A Frequency Matching Method for Generation of a Priori Sample Models from Training Images

Lange, Katrine; Cordua, Knud Skou; Frydendall, Jan; Hansen, Thomas Mejer; Mosegaard, Klaus

Published in:
Proceedings of IAMG 2011

Publication date:
2011

A Frequency Matching Method for Generation of a Priori Sample Models from Training Images

Katrine LANGE[1,2], Knud Skou CORDUA[1], Jan FRYDENDALL[1], Thomas Mejer HANSEN[1], and Klaus MOSEGAARD[1]

[1] Center of Energy Resources Engineering,
Department of Informatics and Mathematical Modeling,
Technical University of Denmark, Denmark,
[2] katla@imm.dtu.dk

Abstract

This paper presents a Frequency Matching Method (FMM) for generation of a priori sample models based on training images and illustrates its use by an example. In geostatistics, training images are used to represent a priori knowledge or expectations of models, and the FMM can be used to generate new images that share the same multi-point statistics as a given training image.

The FMM proceeds by iteratively updating voxel values of an image until the frequency of patterns in the image matches the frequency of patterns in the training image; making the resulting image statistically indistinguishable from the training image.

1. Background

Consider a training image with N voxels (or pixels if the image is only 2D). Let z_k denote the value of the kth voxel of the image, $k = 1, \dots, N$. Here, we shall assume that the training image is a realization of a random process satisfying:

1) Voxel value z_k depends only on the values of the voxels in a certain neighborhood $\mathcal{N}_k$ around voxel k. Voxel k itself is not contained in $\mathcal{N}_k$. Let $\mathbf{z}_k$ be an ordered vector of the values of the voxels in $\mathcal{N}_k$; we then have:

$$f_Z\left(z_k \middle| z_N, \dots, z_{k+1}, z_{k-1}, \dots, z_1\right) = f_Z(z_k|\mathbf{z}_k).$$

2) For an image of infinite size the geometrical shape of all neighborhoods $\mathcal{N}_k$ are identical. This implies that if voxel k has coordinates (k_1, k_2, k_3), and voxel l has coordinates (l_1, l_2, l_3), then:

$$(n_1, n_2, n_3) \in \mathcal{N}_k \Rightarrow (n_1 - k_1 + l_1, n_2 - k_2 + l_2, n_3 - k_3 + l_3) \in \mathcal{N}_l.$$

3) We assume ergodicity, i.e.:

$$\mathbf{z}_k = \mathbf{z}_l \Rightarrow f_Z(z_k|\mathbf{z}_k) = f_Z(z_l|\mathbf{z}_l).$$

The basis of sequential simulation (e.g. Strebelle, 2002) is to exploit the assumptions above to estimate $f_Z(z_k|\mathbf{z}_k)$, and to use these conditions to generate new realizations of the random process from which the training image is a realization. The FMM does not operate by directly using conditional probabilities but it represents images by their frequency distribution, which is derived using neighborhoods of voxels. The frequency distribution is closely related to conditional probabilities.

2. The Frequency Distribution

Before presenting the FMM we need to define what we denote the frequency distribution. To do so we will reuse the concept of neighborhoods from section 1 as well as the notation. Given an image with the set of voxels $\mathcal{Z} = \{1, \dots, N\}$ and voxel values $z_1, \dots, z_N$ we define the template function Ω as a function that takes as argument a voxel k and returns the set of voxels belonging to the neighborhood of voxel k. The neighborhood is denoted $\mathcal{N}_k$, and we will use the notation $\mathcal{N}_k = \Omega(k)$.

In the FMM the neighborhood of a voxel is indirectly given by the statistical properties of the image itself; however, the shape of a neighborhood satisfying the assumptions from section 1 is unknown. For each training image one must therefore define a template function that seeks to correctly describe the neighborhood.

Let $|\mathcal{N}_k|$ denote the number of voxels in $\mathcal{N}_k$. We define the set of inner voxels, $\mathcal{Z}_{in}$, of the image as:

$$\mathcal{Z}_{in} = \left\{ k \middle| |\mathcal{N}_k| = \max_{l \in \mathcal{Z}} |\mathcal{N}_l| \right\}.$$

Typically, voxels on the boundary or close to the boundary of an image will not be inner voxels. It is the choice of template function that determines whether or not a voxel is an inner voxel.

The frequency distribution of an image is computed by scanning through all inner voxels of the image. For each of these we identify first the neighboring voxels and then the values of those. For voxel $k \in \mathcal{Z}_{in}$, the values of the neighboring voxels are denoted by the vector $\mathbf{z}_k$. The length of this vector equals the number of voxels in the neighborhood $\mathcal{N}_k$, which will be constant for all inner voxels; this follows trivially from the definition of inner voxels. We denote this number n. As each voxel can take on m different values, there exists up to m^n different types of neighborhoods; i.e. m^n different combinations for the values in $\mathbf{z}_k$.

Using the above definition of a neighborhood we now introduce the concept of patterns. The kth pattern $\mathcal{P}_k$ of the image is defined as the union of an inner voxel k and the set of its neighboring voxels. We will denote voxel k the center voxel of the kth pattern regardless of the geometrical shape of $\mathcal{P}_k$. Trivially, it follows that there exist m^{n+1} different types of patterns in the image. The type of a pattern is characterized by the (ordered) values of $\mathbf{z}_k$ and the value of the kth voxel itself. It should be stressed that the subindex k of $\mathcal{P}_k$, as well as of $\mathcal{N}_k$, represents the center voxel and thereby the location of the pattern, and it does not contrain any information on the type of the pattern.

Let p_i, for $i = 1, \dots, m^{n+1}$, count the number of times a pattern of type i appears in the image. These counts are used to represent the frequency distribution of an image. After having scanned through all inner voxels exactly once (the order is irrelevant) the frequency distribution is given by the vector $\mathbf{p}$:

$$\mathbf{p} = [p_1 \quad \cdots \quad p_{m^{n+1}}] = p_\Omega(z_1, \dots, z_N).$$

Here p_Ω is the function that, given an image and a template function Ω, computes the frequency distribution of the image with respect to the template as just described.

We notice that, for a given template, the frequency distribution of an image is uniquely determined. The opposite, however, does not hold. Different images can have the same frequency distribution. This is exactly what we seek to exploit by using the frequency distribution to generate multiple new images, at the same time similar to, and different from, our training image.

3 The Frequency Matching Method

The Frequency Matching Method proceeds by iteratively updating voxel values of an image, until the frequency of patterns in the image matches the frequency patterns in the training image. One of the primary tasks when formulating the method is to define a similarity function for how close the frequency distributions of two images are. Below we shall define the similarity function used in the current implementation, and describe the optimization method we have applied to solve the combinatorial optimization problem arising from this.

3.1 The Similarity Function

The similarity function plays the following two important roles:

I. It allows us to determine if the frequency distribution of an image and the frequency distribution of a training image are identical within the accuracy required and we therefore consider the image a valid realization of the random process from which the training image is a realized.

II. Given two different images, no matter how similar they might be, and a training image, the similarity function should determine which of the two images is most similar to a valid realization of the same process as the training image, or if the two images are equally similar. At the same time it should reflect (in some sense) how close the images are to being a valid realization.

Using an iterative solution method, point I is used to determine if the method has converged to an acceptable solution, whereas point II guides the method through the solution space, helping it to converge.

As we do not know the random process of which the training image is a realization, we have chosen the chi-square measure of 'goodness of fit' between two sets of nominal data as a similarity function for our FMM implementation. This measure determines the distance between to frequency distributions by comparing the proportions of types of pattern in the two.

3.2 Applying the χ^2 Measure in the FMM

The chi-square measure can be applied to our situation using the following interpretations (see Bere and Chimedza, 2007):

samples Each frequency distribution is considered a sample, i.e., we have two independent samples; one for the image itself and one for the training image.

categories The samples are categorized with respect to the m^{n+1} exclusive and exhaustive types of patterns.

observations Each appearance or count of a pattern is an observation. For each sample, the number of observations equals the number of inner voxels in the corresponding image.

Given the frequency distributions of an image, $\mathbf{p}$, and of a training image, $\boldsymbol{\pi}$, we can compute what we denote to be the similarity function value of the image:

$$f(\mathbf{p}) = \chi^2(\mathbf{p},\boldsymbol{\pi}) = \sum_{i=1}^{m^{n+1}} \frac{(p_i - e_i)^2}{e_i} + \sum_{i=1}^{m^{n+1}} \frac{(\pi_i - \varepsilon_i)^2}{\varepsilon_i},$$

where e_i and ε_i denote the expected count of patterns of type i of the image and the training image, respectively. These are computed as:

$$e_i = \frac{(p_i + \pi_i)}{n_p + n_\pi} n_p,$$

$$\epsilon_i = \frac{(p_i + \pi_i)}{n_p + n_\pi} n_\pi,$$

and n_p and n_π are the number of inner voxels in the image and the training image, respectively.

Let χ^2 denote the chi-square value of the image computed from the two frequency distributions $\mathbf{p}$ and $\boldsymbol{\pi}$. f is a function of the frequency distribution $\mathbf{p}$ of the image, and the frequency distribution $\boldsymbol{\pi}$ of the training image. The training image and therefore its frequency distribution will remain unchanged when computing a new image; $\boldsymbol{\pi}$ has therefore been omitted as an argument of the similarity function. Furthermore, the frequency distribution $\mathbf{p}$ of the image is derived given a template function, i.e., the argument $\mathbf{p}$ of f depends on a template as well as on $z_1, \dots, z_N$, which means f is in fact a function of the image and a template function. However, to simplify the text, we have chosen to avoid these dependencies in the notation.

3.3 The Optimization Problem

The function f defined in section 3.2 seems to fulfill the two requirements we had, making the FMM a combinatorial optimization problem. The variables are the voxel values of the image. They can take on m different integer values namely $\{0, \dots, m-1\}$. Binary images, for instance, have $m = 2$. Given a template function Ω, the frequency distributions of the solution image, $\mathbf{p}$, and of a training image, $\boldsymbol{\pi}$, are computed by the frequency function p_Ω. Based on the two frequency distributions the similarity function of the image is computed. By minimizing the similarity function with respect to certain constraints, we can create images sharing the same multiple-point statistics as the training image. The resulting optimization problem can be expressed as follows:

$$\begin{aligned} \min_{z_1,\dots,z_N} \quad & f(\mathbf{p}) \\ \text{w.r.t.} \quad & \mathbf{p} = p_\Omega(z_1, \dots, z_N), \\ & z_k \in \{0, \dots, m-1\}, \quad \text{for } k = 1, \dots, N. \end{aligned}$$

If some of the voxel values are known beforehand, and the voxels are therefore not free variables, the last set of constraints can easily be altered, such that the set of values that the kth voxel can take is only a subset of $\{\mathbf{0}, \dots, \boldsymbol{m-1}\}$.

3. Example

We have now introduced the Frequency Matching Method for generating a priori sample models from training images, and this has led us to a combinatorial optimization problem. Our choice of solution method is, for now, the intuitively simple heuristic Simulated Annealing (SA) (e.g. Kirkpatrick et al., 1983). For future work we would also like to explore other solution methods in the hope of finding one better suited for optimization and sampling problems.

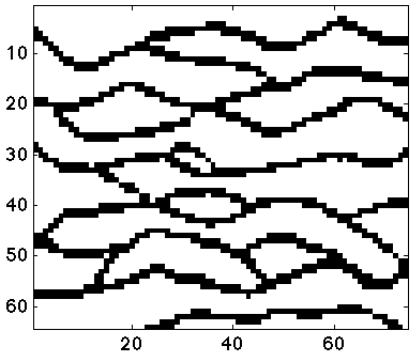

Figure 1: Training image.

The FMM has been implemented in MATLAB. To demonstrate the FMM we will consider a two-dimensional, binary training image with channel structures, see Figure 1.

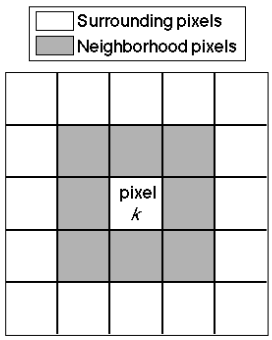

Figure 2: The template

We have defined the template such that the neighborhood of an arbitrary inner pixel k contains exactly the eight nearest pixels, see Figure 2. This relatively small neighborhood size is unlikely to completely satisfy our assumption of a pixel only being conditioned upon the pixels in its neighborhood. However, it will be shown that the method is still able to compute an acceptable solution. Due to the complexity of the method the size of the neighborhood greatly influences the running times, and for using much bigger templates we recommend implementing the method in Fortran, for instance.

We choose the exponential cooling rate for the SA, and the algorithm parameters are chosen manually. Discussing the strategies for choosing these optimally is beyond the scope of this text.

The starting image for SA is chosen to be all white. The SA algorithm searches the solution space consisting of images, and it moves from one image to another by randomly choosing a pixel and changing its value. Figure 3 and Figure 4 show the normalized frequency distributions of the training image and the image computed by the FMM, respectively. By 'normalized' we mean relative to the number of inner pixels in each of the images. Any normalized frequency distribution therefore sums to 1. Here we have truncated the ordinates of Figure 3 and Figure 4, as only one entry is significantly bigger than 0.08. The last entry is approximately 0.42 for both images. This entry is the one representing a white center pixel surrounded by all white neighboring pixels.

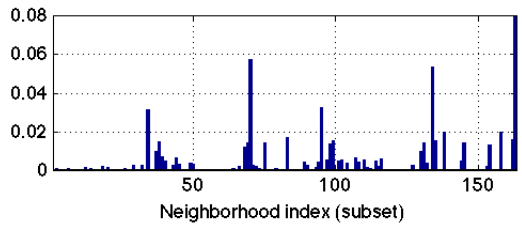

Figure 3: Normalized frequency distribution of the training image.

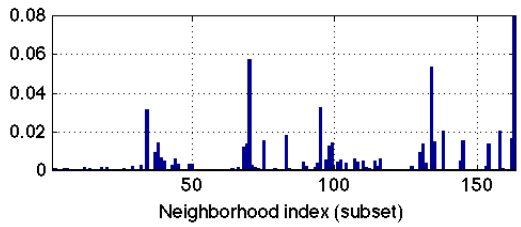

Figure 4: Normalized frequency distribution of the optimal solution image.

Notice that in Figure 3 and Figure 4 indexes corresponding to types of patterns appearing in neither the training image nor the solution image have been omitted

We observe that the FMM in terms of the frequency distributions has managed to match the training image quite well. Summing the bars of Figure 5 reveals that the two images have approximately 96.7% of their patterns in common. This number is likely to be improved by changing the parameters of the SA algorithm.

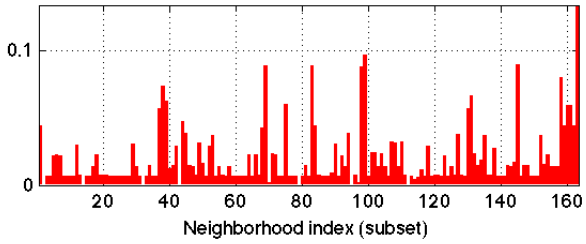

Figure 5: The absolute difference (in percent) between the normalized frequency distributions in Figure 3 and Figure 4.

Keep in mind that matching the frequency distributions only results in a useful image if our assumptions are met; i.e., if we chose a suitable template. Choosing too big a template means very long running times without sufficient gain in accuracy, and choosing too small a template will result in the picture not being similar to the training image. Our choice seems sufficient although not perfect, see Figure 6.

Figure 6 shows the image computed by the FMM. For this test case we have chosen to compute a 60×60 image based on a 64×74 training image but the method can produce images of arbitrary size. We notice that despite the relatively small template size, we have successfully recreated the channel structures. The channels even occasionally form loops, just like the channels of the training image.

One significant difference between the computed image and the training image is that the channels in the computed image are not all horizontally continuous across the image. We expect that this is merely a matter of choice of template and also the number of iterations the algorithm has been allowed to perform.

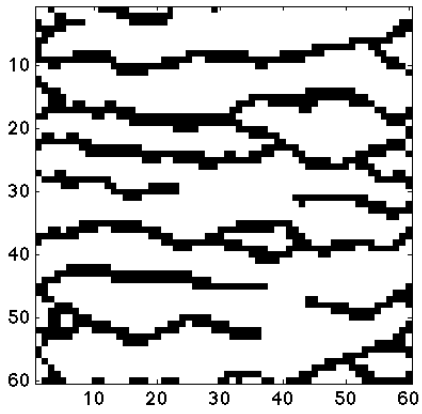

Figure 6: The computed solution image.

Another difference is the boundaries. It seems the method creates some artifacts along the boundaries. The density of channels is much higher on the left and right boundary then in the middle of the image. In the middle it resembles our training image and we therefore could have some issues in the way we treat non-inner pixels.

Notice how matching the frequency distributions indirectly results in the proportion of channels versus background in the computed picture to be in correspondence with the proportion of channels versus background in the training image. As stated, this is merely an example of the performance of the FMM. The method has also been applied to training images with different structures and shown similar results.

4. Conclusions and Future Work

In this paper we have derived the Frequency Matching Method for generation of a priori sample models from training images. We have implemented the method in MATLAB and shown the results of a simplified test case. The test example shows that the method is indeed able to produce an image that shares the same multi-point statistics as the training image.

This paper only scratches the surface of this newly developed method. In order to better understand its potential we would like to:

- Experiment thoroughly with training images with different structures.
- Investigate the convergence rate and performance of the FMM combined with other optimization methods.
- Explain and eventually avoid possible artifacts for non-inner voxels.

References

BERE, A., CHIMEDZA, C. (2007): A Comparative Study of the Accuracy of the Chi-Squared Approximation for the Power-Divergence Statistic and Pearson's Chi-Square Statistic in Sparse Contingency Tables. Journal of Statistical Research, Vol. 41, No. 2, pp. 73-81.

KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P. (1983): Optimization by Simulated Annealing. Science, New Series, Vol. 220, No. 4598, pp. 671-680, May.

STREBELLE, S. (2002): Conditional Simulation of Complex Geological Structures Using Multiple-Point Statistics. Mathematical Geology, Vol. 34, No. 1, January.