Mathematical Geology, Vol. 38, No. 1, January 2006 (© 2006)
DOI: 10.1007/s11004-005-9004-x

Filter-Based Classification of Training Image Patterns for Spatial Simulation[1]

Tuanfeng Zhang,[2] Paul Switzer,[2] and Andre Journel[2]

Multiple-point simulation, as opposed to simulation one point at a time, operates at the pattern level using a priori structural information. To reduce the dimensionality of the space of patterns we propose a multi-point filtersim *algorithm that classifies structural patterns using selected filter statistics. The pattern filter statistics are specific linear combinations of pattern pixel values that represent directional mean, gradient, and curvature properties. Simulation proceeds by sampling from pattern classes selected by conditioning data.*

KEY WORDS: multiple-point simulation, geostatistics, data conditioning, multiple grids.

INTRODUCTION

Multiple-point (mp) simulation aims at capturing local patterns of variability from a training image and anchoring them to the image or numerical model being built. A pattern is defined as a set of values spatially distributed over a given template of spatial locations. The idea is to search within the training image for the pattern most similar to any specific local conditioning data event existing in the numerical model being built. That most similar pattern is then patched in all or part onto the image being built, much alike one would build a puzzle by patching onto the board, sequentially, puzzle pieces chosen to match the neighboring pieces already present on the board.

The mp simulation algorithm here proposed builds and expands on the now well-established *snesim* algorithm initiated by Guardiano and Srivastava (1993) and developed by Strebelle (2000, 2002). The *snesim* (single normal equation simulation) algorithm requires an exact match of the conditioning data event by the training pattern; if no such map is found, the conditioning data event is reduced by dropping the farthest away datum value resulting in a loss of conditioning

[1]Received 12 May 2004; accepted 18 April 2005; Published online: 28 April 2006.
[2]Department of Geological and Environmental Sciences, Stanford University, California 94305; e-mail: tfzhang@pangea.stanford.edu.

information. Also the *snesim* algorithm proceeds by simulating one pixel at a time, the central value of the conditioning data template. This extreme rigor in reproduction of the data event does not allow for any filtering or averaging of the training patterns, it calls for large and rich training images depicting most of the conditioning data events that could be found in the course of the sequential simulation process.

The *filtersim* (simulation using filter scores) algorithm here proposed trades the exact data event reproduction for an approximate reproduction with the benefit of not having to cut out any data in case of mismatch. During an initial processing of the training image all training patterns are classified in a filter space of reduced dimension, six in 2D, nine in 3D, even though the template size could contain up to $N = 1000$ nodes in 3D. Each training pattern is approximated by six to nine universal filter scores, the results of a corresponding number of weighted linear averages applied to its N nodal values. These training patterns, or puzzle pieces, are then grouped into a limited number of bins according to a similarity measure. Simulation proceeds by selecting the bin that matches best (although approximately) the conditioning data event, dig with replacement into that bin for a training pattern which is then patched into the simulation grid. In this regard, the proposed *filtersim* algorithm simulates patterns approximately conditioned to data patterns, as opposed to the *snesim* algorithm which simulates single-point values exactly conditioned to possibly reduced data patterns.

Both *filtersim* and *snesim* algorithms share the objective of building the image or numerical model by conditioning to local data patterns (mp geostatistics) using a prior structural model given under the form of a visually explicit training image. This is as opposed to traditional 2-point simulation algorithms such as *sgsim* (sequential Gaussian simulation) or *sisim* (sequential indicator simulation) which condition to the data considered one at a time, using for prior structural model a 2-point variogram or covariance model, see Deutsch and Journel (1998). If the information brought by the conditioning data is one of patterns involving multiple locations ($\gg 1$) at a time, and if a training image displaying these local patterns is available, then mp simulation algorithm would make a better use of the information available.

ALGORITHM REVIEW

The proposed multiple-point (mp) simulation algorithm proceeds in two steps. The first step consists of a classification of the patterns found in any given training image; this step is performed only once (just like a traditional variogram model is inferred only once). The second step is that of simulation which builds each realization by piecing together training patterns chosen from the previous classification so that they match the conditioning data. The stochastic simulation algorithm can be seen as an image construction from a box of

jigsaw pieces, where each piece can be used repetitively and many different pieces can match (approximately) the same set of local conditioning data. Instead of attempting to reproduce a variogram model, this simulation will reproduce local window mp statistics from the training image by borrowing entire training patterns.

In the first step, the training image is scanned with a template (moving window) of fixed size. Each template location yields a limited series of local statistics resulting from specific linear filters applied to the template data values. These filter values, or scores, describe local and directional mean levels, gradients, and curvatures of the spatial pattern present in that template. Because the filters are few ($K = 6$ in 2D, $= 9$ in 3D) they provide a low-dimensional representation of the window mp statistics. Ignoring border effects, if the training image (TI) is of dimension N and K filters are retained, that TI is represented by N points in a K-dimensional space. Each of the K filters axes can be split into, say, five quintile classes of equal frequency, allowing a summary of the possibly very large and pattern-rich training image into a low-dimensional (5^K) score space. The actual RAM demand for that filter score space is even smaller in practice since many of these classes can be empty. All N training patterns are classified in that filter score space. This is done only once per training image and per template size and geometry.

In the second step, each simulated realization is built sequentially along a random path visiting all nodes of the simulation grid. At each such node,

- collect all conditioning data found in its template neighborhood; these conditioning data include both original (hard) data and previously simulated values. That data template is mostly void in the beginning of the simulation and gets progressively filled as the sequential simulation progresses.
- using a predefined distance, determine the training class whose average pattern (called a training prototype) is closest to the conditioning data pattern; the distance allows comparing templates incompletely filled.
- draw from that class an individual training pattern and patch it onto the simulation grid except at the hard data locations. The inner part of that pattern is frozen and passed as data to the next sequence of simulation. The outer part of that pattern is passed as soft data allowing definition of conditioning data patterns and calculation of distances, but the corresponding points will be visited along the random path, hence re-simulated.

The simulation sequence stops, and one realization has been generated, when all nonhard data nodes have been visited. Another realization can be generated by seeding a complete new random path. This workflow is explained in detail in the following sections, see also the flowcharts of Figures 1 and 2.

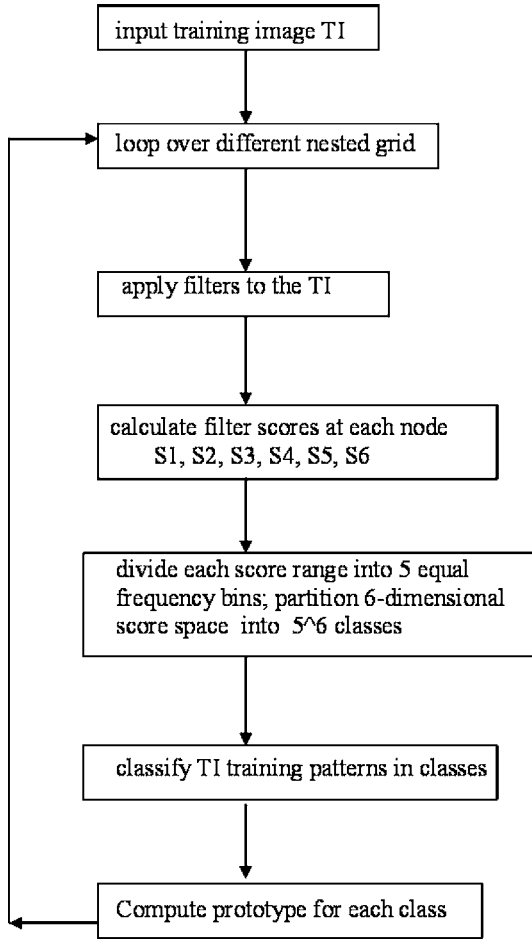

Figure 1. Flowchart for pattern classification in 2D.

PATTERN SCORES

Let $X(i, j)$ denote the datum value (continuous or categorical) at location (i, j) in a 2D training image. A score $S_f(i, j)$ for the training pattern centered at (i, j) is defined by a filter $f(u, v)$ as follows:

$$S_f(i, j) = \sum_{v=-n}^{n} \sum_{u=-n}^{n} f(u, v) X(i + u, j + v)$$

where the dimension of the local template is $(2n + 1) \times (2n + 1)$.

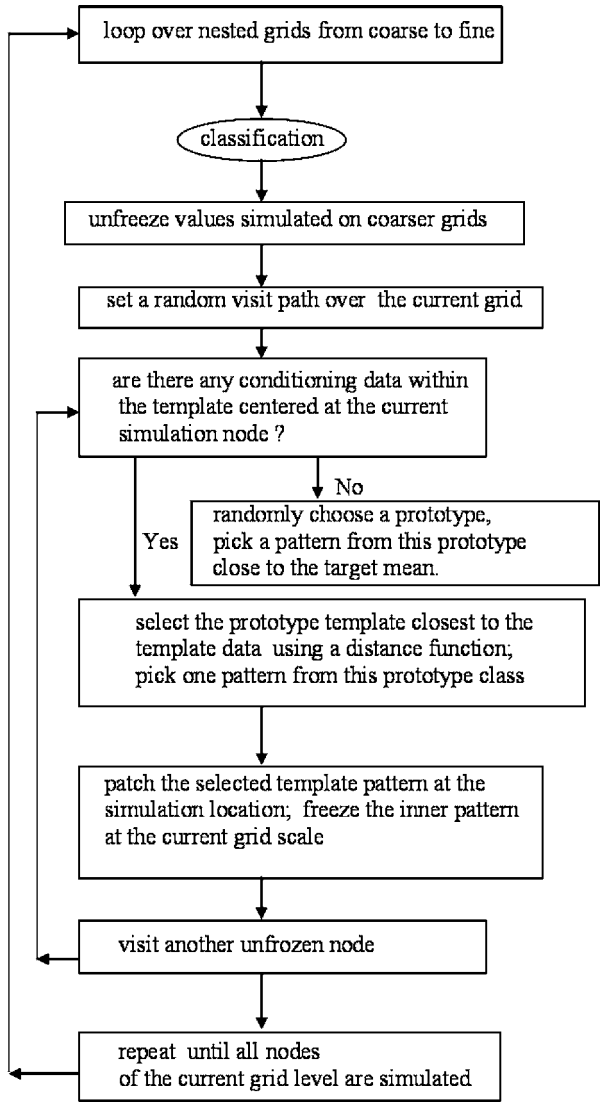

Figure 2. Flowchart for simulation.

In 2D, define six different filters $f_1, \ldots, f_6$ as follows, see Figures 3(a)–(f):
(1) f_1: N–S directional average

$$f_1(u, v) = 1 - \frac{|v|}{n}, \quad v = -n, \ldots, n$$

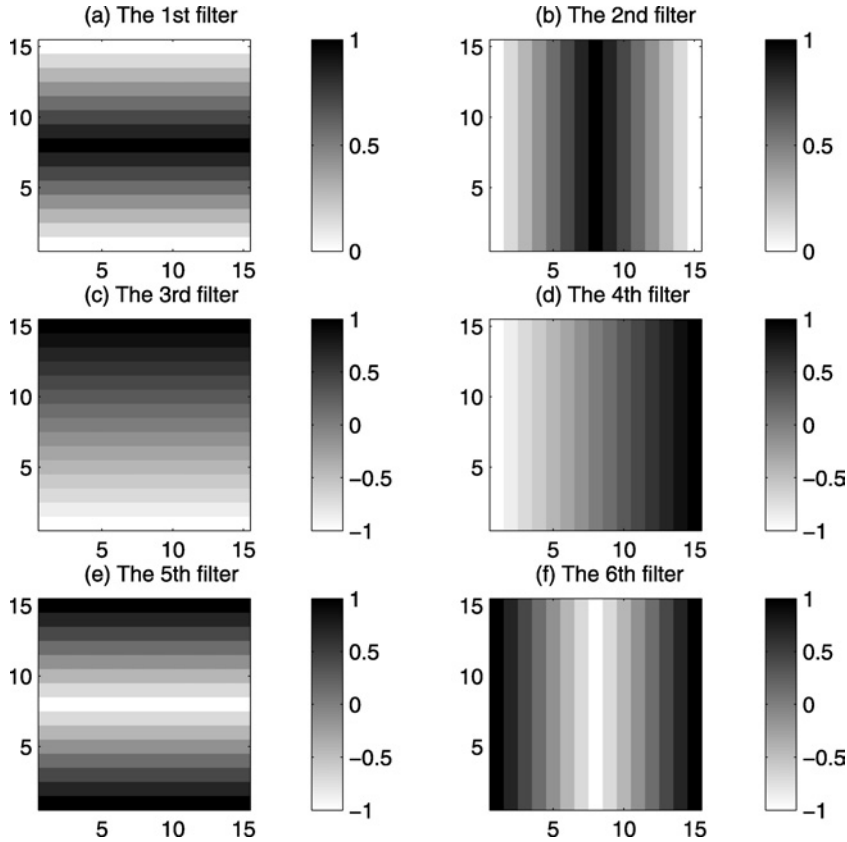

Figure 3. Six 2D local filters: the weights are given in gray scale.

(2) f_2: E–W average, obtained by rotating f_1 by 90°:

$$f_2(u, v) = 1 - \frac{|u|}{n}$$

(3) f_3: N–S directional gradient:

$$f_3(u, v) = v/n$$

(4) f_4: E–W gradient, obtained by rotating f_3 by 90°:

$$f_4(u, v) = u/n$$

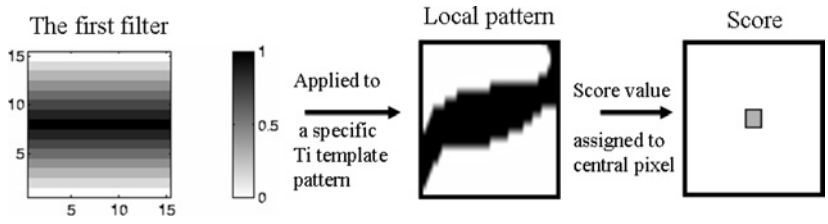

Figure 4. Pattern scores.

(5) f_5: N–S directional curvature:

$$f_5(u, v) = \frac{2|v|}{n} - 1$$

(6) f_6: E–W curvature, obtained by rotating f_5 by 90°:

$$f_6(u, v) = \frac{2|u|}{n} - 1$$

Each of these six filters is used to scan any 2D training image. At each pixel location (i, j), the template of neighborhood data values is weighted by these filters to produce a series of six scores, see Figure 4 for the first score. By assigning these six scores to the pixel (i, j), we obtain six score maps. Figures 5–7 give the three sets of score maps corresponding to the three multiple grids used to scan the training image of Figure 8. The number of nodes for each of the three templates used is 15×15, while the training image is 250×250.

The first two score maps S_1 and S_2 are weighted moving averages of the $15 \times 15 = 225$ template values: they highlight the channel center locations. The third S_3 score provides N to S edge detection, hence that score highlights the channel boundaries. The S_4 score gives E to W edge detection, reflecting any deviation of the channel from the dominant EW channel direction. The last two scores, S_5 and S_6, point to zones of maximum directional curvatures; for this particular application they could have been omitted since they provide little additional information.

PATTERN CLASSIFICATION USING SCORES

Each of the six filter scores is discretized into five equal frequency bins according to their respective quintile thresholds. This results in a partition of the 6-dimensional score space into a maximum of $5^6 = 15{,}625$ bins. In practice, many of these bins may be empty, for example, for categorical data on a fine grid it may

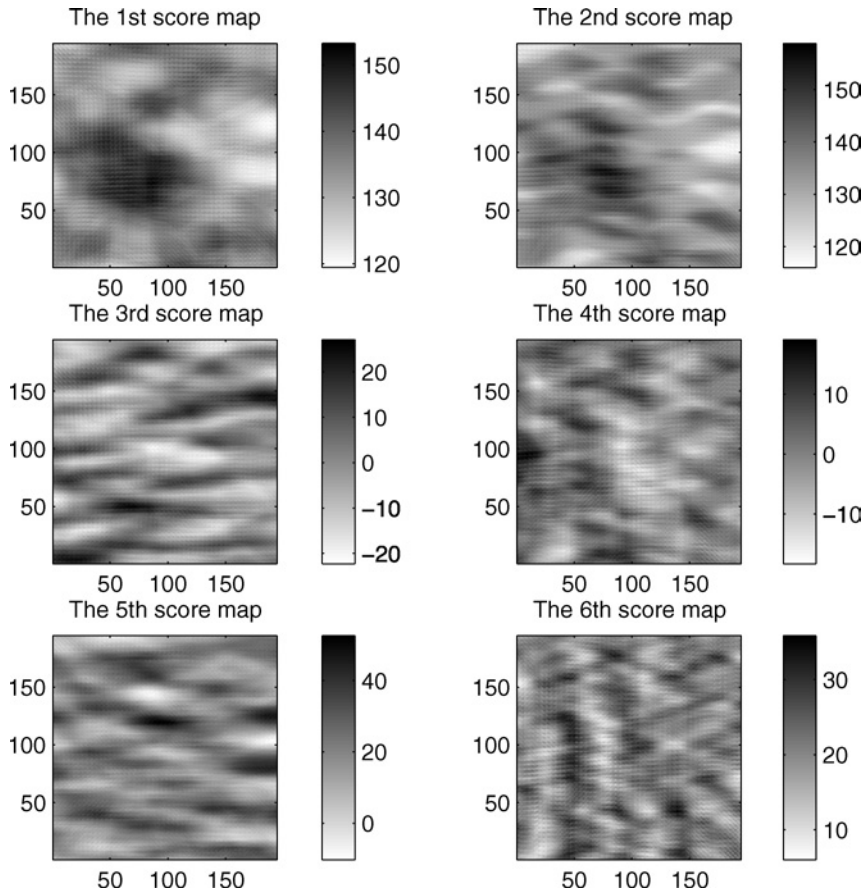

Figure 5. Six score maps for the channel Ti on the coarsest grid.

happen that many training templates are filled with a single category; in which case some of the quintile thresholds are the same, resulting in fewer effective bins and lesser RAM demand.

Through their score transforms, similar training patterns are grouped into the same bin in the score space. Figure 9 shows one such bin containing 56 channel patterns scanned from Figure 8 at the coarsest grid. For each nonempty score bin, a prototype is obtained by averaging all training patterns falling in it; that prototype can be seen as an aggregate of similar patterns. The prototype corresponding to the patterns in Figure 9 is shown at the bottom of that figure. Figure 10 shows the 20 prototypes with most replicates, as defined from the coarsest grid template applied to the channel training image of Figure 8.

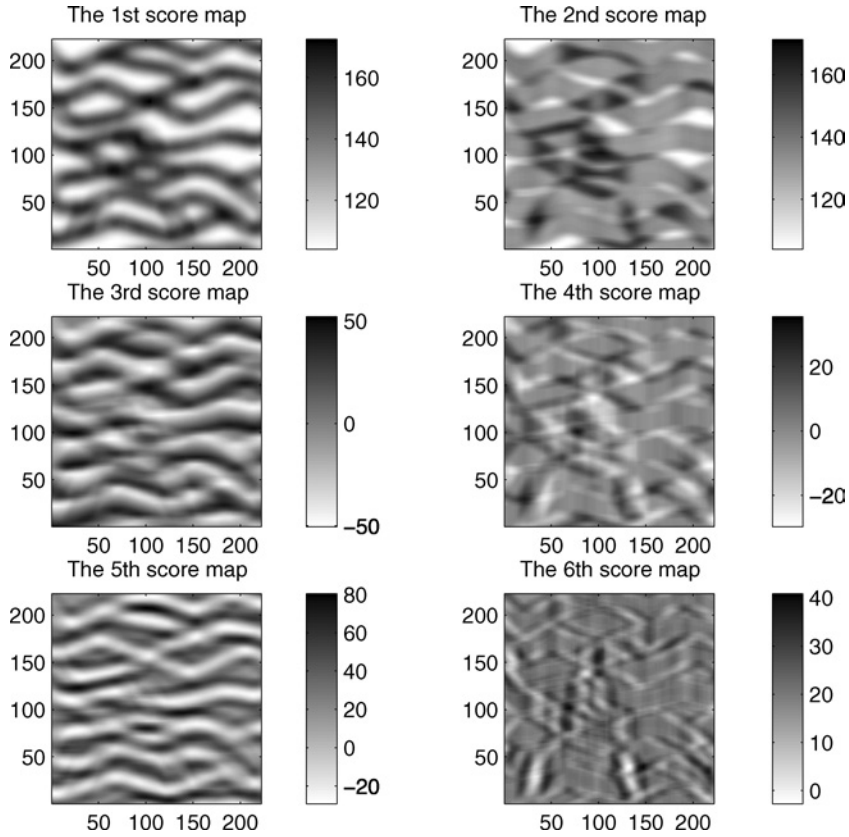

Figure 6. Six score maps for the channel Ti on the second grid.

STOCHASTIC SIMULATIONS OF PATTERNS

Based on the previous classification of the training patterns, sequential simulation can now be applied to build the simulated images.

A sequential simulation approach is performed visiting each node of the current grid along a random path different for each of the nested multiple grids used. Simulation proceeds from the coarsest grid to the finest grid. Except for the coarsest grid, all values simulated at previous grids are used only to calculate the distance between the conditioning data event and the closest training prototype. The simulated values at the coarser grid are revisited and resimulated at the current grid.

At each node to be simulated, the conditioning data are searched within a data template centered at that node. This data template has the same dimensions as that used to scan the training image at the current grid level. If there are no

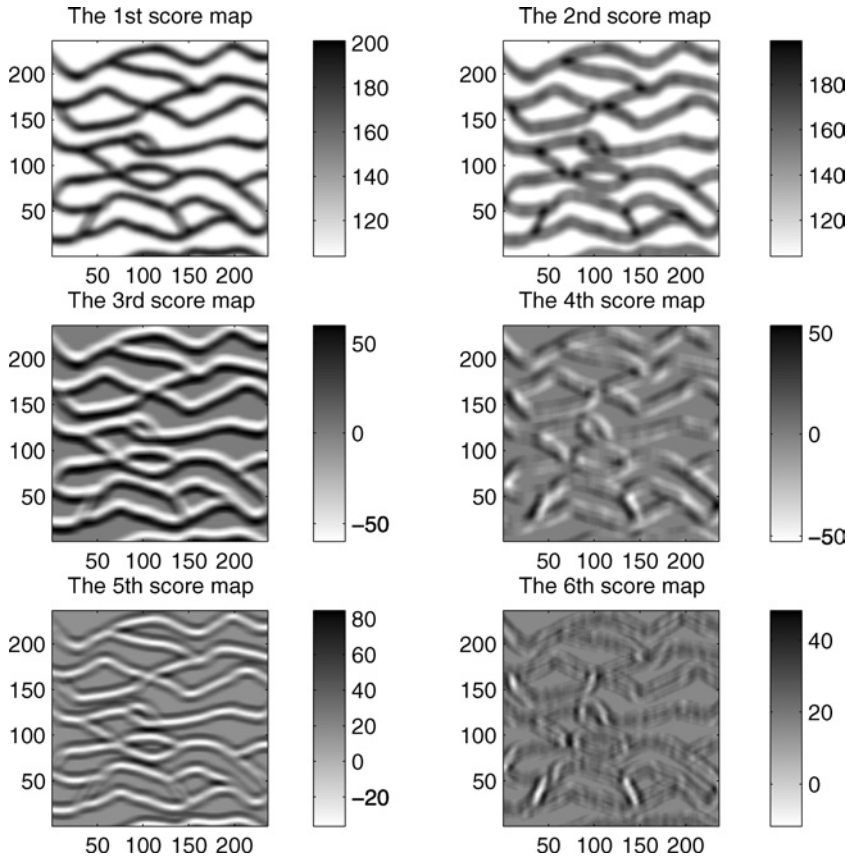

Figure 7. Six score maps for the channel Ti on the finest grid.

conditioning data within the data template, choose the template prototype closest to the target mean attribute value and pick at random a training pattern from this prototype class. If there are conditioning data in the data template, calculate the distance between this data event (DEV) and each training prototype (PROT) template recorded at the current grid level. There are three types of conditioning data: $k = 1$: hard original data; $k = 2$: previously simulated values at the current grid level; $k = 3$: values coming from patterns patched during simulation at the previous coarser grids.

The distance expression is written as

$$d(\text{DEV}, \text{PROT}) = \sum_{k=1}^{3} \omega(k) \frac{\sum_{i_k=1}^{n_k} |x^{(k)}(i_k) - y^{(k)}(i_k)|}{n_k}$$

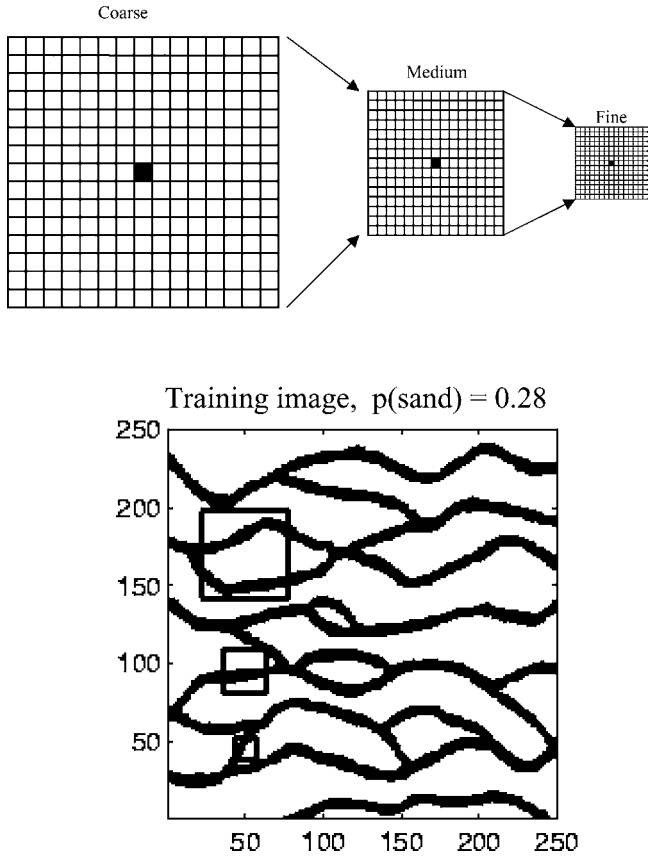

Figure 8. Channel training image and three templates (15 × 15) used for three nested multiple grids, from coars to fine.

where i_k are the pixel locations of information of type k and $\omega(k)$ are weights for the three previous data types with $\omega(1) \geq \omega(2) \geq \omega(3)$. The priority given to the hard original data ($k = 1$) ensures selection of a training pattern matching best these original data.

Once a prototype closest to the data template is identified, one training pattern is drawn from that prototype class. The particular pattern drawn may be determined from a distribution such that the simulated overall mean is gradually forced to approach the target mean. Once a specific pattern is selected, it is patched centered at the current simulation node. A specified inner part of that patch is frozen not to be revisited at the current grid. A larger inner patch makes simulation faster, but may cause discontinuities.

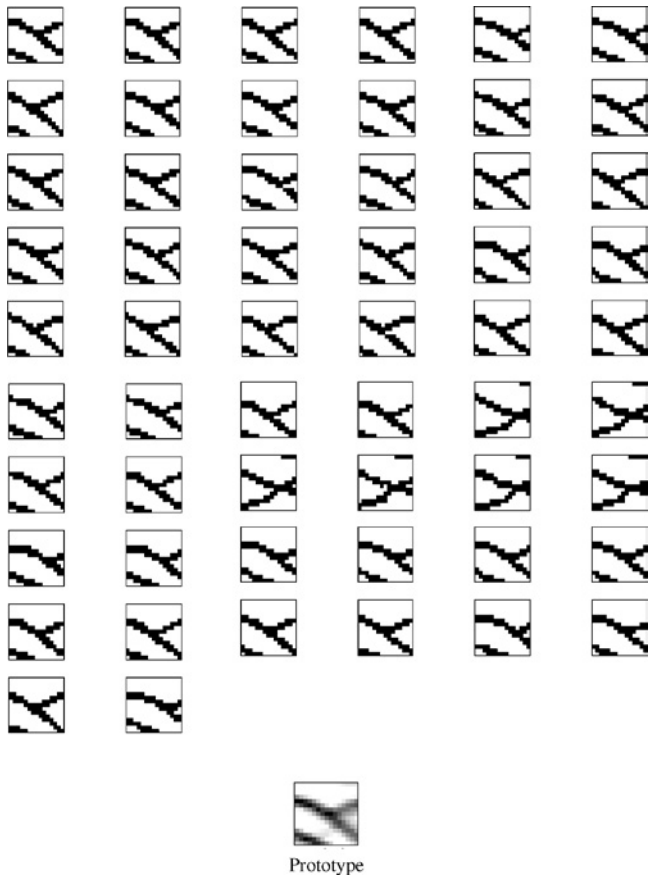

Figure 9. Fifty-six patterns falling into the same bin in the filter score space. Coarse grid 15 × 15 template.

The two flowcharts for pattern classification and simulation are displayed in Figures 1 and 2.

ILLUSTRATIONS

Consider first the simulation of channel patterns from the binary (sand/shale) training image of Figure 11(a) which is of dimension 250×250 with sand proportion $p = 0.28$. For hard data conditioning, 50 values are sampled from the lower left corner region of that training image, see Figure 11(b). A 15×15 data template is used to scan the entire training image over three nested grids from coarsest to finest, see the top row of Figure 8.

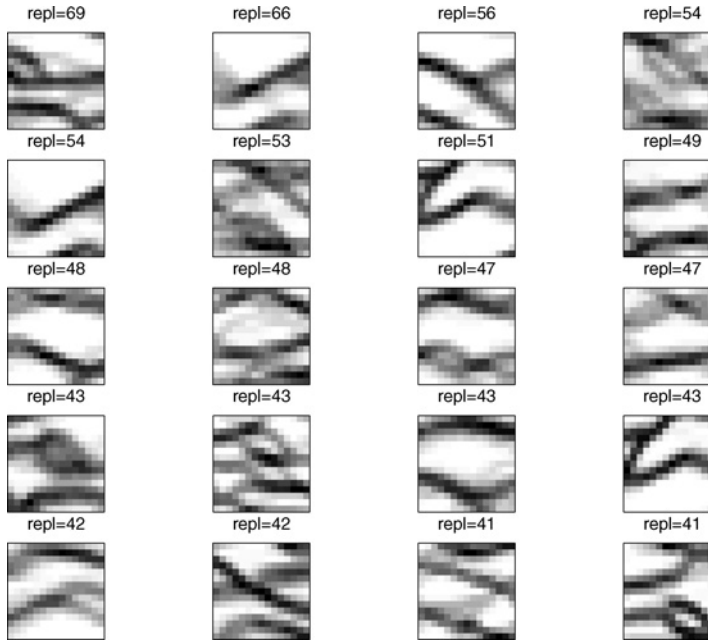

Figure 10. Prototypes with most replicates at the coarset grid.

Figures 11(c)–(e) give the result of one single-conditional simulation progressing over the three nested multiple grids. Examples of the pattern classification and prototypes at the coarsest grid were shown in Figures 9 and 10. For this case example, we used a 15×15 template with a 11×11 inner patch. It can be seen from Figure 11 that the channel structures are reasonably well reproduced from coarse to fine grid. Unfreezing the data values simulated at coarser grids leads to better shape reproduction because poorly matched patterns can be corrected during simulation at the finer grids. To examine the impact of data conditioning, we generated 30 conditional realizations and averaged them to produce an E-type estimation of sand probability, see Figure 11(f). Since all conditional simulations honor exactly the 50 hard data, there is a higher probability of simulating sand when the simulation node is closer to a sand hard datum.

A continuous variable training image was downloaded from a texture synthesis website (http://www.vision.ee.ethz.ch/∼rpaget/nonparaMRFfastContents_Xu_Zhu_Shum_Guo.htm) and displayed in Figure 12(a): it is a mosaic cross section of packed stones, with visible gray scale textures and sharp boundaries. Figures 12(b)–(g) display the six score maps at the finest grid. We used a template of size 21×21 pixels to classify the training patterns over three nested grids. The same nonconditional simulated realization at the three different grids is shown in

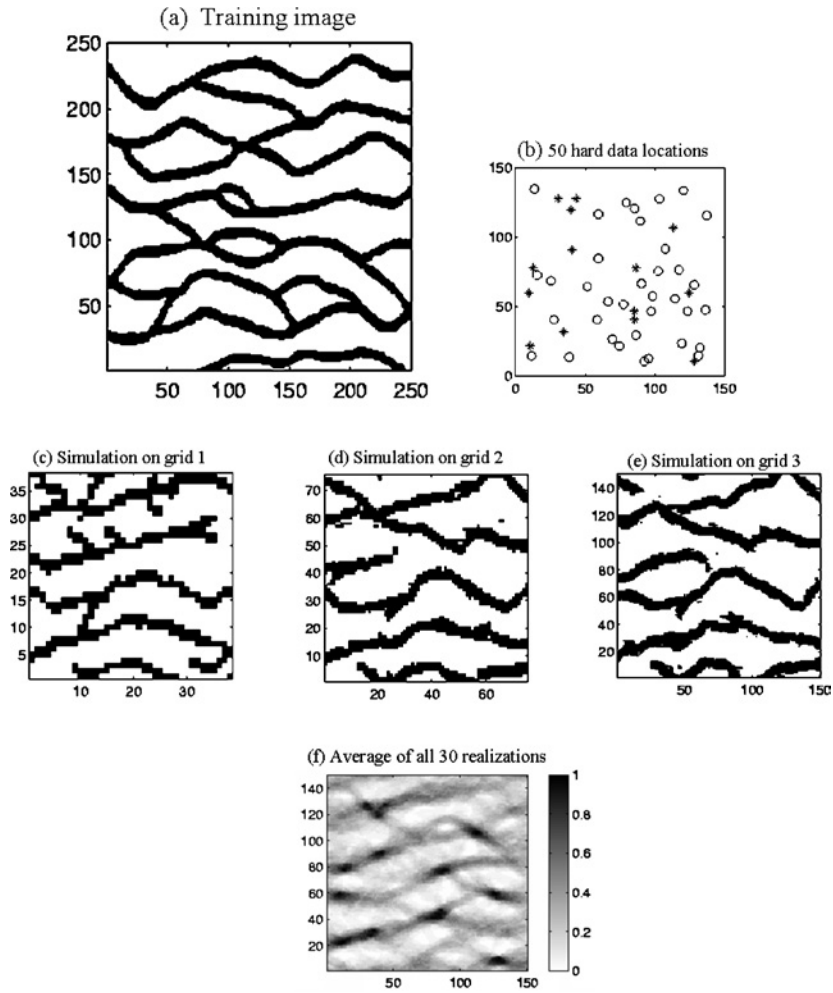

Figure 11. (a) Channel training image; (b) hard data locations; (c)–(e) Conditional simulation progressing over the three nested grids; (f) E-type average of 30 conditional realizations.

Figures 13(b)–(d). A 21 × 21 template with an inner patch of 15 × 15 was used. Figure 14 shows three additional nonconditional realizations on the finest grid.

DISCUSSION AND FURTHER DEVELOPMENTS

The pattern classification algorithm, described above, appeared to work well for our illustrative examples, in the sense that the resulting simulations appear

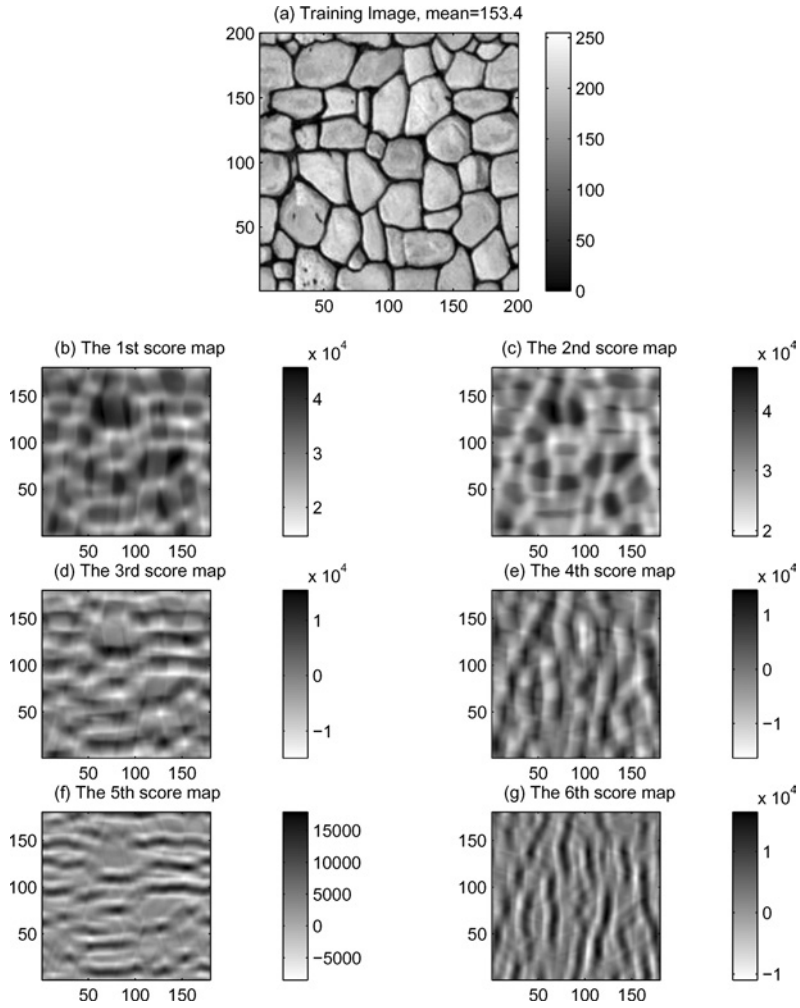

Figure 12. Stone training image and six score maps at the finest grid.

to reproduce important features of the training images. The illustrations that we used were all of limited size and were all two dimensional. In 3D and for more complex training images, we will likely need classification procedures that are more refined than the simple cross-classification induced by partitioning each of the filter scores separately, as done above.

The case of a large number (>2) of nonordered categories still needs investigation, for it is not equivalent to the continuous variable case. Indeed, if category 2 is not necessarily nested in category 1 and does not include category

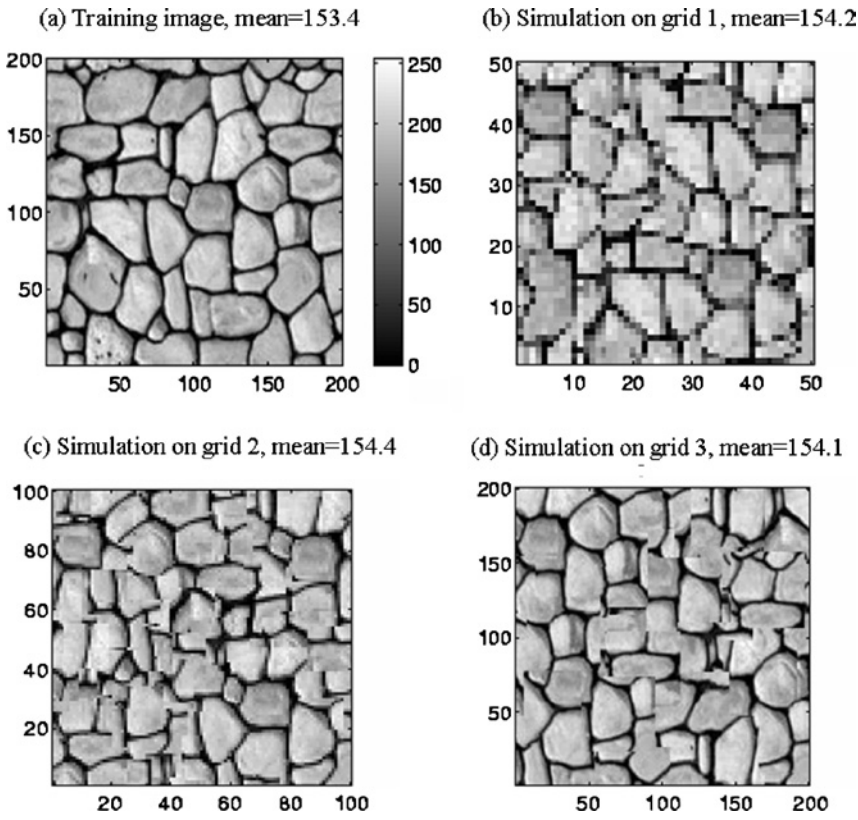

Figure 13. (a) Stone training image; (b)–(d) The same non-conditional realization seen from coarsest to finest grid.

3, the corresponding three category indicators cannot be considered as class indicators of a continuous variable. The weighted distance proposed in the section on "Stochastic Simulation of Patterns" may still be appropriate if adapted to count the number of pixel mismatches within the template. However, linear filters applied to categorical indicators have little significance, therefore other pattern summaries should be considered.

The various implementation parameters (number of nested grids, template size, distance weights) need a careful sensitivity analysis as was done for the *snesim* algorithm by Liu (2003).

At this point, the *filtersim* algorithm only exists as test code (in Matlab) and is being optimized in C++ language. Consequently no fair CPU comparison with other mp simulation algorithms is yet possible. We expect that it will be

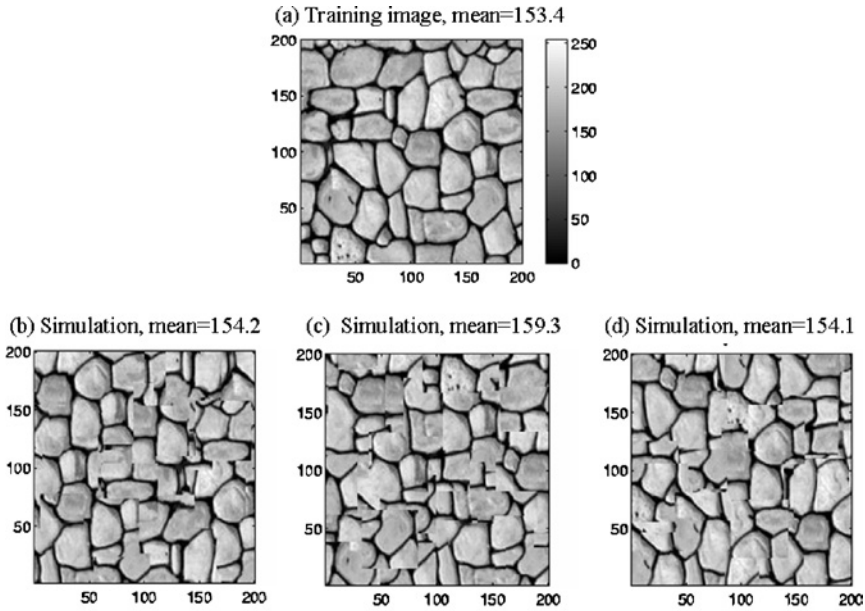

Figure 14. (a) Stone training image; (b)–(d) Three final nonconditional simulations at the finest grid.

comparable to *snesim* in terms of CPU but requiring much less RAM thanks to the dimension reduction brought by the classification in the filter score space.

One major difference with the *snesim* approach (Strebelle 2000, 2002) is that any conditioning data template no matter how sparsely or densely defined could be considered without having to drop any single datum value; this is, of course, obtained at the cost of approximating that complete conditioning data event by a class of training patterns. The tradeoff is approximation of the data event versus its reduction in size.

In practice, only a fraction of the large number of potential classes in the quintile cross-classification are actually populated by patterns occuring in the training image. In 3D with nine filters the potential number of classes would grow to 5^9 or nearly 2 million classes, with only a small fraction actually populated. Therefore, simple quintile cross-classification will need to be replaced by a more intelligent classification procedure. We are now exploring other methods for pattern classification in the filter score space that will not be sensitive to the number of filters used, will not generate spurious classes, and will seek to optimize within-class homogeneity. For example, tree-structured classification algorithms that work with sequential binary divisions could be considered (Hastie, Tibshirani, and Friedman, 2001).

CONCLUSIONS

In this paper, we apply a set of filters to scan the training images. The local training patterns and textures are then classified by the set of filter scores. This leads to a significant dimension reduction of the training patterns and consequent RAM demand. This prior classification allows a storage of the training patterns to be used for simulation. The simulation proceeds by sequentially visiting each simulation grid node and identifying the training pattern class closest to the local data conditioning that simulation node. We sample a specific pattern from the identified pattern class, and patch it centered at the simulation node. Freezing the inner part of the patched pattern not only makes the simulation faster but also ensures better pattern reproduction. Multigrid simulation is implemented, allowing for pattern reproduction at different scales. Examples show that the approach proposed works well for ordered categorical variable and continuous variable training images. As it stands now, the code has been developed only for 2D applications. Graduating into real 3D applications would call for innovative ways to perform classification of complex 3D training patterns.

REFERENCES

Deutsch, C. V., and Journel, A. G., 1998, GSLIB: Geostatistical software library and user's guide, 2nd ed.: Oxford University Press, New York, 368 p.

Guardiano, F., and Srivastava, R. M., 1993, Multivariate geostatistics: Beyond bivariate moments: *in* Soares, A., ed., Geostatistics-Troia, v. 1: Kluwer Academic, Dordrecht, The Netherlands, p. 133–144.

Hastie, T., Tibshirani, R., and Friedman J., 2001, The elements of statistical learning, data mining, inference, and prediction: Springer, New York, p. 266–272.

Liu, Y., 2003, Downscaling seismic data into a geological sound numerical model: Unpublished doctoral dissertation, Stanford University, 193 p.

Strebelle, S., 2000, Sequential simulation drawing structures from training images: Unpublished doctoral dissertation, Stanford University, 187 p.

Strebelle, S., 2002, Conditional simulation of complex geological structures using multiple-point statistics: Math. Geol., v. 34, no. 1, p. 1–21.