A New MPS Simulation Algorithm for Continuous Variables based on NL means filtering

Jeff B. Boisvert

ABSTRACT
Simulation of complex features beyond two point statistics is difficult with traditional geostatistical techniques. A novel algorithm for multiple point statistical simulation is presented. The difference in patterns is used as the similarity measure between the training image and the simulation. A kernel provides the weighting to use. The algorithm is currently computationally intensive and cannot be applied to large 3D problems. 2D examples are shown to highlight the benefits of the technique.

Introduction

Simulation methodologies based on linear relationships and the properties of the Gaussian distribution are common in geostatistical practice but often fail to capture complex spatial relationships known to exist in certain geological formations. In recent years multiple point (geo)statistics (MPS) has become popular for modeling facies that display complex features. The typical example is a fluvial reservoir with the nonlinear channels being difficult to model (Figure 1). Other examples exist, but the motivation for MPS is clear, the complex spatial structure of facies is difficult to model with traditional methods (i.e. sequential indicator simulation, truncated Gaussian, deterministic, etc).

MPS algorithms are typically limited to considering categorical variables for two reasons (1) few MPS algorithms exist for continuous variables and (2) it is difficult to select a training image (TI) for continuous variables. While this paper will address the first concern there will be little discussion as to TI selection for continuous variables.

The proposed methodology is similar to FILTERSIM (Wu et al, 2008) with more emphasis on the simulation of point locations rather than entire patterns. FILTERSIM is currently one of the few implementations of MPS simulation that consider continuous variables. A review of MPS algorithms is not provided here, the interested reader is referred to the works of J. Caers, S. Lyster (in multiple CCG reports) or S. Strebelle for detailed reviews of MPS algorithms.

Background

The proposed MPS algorithm is based on non-local means (NLM) denoising (or filtering) (Buades et al. 2006). The NLM algorithm is intended to remove noise from digital images while maintain the underlying structure of the image to ensure only noise is removed, rather than information. The idea is to visit each location (pixel) of the digital image and determine a smoothed value at the location (**u**). The smoothed value should be informed by similar areas in the image. Thus a pattern (MPS template in geostatistics) is scanned over the image to determine patterns similar to location **u**. Specifically, the following is used to determine the smoothed value at **u**, S(**u**).

$$S(\mathbf{u}) = \frac{1}{C_u}\int f\left(\Delta\big(W(\mathbf{u}), W(\mathbf{v})\big)\right) Z(\mathbf{v}) dv$$

where $\Delta\big(W(\mathbf{u}), W(\mathbf{v})\big)$ is the difference between a window centered at location **u** and **v**, C_u is a normalizing factor and S(**u**) is the resulting smoothed value of a variable Z.

This is implemented by scanning the image, determining the difference between templates, $\Delta\big(W(\mathbf{u}), W(\mathbf{v})\big)$, and using this as the weight to assign to the pixel at location **v**. This difference measure could be the Euclidean difference, or any other metric:

$$\Delta^2\big(W(\mathbf{u}), W(\mathbf{v})\big) = \frac{1}{N^2}\sum_{i=1}^{N^2}\big(Z(\mathbf{u_i}) - Z(\mathbf{v_i})\big)^2$$

The smoothed value at a given location (u) becomes:

$$S(u) = \frac{1}{C_u}\sum Z(\boldsymbol{u})w(\boldsymbol{u},\boldsymbol{v})$$

where the weight is

$$w(\boldsymbol{u},\boldsymbol{v}) = e^{\frac{\Delta^2}{h^2}}$$

where h^2 is an input parameter the user selects to control the level of smoothness.

Implementation of the algorithm involves scanning the digital image with a square pattern and calculating the difference between all such patterns and the location of interest (Figure 2). The value at location **u** is replaced with the smoothed value and the algorithm continues.

The similarity between NL means filtering and MPS simulation as defined in the field of geostatistics is clear. The remainder of this paper will discuss the implementation of this algorithm for geostatistical simulation of continuous or categorical variables using a TI (the digital image in the above discussion). The smoothed value becomes the simulated value and the patterns are the MPS template.

MPS algorithm

The algorithm is sequential and visits each cell of the realization in random order. The nearby conditioning data are used as the template around location **u** while the TI provides the **v** locations in the above formulism (Figure 2). The difference between each **v** pattern in the TI is taken and corrected for the number of data in the pattern. In the NL means algorithm the pattern is always full, however, in a sequential simulation approach the number of conditioning data will vary, thus the $\Delta^2\big(W(\mathbf{u}), W(\mathbf{v})\big)$ must be modified accordingly where *n* is the number of conditioning data rather than the number of cells in the pattern:

$$\Delta^2\big(W(\mathbf{u}), W(\mathbf{v})\big) = \frac{1}{n^2}\sum_{i=1}^{n}\big(Z(\mathbf{u_i}) - Z(\mathbf{v_i})\big)^2$$

The specific steps in the algorithm are:

1) Determine conditioning data and previously simulated nodes (they are treated the same)
2) Scan the TI for this pattern of values and determine the difference ($\Delta^2\big(W(\mathbf{u}), W(\mathbf{v})\big)$) and weight ($w(\boldsymbol{u},\boldsymbol{v})$) for each pattern
3) Simulate a value (there are a number of options for how to do this that are discussed further below).
4) Repeat for all cells in the realization.

The user must supply the *h* value to use as the smoothing parameter. Selecting a higher value of h results in smoother realizations. The effect of *h* will be shown with examples below. There are three options for simulation of a value at a location once $w(\boldsymbol{u},\boldsymbol{v})$ has been determined for each pattern in the TI.

Option 1: CDF. A traditional cdf construction approach is the most intuitive but does result in higher CPU requirements. The cdf is constructed by standardizing $w(\boldsymbol{u},\boldsymbol{v})$ to 1 and using this as the weights for each value located at the center of pattern v_i. A value is drawn from this cdf in a traditional Monte Carlo simulation.

Option 2: Greedy. Alternatively, a greedy algorithm could be considered such that the pattern, v_i with the smallest Δ^2 is used. This results in a faster algorithm but does not allow for the full exploration of the uncertainty.

Option 3: Weighted. A final methodology for simulating a value is an estimate approach where $w(\boldsymbol{u}, \boldsymbol{v})$ is used in the same way as the original NL means algorithm intended:

$$Z(u) = \frac{1}{C_u} \sum Z(\boldsymbol{v_i}) w(\boldsymbol{u}, \boldsymbol{v_i})$$

A note on categorical variables

In this algorithm the only difference when implementing categorical variables is the calculation of Δ^2. When categories are unrelated the difference can be an indicator (0=same, 1=different categories) or the difference can come from a user input matrix (note this was not implemented in the program, however, the modification would be slight):

Table 1: Difference matrix showing the relationship between facies. For example, the difference between facies 1 and 3 would be 0.2 rather than 1.0.

Category	**1**	**2**	**3**	**4**
1	0	0.5	0.2	1
2		0	0.9	1
3			0	1
4				0

Implementation details

The algorithm was implemented in 2D only. The intention of the algorithm was to perform MPS with continuous variables and 3D TI's of continuous variables are difficult to obtain. 2D images based on outcrops or interpretations are used in this paper as TI's. There is no theoretical restriction to 3D; however, the CPU requirements of the algorithm are high and 3D simulation is not currently practical.

The algorithm is implemented with a multigrid approach. First simulation proceeds on the coarse grid with the fine grid values simulated conditional to the coarse grid.

To decrease CPU time, the algorithm is implemented in parallel. The parallelization of the program is accomplished by simultaneously simulating at multiple locations in the model depending on the predetermined random order. For example, if 5 processors are used, 5 random locations in the model are simulated, one location per processor. If the bounding boxes of the patterns overlap one of the simulated values is discarded. Depending on the size of the model, the number of discarded values (i.e. number of additional locations to simulate) ranges from 0% to 30%. Note that even when many locations require resimulation there is still a significant increase in speed because of the large number of processors that can be utilized. Consider a 200x200 model with a 7x7 pattern size, the resulting speed up is nearly optimal (Figure 4). In this case there is little lost time due to resimulating at overlapping locations, with 8 processors there were only 87 locations resimulated (0.2%). The increase in time over the ideal case (Figure 4) is due to processors waiting for other processors to finish after simulating.

There are two versions of the program NLmeans_MPS, one for continuous variables and one for categorical. The parameter file for the continuous version is:

Line 1	data.dat	-file with data
Line 2	1 2 4	-xcol ycol valuecol
Line 3	TIflat.out	-training image file
Line 4	1	-col for data in TI
Line 5	239 154	-size of the TI, nx ny
Line 6	5 5	-size of the MPS template X Y
Line 7	4	-number of mulitgrids (4 is a good number, 5 might be too slow)
Line 8	.132	- smoothing parameter
Line 9	output.out	-file for the realization
Line 10	200 0.5 1	-nx,xmn,xsiz
Line 11	200 0.5 1	-ny,ymn,ysiz

Line 12	3654	-rand seed
Line 13	5	-number of threads
Line 14	.5	-MPS patterns that have this difference and larger will not be considered
Line 15	1	-0 use the greedy algorithm, 1=draw value, 2=weighted average
Line 16	0	- 1=use squared difference, 0 = use abs difference
Line 17	2	-power on the kernel (1=exponential, 2=Gaussian ...)

Line 8: The smoothing parameter is squared in the program
Line 14: patters that have a Δ larger than this value will be ignored in the simulation
Line 15: Type of simulation
Line 16: For the calculation of Δ , Δ^2 or $|\Delta|$ can be used.
Line 17: a Gaussian or exponential kernel can be used.

The parameter file for the categorical version is:

Line 1	data.dat	-file with data
Line 2	1 2 4	-xcol ycol valuecol
Line 3	train.dat	-training image file
Line 4	1	-col for data in TI
Line 5	200 100	-size of the TI, nx ny
Line 6	2 2	-size of the MPS template X Y
Line 7	4	-number of mulitgrids (4 is a good number, 5 might be too slow)
Line 8	.1	- sigma (smoothing parameter)
Line 9	output.out	-file for the realization
Line 10	200 1 1.0	-nx,xmn,xsiz (size of the realization must be same as TI)
Line 11	100 1 1.0	-ny,ymn,ysiz
Line 12	2345	-rand seed
Line 13	5	-nfacies
Line 14	0 2 3 4 5	-facies codes, there cannot be facies codes in the file other than these codes
Line 15	0.68 0.17 0.04 0.02 0.09	-global prop for each facies
Line 16	2	-number of threads
Line 17	2	-2=yes preprocess the patterns...
Line 18	0	-MPS patterns that have this support (and lower) will not be considered

Line 17: sorting the patterns and counting the number of unique patterns in the TI usually reduces CPU time.
Line 18: if the pattern occurs less than this many times in the TI, it will be ignored. Even setting to 1 can change the results significantly. Because the distance metric is used for all patterns, it is best to use all the patterns (set to 0) however this can be CPU intensive.

Examples

The first example is a synthetic example for checking the algorithm. SGS (Figure 4) is used as a TI for the algorithm and the expected result is similar in character to the SGSs run with similar variogram and histogram characteristics (Figure 5).

The second example is a categorical variable TI that has been used in previous testing of MPS algorithms (Figure 6). The resulting realizations are similar in structure (Figure 6) with some reduction in the length of the linear features because of the limited pattern size used (7x7 pattern).

The TI for the third example (Figure 7) shows more complex structure and will be used to demonstrate the various options for continuous variable simulation. Figures 8a and 8b show all three options for simulation as well as various sized templates and options for the difference measure. The largest difference in results is due to the template size.

Note that CPU times have been purposefully ignored as the algorithm is not practical for realistic 3D models (although 2D models of hundreds of thousands of cells can be simulated in a reasonable time period). More work is required to improve CPU time, note that similar pattern scores could be used as presented for the FILTERSIM algorithm (see associated literature).

Future Work

The NL means algorithm shows promise for categorical simulation, but the computational requirements of the difference operations for continuous variables makes a naïve implementation (as presented here) not practical for large 3D models. Parallelization of the program helps, but the CPU requirements are still large. Clustering and other techniques will be explored to reduce CPU requirements.

Generation of TIs for MPS simulation is always an issue. When considering continuous variables the issue is compounded. It is difficult to consider an object based methodology (as with categorical variables). Likely, blasthole data in the mining context or photographs of outcrops will have to be considered for continuous variables.

References

Buades, A., Coll, B., Morel, J.M., 2005, A review of image denoising algorithms, with a new one, Multiscale Modeling and Simulation (SIAM interdisciplinary journal), 4(2):490-530.

Liu, Y, 2006, Using the Snesim program for multiple-point statistical simulation. Computers and Geoscience. 32, 1544-1563.

Wu, J., Boucher, A., and Zhang, T., 2008. A SGeMS code for pattern simulation of continuous and categorical variables: FILTERSIM. Computer & Geoscience. 34(12) 1863-1876.

Figures

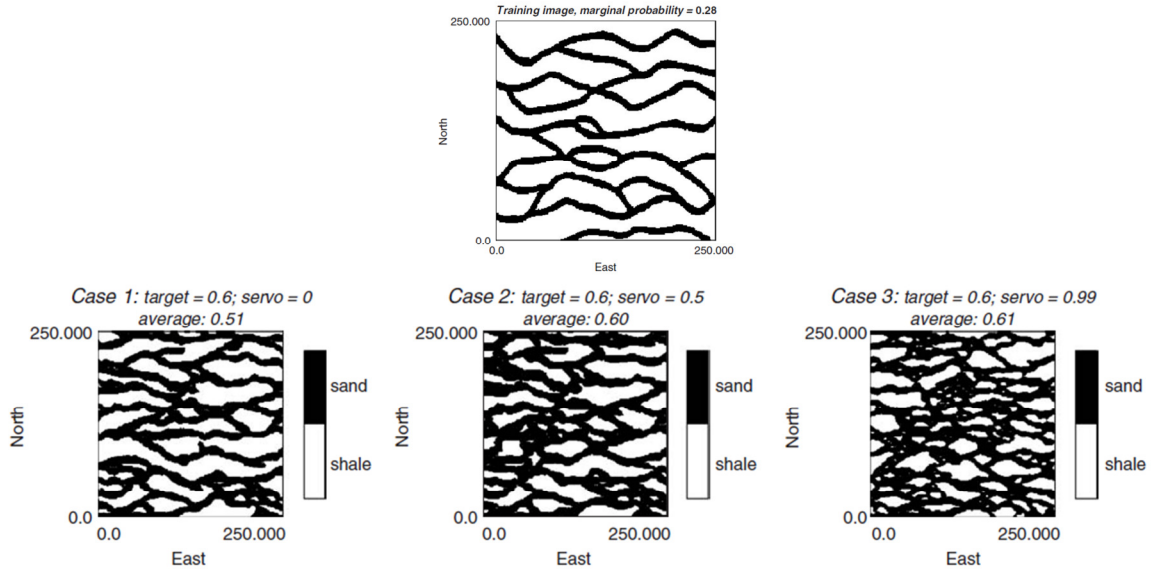

Figure 1: Above: fluvial TI used in MPS. Below: MPS realizations with multiple options in SNESIM. Source: (Liu 2006)

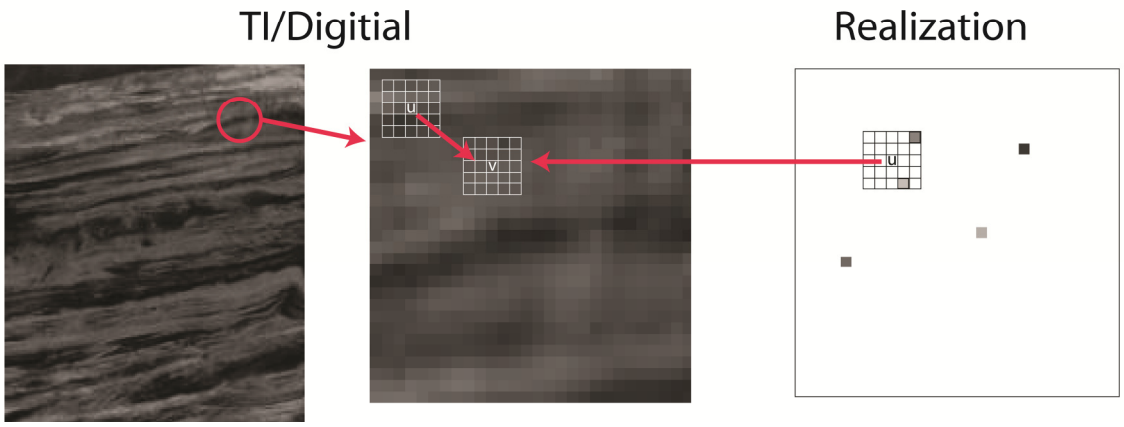

Figure 2: A pattern centered at location **u** and **v**. Left: the TI or digital image. Center: exploded view of pixels in the one area of the TI. Left: simulated model. The difference (delta) would be the difference between the values in the model in each of the patterns. When considering the NL means smoothing algorithm the v location is all other locations in the model. When considering the MPS algorithm (right) the difference is with other previously simulated locations in the pattern centered on location u.

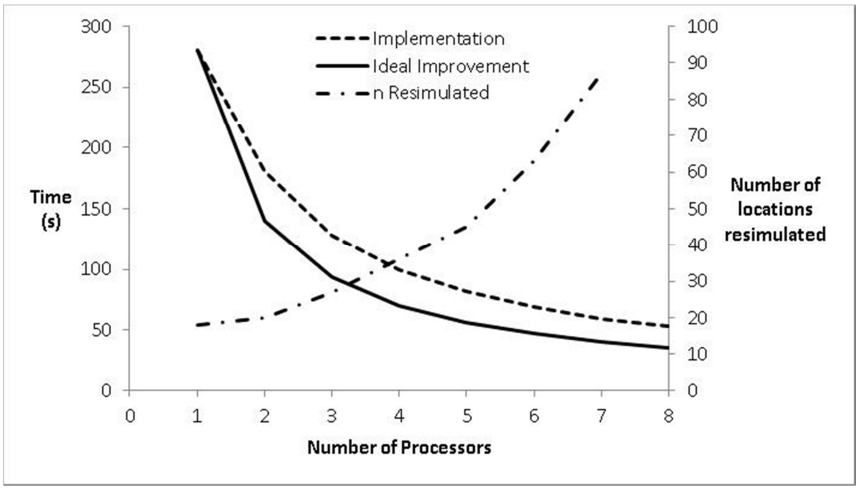

Figure 3: CPU time improvement with additional processors.

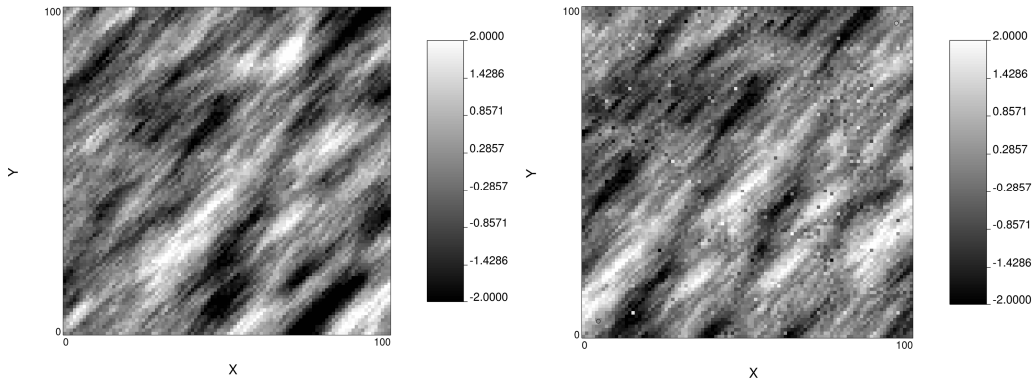

Figure 4: Left: SGS Ti. Right: NL means MPS realization.

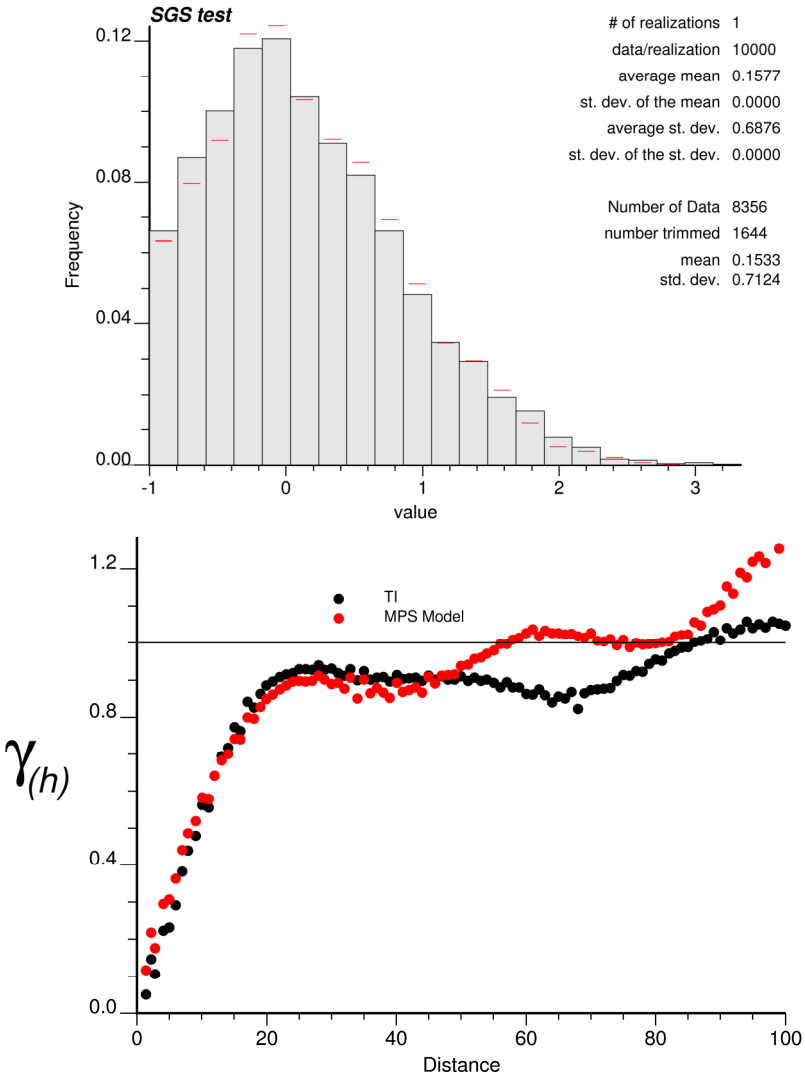

Figure 5: Reproduction of the histogram (red dashes are the MPS realization) and variogram with the SGS TI.

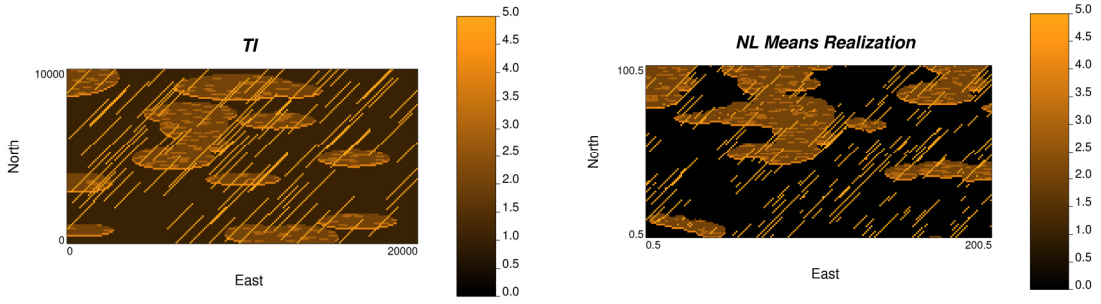

Figure 6: Left: TI used for categorical simulation. Right: NL means realization.

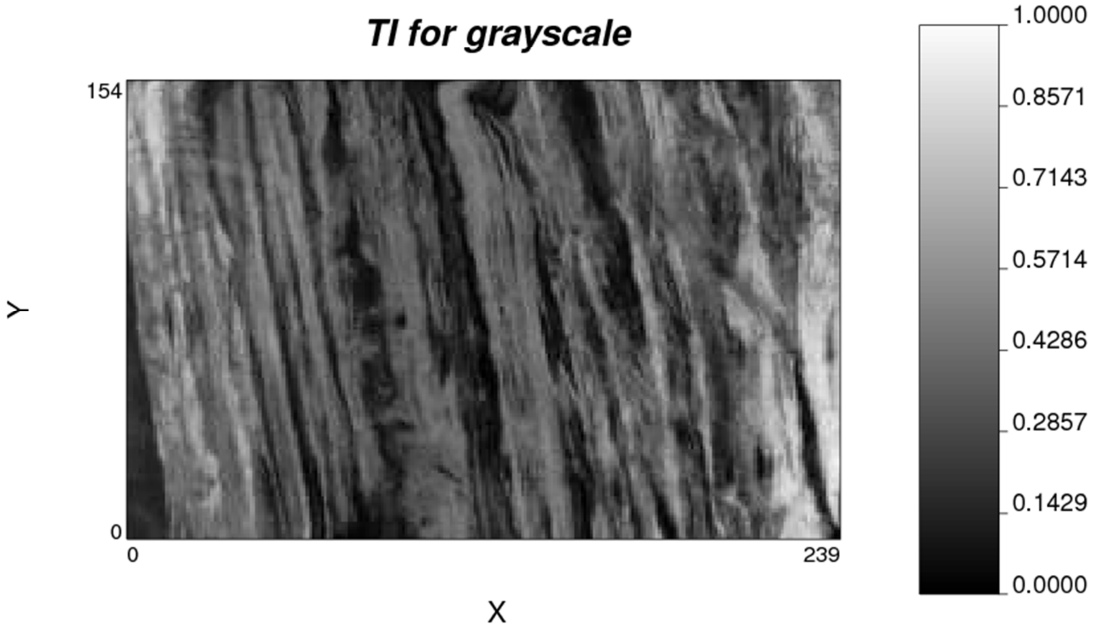

Figure 7: TI used for continuous variable simulation (search.datapages.com).

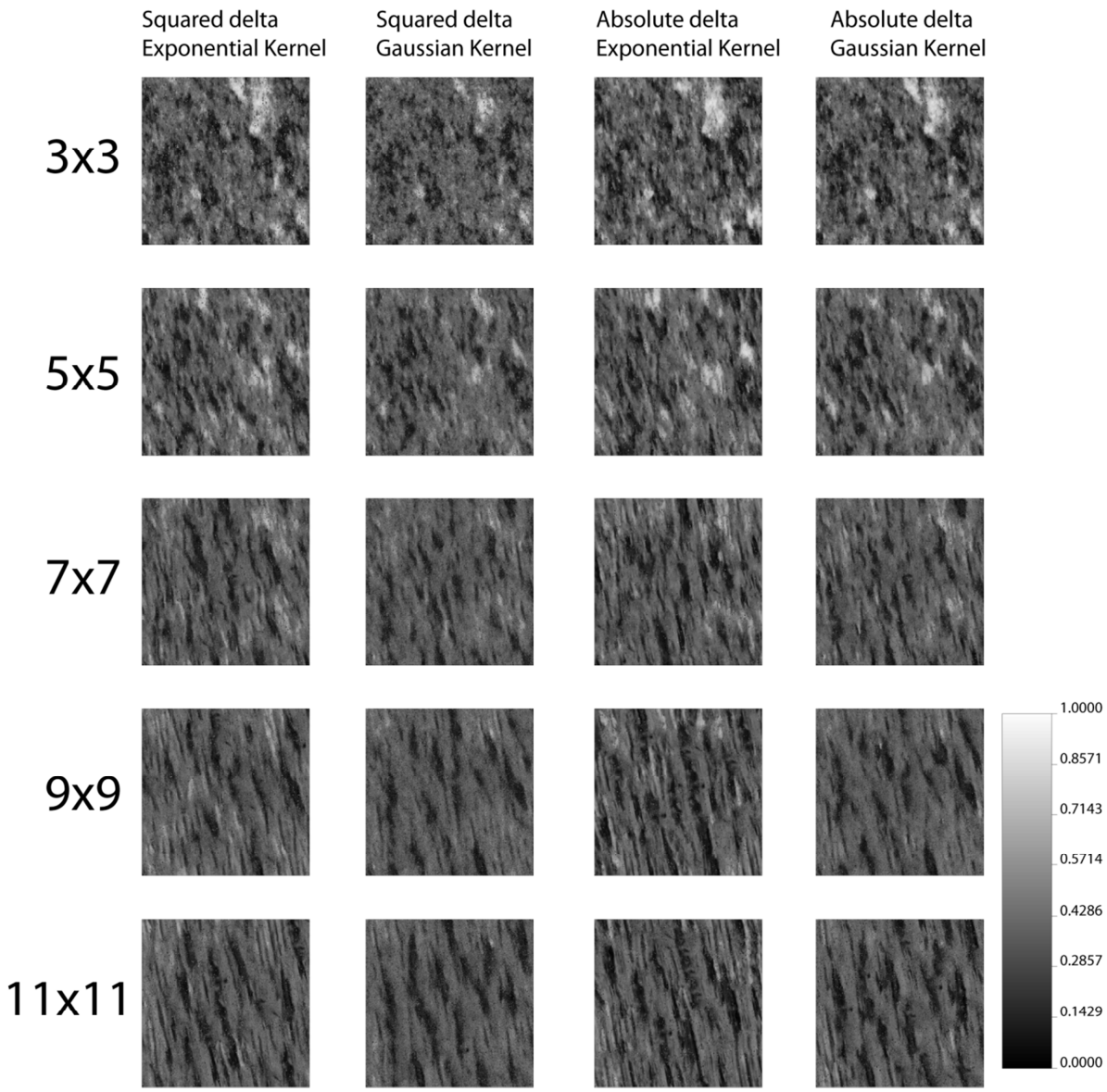

Figure 8a: Multiple realizations of NL means realizations using the 'draw from a cdf' option. Rows represent different pattern sizes and columns represent options available for the Δ function.

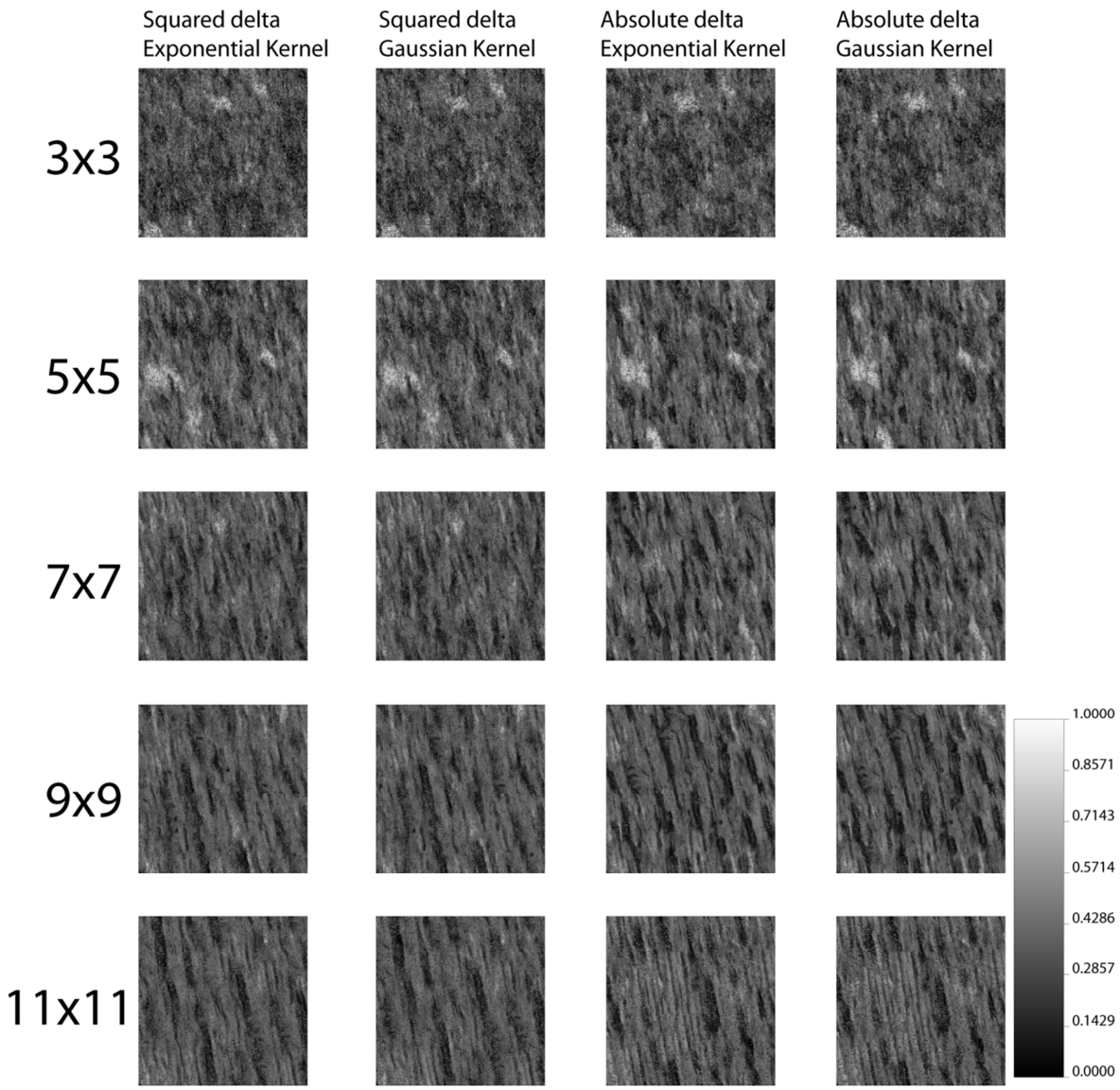

Figure 8b: Multiple realizations of NL means realizations using the 'minimum weight' option. Rows represent different pattern sizes and columns represent options available for the Δ function.