Texture Synthesis with Grouplets

Gabriel Peyré

Texture Synthesis with Grouplets

Gabriel Peyré, *Member, IEEE*

***Abstract*—This paper proposes a new method to synthesize and inpaint geometric textures. The texture model is composed of a geometric layer that drives the computation of a new grouplet transform. The geometry is an orientation flow that follows the patterns of the texture to analyze or synthesize. The grouplet transform extends the original construction of Mallat [1] and is adapted to the modeling of natural textures. Each grouplet atoms is an elongated stroke located along the geometric flow. These atoms exhibit a wide range of lengths and widths, which is important to match the variety of structures present in natural images. Statistical modeling and sparsity optimization over these grouplet coefficients enable the synthesis of texture patterns along the flow. This article explores texture inpainting and texture synthesis, which both require the joint optimization of the geometric flow and the grouplet coefficients.**

***Index Terms*—Texture, grouplets, texture synthesis, inpainting.**

NATURAL images often contain regions composed of locally oriented structures, such as those depicted in figure 1. These anisotropic textures are said to be locally parallel since they are composed of approximately parallel oscillations that propagate over the image plane. These oriented texture patterns provide fundamental features for many problems in computer vision and image processing and are known to be important cues for human vision [2].

Fig. 1. *Examples of locally parallel textures.*

Adaptive methods capture locally parallel patterns by first estimating the geometric flow of the texture and then assembling elongated strokes along this flow. The grouplet framework presented in this paper achieves such a geometric representation of textures and can be applied to various texture processing tasks such as inpainting and synthesis.

Gabriel Peyré is with the CNRS and Ceremade, Université Paris-Dauphine, 75775 Paris Cedex 16 France, email: gabriel.peyre@ceremade.dauphine.fr

I. Introduction

A. Previous Works

Anisotropic texture processing. The analysis of the local geometry of textures is studied extensively in computer vision through the computation of local differential estimators [3], [4], [5], [6]. These local descriptors are integrated into a texture flow that can be used to perform texture recognition [7], [8], [9]. Section II uses such a local orientation descriptor to build the geometric layer of the grouplet representation.

Geometric decompositions. Texture models based on non-adaptive wavelet decompositions fail to represent in a compact manner geometric singularities. Geometrical decompositions improve over the wavelets for the approximation of edges and textures in images. Local oscillating atoms such as Gabor [10], steerable wavelets [11], brushlets [12], curvelets [13] or wave-atoms [14] better capture the directionality of textures. These fixed transforms are however not adaptive and are sub-optimal for texture processing. Adaptive representations such as bandlets [15], [16] and grouplets [1] are tuned for a specific image to process. This adaptivity is achieved by computing an optimized geometry that parameterizes the representation. Section III describes a new grouplet transform that computes the coefficient layer of the grouplet representation.

Image inpainting. Classical methods for inpainting use partial differential equations that propagate the information from the boundary of the missing region to its interior, see for instance [17], [18], [19]. Tensor diffusion makes use of a geometric layer to drive the diffusion, see [20]. Sparsity regularizes the inpainting using a coefficient layer in a redundant frame [21], [22]. A two layers sparse grouplet regularization defined in Section IV bridges the gap between geometrical diffusion and sparse regularization. An early description of this method appeared in [23].

Texture synthesis. Texture synthesis is performed by sampling a texture model that constrains the geometric patterns of the images. Fourier and fractal modeling of textures [24], [25] create textures that are suitable to model some natural phenomena. Multiscale models constrain the distribution of wavelet coefficients that characterizes point-wise singularities in textures. Retaining only marginals of wavelets coefficients generates textures without geometric patterns [26], [27]. Higher order statistical modeling of wavelets coefficients [28], [29], [30], [31] improves the visual quality of synthesis by capturing geometric singularities. The representation of local features in these methods is however implicit and thus hard to analyze or control.

Elongated patterns are difficult to synthesize because they require the generation of coherent geometric structures such as

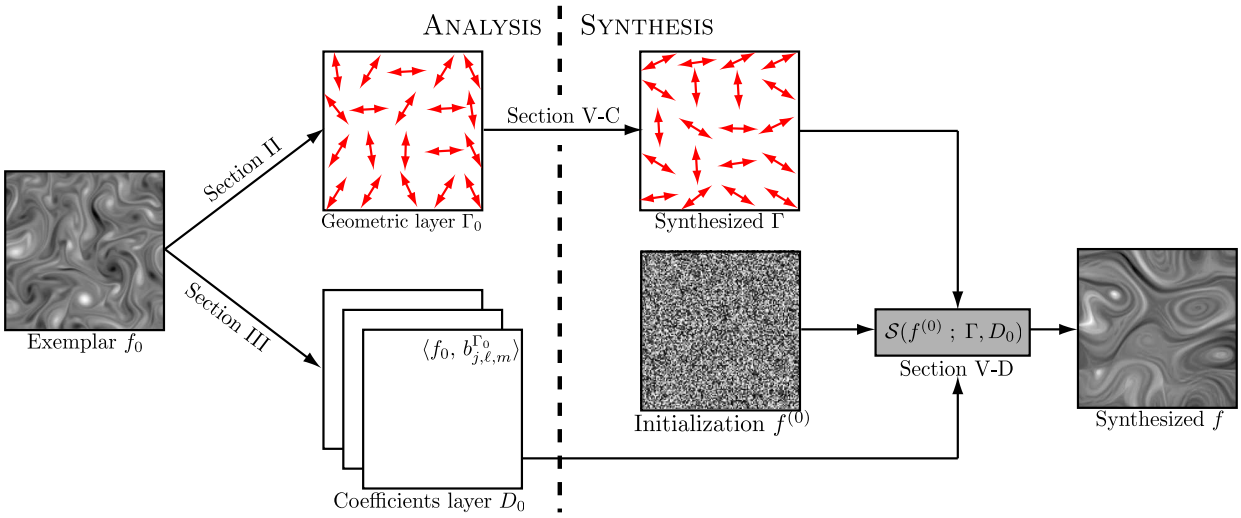


Fig. 2. *Overview of the grouplet texture analysis/synthesis pipeline. During the analysis stage, the image f_0 is decomposed into a geometry layer Γ_0 and an appearance layer, the latter being then decomposed into coefficients D_0 in an adapted grouplet frame $\mathcal{B}^{\mathrm{w}}(\Gamma_0)$. The synthesis stage computes a new geometry Γ and projects an initial image $f^{(0)}$ on constraints to match the statistics of the coefficient layer in the grouplet frame $\mathcal{B}^{\mathrm{w}}(\Gamma)$.*

stripes or vortices. Such a turbulent geometry can be synthesized by filtering noise using spatially varying oriented filters, see [2]. These anisotropic noise patterns can be generated by more advanced diffusion processes such as reaction-diffusion equations [32], [33], curvilinear smoothing [34], particules advection [35] or anisotropic diffusion [36].

Computer graphics methods generate new images of a given texture through careful and consistent copying of pixels from an example image [37], [38], [39]. The quality is further improved using patch recopy [40], [41] and by optimizing the stitching process using graph-cuts [42]. More advanced methods [43], [44], [45] extend this framework to synthesize images and normal maps on surfaces. Despite the high visual quality of the results, these approaches do not provide a parametric model for compact charactization of texture classes.

A method for hair analysis and synthesis is proposed by Chen and Zhu [46] that models oriented hair textures as a Markov random field parameterized by an hidden geometric layer. Orientation singularities is used as a hidden geometric layer to synthesize fingerprint images by Cappelli et al. [47]. Section V introduces a two layers model that makes use of statistics of the grouplet representation.

Dynamic Textures. Fluid texture synthesis generates oriented patterns that are warped along a coherent flow. Such a turbulent flow is animated by discretizing the Navier-Stokes equations of fluid motion [48], [49] or using texture synthesis of vector fields [50]. Dynamic textures are rendered by advecting particules [51], and is used to simulate physically plausible phenomena [52], [53]. Wavelet domain texture synthesis enhances the visual quality of turbulence simulation by adding fine scale details [54]. Dynamic textures can be generated by statistical modeling that captures the space and time homogeneity of natural dynamics [55], [42]. Patch recopy methods in computer graphics can be animated according to a vector field [44], [45] and can be mapped on a fluid in motion [56], [57]. Texture mixing animates an image by interpolating between two texture models [58]. Section VI extends the grouplet model to synthesize dynamic anisotropic textures and perform texture mixing.

B. Overview of the Grouplet Texture Model

The grouplet framework is a two layers model composed of a geometric layer Γ and a coefficient layer that stores the decomposition of the texture in a wavelet-grouplet tight frame $\mathcal{B}^{\mathrm{w}}(\Gamma)$. Figure 2 shows a graphical overview of the bluiding blocks of the grouplet pipeline. The wavelet-grouplet tight frame $\mathcal{B}^{\mathrm{w}}(\Gamma) = \{b^{\Gamma}_{j,\ell,m}\}_{j,\ell,m}$ is composed of elongated atoms. Each $b^{\Gamma}_{j,\ell,m}$ is an anisotropic stroke of width $\sim 2^j$ and length $\sim 2^\ell$, located around a pixel m, and that follows the flow Γ.

The estimation of the model from an exemplar f_0 is performed by computing an optimized flow Γ adapted to f_0, and the collection of coefficients $\{\langle f_0, b^{\Gamma}_{j,\ell,m}\rangle\}_{j,\ell,m}$ of f_0 in the wavelet-grouplet family $\mathcal{B}^{\mathrm{w}}(\Gamma)$. The application of the model is performed by either modifying both the geometry and the coefficient layers (for inpainting purpose) or by synthesizing new layers with statistical properties similar to the original one (for texture synthesis and mixing applications).

C. Contributions

The first contribution of this paper is a new grouplet processing framework. It extends the original grouplet transform of Mallat [1] and is well suited for the analysis of locally parallel textures. The two layers of this framework are the estimation of the geometrical flow described in Section II and the computation of the grouplet coefficients described in Section III. The second contribution is a new inpainting method detailed in Section IV that makes use of this grouplet transform. The resolution of such an inverse problem requires the optimization of both the geometric flow parameterizing the grouplet frame and the grouplet coefficients. The last contribution presented in Section V is a new texture model based on statistics of both the geometric flow and the bandlet coefficients. This model is used to perform realistic synthesis of static and dynamic turbulent textures.

II. Grouplet Geometric Layer Computation

The estimation of the geometric layer computes a local orientation field Γ that follows the patterns of a given input texture $f_0 \in \mathbb{R}^N$ of N pixels. This local orientation is not directional since geometric structures in a locally parallel texture do not have a specific direction. It means that at some pixel x, both $\Gamma(x)$ and $-\Gamma(x)$ are equally suitable to describe the texture.

A. Computation of the Local Orientation Field

The local direction of the edges in an image f_0 is captured by the vector $v(x)$ orthogonal to the gradient $\nabla_x f_0$. This vector is computed numerically on the discrete grid $\{0,\dots,n-1\}^2$ as the convolution against two directional derivative filters

$$v = (f_0 \star h^1, f_0 \star h^2)^\perp = (-f_0 \star h^2, f_0 \star h^1)$$

$$\text{where } h^i(x) = \frac{\partial G_{\mu_0}}{\partial x_i}(x),\ G_{\mu_0}(x) = \frac{1}{\mu\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\mu_0^2}\right)$$

where the scale μ_0 is of the order of one pixel.

While v is suitable to estimate the direction of step edges in images, it cannot be used directly to estimate the orientation of locally parallel textures such as those depicted on Figure 1. Indeed, the gradient vector vanishes at the top of ridges or at the bottom of valleys. This problem is alleviated by pooling locally the orientation information, which corresponds to averaging the orientation of $v(x)$ without its direction.

This local non-linear pooling corresponds to a local covariance analysis, summing the outer products of gradient vectors in a local region to generate a tensor field [3], [5], [6]. The structure tensor T_{f_0} is defined as a local averaging of the rank-1 tensor field vv^{T}

$$T_{f_0}(x) = (G_\mu \star (vv^{\mathrm{T}}))(x) = G_\mu \star \begin{pmatrix} v_1^2 & v_1 v_2 \\ v_1 v_2 & v_2^2 \end{pmatrix}(x), \quad (1)$$

where the convolution is applied on each component of the tensor. Figure 3 shows an example of tensor field T_{f_0} estimated from the gradient at different scales μ. This scale μ should match the width of the texture oscillations, and is set to 5 pixels in the numerical experiments.

Each symmetric tensor $T_{f_0}(x)$ is decomposed as a sum of two rank-1 tensors

$$T_{f_0}(x) = \lambda(x)(\Gamma(x)\Gamma(x)^{\mathrm{T}}) + \lambda^\perp(x)(\Gamma^\perp(x)\Gamma^\perp(x)^{\mathrm{T}}). \quad (2)$$

where the eigenvalues are $\lambda(x) \geqslant \lambda^\perp(x) \geqslant 0$ and $(\Gamma(x), \Gamma^\perp(x))$ are the corresponding orthonormal eigenvectors. The dominant eigenvector $\Gamma(x)$ indicates the local direction of the texture.

This orientation field Γ is unit normed $\|\Gamma(x)\| = 1$ and is not directional since $\Gamma(x)$ or $-\Gamma(x)$ can be used indifferently in the decomposition (2).

B. Computation of the Local Direction Field

The orientation field $\Gamma(x)$ is the geometric layer that describes the geometry of the texture f_0. A directional vector field $\tilde{\Gamma}$ is computed from this geometric descriptor Γ by optimizing a sign field ε

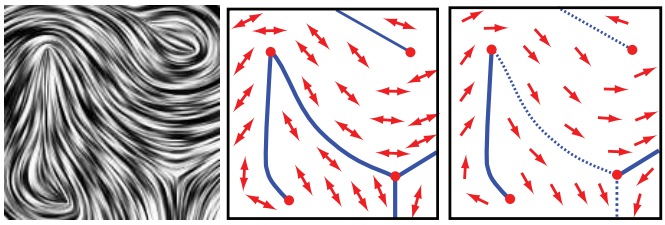
$$\tilde{\Gamma}(x) = \varepsilon(x)\Gamma(x) \quad \text{with} \quad \varepsilon(x) \in \{-1, +1\},$$

such that $\tilde{\Gamma}$ is as smooth as possible. A similar problem is studied by Chen and Zhu for hair synthesis [46]. This direction field is used extensively in the grouplet transform detailed in Section III.

Finding a globally smooth vector field $\tilde{\Gamma}$ is not possible in general because of topological incompatibilities. A smooth vector field can however be computed outside the set of strucutural singularities which are the set of separatrices $S(\Gamma)$. Each separatrix is a flow line γ parallel to Γ

$$\forall\, t, \quad |\langle \gamma'(t), \Gamma(\gamma(t))\rangle| = \|\gamma'(t)\|$$

that connects two singular points of Γ. We compute S by tracing a set of evenly spread flow lines but more efficient methods can be used [59]. Figure 4 (b) shows an example of separatrices $S(\Gamma)$.



(a) Texture f_0 (b) Γ and $S(\Gamma)$ (c) Directions $\tilde{\Gamma}$

Fig. 4. *Computation of a direction field $\tilde{\Gamma}$. The whole set of separatrices $S(\Gamma)$ is shown in (b), whereas only the discontinuities of $\tilde{\Gamma}$ are shown in (c). The texture f_0 is a synthetic example computed using the LIC method [34].*

The sign field ε is found by optimizing the smoothness of $\tilde{\Gamma} = \varepsilon\Gamma$ outside $S(\Gamma)$. In the discrete setting, one solves the following minimization

$$\min_{\varepsilon} \sum_x \left\| \varepsilon(x)\Gamma(x) - \frac{1}{|V_x|} \sum_{y \in V_x} \varepsilon(y)\Gamma(y) \right\|^2.$$

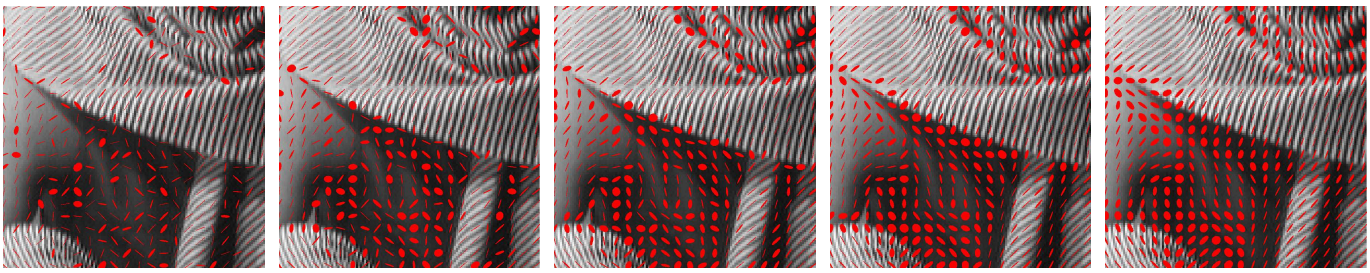
where $y \in V_x$ is a direct neighbor of a pixel x (using the 4 connectivity over the image domain) such that the segment $[x, y]$ does not cross a singularity curve in $S(\Gamma)$. By relaxing the binary constraint $\varepsilon \in \{-1, +1\}$ to $\varepsilon \in \mathbb{R}$, one solves the following linear problem

$$\forall\, x, \qquad \varepsilon(x) = \frac{1}{|V_x|} \sum_{y \in V_x} \varepsilon(y)\langle \Gamma(x), \Gamma(y)\rangle. \quad (3)$$

For the system to be uniquely solvable, one needs to set additional constraints $\{\varepsilon(z_i) = 1\}_{i=1}^I$ to fix the overall sign under-determinacy, where each pixel z_i belongs to a cell of the separatrices segmentation.

Equation (3) requires to solve a sparse linear system and the un-signed orientation field $\Gamma(x)$ is then turned into a signed vector field that defines the local direction of the geometry

$$\forall\, x, \qquad \tilde{\Gamma}(x) = \mathrm{sign}(\varepsilon(x))\Gamma(x).$$



$\mu = 1$ $\mu = 2$ $\mu = 4$ $\mu = 8$ $\mu = 16$

Fig. 3. *Structure tensor fields $T_{f_0}(x)$ computed with an increasing scale μ. The ellipsoid at a pixel x is aligned with the principal axis $\Gamma(x)$ of the tensor $T_{f_0}(x)$ and its aspect matches the anisotropy $\lambda(x)/\lambda^{\perp}(x)$, see equation (2) for the notations.*

Figure 4 shows the computation of a direction field $\tilde{\Gamma}$, whose discontinuities are aligned with a subset of the separatrices.

III. Grouplet Coefficient Layer Computation

Once the geometric layer Γ is known, the coefficient layer is computed with a fast redundant grouplet transform. A redundant grouplet transform is introduced by Mallat in [1]. It corresponds to the decomposition of an image on a redundant family of vectors $\mathcal{B}(\Gamma)$. This family is a tight frame of $\mathbb{R}^N$ chosen adaptively to process in an optimized way some given input image of $N = n^2$ pixels. This section describes a new construction of grouplets frame parameterized by a local geometric flow Γ. The grouplet atoms follow closely the flow lines of Γ. These grouplets are thus efficient to represent textures whose geometry is locally parallel to the orientation field Γ.

A. Association Field Computation and Pruning

A set of non-local discrete association fields $\{A_\ell\}_{\ell=0}^{L-1}$ is computed from the geometrical flow Γ. Each field A_ℓ is then used to compute the grouplet decomposition at a scale 2^ℓ. Each association field A_ℓ groups together two pixels $x \to y = A_\ell(x)$ that are separated by a distance of $\|x - y\| \approx 2^\ell$. The scale L defines the largest width $\sim 2^L$ of a grouplet and is set to $L = 6$ in the numerical experiments, which is enough to capture the elongated patterns of the textures to synthesize.

Our construction of the association fields A_ℓ differs from the original one of [1]. The original construction is based on a fixed directional ordering of the grid pixels, that forbids an arbitrary association field. Such a directional ordering is not well suited to process turbulent textures that might exhibit circular patterns or vortices.

Association Field. The directional vector field $\tilde{\Gamma}$ computed in section II-B from Γ is the seed for a linear differential flow

$$\forall\, x,\ \forall\, t > 0, \quad \frac{\mathrm{d}\varphi_x}{\mathrm{d}t}(t) = \tilde{\Gamma}(\varphi_x(t)) \quad \text{and} \quad \varphi_x(0) = x.$$

This ordinary differential equation is solved numerically by interpolating $\tilde{\Gamma}$. Each integral curve $\{\varphi_x(t)\}_{t>0}$ follows the vector field $\tilde{\Gamma}$.

A raw discrete association field A_ℓ^0 at various scales ℓ is computed by sampling this integral curve at dyadic locations

$$\forall\, x, \quad A_\ell^0(x) = [\varphi_x(2^\ell)]_{n\times n} \in \{0, \ldots, n-1\}^2,$$

where $[\,\cdot\,]_{n\times n}$ is the rounding operator that projects the points on the sampling grid $\{0, \ldots, n-1\}^2$. Although the association fields are quantized in order to link points on the grid $\{0, \ldots, n-1\}^2$, they follow closely the structure of the input texture f_0.

Input: raw association field A_ℓ^0.
Output: pruned association field A_ℓ and coherent ordering V_ℓ.

1) *Initialization:* set $A_\ell = A_\ell^0$ and $k = 0$. Initialize the ordering $\forall\, x,\ V_\ell(x) = -1$. For each x, build the father list $\Phi(x) = \{y \setminus A_\ell(y) = x\}$.
2) *Select a pixel:* choose some x satisfying $V_\ell(x) = -1$.
3) *Back-track:* initialize $W = V_\ell$,
   While $W(A_\ell(x)) < 0$, do
   $x \leftarrow A_\ell(x)$, $W(x) \leftarrow 0$.
4) *Set up the ordering:* initialize the pool $P = \{x\}$.
   While $P \neq \emptyset$, do
   extract $y \in P$, remove $P \leftarrow P \backslash y$,
   set $V_\ell(y) \leftarrow k$, $k \leftarrow k + 1$,
   add fathers $P \leftarrow P \cup (\Phi(y) \cap \{z \setminus V_\ell(z) < 0\})$.
5) *Stop:* if $\{z \setminus V_\ell(z) < 0\} \neq \emptyset$, go back to 2.
6) *Prune the flow:* for all x such that $V_\ell(A_\ell(x)) > V_\ell(x)$, set $A_\ell(x) \leftarrow x$.

Table 1: Greedy algorithm to prune a raw association field.

Association Field Pruning. The grouplet algorithm processes the pixels according to an ordering

$$V_\ell : \{0, \ldots, n-1\}^2 \longrightarrow \{0, \ldots, N-1\} \quad \text{where} \quad N = n^2.$$

At a scale 2^ℓ, the pixels are visited in an order $\{m_0, m_1, \ldots, m_{N-1}\}$ such that $V_\ell(m_i) = i$.

For each ℓ, this ordering V_ℓ is computed to be maximally coherent with the association field A_ℓ^0. This means that a pruned association field A_ℓ close to the raw field A_ℓ^0 is computed together with an ordering V_ℓ that satisfies

$$\forall\, x, \quad A_\ell(x) \neq x \quad \Longrightarrow \quad V_\ell(A_\ell(x)) < V_\ell(x). \tag{4}$$

Excepted at a few singular points where $A_\ell(x) = x$, the association field A_ℓ links pixels with increasing values of the ordering function V_ℓ. This consistent ordering is important for the stability of the grouplet frame $\mathcal{B}(\Gamma)$, since it guarantees that it is a tight frame of $\mathbb{R}^N$.

The computation of such a couple (A_ℓ, V_ℓ) is performed using a greedy algorithm detailed in Table 1. This algorihtm

removes connexions in the field A_ℓ that cause loops in the graph $x \to A_\ell(x)$.

B. Grouplet Transform

A grouplet tight frame

$$\mathcal{B}(\Gamma) = \left\{ b^\Gamma_{\ell,m} \setminus m \in \{0,\dots,n-1\}^2, \ell = 0,\dots,L \right\}$$

is a redundant family of $(L+1)N$ vectors $b^\Gamma_{\ell,m} \in \mathbb{R}^N$ parameterized by the local direction flow Γ. Each grouplet vector $b^\Gamma_{\ell,m}$ is localized around the pixel $m \in \{0,\dots,n-1\}^2$ and follows the flow Γ on a length of 2^ℓ pixels. This family of vectors is thus efficient to compress a texture whose structures follow closely the direction of Γ. This is in particular the case for the image f_0 that has been used to extract the flow Γ as explained in section II, but the tight frame $\mathcal{B}(\Gamma)$ can be used to analyze any image f of N pixels.

Forward decomposition. The forward grouplet transform [1] corresponds to the computation of the decomposition of f onto the vectors of $\mathcal{B}(\Gamma) = \{b^\Gamma_{\ell,m}\}_{\ell,m}$

$$\forall\, \ell, \quad \forall\, m, \quad d_\ell(m) = \langle f,\, b^\Gamma_{\ell,m} \rangle. \tag{5}$$

The grouplet vectors $b^\Gamma_{\ell,m}$ are defined implicitly through a decomposition algorithm that computes directly the coefficients $d_\ell(m)$.

This decomposition is computed with a fast algorithm similar to the Haar wavelet transform. The algorithm keeps track of a coarse approximation $\tilde f$ of f that is progressively filtered with increasing scales 2^ℓ. At a given scale 2^ℓ, the algorithm visits the pixels m by increasing values of $V_\ell(m)$ and computes the coefficient $d_\ell(m)$ as a weighted difference of two values at positions m and $\tilde m = A_\ell(m)$

$$d_\ell(m) = (\tilde f(m) - \tilde f(\tilde m)) \frac{\sqrt{s(m)s(\tilde m)}}{\sqrt{s(m)+s(\tilde m)}} \tag{6}$$

$$\tilde f(\tilde m) \leftarrow \frac{s(m)\tilde f(m) + s(\tilde m)\tilde f(\tilde m)}{s(m)+s(\tilde m)} \tag{7}$$

$$s(\tilde m) \leftarrow s(m) + s(\tilde m). \tag{8}$$

The weight $s(m)$ balances the contribution of each pixel to ensure conservation of energy, even in the case where two flows $A_\ell(m) = A_\ell(m')$ are converging at the same location. Once the process has been performed over L grouplet scales, the recursion is stopped and the remaining coarse scale layer $\tilde f$ is kept with a renormalization $d_L(m) = \tilde f(m)\sqrt{s(m)}$. The grouplet transform maps the original signal $f(x)$ to the set of coefficients $\{d_\ell(m)\}_{\ell,m}$. The overall process is stable and the constraint (4) on the associations implies a conservation of energy

$$\|f\|^2 = \sum_{\ell<L} \sum_m \frac{1}{2^\ell} |d_\ell(m)|^2 + \sum_m \frac{1}{2^{L-1}} |d_L(m)|^2, \tag{9}$$

see [1]. The values of $\{d_\ell(m)\}_{\ell,m}$ are the coefficients of f in a tight frame $\mathcal{B}(\Gamma)$, see (5). These coefficients are defined implicitly by the decomposition algorithm detailed in Table 2, left. Its complexity is $O(N(L+1))$ operations for an image of N pixels and L grouplet scales.

Input: image f and association fields $\{A_\ell, V_\ell\}_\ell$.
Output: coefficients $\{d_\ell(m)\}_{\ell,m}$ and weights s.
Initialization: set $\tilde f = f$ and $\forall m, s(m) = 1$.
Iterations: for each $\ell = 0,\dots,L-1$,
  for each m with increasing $V_\ell(m)$, define $\tilde m = A_\ell(m)$,
    compute $d_\ell(m)$ using (6),
    update $\tilde f(\tilde m)$ and $s(\tilde m)$ according to (7), (8).
Normalize coarse scale: $\forall\, m, d_L(m) = \tilde f(m)\sqrt{s(m)}$.

Table 2: Fast forward grouplet transform algorithm.

Backward decomposition. The backward grouplet transform retrieves an image f from the coefficients $\{d_\ell(m)\}_{m,\ell}$ of the decomposition on the tight frame $\mathcal{B}(\Gamma)$. This requires scanning the pixels by decreasing values of V_ℓ and reversing the steps (6), (7) and (8) for $\tilde m = A_\ell(m)$. The most stable inversion is obtained using the pseudo inverse, which corresponds to computing $S = s(\tilde m) - s(m)$ and

$$v_1 = \tilde f(\tilde m) - \frac{d_\ell(m)\sqrt{s(m)}}{\sqrt{S s(\tilde m)}},\ v_2 = \tilde f(\tilde m) + \frac{d_\ell(m)\sqrt{S}}{\sqrt{s(\tilde m)s(m)}}.$$

and updating the values of $\tilde f$ and s as follow

$$\tilde f(\tilde m) \leftarrow v_1,\ \tilde f(m) \leftarrow \frac{\tilde f(m) + v_2}{2},\ s(\tilde m) \leftarrow S. \tag{10}$$

The fast backward grouplet transform, detailed in Table 3, has a complexity of $O(N(L+1))$ operations.

This pseudo inverse implements the reconstruction formula

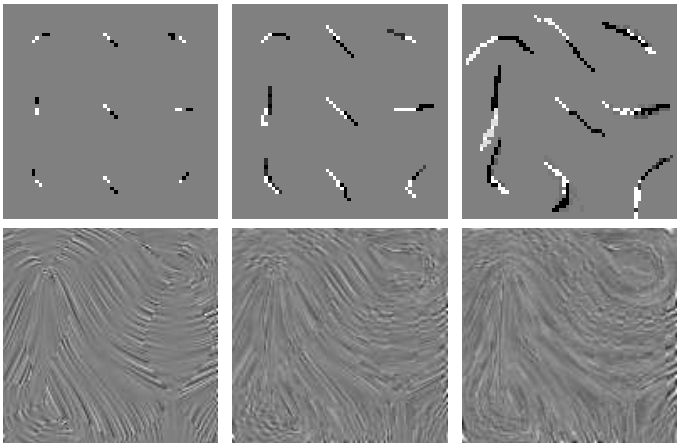
$$f = \sum_{0 \leqslant \ell \leqslant L} \tau_\ell \sum_m d_\ell(m) b^\Gamma_{\ell,m}$$

where $\tau_\ell = 2^{-\ell}$ for $\ell < L$ and $\tau_L = 2^{-L+1}$ accounts for the redundancy of the transform.

Input: coefficients $\{d_\ell(m)\}_{\ell,m}$, weights s
  and association field (A_ℓ, V_ℓ).
Output: image $f = \tilde f$.
Initialization: set $\tilde f(m) = d_L(m)/\sqrt{s(m)}$.
Iterations: For each $\ell = L-1,\dots,0$,
  For each m with decreasing $V_\ell(m)$, define $\tilde m = A_\ell(m)$.
    Update $\tilde f(m), \tilde f(\tilde m)$ and $s(\tilde m)$ using (10).

Table 3: Fast backward grouplet transform algorithm.

Examples of decompositions. Figure 5, top row, shows some examples of grouplet vectors. A vector $b_{\ell,m}$ is computed by applying the backward transform to a set of coefficients $d_{\ell'}(m') = 0$ for $(\ell', m') \neq (\ell, m)$ and $d_\ell(m) = 1$. These vectors follow the geometric structures of the exemplar texture f_0 and are thus efficient to process it. Figure 5, bottom row, shows the coefficients $d_\ell(m)$ in a grouplet tight frame $\mathcal{B}(\Gamma)$ of the texture $f = f_0$. The resulting coefficients $d_\ell(m)$ are highly sparse and compressible since most of them have a small magnitude.



$\ell = 1$ $\ell = 2$ $\ell = 3$

Fig. 5. *Top row: Some examples of grouplets basis vectors $\{b_{\ell,m}(x)\}_m$ for some sampling location m. Bottom row: grouplets coefficients $\{d_\ell(m)\}_m$ of the image f_0 displayed in Figure 4.*

C. Wavelet-Grouplet Transform

A grouplet basis vector $b_{\ell,m}$ has an arbitrary length 2^ℓ but a fixed width of 1 pixel. This is problematic to represent textures with patterns of arbitrary width. Applying the grouplet transform over the wavelet coefficients of an image defines a wavelet-grouplet transform. The corresponding wavelet-grouplet atoms have an arbitrary width 2^j that corresponds to the scale of the wavelet coefficients.

In this wavelet-grouplet setting, one first computes the decomposition of the image f on a set of wavelet vectors

$$\forall\, j = 0, \ldots, J, \forall\, p \in \{0, \ldots, n-1\}^2, \quad f_j(p) = \langle f,\, \psi_{j,p} \rangle,$$

where $\psi_{j,p}$ is a wavelet vector scale 2^j and position p. A Laplacian pyramid tight frame is used in the numerical experiments, which has a redundancy of $J+1$, see [10]. This multiscale transform has the advantage of being non-oriented, which leaves the processing of orientation to the grouplet transform alone. The reconstruction from the coefficients in such a tight frame reads

$$f = \sum_j \kappa_j \sum_p f_j(p) \psi_{j,p},$$

where $\kappa_j = 2^{-2j}$ for $j < J$ and $\kappa_J = 2^{-2(J-1)}$. The set of coefficients f_j is re-transformed using the grouplet transform explained in section III-B.

The projection of each f_j on a grouplet frame $\mathcal{B}(\Gamma)$ generates the following set of coefficients

$$\forall\, j, \ell, m, \quad d_{j,\ell}(m) = \langle f_j,\, b^\Gamma_{\ell,m} \rangle = \langle f,\, b^\Gamma_{j,\ell,m} \rangle$$

where $b^\Gamma_{j,\ell,m}$ is the wavelet-grouplet basis vector defined by

$$b^\Gamma_{j,\ell,m} = \sum_p b^\Gamma_{\ell,m}(p) \psi_{j,p} \in \mathbb{R}^N.$$

A wavelet-grouplet atom $b^\Gamma_{j,\ell,m}$ is an elongated stroke of width $\sim 2^j$ and length $\sim 2^\ell$, that follows the geometrical flow Γ.

This set of vectors $\{b^\Gamma_{j,\ell,m}\}_{j,\ell,m}$ is a wavelet-grouplet tight frame $\mathcal{B}^{\mathrm{w}}(\Gamma)$ of $\mathbb{R}^N$ composed of $(J+1)(L+1)N$ vectors. The pseudo-inverse reconstruction from the wavelet-grouplet coefficients reads

$$f = \sum_{j,\ell} \tau_{j,\ell} \sum_m d_{j,\ell}(m) b^\Gamma_{j,\ell,m} \tag{11}$$

where $\tau_{j,\ell} = \kappa_j \tau_\ell$.

The wavelet-grouplet forward transform corresponds to the computation of the coefficients $\{d_{j,\ell}(m)\}_{j,\ell,m}$ of a given image f in the tight frame $\mathcal{B}^{\mathrm{w}}(\Gamma)$. This algorithm first computes the f_j using the fast translation invariant wavelet transform, see [10] and then applies the fast grouplet transform, Table 2, to each of these f_j. The fast backward wavelet-grouplet transform first applies the backward grouplet transform, Table 3, and then the backward wavelet transform. Both forward and backward algorithms have a complexity of $O((J+1)(L+1)N)$ operations.

IV. Inpainting with Grouplets

Inpainting aims at restoring an image f from which a set $\Omega \subset \{0, \ldots, n-1\}^2$ of pixels is missing. It corresponds to the inversion of the ill posed linear problem

$$y = \Phi f + w \tag{12}$$

where y is the image with missing pixels, w is an additive noise, and Φ is defined as

$$(\Phi f)(x) = \begin{cases} 0 & \text{if} \quad x \in \Omega, \\ f(x) & \text{if} \quad x \notin \Omega. \end{cases} \tag{13}$$

A. Sparse Regularisation in Grouplet Basis

The inpainting problem (12) is regularized by exploiting some smoothness of the solution f. Given some orthogonal basis $\{\psi_k\}_k$ of $\mathbb{R}^N$, a sparsity-regularized solution of the inverse problem is

$$f^\star = \operatorname*{argmin}_{g \in \mathbb{R}^n} \frac{1}{2} \|y - \Phi g\|^2 + \lambda \sum_k |\langle g,\, \psi_k \rangle|, \tag{14}$$

where λ should be adapted to match the noise level $\|w\|$. This prior has been introduced by Donoho and Johnstone [60] with the wavelet basis for denoising purpose. It has then been used to solve more general inverse problems, see for instance [61] and the references therein. It can also be used in conjunction with redundant frames instead of orthogonal bases [21], [22].

Given a wavelet-grouplet basis $\mathcal{B}^{\mathrm{w}}(\Gamma) = \{b^\Gamma_{j,\ell,m}\}_k$, the sparse regularization (14) reads

$$f^\star = \operatorname*{argmin}_{g \in \mathbb{R}^n} \frac{1}{2} \|y - \Phi g\|^2 + \lambda \sum_{j,\ell,m} |\langle g,\, b^\Gamma_{j,\ell,m} \rangle|. \tag{15}$$

The difficulty is that the geometric layer Γ is only partly known outside Ω and in the vicinity of the boundary $\partial\Omega$. The flowing section details an interative algorithm that computes an optimized geometric layer over large missing regions.

Fig. 6. *Iterations $f^{(i)}$ of the inpainting process that modifies the image to obtain a sparse representation in an adapted wavelet-grouplet basis.*

B. Inpainting Algorithm

If the flow Γ is fixed, an iterative thresholding scheme minimizes (15) by updating an estimate $f^{(i)}$

$$f^{(i+1)} = S_\lambda^\Gamma(\bar{f}^{(i)}) \text{ where } \bar{f}^{(i)} = f^{(i)} + \Phi^*(y - \Phi f^{(i)}), \quad (16)$$

where S_λ^Γ soft thresholds the decomposition (11) in the tight frame $\mathcal{B}^{\mathrm{w}}(\Gamma)$

$$S_\lambda^\Gamma(g) = \sum_{j,\ell} \tau_{j,\ell} \sum_m s_\lambda(\langle g,\, b_{j,\ell,m}^\Gamma \rangle)\, b_{j,\ell,m}^\Gamma \quad (17)$$

where $s_\lambda(x) = \max(1 - \lambda/|x|, 0)x$. The computation of $\bar{f}^{(i)}$ in (16) corresponds to imposing the known pixel values

$$\bar{f}^{(i)}(x) = \begin{cases} f^{(i)}(x) & \text{if} \quad x \in \Omega, \\ y(x) & \text{if} \quad x \notin \Omega. \end{cases} \quad (18)$$

These iterations solve an analysis prior that measures the sparsity of the solution as the ℓ^1 norm of the projection of the image on a set of atoms, see [62]. Such an iterative scheme has been introduced to compute the morphological component analysis [21]. For orthogonal bases, it is equivalent to the iterative thresholding algorithm to solve ℓ^1 regularization, see [61].

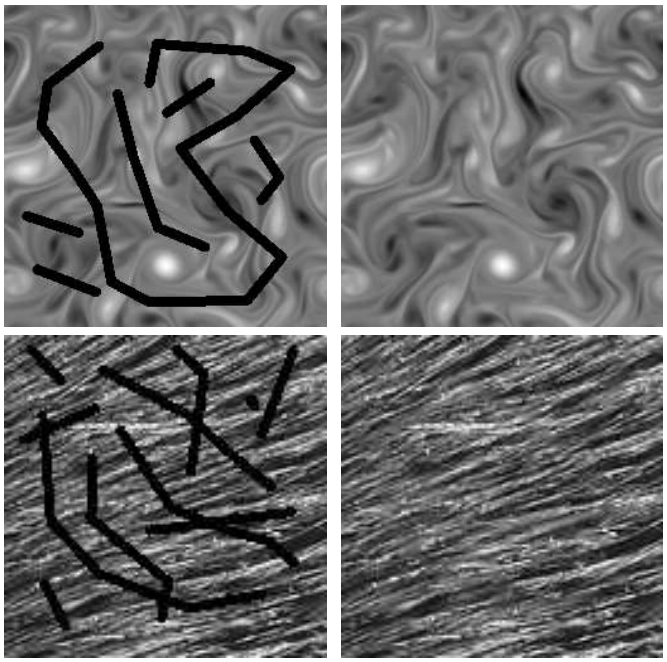
Similarely to the best basis optimization proposed by Peyré [63], the geometric layer Γ is updated iteratively during the iterative thresholding algorithm to improve the sparsity of the solution. Table 4 details the corresponding algorithm. The tolerance parameter η controls the number of iterations. If there is no noise, $w = 0$, the threshold λ is decayed toward 0 during the iterations, see [21].

The thresholding iterations cause the estimate $f^{(i)}$ to be smooth along the flow lines of $\Gamma^{(i)}$. This corresponds to a diffusion of the information from the boundary of the hole Ω to its interior. The flow $\Gamma^{(i)}$ is refined during the iterations in order to connect in a smooth manner the boundaries of the missing region. Figure 6 shows some iterations of the inpainting process in the noiseless case $w = 0$. Figure 7 shows examples of locally parallel textures inpainting[1], also in the noiseless case.

[1]The corral image is obtained from [42] and the hair texture comes from the book of Brodatz [64].

1) *Initialization:* set $f^{(0)} = y$ and $i = 0$.
2) *Enforce the values of known pixels:* compute $\bar{f}^{(i)}$ as defined in (18).
3) *Geometry detection:* compute the flow $\Gamma^{(i)}$ adapted to $\bar{f}^{(i)}$ as detailed in section II-A.
4) *Sparsity enforcing:* update the estimate by thresholding in $\mathcal{B}^{\mathrm{w}}(\Gamma^{(i)})$
$$f^{(i+1)} = S_\lambda^{\Gamma^{(i)}}(\bar{f}^{(i)})$$
where the thresholding operator is defined in (17).
5) *Stop:* while $\|f^{(i+1)} - f^{(i)}\| > \eta$, set $i \leftarrow i + 1$ and go back to 2.

Table 4: Grouplet inpainting algorithm.



Observation y Result $f^\star$

Fig. 7. *Examples of inpainting with grouplets.*

Figure 8 shows a comparison of inpainting with a sparse regularization in a fixed redundant frame and in an adapted grouplet frame computed with the algorithm of Table 4, for a low noise $\|w\| = 0.01\|f\|$. The mask Ω is a random set with $|\Omega|/N = 70\%$ of missing pixels. As detailed by Starck et al. [21] and Fadili et al. [22], the fixed tight frame is the union of a redundant local cosine transform over overlapping blocks of 16×16 pixels and a translation invariant wavelet basis. The local cosine atoms are able to extract the texture part of

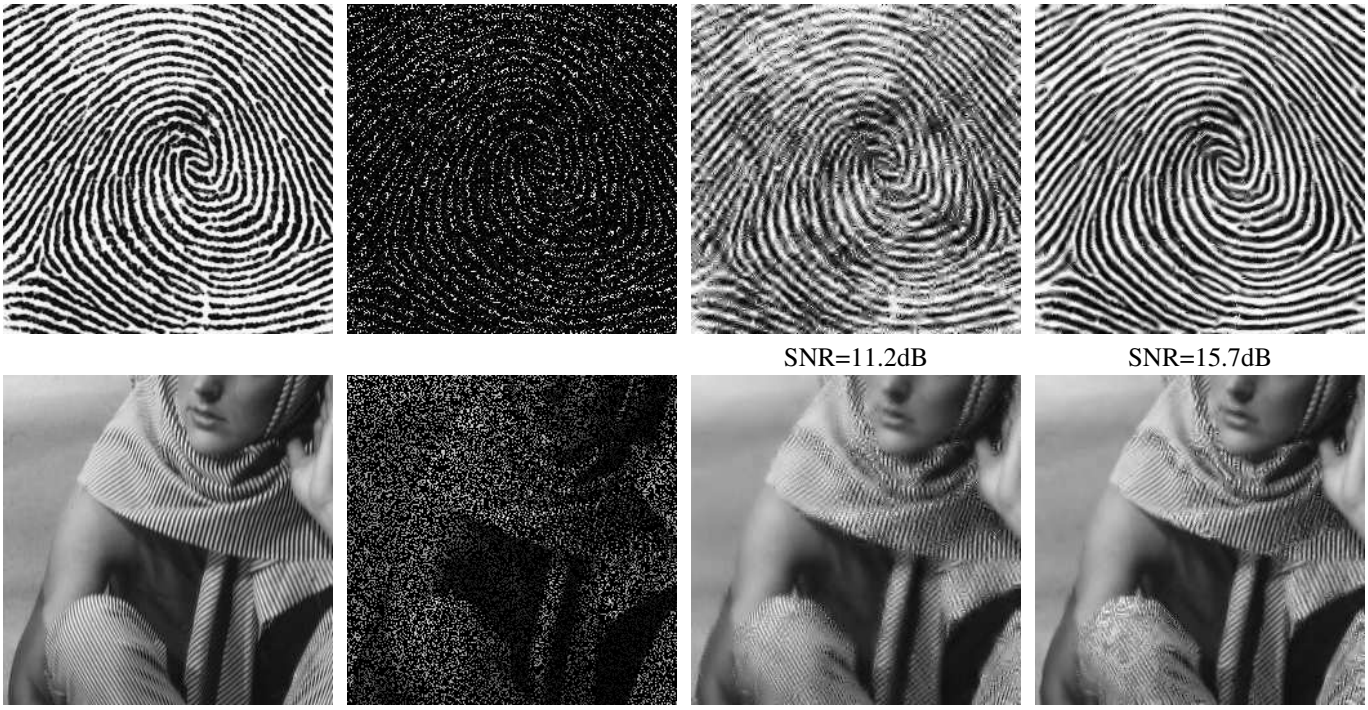


Fig. 8. *Comparison between inpainting with* 70% *(top) and* 80% *(bottom) missing pixels by sparse regularization in DCT+Wavelet redundant frame and grouplet frame.*

the image, while the wavelets takes care of the edges. Figure 8 shows that the adaptivity of the grouplet frame is however able to improve over the fixed sparse regularization.

V. Texture Synthesis with Grouplets

Grouplet texture synthesis creates a new geometric texture f visually similar to a given input exemplar f_0. This requires a statistical modeling of both the geometric layer and the wavelet-grouplet coefficient layer. The grouplet model for locally parallel textures is based on non parameteric statistics of the wavelet-grouplet coefficient layer. Similarly to the work of Heeger and Bergen [26], we retain only marginal statistics of the decomposition. In contrast to texture modeling over multiscale decompositions [26], [30], [31], our texture model also constrains the geometric layer of the texture.

A. Geometric and Coefficients Layers Learning

Grouplet layer learning. The geometric layer of the grouplet model is learned by computing the flow Γ_0 of the exemplar texture f_0 as detailed in Section II-A.

Coefficients layer learning. The coefficient layer D_0 learned from f_0 is described through a the set of coefficients in the frame $\mathcal{B}^{\mathrm{w}}(\Gamma_0)$

$$\forall\,(j,\ell),\quad D_{j,\ell}(f_0,\Gamma_0) = \{\langle f_0,\, b^{\Gamma_0}_{j,\ell,m}\rangle\}_m,$$

together with the set of pixel values $D^{\mathrm{sp}}(f_0) = \{f_0(x)\}_x$. Each $D_{j,\ell}(f_0,\Gamma_0)$ is computed using the fast transform algorithm detailed in Table 2.

The layer D_0 is composed of several independent sets (one per scale (j,ℓ) and the pixel values) that empirical marginals of the statistical distribution of coefficients. They are used by the synthesis algorithm to enforce the synthesized texture to share the same empirical marginals as f_0.

B. Coefficients Layer Synthesis

Texture synthesis with the grouplet model corresponds to synthesizing both a new geometric layer Γ and a new set of coefficients. This section assumes that the geometric layer Γ is known. The synthesized image is drawn at random from a texture ensemble parameterized by Γ. Section V-C studies the synthesis of the geometric layer and shows examples of synthesis with the whole model.

Grouplet texture ensemble. Once both the geometric layer Γ_0 and the coefficient layers D_0 have been learned, a grouplet texture ensemble $\mathcal{T}(\Gamma, D_0)$ is defined for any orientation field Γ. It is the set of textures having the same wavelet-grouplet and pixels marginals as f_0

$$\mathcal{T}(\Gamma, D_0) = \left\{f \in \mathbb{R}^N \setminus \forall\,(j,\ell),\ D_{j,\ell}(f,\Gamma) = D_{j,\ell}(f_0,\Gamma_0)\right\} \cap \left\{f \in \mathbb{R}^N \setminus D^{\mathrm{sp}}(f) = D^{\mathrm{sp}}(f_0)\right\},$$

This texture ensemble is thus parameterized by the geometric layer Γ of the new texture and by the coefficient layer D_0 of the exemplar. The geometry Γ that defines the model might be different from the learned geometry Γ_0. This allows one to synthesize new textures with a different geometry.

Marginal constraint. Our texture model enforces synthesized coefficients to have the same coefficient layer as the one of the exemplar D_0. Each marginal $D_{j,\ell,0} = D_{j,\ell}(f_0,\Gamma_0)$ generates a constraint on the texture to synthesize. The synthesis requires to compute the orthogonal projection $\mathcal{P}(d_{j,\ell}, D_{j,\ell,0})$

of wavelet-groulpet coefficients $d_{j,\ell} \in \mathbb{R}^N$ on

$$\left\{ d \in \mathbb{R}^N \setminus \{d(m)\}_m = D_{j,\ell,0} \right\}.$$

The same projection algorithm applies to the pixel values constraints generated by $D^{\text{sp}}(f_0)$.

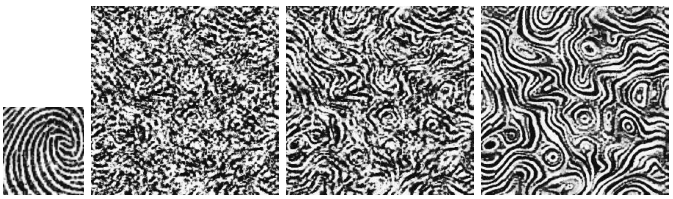
This projection corresponds to an histogram equalization of $d_{j,\ell}$ with the values of $D_{j,\ell,0}$, see [65]. We denote by $d^r_{j,\ell}(i)$ the values of $D_{j,\ell,0}$ ordered by increasing values, and by $d_{j,\ell}(r(i))$ the ordered values of $d_{j,\ell}$, where r is a permutation of the indices. The projection is

$$\mathcal{P}(d_{j,\ell}, D_{j,\ell,0}) = \tilde{d}_{j,\ell} \quad \text{with} \quad \tilde{d}_{j,\ell}(r(i)) = d^r_{j,\ell}(i). \tag{19}$$

If $d_{j,\ell}$ and $D_{j,\ell,0}$ do not have the same number of coefficients, this formula requires interpolation.

Sampling the grouplet texture ensemble. As explained in the FRAME model [30], the uniform distribution on $\mathcal{T}(\Gamma, D_0)$ has maximal entropy, thus leading to the least synthesis bias. Performing an un-biased synthesis is difficult, and Portilla and Simoncelli [31] replace this uniform sampling by a sampling with high entropy. This is achieved by iteratively projecting a white noise image on the set of constraints that define the texture ensemble.

The grouplet coefficients synthesis algorithm detailed in Table 5 uses a similar iterative projection strategy. The initial image $f^{(0)}$ is a random Gaussian white noise image, and the synthesis algorithm converges to a synthesized image $\mathcal{S}(f^{(0)}\,; \Gamma, D_0) \in \mathcal{T}(\Gamma, D_0)$. The procedure iteratively decomposes the image into the wavelet-grouplet frame $\mathcal{B}^{\text{w}}(\Gamma)$, forces the wavelet-grouplet marginal $D_{j,\ell}(f, \Gamma)$ to match the coefficient layer $D_{j,\ell,0}$ of the exemplar, reconstructs an image from the grouplet coefficients and then enforces consistency of the pixel marginal $D^{\text{sp}}(f)$ with the one of the exemplar. The synthesis algorithm handles color images by synthesizing each channel of the image in a color space obtained by applying a PCA to decorrelate the color channels.



f_0 Iter #1 Iter #3 Iter #20

Fig. 9. *Iterations of the synthesis algorithm 5.*

Figure 9 shows the iterations of the synthesis with a user defined geometric layer Γ. The iterations progressively filter the noise along the geometric flow to create texture patterns. The exemplar image f_0 is twice smaller than the synthesized texture f.

Input: orientation flow Γ, initial image $f^{(0)}$,
coefficient layer $D_0 = \{D_{j,\ell,0}\}_{j,\ell} \cup D_0^{\text{sp}}$.
Output: synthesized texture f.

1) *Initialization:* set $i \leftarrow 0$.
2) *Forward transform:* compute the wavelet-grouplet coefficients $\{d^{(i)}_{j,\ell}(m)\}_{j,\ell,m}$ of $f^{(i)}$ in $\mathcal{B}^{\text{w}}(\Gamma)$.
3) *Grouplet constraints:* for each (j,ℓ), perform the histogram equalization (19): $\bar{d}^{(i)}_{j,\ell} \leftarrow \mathcal{P}(d^{(i)}_{j,\ell}, D_{j,\ell,0})$.
4) *Backward transform:* compute $\bar{f}^{(i+1)}$ from the coefficients $\{\bar{d}^{(i)}_{j,\ell}(m)\}_{j,\ell,m}$.
5) *Projection on gray-level constraint:* equalize the pixel histogram: $f^{(i+1)} \leftarrow \mathcal{P}(\bar{f}^{(i+1)}, D_0^{\text{sp}})$.
6) *Stop:* while not converged, set $i \leftarrow i+1$ and go back to 3.

Table 5: Synthesis algorithm to compute $\mathcal{S}(f^{(0)}\,; \Gamma, D_0)$.

C. Geometric Layer Synthesis

Texture synthesis with a user-defined geometry Figure 10 shows examples of synthesis[2] where the geometry is copied from the exemplar f_0, meaning $\Gamma = \Gamma_0$. The turbulent geometry of f_0 is transfered to the synthesized texture f.

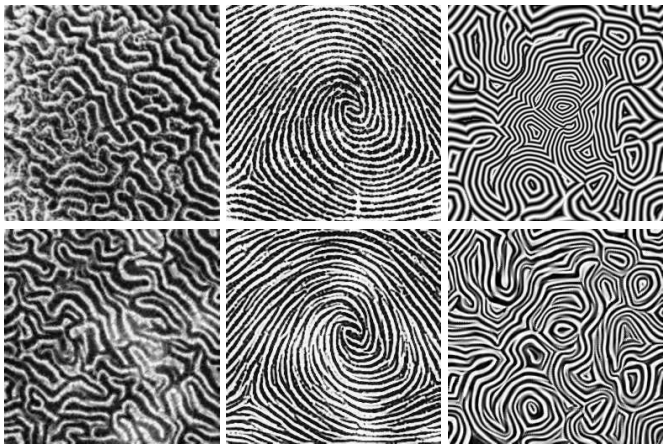


Fig. 10. *Texture synthesis using the same geometry $\Gamma = \Gamma_0$ as the exemplar texture. Top: original f_0 ; bottom : synthesized f.*

Figure 11 shows examples of synthesis with the same exemplar f_0 and several user-defined flows Γ with an increasing complextiy[3]. Some fine scale details are not present in the new texture, because they are not captured by the grouplet model.

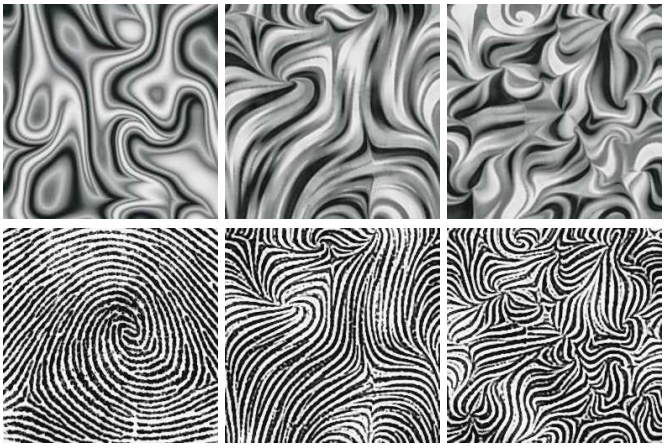
Statistical Modeling of the Geometric Layer The geometric layer is modeled by imposing statistical constraint on the orientation flow Γ. This orientation field is turned into a smooth vector field using the following angle-doubling mapping

$$\Gamma = (\cos(\theta), \sin(\theta)) \;\mapsto\; \beta(\Gamma) = (\cos(2\theta), \sin(2\theta)). \tag{20}$$

The inverse mapping β^{-1} is computed similarly by halving the angle of a unit-norm vector.

[2]The corral is taken from [42], the fingerprint image extracted from a database of fingerprint images [47], the synthetic texture on the right is an example of locally parallel texture obtained from [66].

[3]The texture on the top row is obtained from [40].



Exemplar f_0 Synthesis #1 Synthesis #2

Fig. 11. *Texture synthesis using a synthesized geometry. The complexity of the orientation flow Γ is increased from left to right by adding singular points.*

The resulting field $\beta(\Gamma)$ is smooth excepted at singular points of the orientation field. The wavelet decomposition is able to represent efficiently functions with pointwise singularities, see [10]. Our geometry synthesis algorithm thus iteratively constraints the wavelet marginal distributions of each coordinate of $\beta(\Gamma)$. This corresponds to the use of Heeger and Bergen algorithm [26] to synthesize the flow after the mapping by β. The corresponding geometry synthesis algorithm is detailed in Table 6.

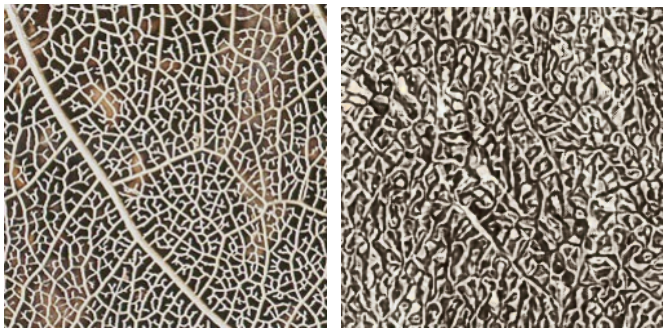
Input: exemplar orientation field Γ_0.
Output: synthesized orientation field Γ.

1) *Initialization:* set $\Gamma^{(0)} \leftarrow$ random, $i \leftarrow 0$, compute the marginals $\beta_j = \{\langle \beta(\Gamma_0), \psi_{j,p} \rangle\}_p$ of the wavelet coefficients of $\beta(\Gamma_0)$.
2) *Wavelet transform:* compute the wavelet coefficients of $\beta(\Gamma^{(i)})$: $\beta_j^{(i)}[p] = \langle \beta(\Gamma^{(i)}), \psi_{j,p} \rangle$.
3) *Impose wavelet marginal:* for each j, perform the histogram equalization (19): $\bar{\beta}_j^{(i)} \leftarrow \mathcal{P}(\beta_j^{(i)}; \beta_j)$ (each of the two coordinates of $\beta_j^{(i)}$ and β_j are equalized separately).
4) *Backward transform:* compute $\beta(\bar{\Gamma}^{(i)})$ and thus $\bar{\Gamma}^{(i)}$ from the wavelet coefficients $\bar{\beta}_j^{(i)}$. Set $\Gamma^{(i+1)} = \bar{\Gamma}^{(i)} / \|\bar{\Gamma}^{(i)}\|$.
5) *Stop:* while not converged, set $i \leftarrow i+1$ and go back to 2.

Table 6: Geometry synthesis algorithm.

Figure 12 shows examples[4] of synthesis that combine the algorithm of Table 6 for the synthesis of the geometry Γ and the algorithm of Table 5 for the synthesis of the texture $f \in \mathcal{T}(\Gamma, D_0)$. The comparison with the wavelet-domain synthesis [26], [31] shows that our method enhances this multiscale synthesis by creating elongated geometrical structures. The comparison with texture quilting [41] shows that the visual quality is slightly below such a patch-recopy approach. The grouplet synthesis is however able to generate new geometric patterns not present in the original texture. The quilting process is performed with patches of size 32×32, and the initial seed patch in the upper-left corner of the synthesis is extracted in the center of the exemplar.

To handle textures with complex geometries, the geometric smoothing parameter σ in (1) should be small, which creates instabilities in the synthesis process. The geometric layer Γ captured with a small σ indeed exhibits many singularities that are difficult to synthesize with the algorithm of Table 6. This shows the fundamental limit of our model to capture geometric fields that are highly irregular, so that we consistently use $\sigma = 6$ in the synthesis experiments. Figure 13 shows a typical example of synthesis failure for a geometric texture whose geometry is not locally parallel. The patterns of the textures exhibit many crossing singularities that are poorly captured both by the geometric layer and by the grouplet transform.



Exemplar f_0 Synthesis f

Fig. 13. *Examples of synthesis failures.*

VI. Dynamic Texture Synthesis

A time dependent texture $\tilde{f}_t$ is obtained by computing a time evolution of both the geometric layer Γ_t and the coefficient layer. Fluid textures are computed by evolving the geometry according to the equations of fluid motion, while keeping the coefficient layer fixed. Texture mixing interpolates in time both layers using two exemplars.

A. Fluid Textures

A dynamic fluid texture $\{\tilde{f}_t\}_{t \geqslant 0}$ is created from a single exemplar image f_0 by evolving in time the direction field $\tilde{\Gamma}_t$ of the geometric layer according to the incompressible Navier Stokes equation

$$\frac{\partial \tilde{\Gamma}_t}{\partial t} = \mathcal{P}_{\text{inc}} \left(-(\tilde{\Gamma}_t \cdot \nabla)\tilde{\Gamma}_t + \nu \Delta \tilde{\Gamma}_t \right),$$

with Neumann or periodic boundary conditions, where ν is the viscosity of the fluid and Δ is the Laplacian operator. The geometry evolution of a fluid texture is governed by a directional vector field $\tilde{\Gamma}_t$ and not directly by the geometric layer Γ_t that is not oriented.

The projector $\tilde{v} = \mathcal{P}_{\text{inc}}(v)$ on the set of divergence free vector fields is computed by solving the Poisson equation for a scalar potential V

$$\tilde{v} = v - \nabla V \quad \text{where} \quad \Delta V = \nabla \cdot v. \tag{21}$$

[4]The wood texture is obtained from [40]. The turbulence image (bottom row) is obtained by a direct simulation of bidimensional turbulent flow computed by K. Schneider, see [67].

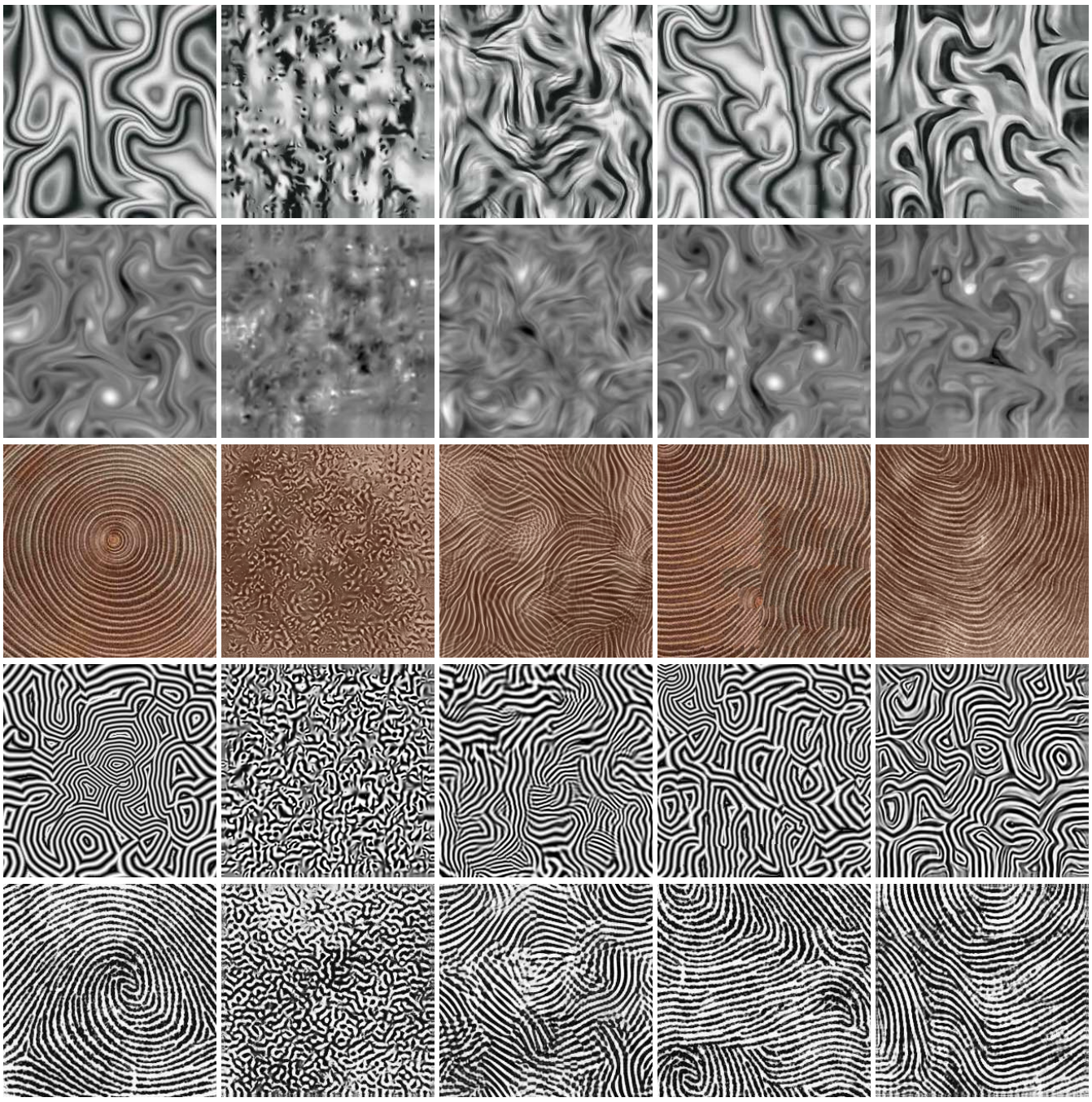


Fig. 12. *Comparison of Heeger and Bergen wavelet domain synthesis [26], Portilla and Simoncelli synthesis [31] and texture quilting [41] with the grouplet synthesis. In this setting, the geometry Γ is synthesized using algorithm 6.*

The synthesized texture $\tilde{f}_t$ for $t > 0$ is obtained by solving an advection equation projected on the grouplet texture model $\mathcal{T}(\Gamma_t, D_0)$ defined in Section V-B by the current geometry $\tilde{\Gamma}_t$ and the coefficient layer D_0 of f_0. This projected advection is discretized in time with a step size τ

$$\tilde{f}_{t+\tau} = \mathcal{S}\left(\tilde{f}_t - \tau(\tilde{\Gamma}_t \cdot \nabla)\tilde{f}_t + \tau\tilde{\nu}\Delta\tilde{f}_t \; ; \; \Gamma_t/\|\Gamma_t\|, D_0\right), \quad (22)$$

where $\tilde{\nu}$ is the viscosity of the texture, that might be different from ν. The projection $\mathcal{S}(\cdot\,; \Gamma, D_0)$ on the grouplet model is computed with a few steps of the iterative algorithm detailed in Table 5.

The advection in both (21) and (22) is computed numerically using the implicit solver of Stam [49]. It makes use of the following flow-warping operator, that is defined for scalar functions and vector fields

$$\mathcal{W}_{\tilde{\Gamma}}(f) = \tilde{f} \quad \text{where} \quad \tilde{f}(x) = f(x - \tilde{\Gamma}(x)). \quad (23)$$

Table 7 details the steps of the fluid texture synthesis algorithm. Figure 14 shows two examples of dynamic texture synthesis. The initial texture $\tilde{f}_0 = \mathcal{S}(f^{(0)}\,; \Gamma_0, D_0)$ is a random white noise texture $f^{(0)}$ projected on the grouplet model.

Input: initial textures f_0 and $\tilde{f}_0$, time step size $\tau > 0$.
Output: synthesized texture $\tilde{f}_i$ for times $0 \leqslant \tau i \leqslant T$.

1) *Initialization:* compute $\tilde{\Gamma}_0$ and D_0 from f_0, set $i \leftarrow 0$.
2) *Advect:* compute
$$\tilde{\Gamma}_i^{(1)} = \mathcal{W}_{\tau\tilde{\Gamma}_i}(\tilde{\Gamma}_i) \quad \text{and} \quad \tilde{f}_i^{(1)} = \mathcal{W}_{\tau\tilde{\Gamma}_i}(\tilde{f}_i).$$
3) *Diffuse:* compute
$$\tilde{\Gamma}_i^{(2)} = \tilde{\Gamma}_i^{(1)} + \tau\nu\Delta\tilde{\Gamma}_i^{(1)} \quad \text{and} \quad \tilde{f}_i^{(2)} = \tilde{f}_i^{(1)} + \tau\tilde{\nu}\Delta\tilde{f}_i^{(1)}.$$
4) *Project:* compute $\tilde{\Gamma}_{i+1} = \mathcal{P}_{\text{inc}}(\tilde{\Gamma}_i^{(2)})$ and
$$\tilde{f}_{i+1} = \mathcal{S}(\tilde{f}_i^{(2)}\,;\, \Gamma_{i+1}/\|\Gamma_{i+1}\|, D_0)$$
5) *Stop:* while $\tau i < T$, go back to 2.

Table 7: Grouplet fluid texture synthesis.

B. Texture Mixing

Texture mixing computes a dynamic texture $\tilde{f}_t$ for $t \in [0,1]$, where $(\tilde{f}_0, \tilde{f}_1)$ are visually similar to a pair of exemplars (f_0, f_1). This is achieved by interpolating between both the geometric layers (Γ_0, Γ_1) and the coefficient layers (D_0, D_1) to obtained new layers Γ_t and D_t for each t. Each coefficient layer is decomposed as $D_k = \{D_{j,\ell,k}\}_{j,\ell} \cup D_k^{\text{sp}}$ for $k = 0, 1$.

The geometric layer interpolation is achieved by a pointwise averaging of the angle-doubled flow
$$\beta(\Gamma_t) = \frac{(1-t)\beta(\Gamma_0) + t\beta(\Gamma_1)}{\|(1-t)\beta(\Gamma_0) + t\beta(\Gamma_1)\|},$$
where β is defined in (20). More complicated interpolation scheme for vector field could be used.

The grouplet coefficient layer interpolation is achieved by partial matching of the marginals. For each (j,ℓ), we denote by $\{d_{j,\ell,k}(r_k(i))\}_i$, for $k = 0, 1$, the coefficients of $D_{j,\ell,k}$ ordered by increasing values, where r_k is a permutations of the indexes. The marginal $D_{j,\ell,t}$ of the interpolated layer D_t is defined by
$$D_{j,\ell,t} = \{(1-t)d_{j,\ell,0}(r_0(i)) + t d_{j,\ell,1}(r_1(i))\}_m$$
This corresponds to a 1D optimal transportation of the measures associated to the marginals.

The texture mixing algorithm then starts from $\tilde{f}_{-\tau}$ being a random noise, and iteratively projecting on the interpolated texture model $\mathcal{T}(\Gamma_{i\tau}, D_{i\tau})$
$$\forall\, i \in [0, 1/\tau], \quad \tilde{f}_{i\tau} = \mathcal{S}(\tilde{f}_{(i-1)\tau}\,;\, \Gamma_{i\tau}, D_{i\tau}).$$
Figure 15 shows two examples of texture mixing, where the patterns of the textures are blended according the parameter$t$.

VII. Conclusion

This paper proposed a new grouplet texture models. The geometric layer is computed with a local orientation analysis while the coefficient layer is computed with a wavelet-grouplet transform. The sparsity of the grouplet representation regularizes inpainting with an iterative thresholding algorithm. Statistical modeling of the geometric and coefficient layer enables the synthesis of anisotropic textures. Time dependent synthesis is possible by coupling the synthesis with a fuild simulation or by interpolating between two grouplets models. More advanced models could be built on top of this grouplet representation. This could include couplings between the geometry and the coefficient layers that certainly occur in natural images, for instance because of occlusion and shading.

Acknowledgements. I would like to thank Stéphane Mallat and Eero Simoncelli for fruitful discussions.

References

[1] S. Mallat, "Geometrical grouplets," *to Appear in Applied and Computational Harmonic Analysis*, 2008.
[2] J. R. Bergen and M. S. Landy, "Computational modeling of visual texture segregation," in *Comp. Models of Visual Proc.*, M. S. Landy and J. A. Movshon, Eds. Cambridge, MA: MIT Press, 1991, pp. 253–271.
[3] M. Kass and A. Witkin, "Analyzing oriented patterns," *Comput. Vision Graph. Image Process.*, vol. 37, no. 3, pp. 362–385, 1987.
[4] C. F. Shu and R. C. Jain, "Vector field analysis for oriented patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 9, pp. 946–950, 1994.
[5] U. Kothe, "Edge and junction detection with an improved structure tensor," in *Proc. DAGM03*, 2003, pp. 25–32.
[6] A. Rao and R. Jain, "Computerized flow field analysis: Oriented texture fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 693–709, 1992.
[7] M. Almeida, "Anisotropic textures with arbitrary orientation," *Journal of Mathematical Imaging and Vision*, vol. 7, no. 3, pp. 241–251, June 1997.
[8] O. Ben-Shahar and S. Zucker, "The perceptual organization of texture flow: A contextual inference approach," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 25, no. 4, pp. 401–417, 2003.
[9] A. Jain, L. Hong, and R. Bolle, "On-line fingerprint verification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 302–314, 1997.
[10] S. Mallat, *A Wavelet Tour of Signal Processing*. San Diego: Academic Press, 1998.
[11] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multi-scale transforms," *IEEE Trans Information Theory*, vol. 38, no. 2, pp. 587–607, Mar. 1992.
[12] F. G. Meyer and R. R. Coifman, "Brushlets: A tool for directional image analysis and image compression," *Journal of Appl. and Comput. Harmonic Analysis*, vol. 5, pp. 147–187, 1997.
[13] E. Candès and D. Donoho, "New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities," *Comm. Pure Appl. Math.*, vol. 57, no. 2, pp. 219–266, 2004.
[14] L. Demanet and L. Ying, "Wave atoms and sparsity of oscillatory patterns," *Applied and Computational Harmonic Analysis*, vol. 23, no. 3, pp. 368–387, 2007.
[15] E. Le Pennec and S. Mallat, "Bandelet Image Approximation and Compression," *SIAM Multiscale Modeling and Simulation*, vol. 4, no. 3, pp. 992–1039, 2005.
[16] S. Mallat and G. Peyré, "Orthogonal bandlets for geometric image approximation," *Communication on Pure and Applied Mathematics*, vol. 61, no. 9, pp. 1173–1212, 2008.
[17] S. Masnou, "Disocclusion: a variational approach using level lines," *IEEE Trans. Image Processing*, vol. 11, no. 2, pp. 68–76, Feb. 2002.
[18] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Trans. Image Processing*, vol. 10, no. 8, pp. 1200–1211, Aug. 2001.
[19] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. Siggraph 2000*, 2000, pp. 417–424.
[20] D. Tschumperlé and R. Deriche, "Vector-valued image regularization with PDEs: Acommon framework for different applications," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 27, no. 4, pp. 506–517, 2005.
[21] M. Elad, J.-L. Starck, D. Donoho, and P. Querre, "Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)," *Journal on Applied and Computational Harmonic Analysis*, vol. 19, pp. 340–358, 2005.
[22] M. Fadili, J.-L. Starck, and F. Murtagh, "Inpainting and zooming using sparse representations," *The Computer Journal (to appear)*, 2007.
[23] S. Mallat and G. Peyré, "A review of bandlet methods for geometrical image representation," *Numerical Algorithms*, vol. 44, no. 3, pp. 205–234, 2007.

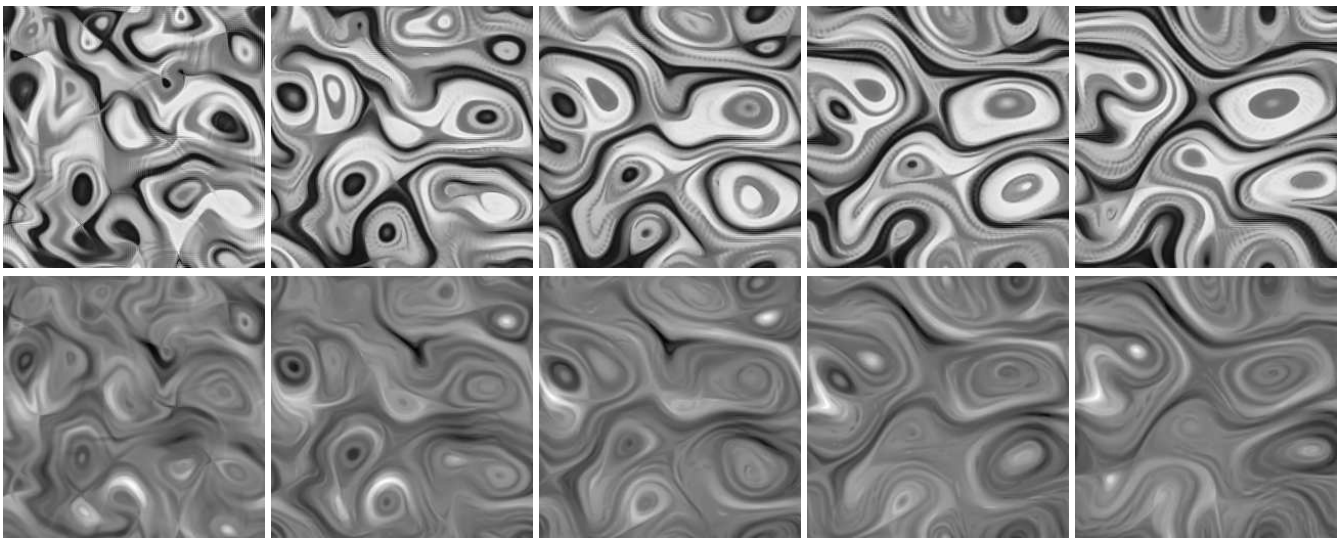


Fig. 14. *Example of dynamic texture synthesis. The exemplar images are the texture f_1 shown on Figure 14, top.*

[24] J.-P. Lewis, "Texture synthesis for digital painting," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 245–252, 1984.

[25] K. Perlin, "An image synthesizer," in *Proc. Siggraph '85*. New York, NY, USA: ACM Press, 1985, pp. 287–296.

[26] D. J. Heeger and J. R. Bergen, "Pyramid-Based texture analysis/synthesis," in *Proc. Siggraph '95*, ser. Annual Conference Series, R. Cook, Ed., ACM SIGGRAPH. Addison Wesley, Aug. 1995, pp. 229–238.

[27] R. Cook and T. DeRose, "Wavelet noise," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 803–811, 2005.

[28] R. Paget and I. D. Longstaff, "Texture synthesis via a noncausal nonparametric multiscale markov random field," *IEEE Trans. Image Processing*, vol. 7, no. 6, pp. 925–931, Jun. 1998.

[29] J. S. D. Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture images," in *Proc. Siggraph '97*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 361–368.

[30] S. C. Zhu, Y. Wu, and D. Mumford, "Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling," *Int. J. Comput. Vision*, vol. 27, no. 2, pp. 107–126, 1998.

[31] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int. J. Comput. Vision*, vol. 40, no. 1, pp. 49–70, 2000.

[32] A. Witkin and M. Kass, "Reaction-diffusion textures," in *Proc. Siggraph '91*. ACM Press, 1991, pp. 299–308.

[33] J. Weickert, "Coherence-enhancing diffusion filtering," *Int. J. Comput. Vision*, vol. 31, no. 2-3, pp. 111–127, 1999.

[34] B. Cabral and L. C. Leedom, "Imaging vector fields using line integral convolution," in *Proc. Siggraph '93*, 1993, pp. 263–272.

[35] J. J. van Wijk, "Image based flow visualization," in *Proc. Siggraph '02*. New York, NY, USA: ACM Press, 2002, pp. 745–754.

[36] U. Diewald, T. Preußer, and M. Rumpf, "Anisotropic diffusion in vector field visualization on Euclidean domains and surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 2, pp. 139–149, 2000.

[37] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*. IEEE Computer Society, 1999, p. 1033.

[38] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. Siggraph '00*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 479–488.

[39] M. Ashikhmin, "Synthesizing natural textures," in *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*. New York, NY, USA: ACM Press, 2001, pp. 217–226.

[40] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Trans. Graph.*, vol. 20, no. 3, pp. 127–150, 2001.

[41] A. Efros and W. Freeman, "Image quilting for texture synthesis and transfer," *Proc. Siggraph '01*, pp. 341–346, August 2001.

[42] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 277–286, Jul. 2003.

[43] S. Lefebvre and H. Hoppe, "Parallel controllable texture synthesis," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 777–786, 2005.

[44] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 795–802, Jul. 2005.

[45] S. Lefebvre and H. Hoppe, "Appearance-space texture synthesis," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 541–548, Jul. 2006.

[46] H. Chen and S. C. Zhu, "A generative sketch model for human hair analysis and synthesis," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 28, no. 7, pp. 1025–1040, 2006.

[47] R. Cappelli, D. Maio, A. Lumini, and D. Maltoni, "Fingerprint image reconstruction from standard templates," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1489–1503, Sep. 2007.

[48] N. Foster and D. N. Metaxas, "Realistic animation of liquids," *Graphical Models and Image Processing*, vol. 58, no. 5, pp. 471–483, Sep. 1996.

[49] J. Stam, "Stable fluids," in *Proc. of Siggraph'99*, A. Rockwood, Ed. N.Y.: ACM Press, Aug. 1999, pp. 121–128.

[50] R. Bridson, J. Houriham, and M. Nordenstam, "Curl-noise for procedural fluid flow," *ACM Trans. Graph.*, vol. 26, no. 3, p. 46, 2007.

[51] F. Neyret, "Advected textures," in *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, D. Breen and M. Lin, Eds. Aire-la-Ville: Eurographics Association, Jul. 26–27 2003, pp. 147–153.

[52] J. Stam and E. Fiume, "Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes," in *Proc. of Siggraph'95*, 1995, pp. 129–136.

[53] A. Lamorlette and N. Foster, "Structural modeling of natural flames," in *Proc. of Siggraph'02*, ser. ACM Transactions on Graphics, S. Spencer, Ed., vol. 21, 3. New York: ACM Press, Jul. 21–25 2002, pp. 729–735.

[54] T. Kim, N. Thürey, D. James, and M. Gross, "Wavelet turbulence for fluid simulation," in *Proc. of SIGGRAPH '08*. New York, NY, USA: ACM, 2008, pp. 1–6.

[55] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, Feb. 2003.

[56] A. W. Bargteil, F. Sin, J. E. Michaels, T. G. Goktekin, and J. F. O'Brien, "A texture synthesis method for liquid animations," in *Proc. of SCA '06*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 345–351.

[57] V. Kwatra, D. Adalsteinsson, T. Kim, N. Kwatra, M. Carlson, and M. C. Lin, "Texturing fluids," *IEEE Trans. Vis. Comput. Graph*, vol. 13, no. 5, pp. 939–952, 2007.

[58] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Texture mixing and texture movie synthesis using statistical learning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 2, pp. 120–135, Apr./Jun. 2001.

[59] X. Tricoche and G. Scheuermann, "Topology simplification of symmetric, second-order 2d tensor fields," *In Geometric Modeling Methods in Scientific Visualization, Springer*, pp. 275–292, 2003.

[60] D. Donoho and I. Johnstone, "Ideal spatial adaptation via wavelet shrinkage," *Biometrika*, vol. 81, pp. 425–455, Dec 1994.

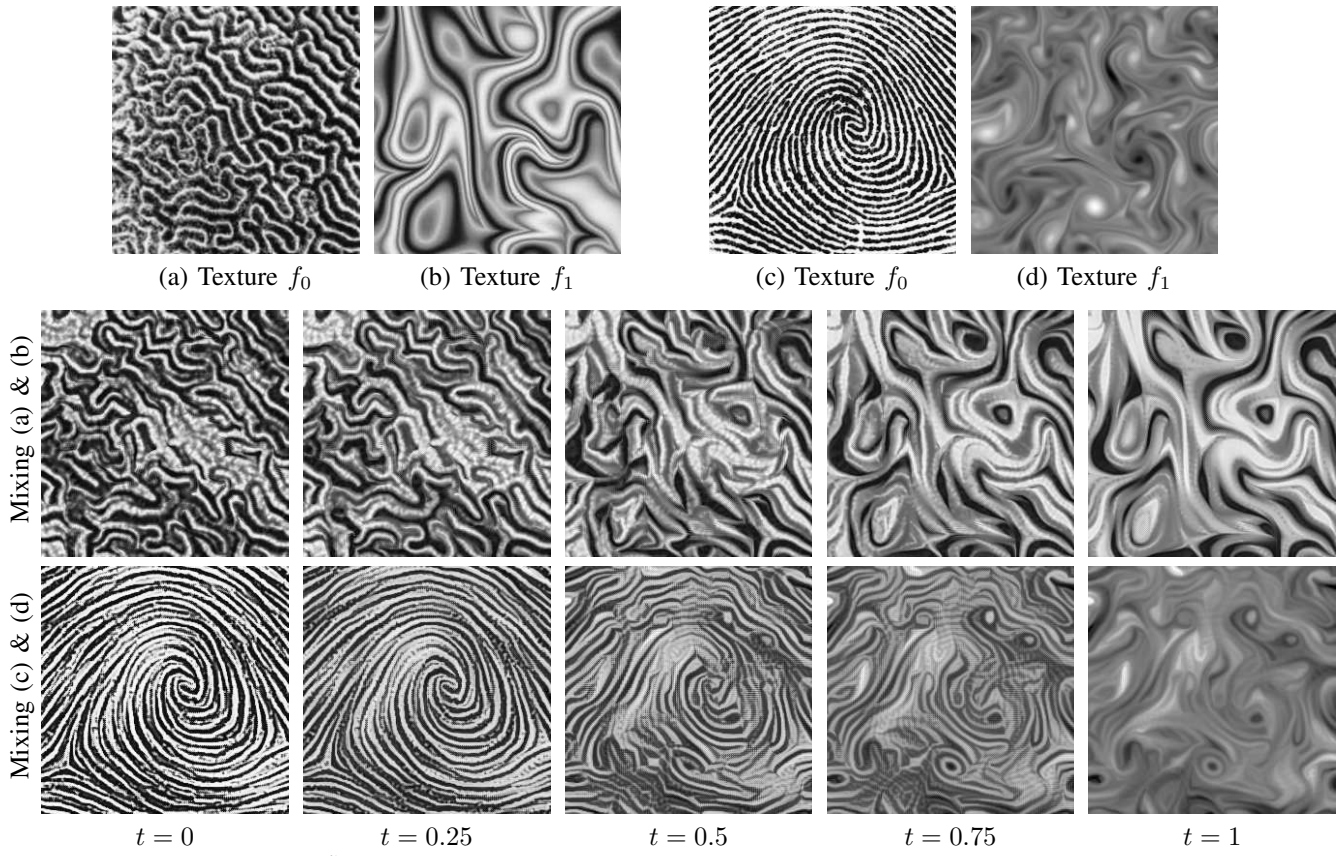


Fig. 15. *Example of mixed textures $\tilde{f}_t$ for various t.*

[61] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math*, vol. 57, pp. 1413–1541, 2004.

[62] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse Problems*, vol. 23, pp. 947–968, Jun. 2007.

[63] G. Peyré, "Best basis compressed sensing," *Preprint Ceremade*, 2007.

[64] P. Brodatz, *Textures: A photographic album for artists and designers*. New York: Dover Publications, 1966.

[65] R. C. Gonzales and R. E. Woods, *Digital Image Processing*. Addison-Wesley Publishing Company, 1993.

[66] G. Peyré, "Manifold models for signals and images," *to appear in Computer Vision and Image Understanding*, 2008.

[67] M. Farge and K. Schneider, *Encyclopedia of Mathematical Physics*. Elsevier, 2006, ch. Wavelets: application to turbulence, pp. 408–420.