Mathematical Geology, Vol. 39, No. 2, February 2007 (© 2007)
DOI: 10.1007/s11004-006-9075-3

Conditional Simulation with Patterns[1]

G. Burc Arpat[2] and Jef Caers[2]

An entirely new approach to stochastic simulation is proposed through the direct simulation of patterns. Unlike pixel-based (single grid cells) or object-based stochastic simulation, pattern-based simulation simulates by pasting patterns directly onto the simulation grid. A pattern is a multi-pixel configuration identifying a meaningful entity (a puzzle piece) of the underlying spatial continuity. The methodology relies on the use of a training image from which the pattern set (database) is extracted. The use of training images is not new. The concept of a training image is extensively used in simulating Markov random fields or for sequentially simulating structures using multiple-point statistics. Both these approaches rely on extracting statistics from the training image, then reproducing these statistics in multiple stochastic realizations, at the same time conditioning to any available data. The proposed approach does not rely, explicitly, on either a statistical or probabilistic methodology. Instead, a sequential simulation method is proposed that borrows heavily from the pattern recognition literature and simulates by pasting at each visited location along a random path a pattern that is compatible with the available local data and any previously simulated patterns. This paper discusses the various implementation details to accomplish this idea. Several 2D illustrative as well as realistic and complex 3D examples are presented to showcase the versatility of the proposed algorithm.

KEY WORDS: Geostatistics, pattern similarity, multiple-point statistics.

INTRODUCTION

Stochastic conditional simulation is an important tool for modeling spatial phenomena. The goal of stochastic simulation is to create multiple, realistic representations, termed realizations, of a studied spatial phenomenon, that are constrained (conditioned) to any available data. Each realization should reflect the spatial continuity believed to exist. In most applications, spatial continuity is quantified through a spatial covariance or variogram. This variogram is modeled from experimental data, then reproduced in each realization. Several publications have pointed out (Journel, 1993; Caers and Journel, 1998; Strebelle, 2000, 2002; Caers and Zhang, 2004) the limitation of the variogram model incapturing spatial

[1]Received 26 January 2005; accepted 22 May 2006; Published online: 12 April 2007.
[2]Department of Energy Resources Engineering, Stanford University, Stanford, CA 94305-2220, USA; e-mail: jcaers@stanford.edu

continuity of actual phenomena. Being a two-point statistic, describing correlation between any two points in space only, the variogram cannot model strongly connected and/or curvilinear geometries of the variable considered. Recent work has suggested an alternative approach to reproducing realistic spatial continuity, based on the concept of "training images." The training image is a conceptual, but explicit, 2D or 3D, depiction of the spatial continuity studied. Training images may come from actual data such as from outcrops or other exhaustive datasets deemed representative for the area being modeled. Alternatively, a training image could be a single but large, unconditional realization of a stochastic simulation method, e.g. using a Boolean or object-based model. Either way, a training image need not be locally constrained to any (hard and soft) data, need not have the same dimensions as the study area, but should reflect a style of spatial continuity interpreted as similar to the actual phenomenon.

Several approaches have been developed for generating conditional realizations reproducing spatial continuity similar as the training image. Consider the simulation of a random function $Z(\mathbf{u})$ over a regularly gridded domain $\mathbf{u} = (x, y, z) \in A$.

When implementing the approach of simulating Markov random fields (MRF) with higher order interaction using McMC (Tjelmeland, 1996), one models the conditional probability of $Z(\mathbf{u})$ given a template of neighboring values around $\mathbf{u}$ through an exponential-type parametric model (known up to a normalization constant) involving higher order (more than two) interactions (termed cliques). Under this form the joint distribution of all $Z(\mathbf{u})$ can be written analytically. The unknown normalization constant requires that simulation is done iteratively, in particular, using the Metropolis-Hasting sampler. While theoretically elegant, the application of the method to practical cases runs into several problems. The parameter estimation of the exponential model using maximum likelihood is tedious because of the unknown normalization constant of the conditional pdfs. Moreover, the model parameters are difficult to interpret in terms of actual patterns or statistics in the training image. Secondly, simulation is slow due to McMC; issues of convergence arise. The methodology is essentially borrowed from Bayesian image analysis and works well for image restoration problems. Few large 3D application in the earth sciences, typified by sparse data and complex priors, has been reported.

Strebelle (2002) proposes a more ad-hoc approach based on an original idea of Guardiano and Srivastava (1993). Strebelle uses a sequential simulation framework, hence is non-iterative, to simulate realizations. Sequential simulation requires one to calculate, at each visited location, the conditional distribution of $Z(\mathbf{u})$ given any direct observations on $Z(\mathbf{u})$ (termed hard data) and previously simulated nodes. Instead of modeling this conditional probability using parametric model as done in MRF, the conditional distribution is estimated directly from the training image as follows. At each nodal location $\mathbf{u}$ along the random path, the data values and configuration of any hard data and previously simulated values

neighboring **u** are extracted. This set of values and their mutual spatial configuration is termed the "data event." The training image is scanned for replicates of this data event. The central value corresponding to each replicate is retained. The histogram of all central values constitutes an empirical model for the conditional pdf.

Based on this idea, a practical algorithm termed *snesim* has been developed, successfully used in modeling large 3D reservoir using large and complex 3D training images (Caers and others, 2001; Caers and others, 2003; Liu and others, 2005; Harding and others, 2005). Despite this success, several limitations of the snesim algorithm are known:

- RAM use: Scanning for replicates of a data event at each node along a random path would be impractical CPU-wise. *snesim* uses a search tree to store, prior to simulation, the conditional probabilities of data events occurring within the training image (see Strebelle, 2002 for details on how this is exactly done). At the sequential simulation stage, the search tree is queried for the conditional probability belonging to each data event along the random path, no scanning is required at this stage. However, for large training images, the size of the search tree may grow to several gigabytes, still impractical for current day RAM (see Strebelle, 2003 for more on the RAM use of *snesim*).
- Issues of stationarity: a translation invariance, hence stationarity, of the training image statistics is implicitly assumed when scanning for replicates of a data event over the entire training image. If the actual spatial phenomenon is known to exhibit trends, then two solutions have been proposed: (1) using a traditional route by decomposing the phenomenon into a known trend and unknown stationary residual, the latter obtained from the training image, (2) using a very large training image that has replicates of that trend. Complex spatial patterns are often difficult to decompose into a trend and residual, moreover, by lack of data, a deterministic trend may not always be easy to model. Hence, solution (1) works well in simple cases, while solution (2) may require an excessive amount of RAM, for the reason listed above.
- Pattern reproduction: several additional tricks (Strebelle and Remy, 2004; Liu and Journel, 2005) have been implemented for *snesim* to reproduce the spatial continuity of the training image adequately in most applications. However, the algorithm has some problems in reproducing strongly curvilinear, strongly non-stationary and complex spatial patterns with low proportion of occurrence. The *snesim* algorithm has yet to be applied to continuous variables.

Motivated by these observations and by personal experience with both MRF and *snesim* method, we take an entirely different approach to stochastically

simulate spatial continuity borrowed from training images. Both MRF and *snesim* take a probabilistic approach, i.e. one aims to reproduce patterns by reproducing explicitly certain statistics (in the MRF case) or conditional probabilities read from the training image. In this paper, the probabilistic framework is abandoned entirely, a decision that has several advantages and also disadvantages as discussed in a separate discussion session. Instead, we propose to look at the problem from its roots and treat it as a pattern reproduction problem, not as a statistics reproduction problem. We propose to use the training image as a database of "patterns," a pattern being defined as a multi-pixel configuration identifying meaningful entities of the underlying geological or spatial continuity.

To simulate realizations, the proposed method still borrows from the sequential simulation idea by randomly visiting nodes along a random path, but now at each node we simulate/paste an entire pattern, instead of drawing a single nodal value from a probability model. The art of designing this new algorithm will lie in how this pasting is done, making sure that the final realization reproduces patterns that are similar to the training image patterns, and, in addition is constrained to any direct observations. Although the proposed method allows to condition to indirect observations (soft data), see Arpat (2005), we consider this outside the scope of this paper and refer to future publications.

The paper borrows ideas and concepts from the pattern recognition and texture synthesis literature, however it should be stated clearly that the problem of conditional simulation is considerably more challenging than a texture synthesis or an image restoration problem. First of all, the simulated realization need to be constrained to a variety of data, often classified as hard (direct) and soft (indirect) data. Moreover, earth science problems are 3D and geological patterns can be considerably more complex than simple 2D textures.

SIMULATION WITH PAATTERNS: SINGLE-GRID, UNCONDITIONAL ALGORITHM

Preprocessing of the Training Image

Define $ti(\mathbf{u})$ as a value of the training image **ti** where $\mathbf{u} \in \mathbf{G}_{ti}$ and $\mathbf{G}_{ti}$ is the regular Cartesian grid discretizing the training image. In general, $ti(\mathbf{u})$ can be a continuous, categorical or vectorial variable. For simplicity of the development, the training image is considered binary (e.g. a sand/non-sand system) and thus an indicator notation is used for $ti(\mathbf{u})$:

$$ti(\mathbf{u}) = \begin{cases} 0 & \text{if at } \mathbf{u} \text{ } \mathbf{ti} \text{ contains non-sand} \\ 1 & \text{if at } \mathbf{u} \text{ } \mathbf{ti} \text{ contains sand} \end{cases} \quad (1)$$

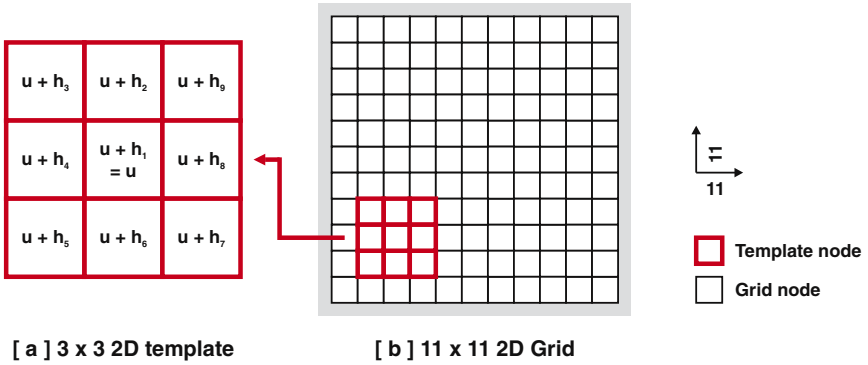

Figure 1. Template definition, example of a 3×3 template on an 11×11 grid.

$\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ indicates a specific multiple-point vector of $ti(\mathbf{u})$ within a template (window) $\mathbf{T}$ centered at node $\mathbf{u}$; i.e., $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ is the vector:

$$\mathbf{ti}_{\mathbf{T}}(\mathbf{u}) = \{ti(\mathbf{u}+\mathbf{h}_1), ti(\mathbf{u}+\mathbf{h}_2), \ldots, ti(\mathbf{u}+\mathbf{h}_\alpha), \ldots, ti(\mathbf{u}+\mathbf{h}_{n_{\mathbf{T}}})\} \quad (2)$$

where the $\mathbf{h}_\alpha$ vectors are the vectors defining the geometry of the $n_{\mathbf{T}}$ nodes of template $\mathbf{T}$ and $\alpha = 1, \ldots, n_{\mathbf{T}}$. The vector $\mathbf{h}_1 = 0$ identifies the central location $\mathbf{u}$. Figure 1 illustrates these concepts for a 3×3 2D template, i.e. $n_{\mathbf{T}} = 9$.

To extract a "pattern database" from the training image $\mathbf{ti}$ a preprocessing is performed by scanning the image using template $\mathbf{T}$ and storing the corresponding multiple-point $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ vectors in a database. Each such $\mathbf{ti}_{\mathbf{T}}(\mathbf{u})$ vector is called a "pattern" of the training image and the database is denoted by $\mathbf{patdb}_{\mathbf{T}}$. Figure 2 illustrates this process for a 3×3 2D template. In fact, this database is never explicitly constructed, instead the patterns remain in the training image, but the location of each pattern in the training image is indexed. Thereby, one avoids storing the patterns in RAM, which could be costly.

A single pattern is represented by the following vector

$$\mathbf{pat}_{\mathbf{T}}^{k} = \left\{pat_{\mathbf{T}}^{k}(\mathbf{h}_1), pat_{\mathbf{T}}^{k}(\mathbf{h}_2), \ldots, pat_{\mathbf{T}}^{k}(\mathbf{h}_\alpha), \ldots, pat_{\mathbf{T}}^{k}(\mathbf{h}_{n_{\mathbf{T}}})\right\} \quad (3)$$

where all $n_{\mathbf{pat}_{\mathbf{T}}}$ patterns are defined on the same template $\mathbf{T}$. The frequency of occurrence of a pattern is not recorded, instead, repeating patterns are listed as many times in the database as they occur in the training image.

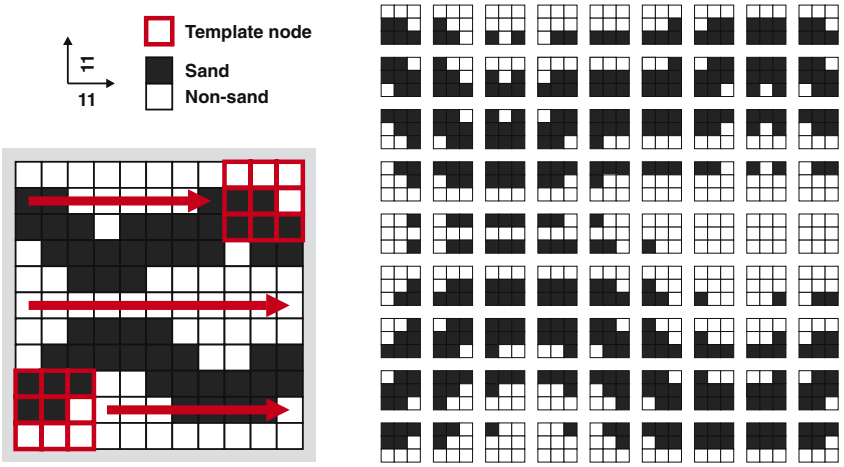

Figure 2. Example of a training image and its patterns obtain by scanning.

Sequential Simulation with Patterns

Once the pattern database $\mathbf{patdb_T}$ is constructed, the algorithm proceeds with the simulation of these patterns on a realization **re**. Initially, **re** is completely uninformed, i.e. nodes $re(\mathbf{u}) \in \mathbf{G}_{re}$ of the realization are unknown. A flag notation $re(\mathbf{u}) = \chi$ is used for such 'unknown' or 'uninformed' nodes:

$$re(\mathbf{u}) = \begin{cases} \chi & \text{if at } \mathbf{u} \text{ } \mathbf{re} \text{ is uninformed (can be sand or non-sand)} \\ 0 & \text{if at } \mathbf{u} \text{ } \mathbf{re} \text{ contains non-sand} \\ 1 & \text{if at } \mathbf{u} \text{ } \mathbf{re} \text{ contains sand} \end{cases} \quad (4)$$

Simulation on the realization **re** follows the footsteps of a sequential simulation algorithm. In sequential simulation, a data event $\mathbf{dev_T}(\mathbf{u})$ is defined as the set of hard data and previously simulated values neighboring the visited location $\mathbf{u}$ within the template $\mathbf{T}$ (the same template used to scan the training image). Thus:

$$\mathbf{dev_T}(\mathbf{u}) = \{dev_{\mathbf{T}}(\mathbf{u}+\mathbf{h}_1), \ldots, dev_{\mathbf{T}}(\mathbf{u}+\mathbf{h}_\alpha), \ldots, dev_{\mathbf{T}}(\mathbf{u}+\mathbf{h}_{n_\mathbf{T}})\} \quad (5)$$

where $dev_{\mathbf{T}}(\mathbf{u}+\mathbf{h}_\alpha) = re(\mathbf{u}+\mathbf{h}_\alpha)$ and **re** is the realization as previously defined. In other words, $\mathbf{dev_T}(\mathbf{u}) = \mathbf{re_T}(\mathbf{u})$. Different from a pattern $\mathbf{pat}_\mathbf{T}^k$, a data event $\mathbf{dev_T}(\mathbf{u})$ is allowed to contain unknown values (values still to be simulated) denoted by χ.

During simulation, the nodes $\mathbf{u} \in \mathbf{G}_{re}$ of the realization **re** are randomly visited and at every node $\mathbf{u}$, the data event $\mathbf{dev_T}(\mathbf{u})$ is extracted. Then, the data

event is compared to all available patterns $\mathbf{pat}_{\mathbf{T}}^{k}$ in the pattern database $\mathbf{patdb}_{\mathbf{T}}$. The aim is to find the 'most similar' pattern $\mathbf{pat}_{\mathbf{T}}^{*}$ to the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$, i.e. $\mathbf{pat}_{\mathbf{T}}^{*}$ is the pattern that minimizes some distance $d\langle \mathbf{dev}_{\mathbf{T}}(\mathbf{u}) \mathbf{pat}_{\mathbf{T}}^{k} \rangle$ for $k = 1, \ldots, n_{\mathbf{pat}_{\mathbf{T}}}$. The concept of distance and similarity will be discussed in the next section. When several patterns $\mathbf{pat}_{\mathbf{T}}^{k}$ are equally similar to the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$, one of these patterns is randomly selected to be the most similar pattern. Once this most similar pattern $\mathbf{pat}_{\mathbf{T}}^{*}$ is found, the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ is replaced by this pattern, i.e. the values of the most similar pattern $\mathbf{pat}_{\mathbf{T}}^{*}$ are pasted on to the realization **re** at the current node $\mathbf{u}$, replacing all values of the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$.

The following algorithm describes the above process:

- Define a random path on the grid $\mathbf{G}_{re}$ of the realization **re** to visit each node $\mathbf{u} \in \mathbf{G}_{re}$ only once.
- At every node $\mathbf{u}$, extract the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ from the realization **re** and find the $\mathbf{pat}_{\mathbf{T}}^{*}$ that minimizes $d\langle \mathbf{dev}_{\mathbf{T}}(\mathbf{u}) \mathbf{pat}_{\mathbf{T}}^{k} \rangle$ for $k = 1, \ldots, n_{\mathbf{pat}_{\mathbf{T}}}$ in the pattern database $\mathbf{patdb}_{\mathbf{T}}$, i.e. $\mathbf{pat}_{\mathbf{T}}^{*}$ is the 'most similar' pattern to $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$.
- If several $\mathbf{pat}_{\mathbf{T}}^{k}$ are equally similar to $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$, randomly select one of such patterns to be the most similar pattern $\mathbf{pat}_{\mathbf{T}}^{*}$.
- Once the most similar pattern $\mathbf{pat}_{\mathbf{T}}^{*}$ is found, assign $\mathbf{pat}_{\mathbf{T}}^{*}$ to $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$, i.e. for all the $n_{\mathbf{T}}$ nodes $\mathbf{u} + \mathbf{h}_{\alpha}$ within the template $\mathbf{T}$, $dev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_{\alpha}) = pat_{\mathbf{T}}^{*}(\mathbf{h}_{\alpha})$.
- Move to the next node of the random path and repeat the above steps until all the grid nodes along the random path are exhausted.

The above algorithm is different from traditional sequential simulation algorithms (such as sequential Gaussian simulation and *snesim*) in two ways:

1. Patterns (set of nodes) are simulated instead of one single node at a time;
2. Previously simulated values can be 'updated,' i.e. modified if the most similar pattern $\mathbf{pat}_{\mathbf{T}}^{*}$ dictates as such.
3. There is no random drawing from probability distributions.

During the simulation, if any data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ is completely uninformed (typically during the initial steps of the random path), all patterns $\mathbf{pat}_{\mathbf{T}}^{k}$ of the pattern database $\mathbf{patdb}_{\mathbf{T}}$ are considered to be equally similar to the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$. In such a case, the algorithm simply selects randomly a pattern from the pattern database $\mathbf{patdb}_{\mathbf{T}}$ as the most similar pattern $\mathbf{pat}_{\mathbf{T}}^{*}$ and proceeds to the next node.

Similarity

Critical to the algorithm are the size of the template and the definition of a measure of similarity between patterns and data event. The template size sensitivity

will be discussed at length with examples next. As for similarity, one can rely on many different similarity measures available in the pattern recognition literature (Rubner and others, 1998; Hagedoorn, 2000; Duda and others, 2001; Arpat, 2005). A simple measure of similarity that often works well is the Manhattan distance defined as

$$d\left\langle \mathbf{dev_T}(\mathbf{u}),\ \mathbf{pat}_{\mathbf{T}}^{k} \right\rangle = \sum_{\alpha=0}^{n_{\mathbf{T}}} \left| dev_{\mathbf{T}}(\mathbf{u})(\mathbf{u}+\mathbf{h}_\alpha) - pat_{\mathbf{T}}^{k}(\mathbf{h}_\alpha) \right| \tag{6}$$

The smaller the Manhattan distance, the more similar are $\mathbf{dev_T}(\mathbf{u})$ and $\mathbf{pat}_{\mathbf{T}}^{k}$. By definition, the Manhattan distance requires each vector entry to be fully defined, i.e. no missing values are allowed. However, a data event $\mathbf{dev_T}(\mathbf{u})$ might contain unknown nodes, i.e. $dev(\mathbf{u}+\mathbf{h}_\alpha) = \chi$ for some $\mathbf{h}_\alpha$. In other words, for the distance function given in Equation (6), if $\mathbf{u}+\mathbf{h}_\alpha = \chi$, the node $\mathbf{h}_\alpha$ is simply skipped during the summation.

It is important to note that, due to the way the Manhattan distance is defined, the similarity function of Equation (6) can be applied to any type of variable, continuous or categorical. For very complex patterns, the simple Manhatten distance may not adequately measure similarity between patterns. In that case, one can rely other distance measures such as the proximity transform distances (Arpat, 2005).

2D Examples

This section shows the application of the single-grid, unconditional algorithm to simple, 2D training images. The training images are chosen to demonstrate the use of the algorithm when applied to binary, multiple-category, and continuous variables.

Figure 3 shows the results of the algorithm when applied to a 250× 250 binary training image using templates of size 9×9 and 35×35. When using the smaller 9×9 template, the algorithm successfully reproduces all the small-scale details of the training image but large-scale patterns, i.e. the overall structure of the channel network, is not reproduced. The larger template of 35×35 'sees' this overall structure better and thus reproduces better results (see Fig. 3C). However, using this larger template increases the total run-time of the single-grid algorithm approximately 15 fold over the 9×9 template simulation.

Figure 4A demonstrates the case of a multiple-category image as a training image (in this case, 3 facies with shale, sand and levee). Figure 4A gives similar results as Figure 3 in that, the small 5×5 template fails to capture the large-scale patterns of the training image whereas the large 15×15 template produces better results but runs very slowly (approximately 10 times slower).

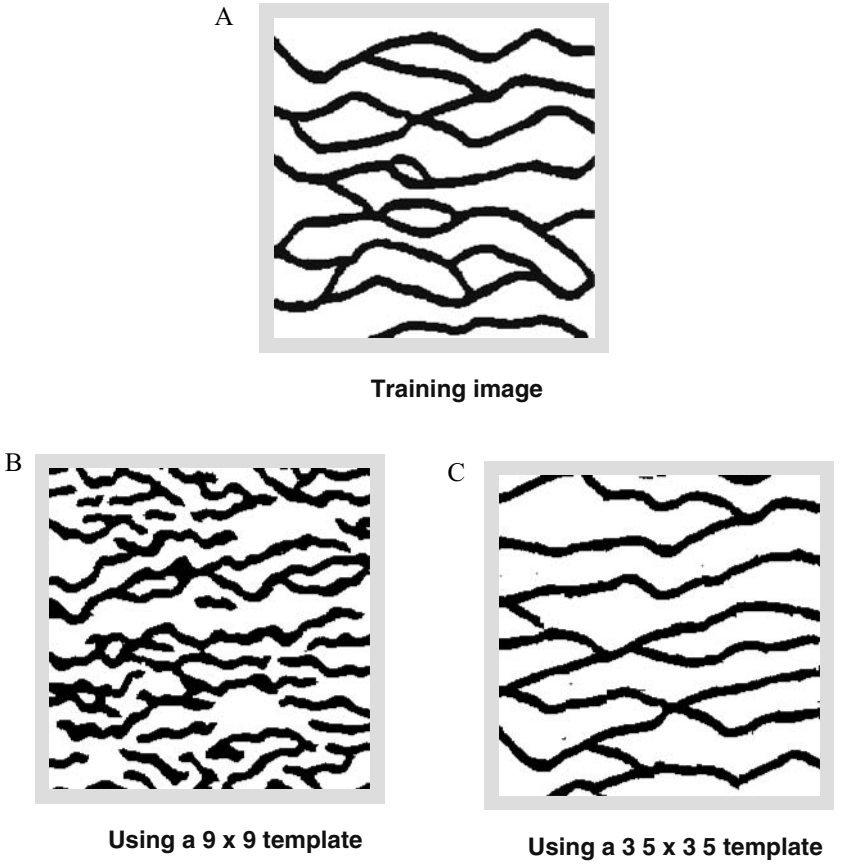

Figure 3. (A) Training image, (B) single realization using a 9×9 template, (C) single realization using a 35×35 template.

Figure 4B applies the single-grid algorithm to a continuous training image. The training image of the example is obtained through an unconditional *SGSIM* realization i.e. an unconditional realization of a multi-Gaussian model (Deutsch and Journel, 1998). Similar to the multiple-category case, the simulation algorithm remains the same but the values **ti**(**u**) of the training image are now continuous. Once again, the larger 15×15 template successfully captures the large-scale patterns (in this case, the long range of the variogram) but runs slow (approximately 10 times slower).

The above examples all show that the single-grid algorithm works well when a sufficiently large template is used to capture the large-scale patterns of a sufficiently large training image. However, it is also evident that using such a large template

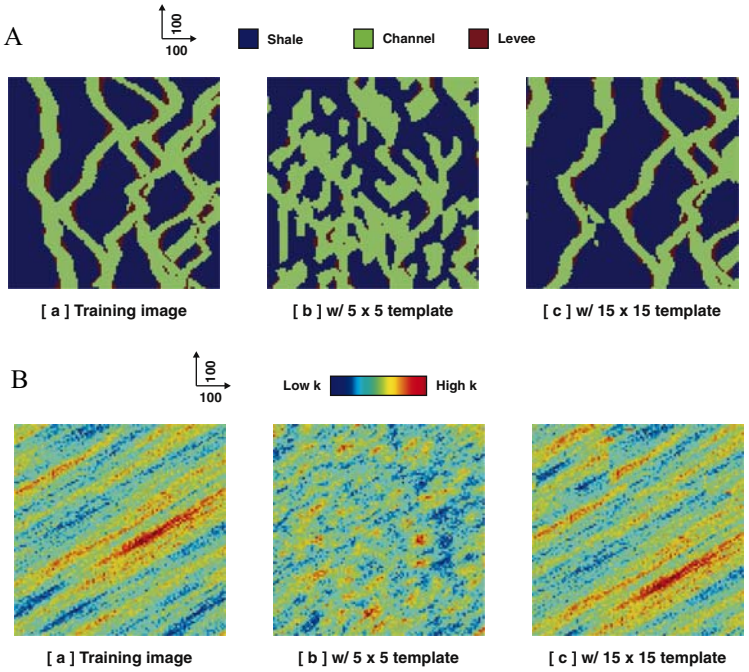

Figure 4. (A) Categorical variable training image generated using a Boolean method. (B) Continuous variable training image generated using *sgsim*.

in combination with a large training image is not practical due to the excessive CPU-demand, which is likely to explode for 3D cases. The next section discusses a multiple-grid approach that aims to remain efficient in terms of CPU by replacing a single large template by several cascading sparse templates and coarse grids.

MULTIPLE-GRID, UNCONDITIONAL ALGORITHM

Basic Method

A well-established method for reducing the amount of conditioning data (the template size in this case) in sequential simulation is the multiple-grid approach (Tran, 1994). In multiple-grid simulation, the multiple-grid view of a grid $\mathbf{G}$ is defined by n_g cascading grids $\mathbf{G}^g$ and templates $\mathbf{T}^g$. Figure 5 illustrates this. The coarse template $\mathbf{T}^g$ used for the grid $\mathbf{G}^g$ is defined by $\mathbf{h}_\alpha^g = 2^{g-1}.\mathbf{h}_\alpha$, i.e. the number of nodes in the template $\mathbf{T}^g$ is same as $\mathbf{T}$ $(= n_\mathbf{T} = n_{\mathbf{T}^g})$ but the nodes are

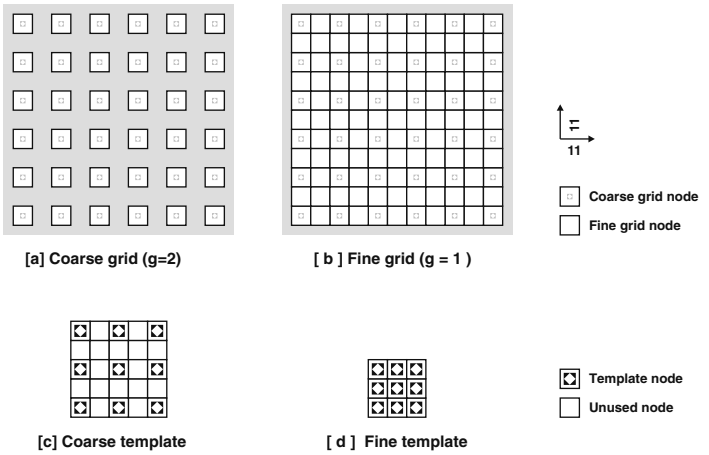

Figure 5. Definition of a coarse and fine template and grid.

'expanded' with 2^{g-1} spacing. Due to this expansion logic, $\mathbf{u} \in \mathbf{G}^{g-1}$ if $\mathbf{u} \in \mathbf{G}^{g}$. i.e. coarser nodes are always included in a finer grid.

When referring to coarse patterns extracted from a training image **ti** using the coarse template $\mathbf{T}^g$, the notation $\mathbf{pat}^k_{\mathbf{T}^g}$ is used. The preprocessing of the training image to extract such coarse patterns and to store them in a pattern database remains the same as the single-grid algorithm. For each coarse grid one uses the template $\mathbf{T}^g$, i.e. the coarse template $\mathbf{T}^g$ is used on the fine grid $\mathbf{G}_{ti}$ to capture all coarse patterns of the training image, not only the ones occurring on the coarse grid $\mathbf{G}^g_{ti}$. Similar to the $\mathbf{pat}^k_{\mathbf{T}^g}$ notation, a coarse data event is denoted by $\mathbf{dev}_{\mathrm{Tg}}(\mathbf{u})$. The distance function remains the same as calculating similarity is a task independent from the underlying grid.

Application of the simple multiple-grid method starts with the construction of the coarsest pattern database using the template $\mathbf{T}^{g=n_g}$. Then, the single-grid algorithm is applied to the coarsest realization grid $\mathbf{G}^{g=n_g}_{re}$ using the coarsest template $\mathbf{T}^{g=n_g}$. Once the coarsest realization is fully informed, g is set to $g-1$ and the process is repeated until the finest grid $\mathbf{G}^{g=1}_{re}$ simulation is complete. In essence, the simple multiple-grid method is nothing but the successive application of the single-grid algorithm to increasingly finer grids such that, with the exception of the coarsest grid, all other simulations start with an already partially populated realization.

The following unconditional, multi-grid algorithm describes this process:

- Set $g = n_g$, i.e. the first simulation is to be performed on the coarsest grid $\mathbf{G}^{g=n_g}_{re}$ (which is initially completely uninformed).

- Using the coarse template $\mathbf{T}^g$ on the finest grid $\mathbf{G}_{ti}$ of the training image **ti**, construct the coarse pattern database.
- Using the coarse template $\mathbf{T}^g$, run a single-grid simulation on the coarse grid $\mathbf{G}^g_{re}$ of the realization **re**.
- Set $g = g - 1$ and repeat from Step 2 until $g = 0$, i.e. the finest grid $\mathbf{G}^1_{re} = \mathbf{G}_{re}$ simulation is complete.

The unconditional, multi-grid algorithm is essentially a modified form of the traditional multiple-grid approach (Tran, 1994). In the traditional approach, the simulation algorithm that runs on grid $\mathbf{G}^g_{re}$ is not allowed to modify the values $re(\mathbf{u})$ of the realization **re** where $\mathbf{u} \in \mathbf{G}^{g+1}_{re}$. In other words, values simulated on coarser grids are frozen during the finer grid simulations. The method described in this section removes this restriction, i.e. any value $re(\mathbf{u})$ of **re** can be 'updated' at any time during the entire course of the multiple-grid simulation. Hence, any 'mistake' made on the coarse grid can still be corrected in subsequent finer grids. Related to this property, another difference between the unconditional, multi-grid algorithm and the traditional approach is how the nodes of **re** are visited. During the simulation of grid $\mathbf{G}^g_{re}$, the traditional approach does not visit the nodes $\mathbf{u} \in \mathbf{G}^g_{re}$ if $\mathbf{u}$ is also in $\mathbf{G}^{g+1}_{re}$, i.e. if $\mathbf{u} \in \mathbf{G}^{g+1}_{re}$. In other words, the traditional approach does not revisit the nodes simulated on the coarser grid during a finer grid simulation. In the proposed approach all nodes of $\mathbf{G}^g_{re}$ including those nodes $\mathbf{u} \in \mathbf{G}^{g+1}_{re}$ are visited hence are allowed to be modified during the finer grid simulation.

Dual Templates

To further improve multiple-grid simulation, the concept of "dual templates" is introduced. A dual template is defined as a template $\widetilde{\mathbf{T}}^g$ such that it has the same convex (bounding) volume as $\mathbf{T}^g$ on $\mathbf{G}^g$ but with all the finest nodes of the finest grid $\mathbf{G}$, see Figure 6. $\widetilde{\mathbf{T}}^g$ is considered "dual" with respect to the "primal" template $\mathbf{T}^g$. As a result, the total number of nodes within a dual template $\widetilde{\mathbf{T}}^g$ (denoted by $n_{\widetilde{\mathbf{T}}^g}$) is always greater than the number of nodes $n_{\mathbf{T}^g}$ within the primal template $\mathbf{T}^g$, i.e. $n_{\mathbf{T}^g} \ll n_{\widetilde{\mathbf{T}}^g}$.

By definition, a dual template $\widetilde{\mathbf{T}}^g$ always shares the center node $\mathbf{h}^g_1 = \mathbf{h}_1 = 0$ with its corresponding primal template $\mathbf{T}^g$. Using this property, the dual of a pattern $\mathbf{pat}^k_{\mathbf{T}^g}$, $\mathbf{pat}^k_{\widetilde{\mathbf{T}}^g}$ is defined as the pattern extracted at the same location $\mathbf{u}$ in the training image **ti** as $\mathbf{pat}^k_{\mathbf{T}^g}$ but using the template $\widetilde{\mathbf{T}}^g$ instead of $\mathbf{T}^g$, i.e. $\mathbf{pat}^k_{\widetilde{\mathbf{T}}^g} = \mathbf{ti}_{\widetilde{\mathbf{T}}_g}(\mathbf{u})$ when $\mathbf{pat}^k_{\mathbf{T}^g} = \mathbf{ti}_{\mathbf{T}_g}(\mathbf{u})$ for the same location $\mathbf{u}$. Figure 6 shows two primal and the corresponding dual patterns extracted from the training image shown in Figure 2 for the coarse grid $g = 2$.

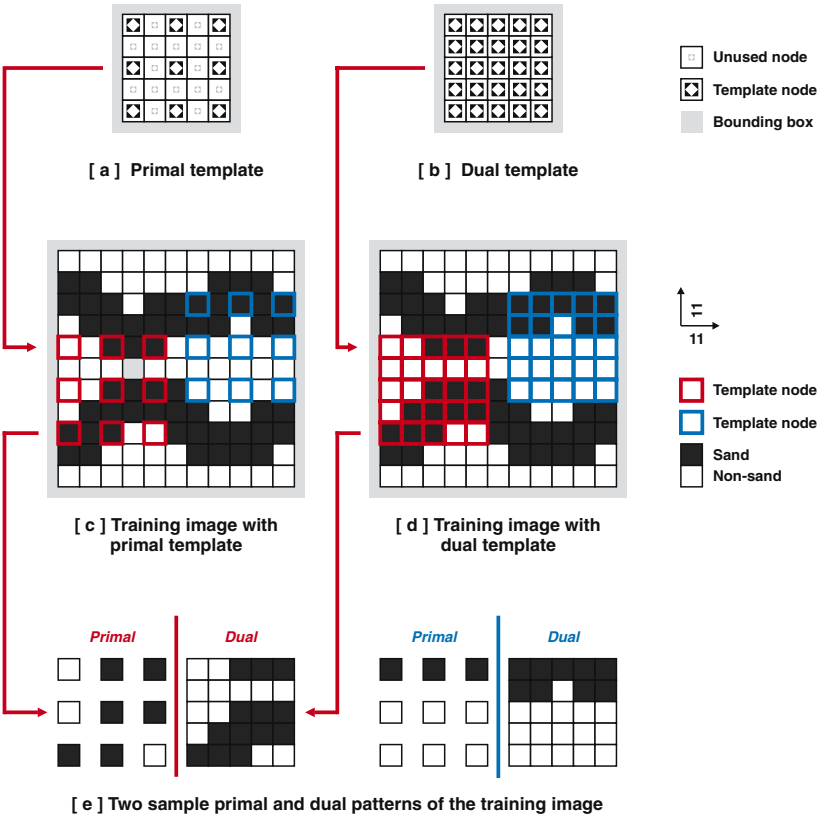

Figure 6. Example of a primal and corresponding dual template and pattern.

To extract the primal and dual patterns jointly, the training image preprocessing algorithm is modified such that the pattern database $\mathbf{patdb}_{\mathbf{T}}$ now stores two patterns per index k, i.e. the primal pattern $\mathbf{pat}^{k}_{\mathbf{T}_g}$ and the dual pattern $\mathbf{pat}^{k}_{\widetilde{\mathbf{T}}_g}$ are stored together as a pair for the same k index. Similar to the $\mathbf{pat}^{k}_{\widetilde{\mathbf{T}}_g}$ notation, a dual data event captured using a dual template $\widetilde{\mathbf{T}}^{g}$ is denoted by $\mathbf{dev}_{\widetilde{\mathbf{T}}_g}(\mathbf{u})$.

Using dual templates, the unconditional multi-grid algorithm is modified. On node $\mathbf{u}$, first, the most similar pattern $\mathbf{pat}^{*}_{\mathbf{T}_g}$ is found by minimizing the distance $d\langle \mathbf{dev}_{\mathbf{T}_g}(\mathbf{u}), \mathbf{pat}^{k}_{\mathbf{T}_g} \rangle$. Then, the corresponding dual pattern $\mathbf{pat}^{*}_{\widetilde{\mathbf{T}}_g}$ is retrieved from the pattern database. Finally, instead of populating the nodes of $\mathbf{dev}_{\widetilde{\mathbf{T}}_g}(\mathbf{u})$ the algorithm populates all the nodes of the dual data event $\mathbf{dev}_{\widetilde{\mathbf{T}}_g}(\mathbf{u})$, i.e. not only the coarse nodes but also the finest nodes of **re** are updated. Since the finest grid $\mathbf{G}_{re}$ includes all nodes of the realization **re**, updating $\mathbf{dev}_{\widetilde{\mathbf{T}}_g}(\mathbf{u})$ amounts to updating **re** at all grids.

In the above process, dual templates are used only when populating the nodes of the dual data event and not used for similarity calculations, i.e. for selecting the most similar pattern $\mathbf{pat}^*_{\mathbf{T}_g}$.

The difference between the traditional multiple-grid approach (Tran, 1994) and the dual template simulation lies in how the information is passed from a coarse grid $\mathbf{G}^g_{re}$ to a finer grid $\mathbf{G}^{g-1}_{re}$. The dual template algorithm essentially guesses the entire, finest realization **re** based only on the coarse information and then lets the finer grid simulations to successively correct this guess.

Examples

Figure 7 shows the application of the dual template simulation to the same example presented in Figure 3. In the figure, using 4 multiple-grids, the algorithm successfully reproduces the large-scale patterns of the training image. Furthermore, the total run-time is only 25% longer than the single-grid 9×9 template simulation.

The choice of template size is one of the most critical decisions when using this method, impacting the reproduction of patterns as well as the space of uncertainty generated. Figure 8 illustrates both aspects. A too small template cannot fully capture all the variability of the training image at large scales leading to poor pattern reproduction. Choosing a very large template will necessarily reduce the space of uncertainty generated, in the limit, when the template size is the same as the training image, only one realization is generated, namely the training image itself. Figure 8 shows various single realizations (plots A, C, E, G) as well as ensemble averages (plots B, D, F, H) when applying an increasing template size. The ensemble average is in some geostatistical literature also termed: E-type estimate (Deutsch and Journel, 1998). As expected, the ensemble averages become increasingly structured (less uncertainty) when the template size increases.

Since the dual template simulation allows use of smaller templates while still successfully reproducing large-scale patterns, it now becomes possible to use larger, 3D training images for simulation. Figure 9A and B show a 3D multiple-category and a 3D continuous training image. The multiple-category training image is obtained via an unconditional object-based simulation and the continuous training image is obtained via an unconditional processed-based simulation (Wen and others, 1998). In the figures, two realizations are shown for each training image; one obtained using a $7 \times 7 \times 3$ template and 4 multiple-grids and the other obtained using a $9 \times 9 \times 3$ template and 3 multiple-grids. The figures demonstrate that the dual template simulation can adequately model complex, 3D geological scenarios. Note that, all simulations use the Manhattan similarity measure of Eq. (6).

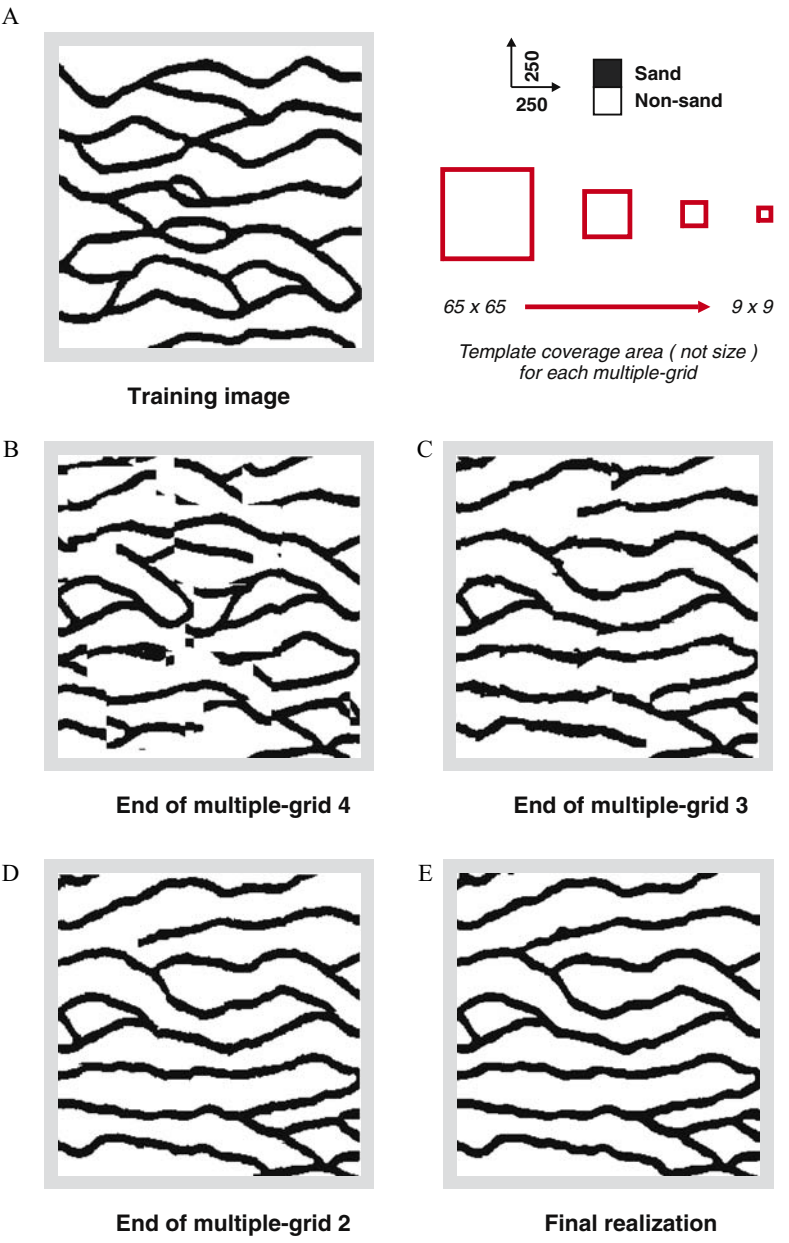

Figure 7. Definition of a primal and dual template and pattern (A) Training image, (B) The corresponding realization on the fine grid after completing the first (coarsest) multi-grid, (C) after the second multi-grid, (D) after the third multi-grid, (E) final realization.

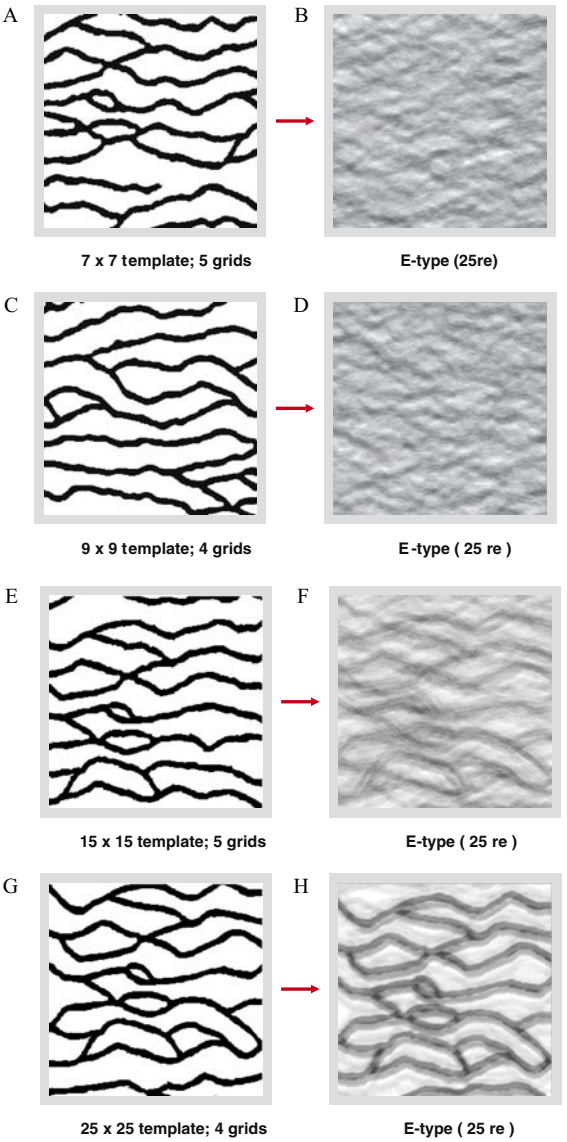

Figure 8. (A) single realization with a 7 × 7 template and 5 multiple coarse grids, (B) corresponding E-type, (C) single realization with a 9 × 9 template and 4 multiple coarse grids, (D) corresponding E-type, (E) single realization with a 15 × 15 template and 5 multiple coarse grids, (F) corresponding E-type, (G) single realization with a 25 × 25 template and 4 multiple coarse grids, (H) corresponding E-type.

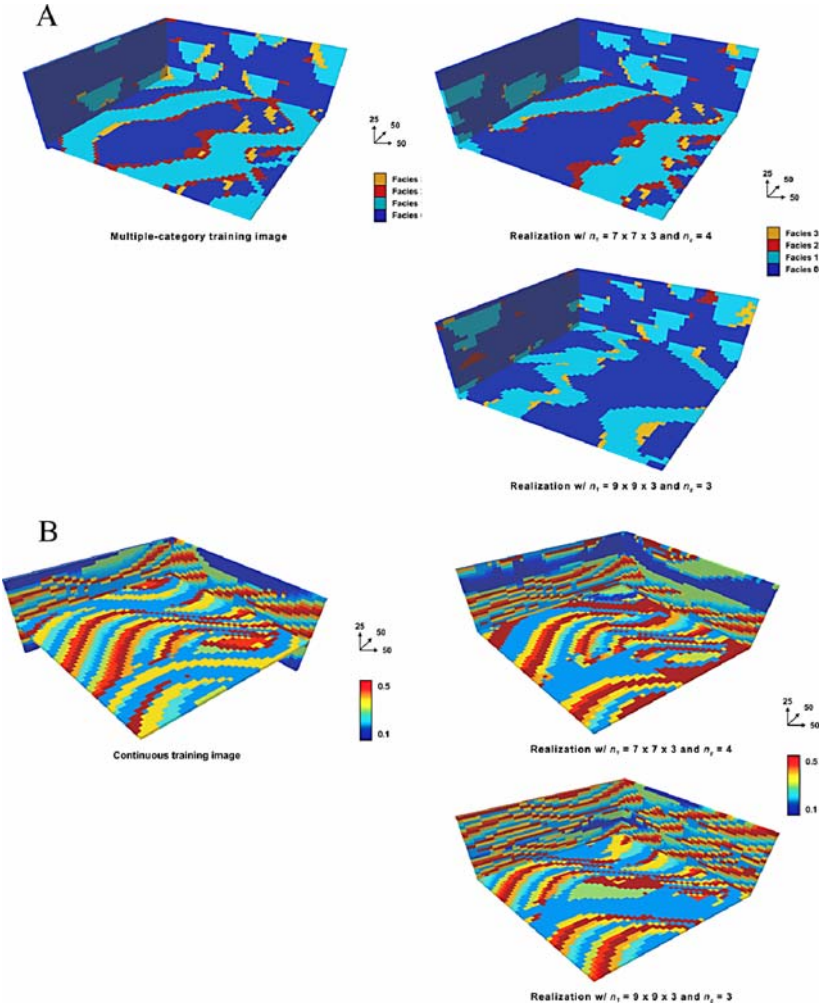

Figure 9. (A) Training image with four facies generate with a Boolean method, two realizations with different template sizes, (B) Training image of wavy bed laminations generated with SBED (Wen et al., 1998) and two realizations.

CONDITIONING TO HARD DATA

Method

To generate realizations that are conditioned to direct observations (hard data), any such hard data are first copied on to the realization **re**, i.e. the realization

re is no longer fully uninformed at the start of simulation. Since the hard data is copied on to the realization **re**, a data event $\mathbf{dev_T(u)}$ automatically contains as a subset any hard data that falls within the template. This subset is termed the hard data event $\mathbf{hdev_T(u)}$.

Conditioning to hard data in a single-grid setting requires a two-stage lookup of the pattern database $\mathbf{patdb_T}$ using first the hard data event $\mathbf{hdev_T(u)}$ and then the data event $\mathbf{dev_T(u)}$.

The algorithm starts with the first lookup where the actual conditioning to hard data occurs. At this stage, called "preconditioning," the algorithm finds all patterns $\mathbf{pat}_\mathbf{T}^k$ such that $d\langle \mathbf{hdev_T(u)}, \mathbf{pat}_\mathbf{T}^k \rangle \leq \epsilon_{hd}$ where $\epsilon_{hd} \geq 0$ is a threshold and is typically taken as zero (attempt to perform exact conditioning). In other words, during the first lookup, all patterns $\mathbf{pat}_\mathbf{T}^k$ of the pattern database $\mathbf{patdb_T}$ that are similar to the hard data event $\mathbf{hdev_T(u)}$ within a given threshold ϵ_{hd} are found. Setting $\epsilon_{hd} = 0$ amounts to searching only for patterns $\mathbf{pat}_\mathbf{T}^k$ that exactly match the hard data event $\mathbf{hdev_T(u)}$.

There are three possible outcomes of the first stage

1. The hard data event $\mathbf{hdev_T(u)}$ is completely uninformed, i.e. there is no hard data near the vicinity of the node **u** (defined by template **T**). Then, all patterns $\mathbf{pat}_\mathbf{T}^k$ fulfill the condition and are considered for the next stage;
2. The hard data event $\mathbf{hdev_T(u)}$ is at least partially informed and at least one $\mathbf{pat}_\mathbf{T}^k$ fulfills the condition $d\langle \mathbf{hdev_T(u)}, \mathbf{pat}_\mathbf{T}^k \rangle \leq \epsilon_{hd}$.
3. None of the patterns $\mathbf{pat}_\mathbf{T}^k$ fulfill the condition $d\langle \mathbf{hdev_T(u)}, \mathbf{pat}_\mathbf{T}^k \rangle \leq \epsilon_{hd}$. In such a case, another search is performed to find a most similar pattern $\mathbf{pat}_\mathbf{T}^*$ to the hard data event $\mathbf{hdev_T(u)}$ by minimizing $d\langle \mathbf{hdev_T(u)}, \mathbf{pat}_\mathbf{T}^k \rangle$. This most similar pattern $\mathbf{pat}_\mathbf{T}^*$ is then assumed to fulfill the condition. In other words, the algorithm settles for a pattern that is most similar to the hard data event $\mathbf{hdev_T(u)}$.

The last item of the above list occurs typically when the training image **ti** is not representative of the hard data. In other words, the scenario described by the training image **ti** (and the pattern database $\mathbf{patdb_T}$) conflicts with the available data, i.e. the **ti** is not rich enough in that it does not contain patterns that agree with the hard data patterns.

The patterns found in the first lookup form another pattern database $\mathbf{hpatdb_T}$ called the "preconditioned pattern database." The second lookup is performed on this preconditioned pattern database using the data event $\mathbf{dev_T(u)}$. The algorithm finds the most similar pattern $\mathbf{pat}_\mathbf{T}^* \in \mathbf{hpatdb_T}$ that minimizes $d\langle \mathbf{dev_T(u)}, \mathbf{pat}_\mathbf{T}^{hk} \rangle$

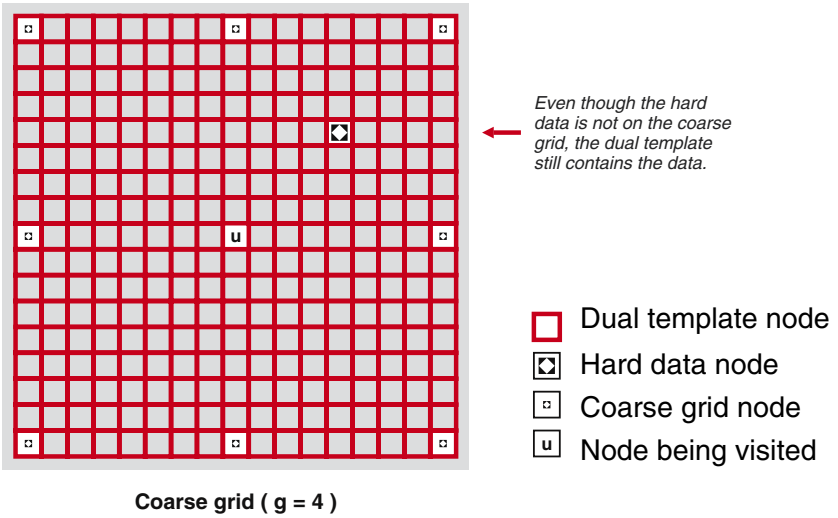

Figure 10. This Figure shows the vicinity of nodes near a location **u** of a simulation grid that is covered by the coarse (primal) template and by the corresponding fine (dual) template. Within this vicinity a single hard data location is shown. This location does not fall on the coarse grid.

for all patterns $\mathbf{pat}_{\mathbf{T}}^{hk} \in \mathbf{hpatdb}_{\mathbf{T}}$ and hk indicates an index of the preconditioned pattern database $\mathbf{hpatdb}_{\mathbf{T}}$.

Once the most similar pattern $\mathbf{pat}_{\mathbf{T}}^{*}$ is found, it is still pasted on to the data event $\mathbf{dev}_{\mathbf{T}}(\mathbf{u})$ but now only for nodes $\mathbf{u} + \mathbf{h}_\alpha$ that does not contain a hard datum, i.e. $dev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_\alpha) = pat_{\mathbf{T}}^{*}(\mathbf{h}_\alpha)$ only if $hdev_{\mathbf{T}}(\mathbf{u} + \mathbf{h}_\alpha) = \chi$.

Conditioning to dense hard data might be problematic when using a multi-grid approach. A particular multiple-grid $\mathbf{G}_{re}^{g}$ might not include the locations containing the hard data, i.e. the data might be located on one of the finer grids $\mathbf{G}_{re}^{g'}$ with $g > g'$. Figure 10 illustrates this. A possible solution to this (Strebelle, 2002) is to relocate the hard data to the nearest coarse grid node. However this means that the hard data is moved from its original location, hence the conditioning is no longer 'exact.' Moreover, relocation is problematic for dense hard data, since they could all be relocated to the same coarse grid node.

An alternative approach to data relocation is to utilize the dual template $\widetilde{\mathbf{T}}$ for hard data conditioning as follows. In the second step of the single grid conditional algorithm, instead of using the primal hard data event $\mathbf{hdev}_{\mathbf{T}}(\mathbf{u})$, one can use the dual hard data event $\mathbf{hdev}_{\widetilde{\mathbf{T}}^g}(\mathbf{u})$ to construct the preconditioned pattern database $\mathbf{hpatdb}_{\mathbf{T}}$. Since the dual template $\widetilde{\mathbf{T}}$ contains the hard data, finding all patterns

$\mathbf{pat}^k_{\widetilde{\mathbf{T}}^g}$ that fulfills the condition,

$$d\left\langle \mathbf{hdev}_{\widetilde{\mathbf{T}}^g}(\mathbf{u})\mathbf{pat}^k_{\widetilde{\mathbf{T}}^g} \right\rangle \leq \epsilon_{hd} \tag{7}$$

guarantees that all hard data is considered during any coarse grid $\mathbf{G}^g_{re}$ simulation even when the data is located on the finest grid $\mathbf{G}^0_{re}$. Then, the preconditioned pattern database $\mathbf{hpatdb}_\mathbf{T}$ is still constructed from primal patterns $\mathbf{pat}^k_{\mathbf{T}^g}$, retaining only the $\mathbf{pat}^k_{\mathbf{T}^g}$ for which the corresponding dual pattern $\mathbf{pat}^k_{\widetilde{\mathbf{T}}^g}$ fulfills Eq. (7). If no such dual pattern(s) $\mathbf{pat}^k_{\widetilde{\mathbf{T}}^g}$ can be found, then the algorithm minimizes $d\langle\cdot,\cdot\rangle$ of Eq. (7) to find a most similar dual pattern $\mathbf{pat}^*_{\widetilde{\mathbf{T}}^g}$ to the dual hard data event $\mathbf{hdev}_{\widetilde{\mathbf{T}}^g}(\mathbf{u})$ and pastes this pattern onto **re**.

To make conditioning with hard data CPU efficient a location specific preconditioned pattern database $\mathbf{hpatdb}_\mathbf{T}(\mathbf{u})$ is constructed before performing the first simulation. This pattern database is re-used for all realizations, i.e. the first lookup is performed ahead of time and the actual simulations only perform the second lookup.

Examples

Consider the reference case of Figure 11A. Using this reference different scenarios of available hard data are extracted, as shown in Figure 11B, C and D. The training image is the same as in Figure 7.

Figure 12 shows sample realizations and the ensemble averages of 25 realizations generated using the training image of Figure 11E for the different hard data sets. In the figure, the ensamble averages are used to check the quality of conditioning. If the algorithm conditions properly, an ensemble average of a binary indicator variable should give less uncertainty (values close to 0 or 1) near the hard data points. Furthermore, the overall uncertainty of the final model should decrease with increasing number of hard data. Figures 12B, D and F demonstrate this. Since the training image used for this example reflects a geological scenario consistent with the reference, the results do not conflict with the data and, as expected, with more data, the realizations converge to the reference itself.

As a sensitivity study, Figure 13 shows a realization conditioned to the same hard data as Figure 11 but with another, but conflicting training image shown in Figure 13. Despite this apparent conflict between the training image and the source of the hard data, the conditional algorithm successfully fits the data to the training image patterns.

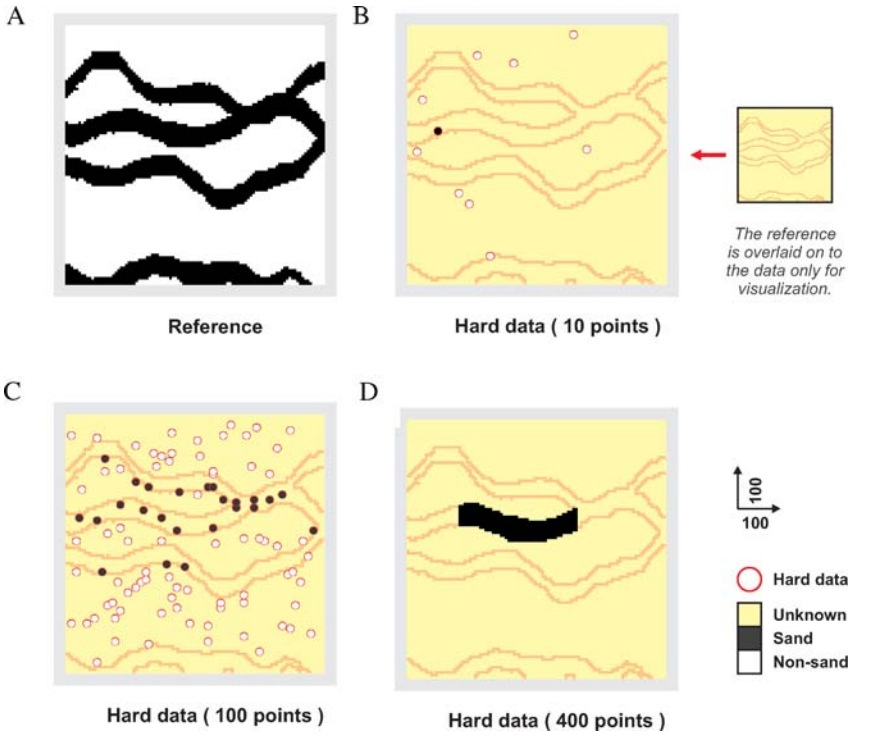

Figure 11. (A) Reference model, (B) hard data set with 10 samples, (C) with 100 samples, (D) object piece as hard data.

Finally we provide an example of a 3D application. A rather dense data set is used with a training image shown in Figure 14B. Since the patterns of the training image agree well with the reference, hence also with the data set extracted from it, the conditioning works well in this case.

EFFICIENCY

As mentioned before, this method does not require a significant amount of RAM, patterns remain stored in the training image. The CPU-time however increases linearly with the number of patterns and also linearly with the number of nodes in the template. The largest cost is that of searching for a most similar pattern, a task repeated at each location along the random path. Several implementation tricks speed up the search considerably and are discussed in more detail in Arpat (2005). The most important ones are:

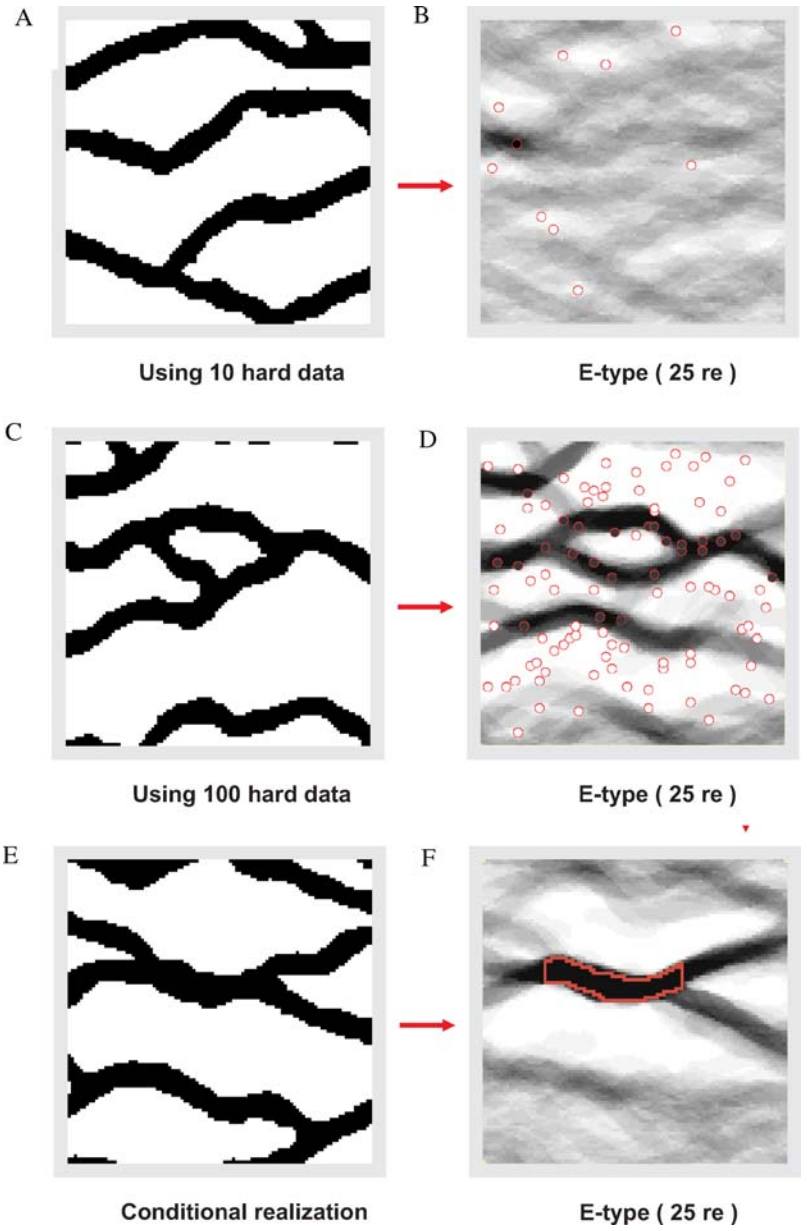

Figure 12. Results for hard data conditioning (A) single realization conditioned to 10 samples, (B) corresponding ensemble average, (C) single realization conditioned to 100 samples, (D) corresponding ensemble average, (E) single realization conditioned to object piece, (F) corresponding ensemble average.

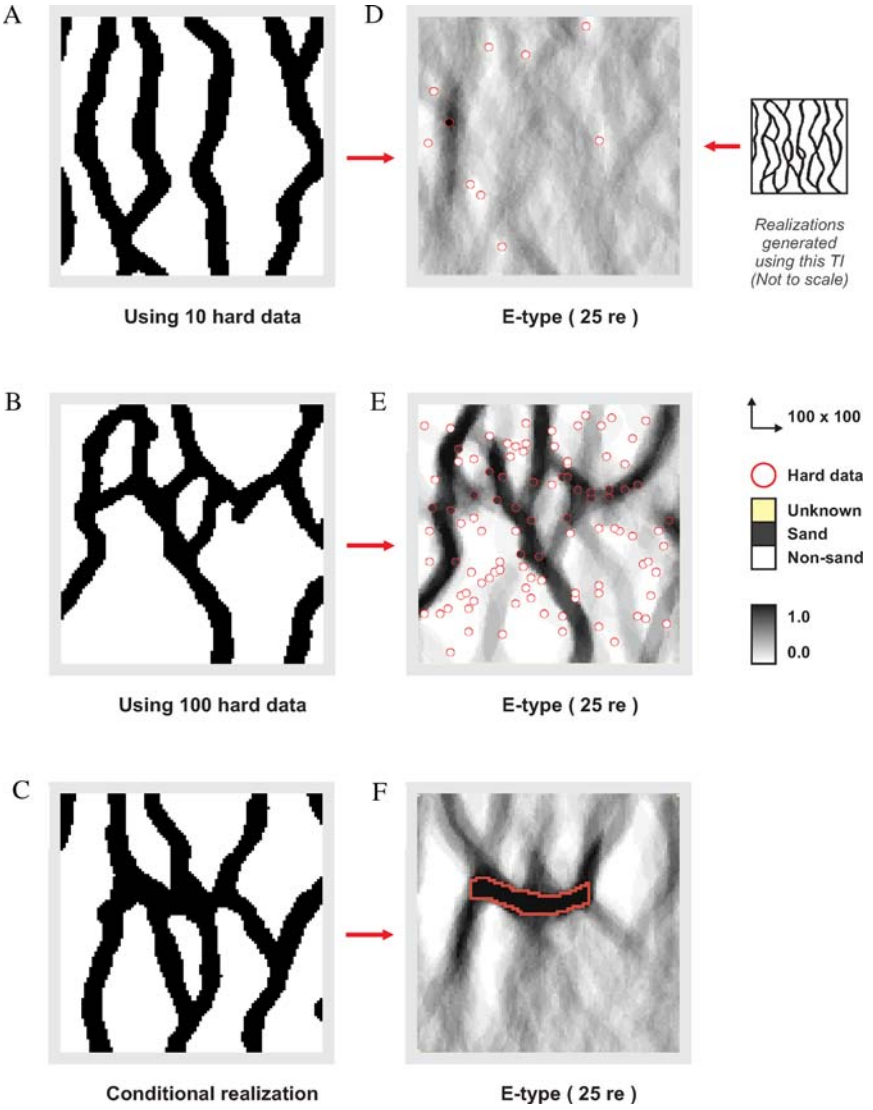

Figure 13. Same as Figure 12 but using a training image with channels running NS instead of EW.

- Not all nodes on the grid need to be visited, one can skip every other node in most cases.
- The search problem is also known as the post-office problem in Knuth (1997). This problem can be perfectly parallelized on a multi-CPU machine.

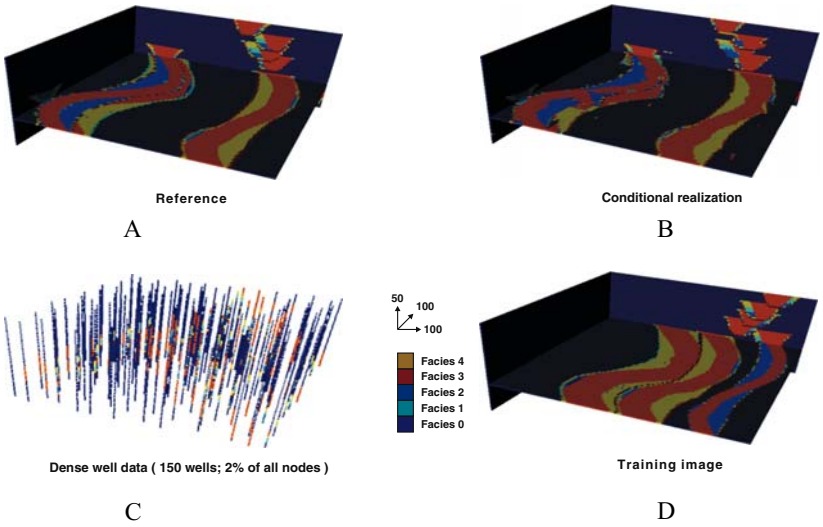

Figure 14. 3D example of data conditioning: (A) Reference model, (B) single conditional realizations (C) data set, (D) training image.

- Fast calculations of similarity are possible in case of the Manhattan distance.

DISCUSSION AND CONCLUSIONS

In many applications, the goal of stochastic simulation is to produce realizations that mimic the actual true spatial pattern of variability of a studied phenomenon, be it a mining grade, reservoir porosity or aquifer hydraulic conductivity. The traditional approach to stochastic simulation is to infer statistics, either from data or from a training image, then reproduce these statistics in a simulated realization. In this paper, we present a more direct approach, namely that of direct pattern reproduction instead of statistics reproduction. The proposed technique recognizes that an image contains patterns at multiple scales, from coarse scale connectivity or trend information to smaller scale local geometries. A multiple-grid approach is therefore proposed that simulates first the coarse scale, and is sequentially updates the coarse scale simulation at increasingly finer level. The simulation at each multigrid level consists of pasting patterns onto the grid, making sure that they are consistent with each other, in other words, they interlock each other properly. By defining a measure of similarity or 'distance' one ensures that this is the case

The approach has some evident advantages:

- There is less reliance on formal probabilistic methods and their inherent limitations. One of these limitations is the stationarity requirement for inferring various statistics. In the proposed approach patterns can be unique and such unique patterns can be pasted directly into the realization. An individual pattern may contain pixel values that depict a trend.
- The simulated realizations contain the patterns of the training image, the reproduction is almost perfect, as shown in the examples. Training images with complex patterns, discrete or continuous can now be handled. One can consider the method to be similar to that of solving a sophisticated jig-saw puzzle: the training image provides the (multi-scale) puzzle pieces, the algorithm puts the pieces together in a randomized fashion, constrained to local data.
- The only major tuning parameter is the size of the template.

Despite these obvious advantages, some equally obvious limitations can be stated:

- There is no formal theory, no explicit modeling. The proposed algorithm is mostly built from heuristic arguments. There is no proof that it works, why it works or when it would not work.
- The method does not resolve the conflict that may exist between the training image patterns and the patterns of the available hard data (conflict between prior and data). Such conflict will exist in most practical applications.
- The variability between realizations (in geostatistical jargon: the uncertainty space) is often smaller than for other similar algorithms, such as *snesim*. The only major random component in the proposed procedure is the random path. The variability is, as indicated, also function of the chosen template size. This may pose a problem in applications where such variability is important. However in many applications, the choice of the training image itself is often a much greater source of uncertainty than the uncertainty created by freezing the training image and generating multiple realizations with the same training image.
- As with any other multi-point geostatistical method, there exist currently no check to verify which of the training image statistics or patterns are actually reproduced. Right now, one simply relies on a visual check.
- The method is less CPU-efficient than comparable sequential simulation methods: based on experience, roughly 3 to 10 times slower depending on the complexity of the problem (presence of conditioning data, complexity of patterns in the training image). A solution to this problem is proposed in Zhang et al. (2006) where the dimensionality of pattern with several

hundreds of pixel values is reduced through the use of filters and their corresponding filterscores.

REFERENCES

Arpat, G. B., 2005. Stochastic simulation with patterns: doctoral dissertation, Stanford University, Stanford, USA, 184 p.

Caers, J., Avseth, P., and Mukerji, T., 2001, Geostatistical integration of rock physics, seismic amplitudes and geological models in North-Sea turbidite systems: The Leading Edge, v. 20, no. 3, p. 308–312.

Caers, J., and Journel, A. G., 1998, Stochastic reservoir simulation using neural networks trained on outcrop data, *in* Proceedings to the SPE Annual Technical Conference and Exhibition, New Orleans, SPE 49026: Society of Petroleum Engineers, 16 p.

Caers, J., Strebelle, S., and Payrazyan, K., 2003, Stochastic integration of seismic and geological scenarios: A submarine channel saga: The Leading Edge, v. 22, no. 3, p. 192–196.

Caers, J., and Zhang, T., 2004, Multiple-point geostatistics: a quantitative vehicle for intergration geologic analogs into multiple reservoir model, Integration of outcrop and modern analog data in reservoir models, AAPG memoir 80, p. 383–394.

Deutsch, C. V., and Journel, A. G., 1998, GSLIB: The geostatistical software library: Oxford University Press, 368 p.

Duda, O. R., Hart, P. E., and Stork, D. G., 2001, Pattern clasification, 2nd ed.: Wiley, New York, 278 p.

Guardiano, F., and Srivastava, M., 1993, Multivariate geostatistics: Beyond bivariate moments, *in* Soares, A., eds., Geostatistics Troia: Kluwer Academic Publications, Dordrecht, p. 133–144.

Hagedoorn, M., 2000, Pattern matching using similarity measures: doctoral dissertation, Universiteit, Utrecht, Holland, 157 p.

Journel, A. G., 1993, Geostatistics: Roadblocks and challenges, *in* Soares, A., eds., Geostatistics troia: Kluwer Academic Publications, Dordrecht, p. 213–224.

Harding, A., Strebelle, S., and Levy, M., 2005, Reservoir facies modeling: New advances in MPS, *in* Leuangthong, O., and Deutsch, C. V., eds., Geostatistics Banff 2005, vol. 2: Springer, Dordrecht, p. 559–568.

Knuth, D., 1997. Art of computer programming, 3rd ed.: Addison-Wesley Pub. Co, 3 vol.

Liu, Y., Harding, A., Journel, A. G., and Gilbert, R., 2005, A workflow for multiple-point geostatistical simulation, *in* Leuangthong, O., and Deutsch, C. V., eds., Geostatistics Banff 2004, vol. 1: Springer, Dordrecht, p. 245–254.

Liu, Y., and Journel, A. G., 2005, Improving sequential simulation with a structured path guided by information content: Math. Geol., v. 38, p. 945–964.

Rubner, Y., Tomasi, C., and Guibas, L., 1998, The earth mover's distance as a metric for image retrieval, Computer Science Department, Stanford University, report STAN-CS-TN-98-86, Stanford, USA.

Strebelle, S., 2000, Sequential simulation drawing structures from training images, Doctoral dissertation, Stanford University, USA, 164 p.

Strebelle, S., 2002, Conditional simulation of complex geological structures using multiple-point statistics: Math. Geol., v. 34, p. 1–22.

Strebelle, S., 2003, New multiple-point statistics simulation implementation to reducememory and CPU-demand, *in* Proceedings to the IAMG 2003, Portsmouth, UK, September 7–12.

Strebelle, S., and Remy, N., 2005. Post-processing of multiple-point geostatistical models to improve reproduction of training patterns, *in* Leuangthong, O., and Deutsch, C. V., eds., Geostatistics Banff 2004, vol. 2: Springer, Dordrecht, p. 979–987 .

Zhang, T., Switzer, P., and Journel, A. G., 2006, Filter-base classification of training image patterns for spatial simulation: Math. Geol., v. 38, no. 1, p. 63–80.

Tjelmeland, H., 1996, Stochastic models in reservoir characterization and Markov random fields for compact objects: Doctoral Dissertation, Norwegian University of Science and Technology, Trondheim, Norway, 71 p.

Tran, T., 1994, Improving variogram reproduction on dense simulation grids: Comput. & Geosci., v. 20, p. 1161–1168.

Wen, R., Martinius, A., Nass, A., and Ringrose, P., 1998, Three-dimensional simulation of small-scale heterogeneity in tidal deposits—A process-based stochastic simulation method, *in* IAMG 1998 Proocedings, International Association for Mathematical Geology.