Non-negative Sparse Modeling of Textures

Gabriel Peyré

Non-negative Sparse Modeling of Textures

Gabriel Peyré

Ceremade, Université Paris Dauphine,
Place du Marchal De Lattre De Tassigny,
75775 Paris Cedex 16 France
gabriel.peyre@ceremade.dauphine.fr,
http://www.ceremade.dauphine.fr/~peyre/

Abstract. This paper presents a statistical model for textures that uses a non-negative decomposition on a set of local atoms learned from an exemplar. This model is described by the variances and kurtosis of the marginals of the decomposition of patches in the learned dictionary. A fast sampling algorithm allows to draw a typical image from this model. The resulting texture synthesis captures the geometric features of the original exemplar. To speed up synthesis and generate structures of various sizes, a multi-scale process is used. Applications to texture synthesis, image inpainting and texture segmentation are presented.

1 Statistical Models for Texture Synthesis

The characterization of textures is a central topic in computer vision and graphics, mainly approached from a probabilistic point of view.

Spatial domain modeling. The works of both Efros and Leung [1] and Wei and Levoy [2] pioneered a whole area of greedy approaches to texture synthesis. These methods copy pixels one by one, enforcing locally the consistence of the synthesized image with the exemplar. Recent approaches such as the method of Lefebvre and Hoppe [3] are fast, multiscale and give impressive results.

Transformed domain modeling. Julesz [4] stated simple axioms about the probabilistic characterization of textures. A texture is described as a realization of a random process characterized by the marginals of responses to a set of linear filters. Zhu, Wu and Mumford [5] setup a Gibbs energy to learn both the filters and the marginals. They use a Gibbs sampler to draw textures from this model.

A fast synthesis can be obtained by fixing the analyzing filters to be steerable wavelets as done by Heeger and Bergen [6]. The resulting textures are similar to those obtained by Perlin [7]. They exhibit isotropic cloud-like structures and fail to reproduce long range anisotropic features. This is because wavelets decompositions represent sparsely point wise singularities but do not compress enough long edge features. Higher order statistics such as local correlations are used by Portilla and Simoncelli [8] to synthesize high quality textures.

Sparse image decompositions. Representing a complex image with few meaningful elements is at the core of the visual processing made by the human cortex. Atteneave [9] and Barlow [10] first stated that efficient high level computations should be performed over a representation of reduced complexity.

This biological processing suggests a sparse description of a patches $y \in \mathbb{R}^N$ of N pixels extracted from a natural image as

$$y[n] = \sum_{k=0}^{p-1} x^k \, d_k[n] \qquad \text{where} \qquad \|x\|_{\ell^0} \stackrel{\text{def.}}{=} \#\left\{k \setminus x^k \neq 0\right\} \leqslant \tau \ll N, \tag{1}$$

where $x = [x^0, \ldots, x^{p-1}]$ are the coefficients of the decomposition, and where typically $p \geqslant N$. This simple linear model uses as prior a dictionary $D = [d_0, \ldots, d_{p-1}] \in \mathbb{R}^{N \times p}$ of atoms.

This generative process is appealing as it involves p degrees of freedom to perform statistical modeling while generating a large amount of different images. Classical works in computational harmonic analysis describe the set of candidate images y as belonging to some functional space and study the sparsity of the decomposition of y in some fixed basis or dictionary. Fourier decomposition is only suitable for smooth images and a wavelet decomposition [11] allows to model sparsely data with pointwise singularities. Images with geometrical singularities such as edges should be analyzed with more involved constructions such as the frame of curvelets [12] or a bandelet best basis [13].

These tools however are not efficient to capture the complex geometry of textures, which might include turbulent structures or complex overlapping junctions. To sparsely represent these structures, we use a exemplar-based approach where the dictionary D is learned from a single input texture.

Learning the dictionary. Given a set of m typical patches $Y = [y_0, \ldots, y_{m-1}] \in \mathbb{R}^{N \times m}$, one needs to compute both the dictionary D and the coefficients $X = [x_0, \ldots, x_{m-1}]$ of the decomposition $Y \approx DX$ that leads to the sparsity $\|x_j\|_{\ell^0} \leqslant \tau$ of each column x_j of X required by equation (1).

Olshausen and Field [14] first proposed a learning scheme to build a dictionary of atoms to represent each signal of a data set using few elements. For natural images, edge filters emerge to efficiently represent the geometry of images. Algorithms have been proposed in signal processing such as the K-SVD of Aharon et al [15] and the tight frames construction of Tropp et al. [16].

ICA and sparse dictionaries have been applied in texture modeling mainly for features extraction in classification [17,18]. An ICA decomposition is used as a post-processing step by Manduchi and Portilla [19] to enhance the synthesis results of Heeger and Bergen multiscale approach [6].

An alternative way to perform sparse coding is to enforce positivity of both the coefficients X and the dictionary D. Non-negative factorization, as proposed by Lee and Seung [20] tends to to decompose an image into its meaningful parts, see the theoretical study of Donoho and Stodden [21].

Contributions. In this paper, we propose a signal-processing approach to the sparse modeling of textures. It is based on the following ingredients:

- Only marginal responses of a decomposition are used. It provides a simple modeling of textures ensembles in the transformed domain.
- An additive generative model based on positive atoms is used. Although our method works with traditional linear decompositions, positive atoms are more localized and decompose the exemplar texture into it constitutive elements.
- A fast signal-processing based method is used to learn the set of atoms adapted to a given texture. This is motivated by the sparse description of textures and does not require complex solvers such as the one of [5].
- A fast iterative scheme is used to sample a typical texture that matches the same marginal statistics. It does not ensure a sampling with maximum entropy distribution [5] but initializing the iterations with a random noise is good enough in typical applications, as noticed in [6,8].

2 Non-negative Atoms for Texture Decomposition

Non-negative matrix factorization. The process of learning a dictionary D of atoms $D = [d_0, \dots, d_{p-1}]$ to represent accurately a set $Y = [y_0, \dots, y_{m-1}]$ of exemplars is equivalent to performing a factorization $Y = DX$. This problem is underconstrained and the non-negative factorization [20] enforces positivity of both X and D to constrained the learning problem.

The problem of learning D and X requires to minimize the reconstruction error $\|Y - DX\|$ subject to the constraints $D, X \succeq 0$. This minimization is not convex on both D and X, but following [22], an alternate minimization on each matrix can be carried using iterations of the following two steps

$$X_{ab} \leftarrow X_{ab} \frac{\sum_i D_{ia} Y_{ib} / (DX)_{ib}}{\sum_i D_{ia}} \qquad \text{and} \qquad D_{ia} \leftarrow D_{ia} \frac{\sum_b X_{ab} Y_{ib} / (DX)_{ib}}{\sum_b X_{ab}},$$

which converge to a local minimum of $\|Y - DX\|$. To enforce the dictionary elements y_i to be of unit norm, the columns of D are normalized at each iteration. An explicit sparsity prior can be added into this process [23] to enforce the constraints $\|x_j\|_{\ell^0} \leqslant \tau$ required by equation (1). In practice, this did not result in a noticeable enhancement for texture modeling.

Patch-based decomposition. Our sparse modeling of textures is based on the assumption of local ergodicity common to many previous works, see for instance [8]. It assumes that the texture is a realization of a random vector whose statistics are invariant under translation.

Starting from an exemplar texture $f_e \in \mathbb{R}^N$ of N pixels given by the user, one works at a fixed scale $w > 0$ that sets the size of the typical structures present in the texture. Section 4 describes how a multiscale procedure can alleviate the issue of using a fixed scale. If n is a pixel in the image, we denote by $p_n = p_n(f_e)$ the patch of $w \times w$ pixels centered at n in f_e.

The ergodicity assumption leads us to model each patch $p_n(f_e)$ as being sparsely represented in some dictionary D. In practice, a set of m square patches $Y = [y_0, \ldots, y_{m-1}]$ of $N = w \times w$ pixels is extracted at random from the exemplar f_e. The non negative factorization $Y = DX$ is used to learn the dictionary from the exemplar. Figure 1 shows some examples of atoms d_j of D learned from a texture.

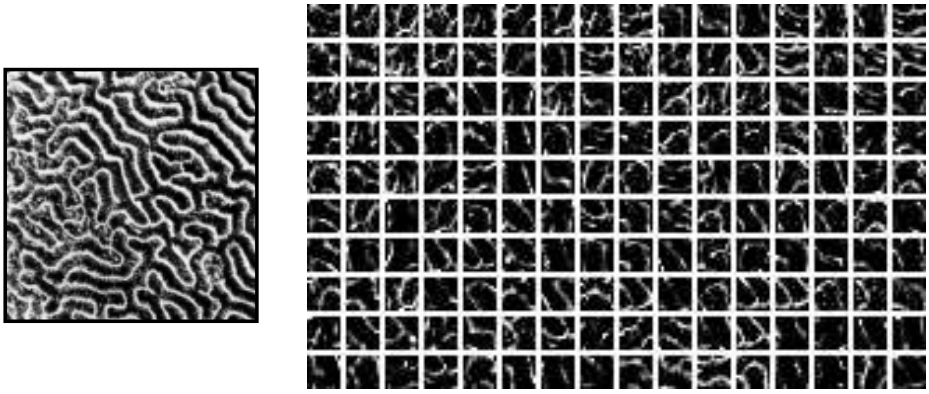

Fig. 1. *Example of dictionary for patches of* 12×12 *pixels. We used* $m = 20w^2$ *random patches for the learning stage.*

3 Parametric Modeling Over a Learned Dictionary

Once a dictionary D is learned to efficiently represent the patches Y of size $w \times w$ extracted from f_e, a statistical model is built using the marginal of the decomposition coefficients X such that $Y = DX$. Ergodicity allows to use the sets of coefficients $x^k = \{X_{k,j}\}_{j=0}^{m-1}$ to estimate the marginals of the underlying probabilistic model that generates the patches of f_e.

The marginal distributions are both on-sided and highly concentrated near 0. We thus keep track of the empirical variance and kurtosis of the decomposition of Y onto the dictionary D defined by

$$\sigma^k(Y) \stackrel{\text{def.}}{=} \sigma(x^k) \stackrel{\text{def.}}{=} M_2(x^k), \quad \text{and} \quad \kappa^k(Y) \stackrel{\text{def.}}{=} \kappa(x^k) \stackrel{\text{def.}}{=} \frac{M_4(x^k)}{(M_2(x^k))^2}, \tag{2}$$

$$\text{where} \qquad M_s(x^k) \stackrel{\text{def.}}{=} \frac{1}{N} \sum_{j=1}^{m} (x^k[j])^s. \tag{3}$$

A texture f_e is characterized, at a scale w, by
– its adapted dictionary D,

– the empirical variance $\sigma^k(Y)$ and kurtosis $\kappa^k(Y)$.
Both are computed from the decomposition of a large enough set of patches Y extracted from f_e.

4 Sampling from the Positive Texture Model

Texture ensembles. A dictionary D learned from the exemplar defines an equivalence relationship $\overset{D}{\sim}$ between two sets of patches $P = \{p_n\}_n$ and $Q = \{q_n\}_n$ that shares the same statistics

$$P \overset{D}{\sim} Q \qquad \Longleftrightarrow \qquad \forall k, \qquad \sigma^k(P) = \sigma^k(Q) \quad \text{and} \quad \kappa^k(P) = \kappa^k(Q).$$

Note that these statistics are defined over a transformed domain, which means that one first has to factor the matrix $P = [p_0, p_1, \ldots]$ as $P = DX$ and then extract the statistics of the rows x^k, as explained in the previous section.

The set of images f of N_1 pixels whose local decompositions in D share the same marginal statistics as f_e defines an ideal texture ensemble

$$\widetilde{\mathcal{T}}(f_e) \overset{\text{def.}}{=} \left\{ f \setminus \{p_n(f)\}_{n\in\mathcal{P}} \overset{D}{\sim} \{p_n(f_e)\}_{n\in\mathcal{P}} \right\} \subset \mathbb{R}^{N_1}.$$

where $\mathcal{P}$ denotes the set of pixels. Note that although f_e contains N pixels, the texture ensemble can be defined for any size N_1. We use a model based on overlapping patches to have a translation invariant description that avoids blocking artifact.

Imposing simultaneously all the statistical constraints that define $\mathcal{T}(f_e)$ is a complex non-linear process due to the overlapping between patches $p_n(f)$. We approximate this texture ensemble using the following decomposition

$$\widetilde{\mathcal{T}}(f_e) \approx \mathcal{T}(f_e) \overset{\text{def.}}{=} \bigcap_{\delta\in\Delta} \mathcal{T}_\delta(f_e) \qquad \text{where} \qquad \Delta \overset{\text{def.}}{=} \{0, \ldots, w-1\}^2. \tag{4}$$

Each $\mathcal{T}_\delta(f_e)$ imposes constraints on a sub-set of non-overlapping patches

$$\mathcal{T}_\delta(f_e) \overset{\text{def.}}{=} \left\{ f \setminus \{p_n(f)\}_{n\in\mathcal{P}_\delta} \overset{D}{\sim} \{p_n(f_e)\}_{n\in\mathcal{P}_\delta} \right\}$$

where $\mathcal{P}_\delta$ with $\delta = (\delta_1, \delta_2)$ is the sub-lattice of pixels n defined by

$$\mathcal{P}_\delta = \{n = (n_1, n_2) \setminus n_1 \% w = \delta_1 \quad \text{and} \quad n_2 \% w = \delta_2\},$$

where $\%$ is the modulo operator.

Projection on the texture ensemble. The approximation of the texture ensemble of equation (4) describes $\mathcal{T}(f_e)$ as the intersection of convex sets $\mathcal{T}_\delta(f_e)$. Following [8], an image f can be approximately projected onto $\mathcal{T}(f_e)$ by iterating projections onto each $\mathcal{T}_\delta(f_e)$. Since each set $\mathcal{T}_\delta(f_e)$ involves constraints on independent patches of f, this projection can be carried by local ajustements on the decomposition of each patch $p_n(f)$ for $n \in \mathcal{P}_\delta$.

We denote by $\pi_\delta(f)$ the approximate projection of f onto $\mathcal{T}_\delta(f_e)$, which is computed using the following steps

- The set of patches $P = \{p_n(f)\}_{n \in \mathcal{P}_\delta}$ are gathered from f.
- The positive factorisation $P = DX$ is performed using the algorithm described in section 2. We now adjust the statistics of each vector of coefficients $x^k = \{X_{k,j}\}_j$ representing the decomposition on a single atom d_k of D.
- The projection on variance constraints is performed by $\widetilde{x}^k \leftarrow x^k \sigma^k(f_e)/\sigma(x^k)$.
- As done in [8], enforcing the kurtosis is performed using a gradient descent of the potential $\widetilde{x}^k \mapsto |\kappa(\widetilde{x}^k) - \kappa^k(f_e)|^2$ while keeping $\sigma(\widetilde{x}^k)$ constant.
- The updated patches $\widetilde{P}$ are reconstructed using the dictionary $\widetilde{P} = D\widetilde{X}$ where the rows of $\widetilde{X}$ are the new coefficients $\widetilde{x}^k$ for $k = 0, \ldots, p-1$. The projection $\pi_\delta(f)$ is computed by rearranging the non-overlapping patches of $\widetilde{P}$.

Sampling from the texture ensemble. The set $\mathcal{T}(f_e)$ is compact and the uniform distribution on $\mathcal{T}(f_e)$ thus defines the probability measure with maximum entropy. Zhu et al. [5] sample this distribution in order to synthesize textures without bias.

Rather than performing an exact sampling of the uniform distribution on $\mathcal{T}(f_e)$, we follow [8] and use a sampling strategy that finds a point in $\mathcal{T}(f_e)$ by iterating projections on each of set $\mathcal{T}_\delta(f_e)$. Starting from an initial point with high entropy such as a gaussian noise ensures that the set of generated images span $\mathcal{T}(f_e)$ with a minimum bias. This leads to the following synthesis algorithm:

- Preprocessing: extract m random patches $Y = [y_0, \ldots, y_{m-1}]$ of width $w \times w$ from f_e. Compute the positive factorization $Y = DX$ and record the parameters $\sigma^k(f_e)$ and $\kappa^k(f_e)$ of the marginals defined in equation (2).
- Initialization: set f as a realization of a gaussian white noise on N_1 pixels.
- Repeat for random shift $\delta \in \{0, \ldots, w-1\}^2$ until convergence: $f \leftarrow \pi_\delta(f)$.

5 Texture Synthesis

Mono-scale synthesis. Starting from an exemplar f_e of N pixels, the mono-scale texture synthesis process consists in computing an image $f \in \mathcal{T}(f_e)$ of N_1 pixel. This sampling is carried out using the iterative projection method exposed in the previous section.

Note that this method generates textures of arbitrary size. Furthermore, we use cyclic boundary conditions when extracting patches $\{p_n(f)\}_{n \in \mathcal{P}_\delta}$ from the output texture, which results in periodic textures that tile the plane. Figure 2 shows two exemples of synthesis. The short range structures are well synthesized, but the algorithm fails to capture long range fiber-like structures.

Color texture are synthesized by applying the algorithm on each channel independently. Moving from the RGB color representation to the HSV representation improves synthesis quality since the intensity channel tends to have more distinct structures than the remaining channels. Figure 2 compares our method with the method of [6] and [8]. Both the input texture f_e and output f are of size 256×256 pixels. The multiscale histogram matching of Heeger and Bergen [6]

is not able to reproduce textures with geometric features. In contrast, both the higher order model of Portaill and Simoncelli [8] and our method can synthesize textures with complex structures.

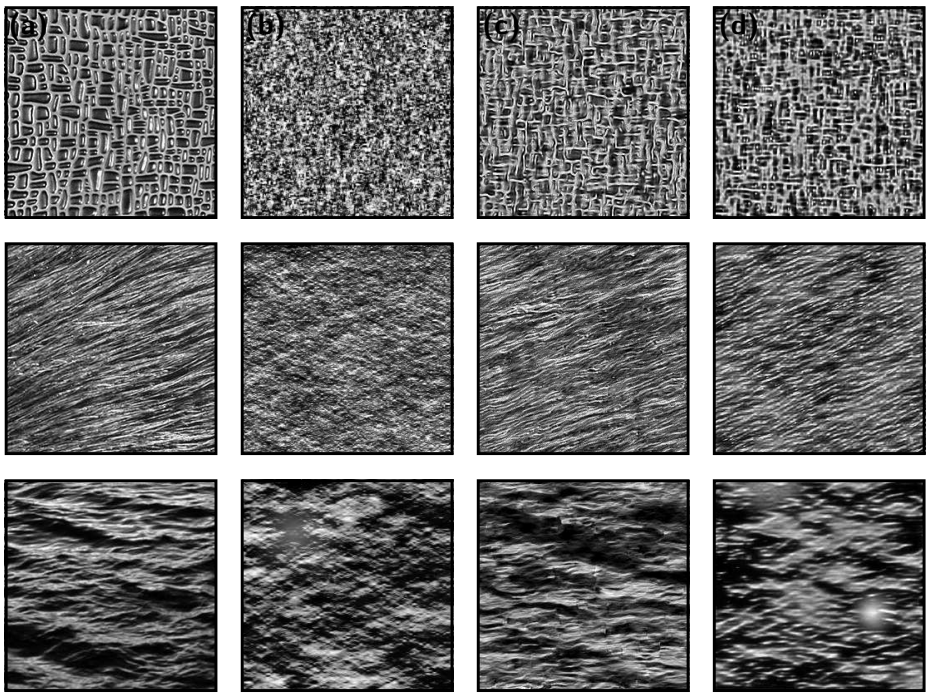

Fig. 2. *(a) Original texture. (b) Textures synthesized with the method of Heeger and Bergen [6] (c) Textures synthesized with the method of Portilla and Simoncelli [8]. (d) Textures synthesized with our method.*

Multiscale synthesis. In order to cope with the fixed scale w used in previous section, one can use a multiscale synthesis strategy. We use a fixed number of pixels w_0 but consider textures with increasing resolutions. This allows to capture first elongated low frequencies structures and then fine scale geometric details. A simple interpolation is used to switch between the various resolutions. At each scale, the synthesis algorithm manipulates only small patches of size $w_0 \times w_0$. This leads to the following algorithm that handles J scales.

- Initialization: Set $j = J$ to be the coarser scale. Initialize the synthesis with a random noise f of $N_1/2^J \times N_1/2^J$ pixels.
- Step 1: Set $w = 2^j w_0$. Smooth the exemplar $f_e^j = f_e * h_j$ where h_j is a gaussian kernel of width 2^j pixels. Extract a set of m patches Y_j from f_e^j. Sub-sample these squares by a factor 2^j so that vectors in Y_j are of size $w_0 \times w_0$.

– Step 2: Perform the mono-scale synthesis algorithm using patches Y_j to train the dictionary and with the current f as initialization.
– If $j = 0$ then stop the algorithm. Otherwise, upsample the current synthesized texture using linear interpolation from $N/2^j \times N/2^j$ pixels to $2N/2^j \times 2N/2^j$ pixels. Set $j \to j-1$ and go back to step 1.

In our implementation, we have used 2 scales $j = 0, 1$ and a base width $w_0 = 8$. Figure 3 compares the fixed scale synthesis and the multiscale synthesis which is able to creates elongated singularities. Figure 4 shows additional synthesis results.

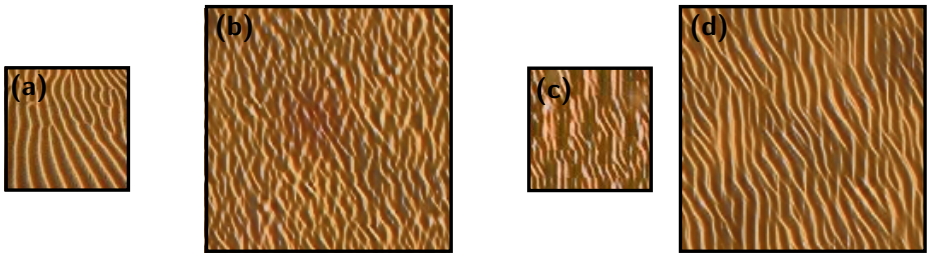

Fig. 3. *(a) Original texture f_e. (b) Texture synthesized with the mono-scale procedure with patches of width $w = 8$. (c) Texture synthesized at scale $2^j = 2$ with $w = 16$. (d) Texture synthesized at scale $2^j = 1$ with $w = 8$.*

6 Texture Inpainting

The inpainting problem consists in filling a set of missing pixels Ω in a given image f_e. This problem has been approached using evolution equation derived from fluid dynamics by Bertalmio et al. [24] however diffusion-based approaches fail to reproduce texture patterns. Using a sparsity prior in a set of fixed bases such as Curvelets and local DCT, Fadili and Starck [25] are able to inpaint oscillatory and elongated texture structures.

Our synthesis algorithm can be slightly modified to cope with missing data as follow.

– Extract a set of m patches Y from f_e that are as close as possible from Ω without intersecting it.
– Set as initial inpainted image f the original f_e with values at random inside Ω.
– Step 1: for a random shift δ, perform one step of synthesis $f \leftarrow \pi_\delta(f)$. The projection needs only to be performed for patches $p_n(f) \cap \Omega \neq \emptyset$.
– Step 2: impose the known values, $\forall n \notin \Omega, \quad f[n] \leftarrow f_e[n]$. Go back to step 1.

Figure 5 shows some step of this inpainting process and figure 6 shows additional results. A limitation of this method is that it works well for homogeneous textures

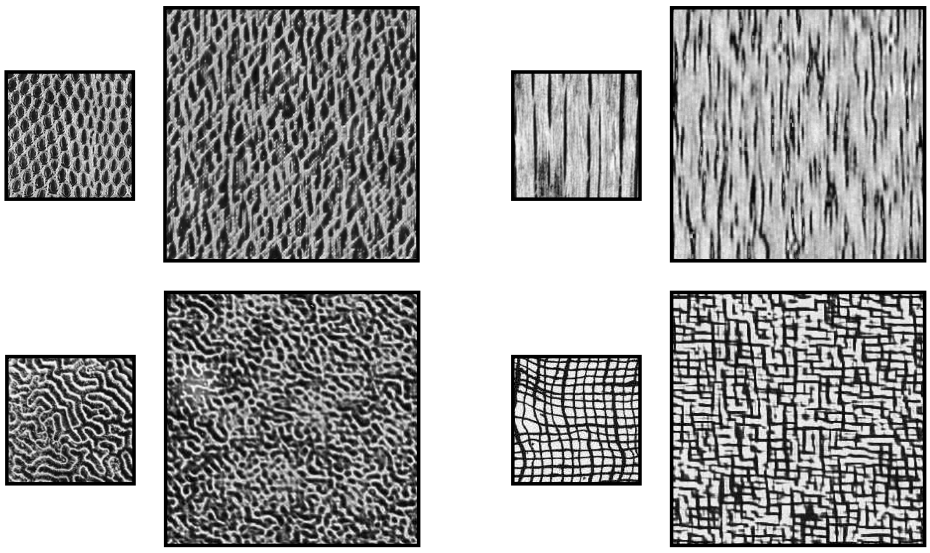

Fig. 4. *Examples of multiscale texture synthesis.*

and the inpainting tends to give poor results if Ω intersects a broad range of structures.

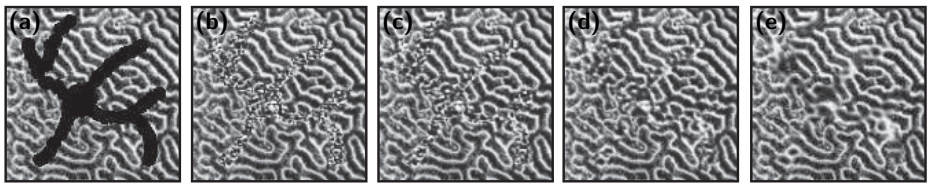

Fig. 5. *(a) Texture to inpaint, the missing region Ω is depicted in black. (b,c,d) Evolution of the inpainting for step 1,2,4. (e) Final result.*

Texture segmentation. Our model can be used to perform segmentation of a given texture f into components corresponding to patterns similar to exemplars $\{f_e^1, \ldots, f_e^s\}$. Learned dictionaries have already been used for segmentation [18], and we recast this approach into our patch-based non-negative model. The idea is to project the texture f onto each texture ensemble $\mathcal{T}(f_e^\ell)$ and select locally the class ℓ that generates the least deviation from f. This leads to the following algorithm.

- Learn the statistical model $\mathcal{T}(f_e^\ell)$ for each class ℓ.
- Compute the projection f_ℓ of f on each ensemble $\mathcal{T}(f_e^\ell)$ using the algorithm of section 4.

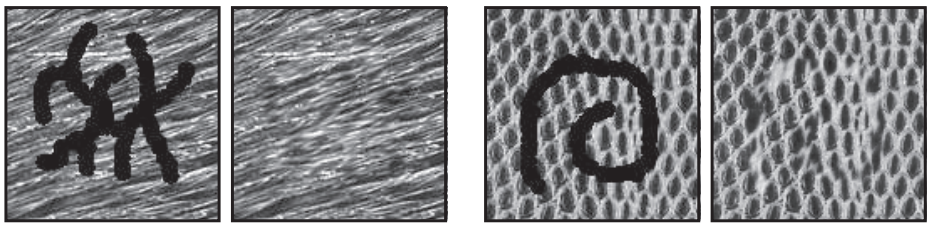

Fig. 6. *Examples of inpainting.*

– Compute the class-wise error for each pixel and smooth it with a gaussian kernel G_{s_0} of with s_0

$$\widetilde{E}_\ell[n] \overset{\text{def.}}{=} |f[n] - f_\ell[n]|^2 \qquad \text{and} \qquad E_\ell = \widetilde{E}_\ell * G_{s_0}.$$

The smoothing removes estimation noise and reflects the prior knowledge that class boundary should be smooth curves.

– Compute the segmentation into classes using

$$\ell[n] = \operatorname*{argmin}_\ell E_\ell[n].$$

We have tested this segmentation using a set of $s = 5$ exemplar textures. The input image f of 256×256 pixels is a patchwork of five textures extracted from the upper left corner of the original images $\widetilde{f}_e^\ell$ of 512×512 pixels. The exemplars f_e^ℓ are extracted from the lower right corner of $\widetilde{f}_e^\ell$. Figure 7 shows the segmentation process.

7 Conclusion

We have proposed a statistical model for textures built out of marginal distributions of a positive decomposition. The statistical model is parametric since we use only low order moments of the distribution. This is permitted thanks to the sparsity provided by a learned dictionary. Such a positive dictionary captures with few atoms the structures of the textures. This simple model allows to perform multiscale texture synthesis and can be fitted into various applications such as texture inpainting or texture segmentation.

An important parameter of our model is the redundancy factor p/N. Redundancy brings invariances to various factors such as translations or local illumination changes. These parameters are however hard to control. Representations that can explicitly capture this invariances include bilinear decompositions [26] that could improve our model.

References

1. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2, IEEE Computer Society (1999) 1033

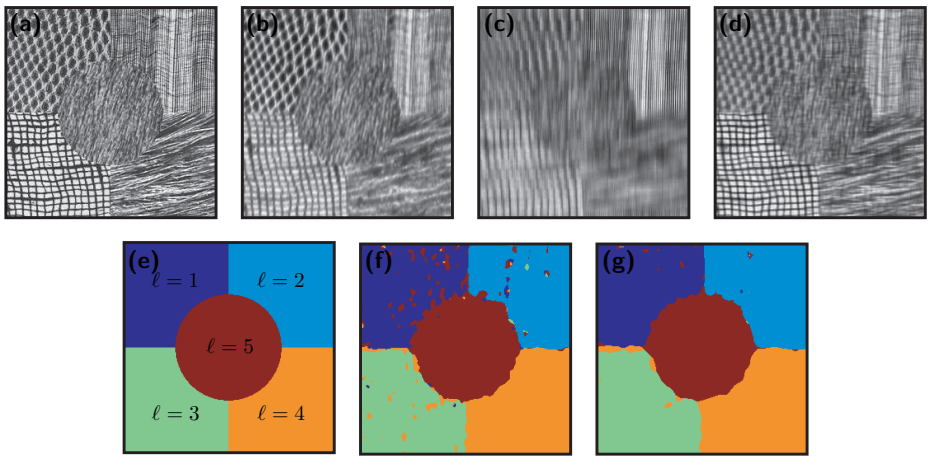

Fig. 7. *(a) Original texture f. (b) Projected texture f_1 of f onto $\mathcal{T}(f_e^1)$. Note how the upper left corner is well preserved. (c) Projected texture f_2. (d) Projected texture f_3. (e) Ground trust segmentation. (f) Segmentation $\ell[n]$ computed with $s_0 = 3$ pixels. (g) Segmentation computed with $s_0 = 6$ pixels.*

2. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (2000) 479–488
3. Lefebvre, S., Hoppe, H.: Parallel controllable texture synthesis. ACM Trans. Graph. **24**(3) (2005) 777–786
4. Julesz, B.: Visual pattern discrimination. IRE Trans. Inform. Theory **8**(2) (1962) 84–92
5. Zhu, S.C., Wu, Y., Mumford, D.: Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. Int. J. Comput. Vision **27**(2) (1998) 107–126
6. Heeger, D.J., Bergen, J.R.: Pyramid-Based texture analysis/synthesis. In Cook, R., ed.: SIGGRAPH 95 Conference Proceedings. Annual Conference Series, ACM SIGGRAPH, Addison Wesley (1995) 229–238
7. Perlin, K.: An image synthesizer. In: SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press (1985) 287–296
8. Portilla, J., Simoncelli, E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. Int. J. Comput. Vision **40**(1) (2000) 49–70
9. Attneave, F.: Some informational aspects of visual perception. Psychological Review **61** (1954) 183–193
10. Barlow, H.B.: Possible principles underlying the transformation of sensory messages. In Rosenblith, W.A., ed.: Sensory Communication, MIT Press (1961) 217–234
11. Mallat, S.: A Wavelet Tour of Signal Processing. Academic Press, San Diego (1998)
12. Candès, E., Donoho, D.: New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities. Comm. Pure Appl. Math. **57**(2) (2004) 219–266

13. Le Pennec, E., Mallat, S.: Bandelet Image Approximation and Compression. SIAM Multiscale Modeling and Simulation **4**(3) (2005) 992–1039
14. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive-field properties by learning a sparse code for natural images. Nature **381**(6583) (1996) 607–609
15. Aharon, M., Elad, M., Bruckstein, A.: The k-svd: An algorithm for designing of overcomplete dictionaries for sparse representation. IEEE Trans. On Signal Processing (to appear) (2006)
16. Tropp, J., Dhillon, I., Heath, R., Strohmer, T.: Designing structured tight frames via an alternating projection method. IEEE Trans. on Information Theory **51**(1) (2005) 188–209
17. Zeng, X.Y., Chen, Y.W., van Alphen, D., Nakao, Z.: Selection of ica features for texture classification. In: ISNN (2). (2005) 262–267
18. Skretting, K., Husoy, J.: Texture classification using sparse frame based representations. EURASIP Journal on Applied Signal Processing, to appear (2006)
19. Manduchi, R., Portilla, J.: Independent component analysis of textures. In: ICCV. (1999) 1054–1060
20. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401** (1999) 788–791
21. Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into parts? In Thrun, S., Saul, L., Schölkopf, B., eds.: Advances in Neural Information Processing Systems 16, Cambridge, MA, MIT Press (2004)
22. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems 13, Cambridge, MA, MIT Press (2001)
23. Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. Journal of Machine Learning Research **5** (2004) 1457–1469
24. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Siggraph 2000. (2000) 417–424
25. Fadili, M., Starck, J.L.: Em algorithm for sparse representation-based image inpainting. In: IEEE International Conference on Image Processing, Vol. II. (2005) 61–63
26. Grimes, D.B., Rao, R.P.N.: Bilinear sparse coding for invariant vision. Neural Computation **17**(1) (2005) 47–73