A BAUM-WELCH ALGORITHM FOR NOISY VECTOR FIELDS FOR CLASSIFICATION AND SYNTHESIS OF TEXTURES USING NON-SYMMETRIC HALF-PLANE

Paul M. Baggenstoss

Naval Undersea Warfare Center
Newport RI, 02841, USA
phone: (+001) 401-832-8240
email: p.m.baggenstoss@ieee.org web: www.npt.nuwc.navy.mil/csf
This work was supported by Office of Naval Research ONR 321 US

ABSTRACT

In this paper we present a statistical model with a non-symmetric half-plane (NSHP) region of support for two-dimensional continuous-valued vector fields. It has the simplicity, efficiency, and ease of use of the well-known hidden Markov model (HMM) and associated Baum-Welch algorithms for time-series and other one-dimensional problems. At the same time, it is able to learn textures on a two-dimensional field. We describe a fast approximate forward procedure for computation of the joint probability density function (PDF) of the vector field as well as an approximate Baum-Welch algorithm for parameter re-estimation. We test the method using synthetic textures.

1. INTRODUCTION

1.1 Goal

Let $s_{i,j}$ be a discrete random variable taking values between 1 and M on a two-dimensional field $1 \le i \le N_1$, $1 \le j \le N_2$. Assume that we cannot observe $s_{i,j}$ directly but instead observe a D-dimensional continuous valued random vector "pixel" $\mathbf{x}_{i,j} \in \mathscr{R}^D$ whose probability density function (PDF) depends on the state at pixel i,j and is denoted by $p(\mathbf{x}_{i,j}|s_{i,j})$. We would like to (1) model the statistical behavior of the discrete random field (2) estimate the PDFs $p(\mathbf{x}|s)$ and (3) determine the joint PDF of the entire field $\mathbf{X} = \{x_{i,j} : 1 \le i \le N_1,\ 1 \le j \le N_2$ given the model parameters.

1.2 Previous Work

Real-world patterns and textures exhibit complex statistical dependencies between neighboring pixels. In order to model these dependencies in a tractable way, we often assert the *Markov property*. This property holds that the state of the system at a given pixel conditioned on the entire field or part of the field can be expressed in terms of only the states of the of neighboring pixels. The choice of the *neighborhood* has a profound affect on the properties of the model [1] [2], [6]. We can classify the models into two subclasses: *causal* and *non-causal*.

In non-causal models, the conditional statistical dependence extends in all directions and the neghborhood of a pixel surrounds the pixel itself. Although this is probably a better representation of real-world processes, the statistical analysis of such fields requires the mathematics of Gibbs fields and its associated limitations. Limitations include (1) the necessity of using iterative methods to generate synthetic fields from the model, (2) the inability to find a closed-form expression for the constant of proportionality of the probability distributions - a necessity for classification, and (3) reliance on energy "clique" functions which are non-intuitive and indirect representations of the statistical dependencies.

In causal models, the neighborhood system is one-sided with statistical dependence expressed only in terms of "past" data. This allows computation of the joint probability density function (PDF) of the entire field of vectors in a recursive manner. Additional advantages are the reliance on direct intuitive conditional dependencies and the ability to synthesize random fields in one pass. Among causal models the Markov Mesh is a very tractable model that expresses dependencies in terms of just two neighboring pixels [4], [5]. The simplicity of the Markov mesh models, however, may limit the usefulness of the model [1], [6]. The non-symmetric half-plane (NSHP) region of support (ROS) consists of four neighboring pixels and provides the most general causal model [6].

The discussion up to now has assumed that $s_{i,j}$ are directly observable. When statistically modeling $\mathbf{X}$, another layer of complexity is added especially when trying to estimate model parameters. In the case of one-dimensional problems such as in speech recognition, the parameter estimation problem is handled very efficiently using the Baum-Welch algorithm [3]. Unfortunately, the *forward procedure* and *backward procedure*, components of the Baum-Welch algorithm which recursively and efficiently compute the joint PDF of the field of noisy measurments, do not generalize to the NSHP two-dimensional model.

In this paper, we demonstrate approximate equivalents to the forward procedure and backward procedure. The result is an approximate Baum-Welch algorithm for estimating the parameters of the NSHP Markov model as well as the observation PDFs.

1.3 Non-symmetric half-plane (NSHP) Model.

The NSHP neighborhood system has been used for two-dimensional autoregressive analysis and image and texture analysis [7],[8], [6]. The NSHP model is based on ordering the pixels in an image by scanning in a raster pattern (see figure 1). As we scan the plane from left to right, and bottom to top, when we reach pixel "A", we have already visited "B", "C", "D", and "E". Because each of the pixels in the ROS are in the "past", we can construct a recursive *forward procedure* that computes state probabilities based on "past" pixels.

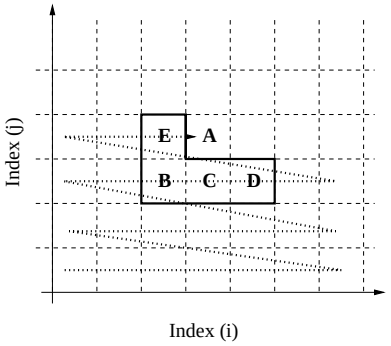

Figure 1: Illustration of the neighbohood system of the non-symmetric half plane model (NSHP). Pixel A is represented statistically in terms of neighbors B,C,D, and E.

2. MATHEMATICAL ALGORITHM DESCRIPTION

2.1 Notation and mathematical definition of the NSHP model.

Let $\mathbf{x}_{i,j} \in \mathscr{R}^D$ be the dimension D data (feature vector) at pixel i,j where $1 \le i \le N_1$, $1 \le j \le N_2$. We define the "past data" $\mathbf{X}_{<i,j>}$ as all data occurring "before" $\mathbf{x}_{i,j}$ that is either to the left or below and not including $\mathbf{x}_{i,j}$. More precisely,

$$< i,j >= \{i',j'\} \ : \ (i' < i \ \text{AND} \ j' = j) \ \text{OR} \ (j' < j).$$

We also define the "past and current data" $\mathbf{X}_{i,j}$ as the union of $\mathbf{X}_{<i,j>}$ and $\mathbf{x}_{i,j}$. We define $p_{i,j}(m|\mathbf{X}_{<i,j>})$ as the *a priori* probability of state m at pixel i,j given all data up to but not including $\mathbf{x}_{i,j}$. Similarly, we define $p_{i,j}(m|\mathbf{X}_{i,j})$ as the *a posteriori* probability of state m at pixel i,j given all data up to and including $\mathbf{x}_{i,j}$. Let the index variables r,s,t,u index over the states of the neighbor pixels ("B", "C", "D", and "E", respectively in figure 1). Let M be the number of possible states.

2.2 NSHP forward procedure.

The forward procedure in HMM terminology [3] recursively computes the state probabilities at each pixel. It also computes the joint probability of all the data given the model, which is needed in a classifier. We can recursively compute the state probabilities as follows. In the first step of the forward procedure we compute the *a priori* probability

$$\begin{aligned} p_{i,j}(m|\mathbf{X}_{<i,j>}) &= \sum_{r=1}^{M}\sum_{s=1}^{M}\sum_{t=1}^{M}\sum_{u=1}^{M} p_{i,j}(m,r,s,t,u|\mathbf{X}_{<i,j>}) \\ &\simeq \sum_{r=1}^{M}\sum_{s=1}^{M}\sum_{t=1}^{M}\sum_{u=1}^{M} p_{i-1,j-1}(r|\mathbf{X}_{i-1,j-1}) \\ &\quad p_{i,j-1}(s|\mathbf{X}_{i,j-1})\ p_{i+1,j-1}(t|\mathbf{X}_{i+1,j-1}) \\ &\quad \cdot p_{i-1,j}(u|\mathbf{X}_{i-1,j})\ \mathbf{C}_{r,s,t,u}(m), \end{aligned} \tag{1}$$

where $\mathbf{C}_{r,s,t,u}(m)$ is the $M^4 \times M$ state transition matrix. The expression that replaces $p_{i,j}(m,r,s,t,u|\mathbf{X}_{<i,j>})$ uses only quantities that have been previously computed. It is approximate because (a) the data conditions used in the terms on the right hand side (RHS) are not equivalent to the data condition $\mathbf{X}_{<i,j>}$ used on the left hand side (LHS) and (b) the joint probability $p_{i,j}(m,r,s,t,u|\mathbf{X}_{<i,j>})$ is approximated by the product of the marginals. Although the RHS expression is approximate, it (a) depends only upon data that is strictly contained in the condition $\mathbf{X}_{<i,j>}$ and (b) is a probability on (m,r,s,t,u). Thus, the result is still a valid PDF and produces in the end a valid PDF on $\mathbf{X}$. This is an important observation because the maximization of the resulting approximate data field PDF over the model parameters would not be a valid approach if this were not true.

The *a posteriori* probability is denoted by $\alpha_m^{i,j}$ and is computed by updating (1) using the data at pixel i,j

$$\alpha_m^{i,j} = p_{i,j}(m|\mathbf{X}_{i,j}) = \frac{p_{i,j}(m|\mathbf{X}_{<i,j>})p(\mathbf{x}_{i,j}|m)}{\sum_{n=1}^{M} p_{i,j}(n|\mathbf{X}_{<i,j>})p(\mathbf{x}_{i,j}|n)}, \tag{2}$$

where $p(\mathbf{x}|m), \quad m = 1 \ldots M$, are the state PDFs represented as Gaussian mixtures [3].

The joint PDF of the data up to and including $\mathbf{x}_{i,j}$ can be recursively computed in parallel:

$$\begin{aligned} p(\mathbf{X}_{i,j}) &= p(\mathbf{X}_{i-1,j})\ p(\mathbf{x}_{i,j}|\mathbf{X}_{i-1,j}) \\ &= p(\mathbf{X}_{i-1,j})\ p(\mathbf{x}_{i,j}|\mathbf{X}_{<i,j>}) \\ &= p(\mathbf{X}_{i-1,j}) \sum_{m=1}^{M} p(\mathbf{x}_{i,j}|m)\ p_{i,j}(m|\mathbf{X}_{<i,j>}). \end{aligned} \tag{3}$$

When the algorithm reaches the last pixel ($i = N_1,\ j = N_2$), the joint PDF of the entire field is given by

$$L(\mathbf{X}) = \log p(\mathbf{X}_{N_1,N_2}).$$

2.2.1 *Initialization*

The forward procedure (1), (2), (3) assumes that the $\alpha_m^{i,j}$ at all four neigbors have already been calculated. To initialize the algorithm, we begin with the first pixel ($i = 1, j = 1$). For this, we need the *a priori* state probabilities $\pi(m)$, $1 \le m \le M$. Then, we can progress through the first row of pixels using the well-known HMM forward procedure. For this we need the 1-D state transition matrix (STM) $A_u(m) = \Pr(s_t = m|s_{t-1} = u)$. When we reach the first pixel of the second row, we need a reduced version of $\mathbf{C}_{r,s,t,u}(m)$ since there is no pixel immediately to the left (pixel "E" in figure 1). There is also no pixel underneath *and* to the left pixel "B" in figure 1), however if we choose, we can wrap data around to that the last pixel on the first row takes the place of pixel "B". Using this approach of wrapping around pixel "B", the reduced STM is $M^3 - by - M$ and is denoted by $\mathbf{B}_{r,s,t}(m)$. The initialization is a straight-forward simplification of (1), (2), (3) made by removing the index u and associated factors.

2.2.2 *Synthesis of Random fields.*

Just as the forward procedure is analogous to the forward procedure of the HMM, the synthesis of discrete random fields is directly analogous to the Markov chain. To synthesize a random field from the model, we first synthesize a discrete random field, then as a second step, synthesize feature values from the state PDFs given the chosen states at each pixel. State synthesis is an algorithm directly parallel to the forward procedure: we use the STM to determine the

state probabilities of the M states at a given pixel given the already-chosen states of the "past" neighbors, select the discrete random variable, then continue to the next pixel.

2.3 NSHP backward procedure.

Only the forward procedure is needed to compute total field probability given the model. The time-reversed version, called the *backward procedure* [3], is needed as a component of the *Baum Welch* algorithm to re-estimate the model parameters [3]. Unfortunately, it is impractical to create a NSHP version of the backward procedure that follows the spirit of the one-dimensional version because the NSHP neighborhood system is not a simply ordered Markov model. The state transition matrix, $\mathbf{C}_{r,s,t,u}(m)$ in eq. 1), is of dimension M^4-by-M. To implement the backward procedure, we would require the backward version of $\mathbf{C}_{r,s,t,u}(m)$, denoted by $\mathbf{C}^b_{r,s,t,u}(m)$, which must also be of the same dimensions. In the one-dimensional backward procedure, it is simply the transpose of the forward STM. The backward procedure uses backward versions of the reduced STMs for initialization analogous to the forward procedure (denoted by $\pi^b(m)$, $A^b_u(m)$, $\mathbf{B}^b_{r,s,t}(m)$). For the NSHP backward procedure, we have two choices, resulting in two different algorithms:

1. **NSHP-Symmetric algorithm**: Assume that the problem is 180-degree symmetric and use $\mathbf{C}^b_{r,s,t,u}(m) = \mathbf{C}_{r,s,t,u}(m)$. This can be used for textures that are symmetric (in the sense of rotating them by 180 degrees).
2. **NSHP-Non-symmetric algorithm**: Estimate $\mathbf{C}^b_{r,s,t,u}(m)$ separately. This adds to the number of parameters that must be estimated but allows applicability to non-symmetric textures.

Let $\mathbf{Y}_{i,j}$ and $\mathbf{Y}_{<i,j>}$ represent *future data* with respect to $\mathbf{x}_{i,j}$, inclusive and exclusive of $\mathbf{x}_{i,j}$, respectively. The NSHP backward procedure, analogous to (2) is given by

$$\begin{aligned} \beta^{i,j}_m &= p_{i,j}(m|\mathbf{Y}_{<i,j>}) \\ &\simeq \sum_{r=1}^{M}\sum_{s=1}^{M}\sum_{t=1}^{M}\sum_{u=1}^{M} p_{i+1,j+1}(r|\mathbf{Y}_{i+1,j+1}) \\ &\quad \cdot p_{i,j+1}(s|\mathbf{Y}_{i,j+1})\; p_{i-1,j+1}(t|\mathbf{Y}_{i-1,j+1}) \\ &\quad \cdot p_{i+1,j}(u|\mathbf{Y}_{i+1,j})\; \mathbf{C}^b_{r,s,t,u}(m). \end{aligned} \tag{4}$$

To continue the recursion, we also need the data-updated probabilities of $\beta^{i,j}_m$, denoted by $\delta^{i,j}_m$:

$$\delta^{i,j}_m \triangleq p_{i,j}(m|\mathbf{Y}_{i,j}) = \beta^{i,j}_m\; p(\mathbf{x}_{i,j}|m).$$

Equation (4) can then be rewritten

$$\begin{aligned} \beta^{i,j}_m &= \sum_{r=1}^{M}\sum_{s=1}^{M}\sum_{t=1}^{M}\sum_{u=1}^{M} \delta^{i+1,j+1}_r\; \delta^{i,j+1}_s\; \delta^{i-1,j+1}_t \\ &\quad \delta^{i+1,j}_u\; \mathbf{C}^b_{r,s,t,u}(m). \end{aligned} \tag{5}$$

Initialization is analogous to the forward procedure.

2.4 NSHP Baum-Welch algorithm.

We now describe an algorithm that parallels the well-known Baum-Welch algorithm for the HMM [3]. The state probabilities given all the data are denoted by γ in HMM terminology [3]. Let

$$\gamma^{i,j}_m = p_{i,j}(m|\mathbf{X}),$$

where $\mathbf{X}$ is all the data. We have

$$\gamma^{i,j}_m \simeq \frac{\alpha^{i,j}_m\; \beta^{i,j}_m}{\sum_k \alpha^{i,j}_k\; \beta^{i,j}_k}.$$

Note that denominator is just the numerator summed over m, so computing $\gamma^{i,j}_m$ can be done in two steps: (a) multiplying $\alpha^{i,j}_m$ and $\beta^{i,j}_m$ together, then (b) normalizing so that $\gamma^{i,j}_m$ sums to 1 over m.

We can use γ to detect likely instances of certain states in the data. We can also use it to re-estimate state transition probabilities $\mathbf{C}_{r,s,t,u}(m)$ as well as the state likelihood functions $p(\mathbf{x}_{i,j}|m)$, which are represented by Gaussian mixtures. We can approximate the *Baum Welch* algorithm to re-estimate $\mathbf{C}_{r,s,t,u}(m)$ using

$$\begin{aligned} \hat{\mathbf{C}}_{r,s,t,u}(m) &= \textstyle\sum_i \sum_j\; \gamma^{i,j}_m\; \gamma^{i-1,j-1}_r\; \gamma^{i,j-1}_s \\ &\quad \cdot\; \gamma^{i-1,j-1}_t\; \gamma^{i-1,j}_u. \end{aligned} \tag{6}$$

followed by a normalization operation to insure that

$$\sum_m \hat{\mathbf{C}}_{r,s,t,u}(m) = 1.$$

The same formula, applied on the 180-degree rotated data field can be used to estimate $\mathbf{C}^b_{r,s,t,u}(m)$ if for the non-symmetric option. Re-estimation of state PDFs, which can be represented by Gaussian mixtures, follows the 1-D approach [3].

3. EXPERIMENTAL RESULTS

3.1 Synthetic Data

We created a 180-degree symmetric synthetic data class from a notional blue ellipse filled with green on a red background. Examples of the data are shown in figure 2. The data are three-dimensional (RGB) feature vector with additive Gaussian noise. The ellipse is randomly positioned in the plane. We also created a non-symmetric synthetic texture data class from repeating triangles in a green field (figure 3).

3.2 NSHP-Symmetric Model

We trained the NSHP-symmetric model with $M = 3$ discrete states on 30 samples of the symmetric texture from figure 2. The state PDFs were represented by a single Gaussian (mixture with 1 component) and was initialized randomly in thirty-two random trials. The algorithm was halted if the log-likelihood began to decrease. The best random trial was used. A representation of the γ probabilities for one data sample is shown in figure 4. This can be compared with the data in figure 2. The NSHP model easily generates random fields as shown by the example in figure 5. The random fields can be compared with figure 4. Clearly the NSHP model has succeeded in learning the synthetic pattern. When we trained the

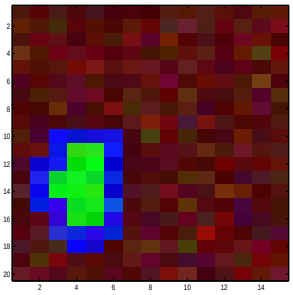

Figure 2: One example of the symmetric "ellipse" synthetic data field. Rotating the image by 180 degrees produces an equally valid sample. Data is a three-dimensional feature vector rendered as an RGB image.

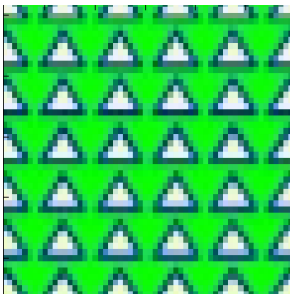

Figure 3: One example of the non-symmetric synthetic data field. Rotating the image by 180 degrees does not produce a valid sample. Data is a three-dimensional feature vector rendered as an RGB image.

symmetric model on the non-symmetric data and synthesized random fields, we obtained figure 6. As we would expect, we see character in the synthetic pattern of triangles, but there is no apparent up or down.

3.3 NSHP Non-Symmetric Model

When we trained the non-symmetric model on the non-symmetric texture in figure 3 and synthesized random fields, we obtained figure 7. Here, we see clearly the upward-pointing triangles.

3.4 Extended feature (EF) vector model.

In hidden Markov models, to which the algorithm we have described belongs, the statistical relationship between a pixel and its neighbors is based on discrete states only. If the number of states is small, we do not have the benefit of accurately predicting the color of a given pixel from the color of its neighbors except through their discrete states. This can limit the usefulness of the models. From a strict application of Bayes theorem to develop a joint PDF of the noisy vector field, there is no reason we cannot also use the feature values in the neighboring pixels to assist in the prediction. To this end, we create at each pixel an extended feature vector $\mathbf{x}^e_{i,j} \in \mathscr{R}^{5D}$ which is of dimension $5D$ because it includes the feature vectors of the current pixel plus the four neighbors.

It must be kept in mind that by using extended feature vectors, we are in effect creating a new, higher-dimension

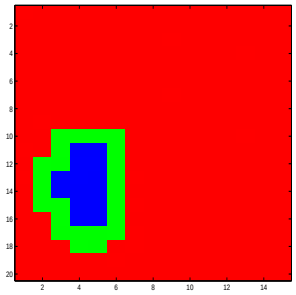

Figure 4: Gamma probabilities for the example in figure 2. The image is coded with a color representing the most likely state as determined by $\gamma^{i,j}_m$. Red: state 1, green: state 2, blue: state 3.

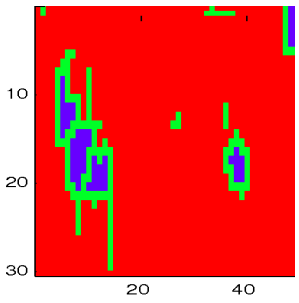

Figure 5: Synthetic random field generated by the NSHP symmetric model trained on the data in figure 2.

random field. Since each feature vector from the original field is replicated 5 times, the new higher-dimensional field is redundant. Nevertheless, the forward procedure just described provided a PDF value for the field of extended vectors. When we synthesize random fields from the non-symmetric model on extended feature vectors, and obtained figure 8, there is the appearance of better texture synthesis than figure 7.

3.5 Conditional-Extended feature (CEF) vector model.

As we explained, the forward procedure operating on the extended feature vectors produces an exact PDF of the field of

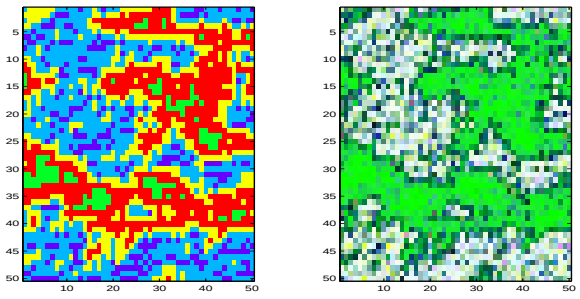

Figure 6: Synthetic random field generated by symmetric NSHP model trained on the data in figure 3. Left: states, Right: color image. There is no apparent up or down evident in the synthetic field.

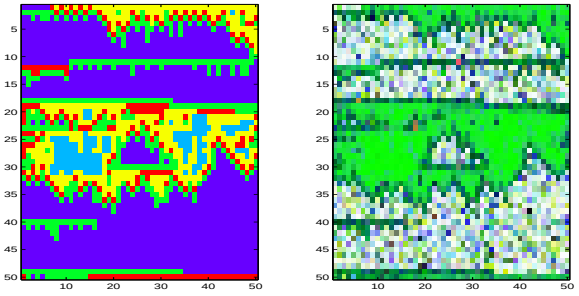

Figure 7: Synthetic random field generated by non-symmetric NSHP model trained on the data in figure 3. Left: states, Right: color image. In contrast to figure 6, there is a clear notion of up and down evident in the synthetic field.

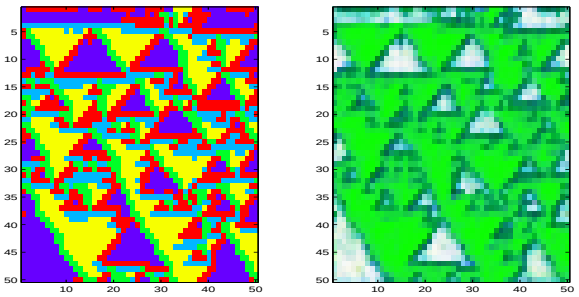

Figure 8: Synthetic random field generated by non-symmetric NSHP model with extended feature (EF) vectors trained on the data in figure 3. Left: states, Right: color image.

extended feature vectors. Exact is meant in the sense that for a fixed set of parameters, it integrates to exactly 1. This is useful in classification of textures provided all texture models being compared use the same EF model. However, if we need the PDF of the original field, we must use the conditional PDFs. Let

$$\mathbf{z}_{i,j} = \begin{bmatrix} \mathbf{x}_{i,j} \\ \mathbf{w}_{i,j} \end{bmatrix},$$

where $\mathbf{x}_{i,j}$ is the feature vector at pixel i,j and $\mathbf{w}_{i,j}$ is the $4D$-by-1 vector of features from the four NSHP neighbors. In the forward procedure, we use in place of the state PDFs the conditional PDF $p(\mathbf{x}_{i,j}|m,\mathbf{w}_{i,j})$. The necessary conditional PDFs can be generated easily from Gaussian mixtures in closed form, even with more than one mixture component. We synthesize random fields from the CEF model and obtained figure 9. We see a degradation wih respect to figure 8.

4. CONCLUSIONS

We have presented a statistical model for two-dimensional fields of pixels where each pixel is represented by a continuous-valued feature of arbitrary dimension. The model is directly analogous to the one-dimensional hidden Markov model (HMM). We have described an fast approximate forward procedure that recursively computes the joint PDF of the entire data field as well as an algorithm for re-estimation of model parameters modeled after the well-known Baum-

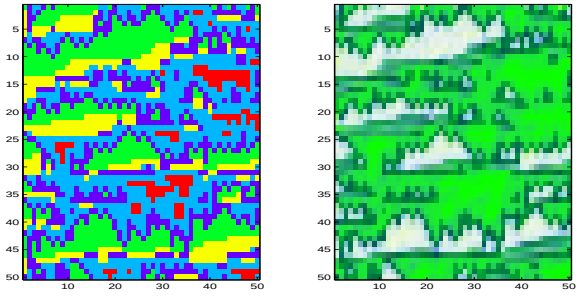

Figure 9: Synthetic random field generated by non-symmetric NSHP model with conditional extended feature (CEF) vectors trained on the data in figure 3. Left: states, Right: color image.

Welch algorithm for the one-dimensional HMM. We have also described variations that utilize extended feature vectors to permit effective feature prediction that would not otherwise be possible with doubly-stochastic models. We have also demonstrated the algorithm's ability to learn textures. Based on texture synthesis, the extended feature vector (EF) model appears to be superior. We found this also to be the case in texture classification experiments which space limitations do not allow us to describe.

REFERENCES

[1] A. J. Gray, J. W. Kay, and D. M. Titterington, "An empirical study of the simulation of various models used for images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 507–513, 1994.

[2] W. Qian and D. M. Titterington, "Multidimensional markov chain models for image textures," *J. Roy. Statist. Soc. B*, vol. 53, pp. 661–674, 1991.

[3] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, February 1989.

[4] D. DeMenthon, M. Vuilleumier, and D. Doermann, "Hidden markov models for images," 2000.

[5] D. DeMenthon, M. Stuckelberg, and D. Doermann, "Image distance using hidden markov models," 2000.

[6] F.C.Jeng and J.W.Woods, "On the relationship of the markov mesh to the nshp markov chain.," *Pattern Recognition Letters*, vol. 5, pp. 273–279, 1987.

[7] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[8] M. Eckstrom and J. Woods, "Two-dimensional spectral factorization with applications in recursive digital filtering," *IEEE Trans. ASSP*, vol. 24, pp. 115–128, April 1976.