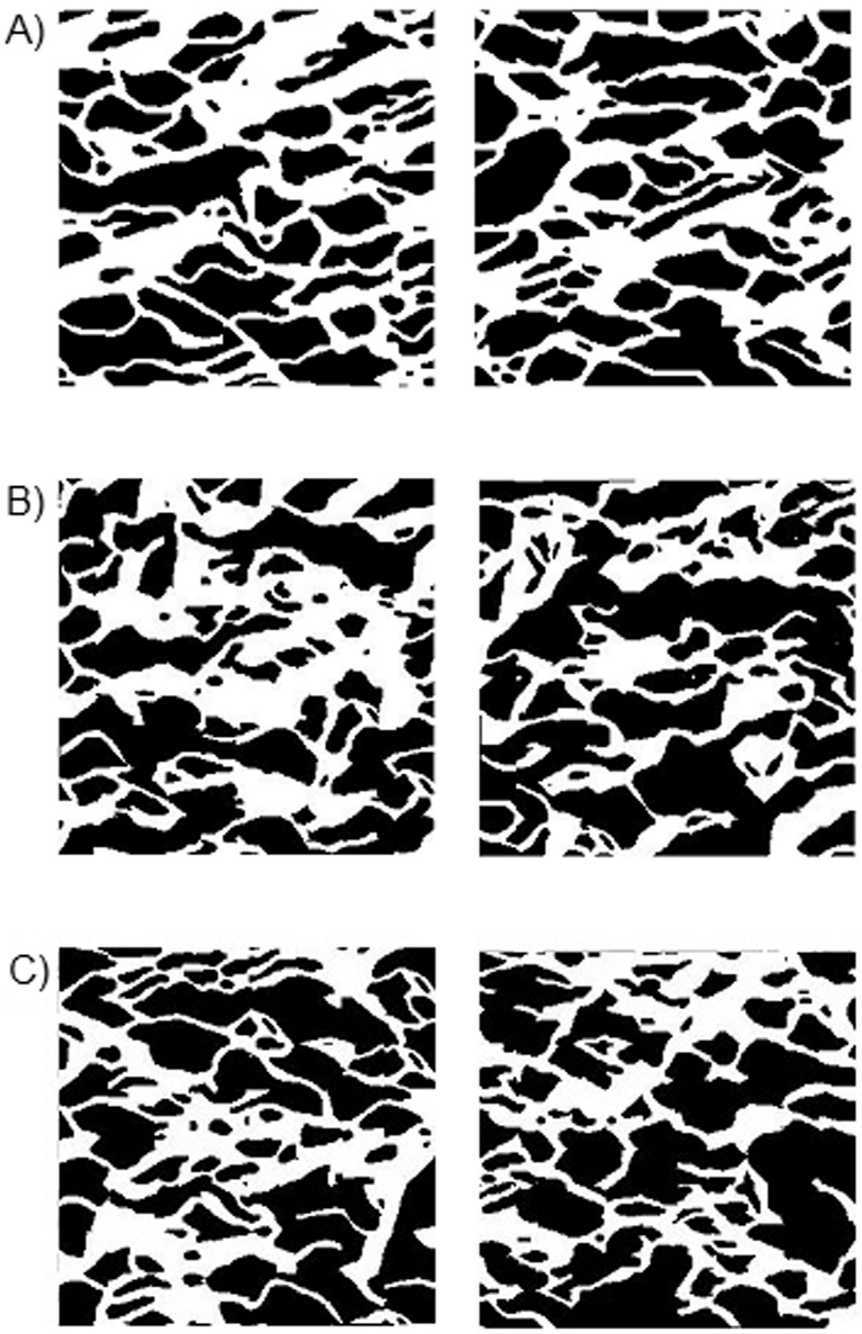


Fig. 8. Unconditional realizations for the TI of Fig. 6 (B): using LSHSIM, using MS-CCSIM with 3 scales (B) and using MS-CCSIM with 1 scale (C).

5.3. *Comparing uncertainty space*

We now analyse LSHSIM's uncertainty space following the analysis of distance (ANODI) method proposed by Tan et al. (2014).

We focus on ANODI's visual approach, which consists of the MDS technique with the Jensen-Shannon divergence as a measure of distance. It represents the realizations and the TI as points in a two or three-dimensional space, where the relative distances between each realization and the TI are preserved as much as possible. We used a MATLAB implementation of ANODI available in ANODI (2016).

We generated 50 realizations with both methods for two TI's. Fig. 13 shows the MDS plot for the Strebelle TI with the following settings: realization size of 256 × 256 pixels, template size of 32 × 32 and overlap of 4. Similarly, Fig. 14 shows the MDS plot for C_Wlticat, which is a ternary image, using a realization of 400 × 400 pixels, a template size of 16 × 16 and an overlap of 4. In each plot, the black dot denotes the TI, while the green and blue points represent realizations generated with LSHSIM and MS-CCSIM, respectively. The numbers close to some points indicate the rank of that realization, among the ones generated by the same method, with respect to the distance to the TI. Note that the axis are not shown because the focus is on the relative distances between points.

One can observe that, for Fig. 13, LSHSIM achieved a good pattern reproduction such that its realizations are close to the TI. In addition, both LSHSIM and MS-CCSIM had similar spreading of their realizations. Concerning the plot depicted in Fig. 14, LSHSIM generated realizations close to the TI, thus reproducing well the TI patterns. Again, both methods had a similar variability, since LSHSIM's space of uncertainty is almost as large as MS-CCSIM's.

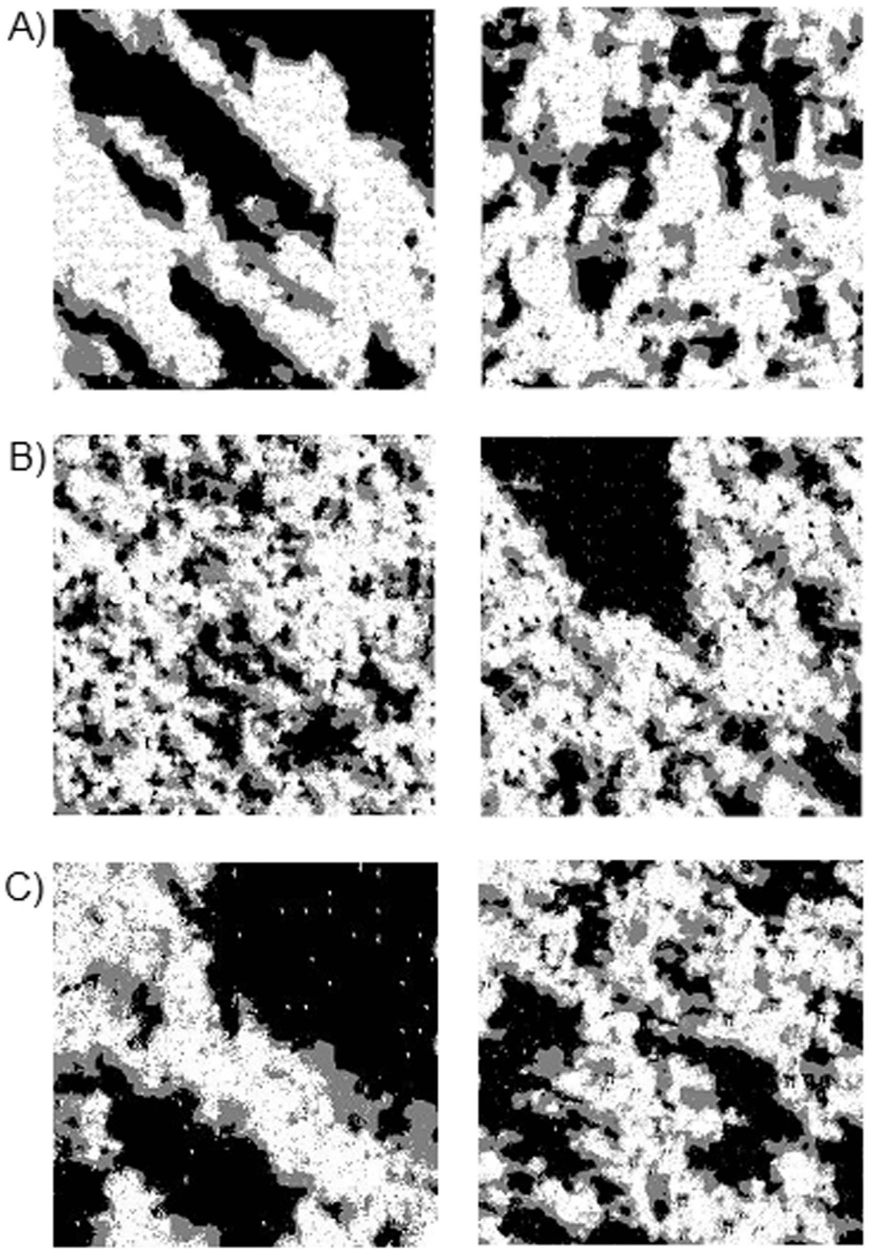


Fig. 9. Unconditional realizations for the TI of Fig. 6 (D): using LSHSIM, using MS-CCSIM with 3 scales (B) and using MS-CCSIM with 1 scale (C).

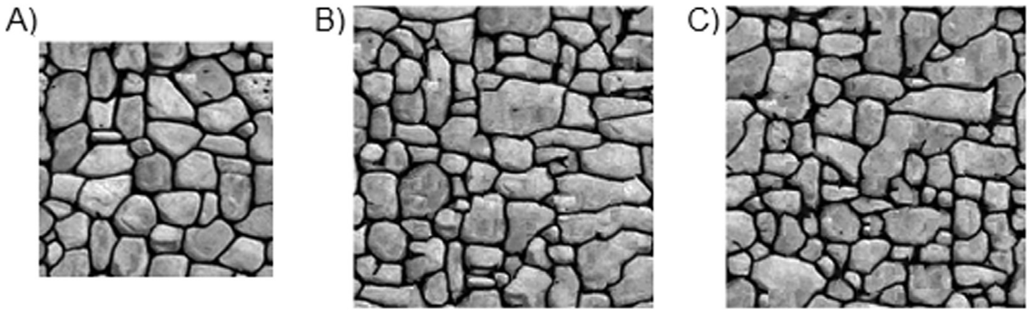


Fig. 10. Unconditional realizations using LSHSIM for continuous data: the Stonewall TI (A) and two generated realizations (B) and (C).

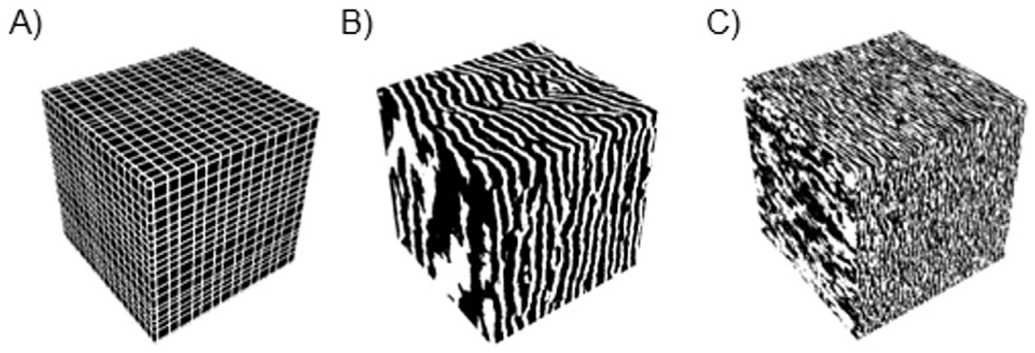


Fig. 11. Unconditional realizations using LSHSIM for 3D data: for the Checker TI (A), for the Fold_Categorical TI (B) and for the Maules_Creek TI (C).

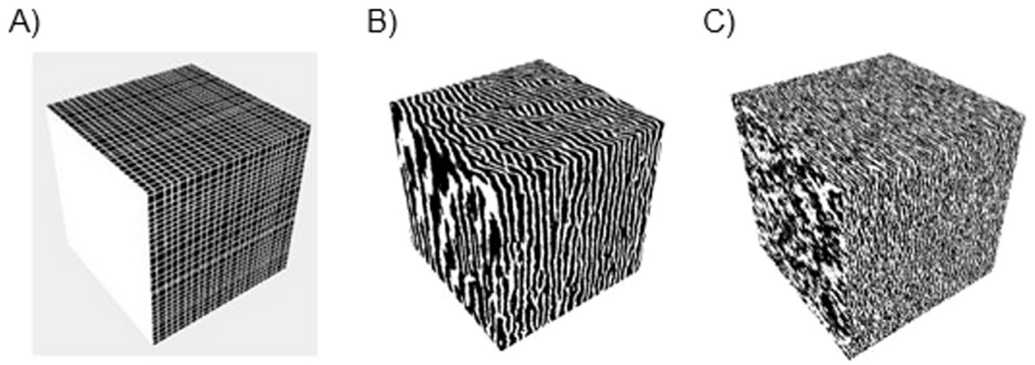


Fig. 12. Unconditional realizations using LSHSIM for 3D data: for the Checker TI (A), for the Fold_Categorical TI (B) and for the Maules_Creek TI (C).

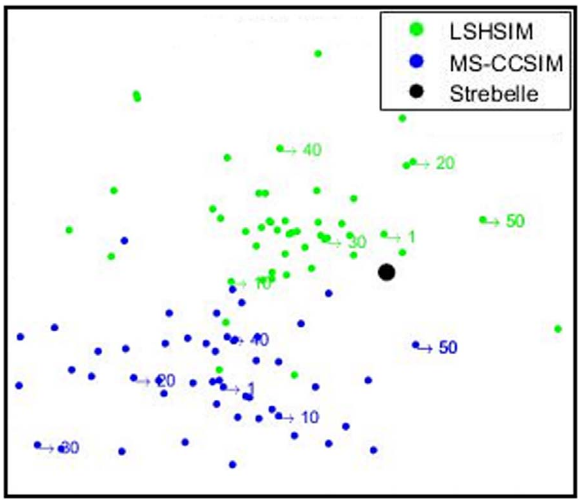


Fig. 13. MDS plot illustrating the variability of LSHSIM and MS-CCSIM methods by using the TI in Fig. 6 (A). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

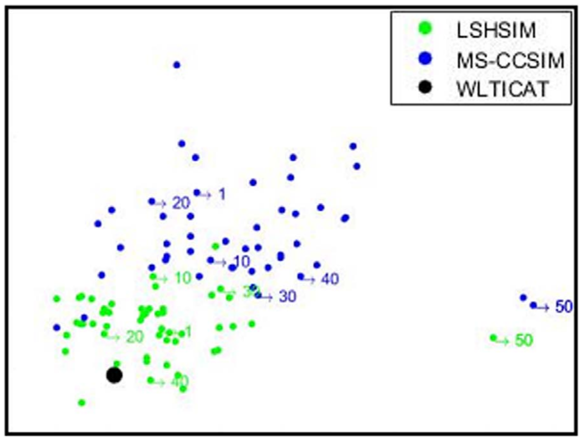


Fig. 14. MDS plot exposing the variability of both methods by using the TI in Fig. 6 (C). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

6. Conclusions

In this paper, we presented LSHSIM, a new and fast method to generate realizations that are based on the characteristics of a given training image. The method introduces new ideas to accelerate the simulation process such as the use of the LSH technique and the RLE based similarity computation. Experiments carried over a set of 7 selected TI's indicate that LSHSIM is almost one order of magnitude faster than MS-CCSIM for categorical images. In addition, the quality of our realizations is competitive with those generated by MS-CCSIM, in the sense that the spatial continuity of the TI's was well expressed. Our MDS plots depicted that LSHSIM's space of uncertainty has a good spread and the realizations are close to the TI. Lastly, as a potential disadvantage, LSHSIM may not be suitable for quickly generating a small number of realizations due to its non-negligible preprocessing time.

Acknowledgement

The authors are grateful to PETROBRAS for partially sponsoring this research under the contract PT120.60.10085 – SAP4600454071.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cageo.2017.06.013.

References

Abdollahifard, M.J., 2016. Fast multiple-point simulation using a data-driven path and an efficient gradient-based search. Comput. Geosci. 86 (January), 64–74, (C).
Abdollahifard, M.J., Nasiri, B., 2017. Exploiting transformation-domain sparsity for fast query in multiple-point geostatistics. Comput. Geosci. 21 (2), 289–299.
ANODI, 2016. Matlab Code of the Anodi Method. ⟨https://github.com/SCRFpublic/ANODI⟩. (Accessed 10 April 2016).
Arpat, G.B., Caers, J., 2007. Conditional simulation with patterns. Math. Geol. 39 (2), 177–203.
Caers, J., 2011. Modeling Uncertainty in the Earth Sciences. John Wiley & Sons, Hoboken, New Jersey.
Chilès, J.P., Delfiner, P., 2012. Geostatistics: Modeling Spatial Uncertainty 2nd ed.. John Wiley & Sons, Hoboken, New Jersey.
Cooley, J.M., Tukey, J.W., 1965. An algorithm for the machine calculation of complex fourier series. Math. Comp. 19, 297.
Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2009. Introduction to Algorithms 3rd ed.. The MIT Press, Cambridge, Massachusetts.
Efros, A.A., Freeman, W.T., 2001. Image quilting for texture synthesis and transfer. In: Proceedings of the SIGGRAPH 2001. August, pp. 341–346.
Gardet, C., Le Ravalec, M., Gloaguen, E., 2016. Pattern-based conditional simulation with a raster path: a few techniques to make it more efficient. Stoch. Environ. Res. Risk Assess. 30 (2), 429–446.
Gionis, A., Indyk, P., Motwani, R., et al., 1999. Similarity search in high dimensions via hashing. In: Proceedings of the International Conference on Very Large Data Bases. No. 6, pp. 518–529.
Guardiano, F.B., Srivastava, R.M., 1993. Geostatistics Tróia'92: Volume 1. Springer Netherlands, Dordrecht, Ch. Multivariate Geostatistics: Beyond Bivariate Moments, pp. 133–144.
Hamming, R., 1950. Error detecting and error correcting codes. Bell Syst. Tech. J. 29, 147–160.
Honarkhah, M., Caers, J., 2010. Stochastic simulation of patterns using distance-based pattern modeling. Math. Geosci. 42 (July), 487–517.
Indyk, P., Motwani, R., 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, ACM. pp. 604–613.
Laber, E., Moura, P., Pavanelli, L., 2016. On compression techniques for computing convolutions. In: Proceedings of the Data Compression Conference. IEEE, pp. 359–368.
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 1st ed.. John Wiley & Sons, Hoboken, New Jersey.
MS-CCSIM, 2016. Matlab Code of the MS-CCSIM Method. ⟨https://github.com/SCRFpublic/MS_CCSIM⟩. (Accessed 10 April 2016).
OPENCV, 2016. Opencv – Open Source Computer Vision. ⟨http://opencv.org⟩. (Accessed 15 July 2016).
Parra, Á., Ortiz, J.M., 2011. Adapting a texture synthesis algorithm for conditional multiple point geostatistical simulation. Stoch. Environ. Res. Risk Assess. 25 (8), 1101–1111.
Strebelle, S., 2002. Conditional simulation of complex geological structures using multiple-point statistics. Math. Geol. 34 (1), 1–21.
Tahmasebi, P., Sahimi, M., 2016a. Enhancing multiple-point geostatistical modeling: 1. Graph theory and pattern adjustment. Water Resour. Res. 52, 2074–2098.
Tahmasebi, P., Sahimi, M., 2016b. Enhancing multiple-point geostatistical modeling: 2. Iterative simulation and multiple distance function. Water Resour. Res. 52 (3), 2099–2122. http://dx.doi.org/10.1002/2015WR017807.
Tahmasebi, P., Hezarkhani, A., Sahimi, M., 2012. Multiple-point geostatistical modeling based on the cross-correlation functions. Comput. Geosci. 16 (3), 779–797.
Tahmasebi, P., Sahimi, M., Caers, J., 2014. MS-CCSIM: accelerating pattern-based geostatistical simulation of categorical variables using a multi-scale search in fourier space. Comput. Geosci. 67, 75–88.
Tan, X., Tahmasebi, P., Caers, J., 2014. Comparing training-image based algorithms using an analysis of distance. Math. Geosci. 46 (2), 149–169.
TrainingImagesLibrary, 2016. Training Images Library. ⟨http://www.trainingimages.org/training-images-library.html⟩. (Accessed 2 March 2016).
Yang, L., Hou, W., Cui, C., Cui, J., 2016. Gosim: a multi-scale iterative multiple-point statistics algorithm with global optimization. Comput. Geosci. 89, 57–70.
Zhang, T., Switzer, P., Journel, A., 2006. Filter-based classification of training image patterns for spatial simulation. Math. Geol. 38 (1), 63–80.