Texture Synthesis with Grouplets
Gabriel Peyré

To cite this version:
Gabriel Peyré. Texture Synthesis with Grouplets. IEEE Transactions on Pattern Analysis and Machine Intelligence, Institute of Electrical and Electronics Engineers, 2009, 32 (4), pp.733-746. 10.1109/TPAMI.2009.54 . hal-00360795

HAL Id: hal-00360795
https://hal.archives-ouvertes.fr/hal-00360795
Submitted on 12 Feb 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Texture Synthesis with Grouplets

Gabriel Peyré, Member, IEEE

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

The estimation of the model from an exemplar $f_0$ is performed by computing an optimized flow $\Gamma$ adapted to $f_0$, and the collection of coefficients $\{b_{j,\ell,m}^\Gamma\}_{j,\ell,m}$ of $f_0$ in the wavelet-grouplet family $\mathcal{F}^w(\Gamma)$. 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 \( \Gamma \) 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 \( \{0, \ldots, n-1\}^2 \) as the convolution against two directional derivative filters

\[
v = (f_0 \ast h^1, f_0 \ast h^2) = (-f_0 \ast h^2, f_0 \ast h^1)
\]

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

where the scale \( \mu_0 \) 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 \( vv^T \)

\[
T_{f_0}(x) = (G_{\mu} \ast (vv^T))(x) = G_{\mu} \ast \left( \begin{array}{cc} v_1^2 & v_1 v_2 \\ v_1 v_2 & v_2^2 \end{array} \right)(x),
\]

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 \( \mu \). This scale \( \mu \) 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

\[
T_{f_0}(x) = \lambda(x)(\Gamma(x)\Gamma(x)^T) + \lambda^\perp(x)(\Gamma^\perp(x)\Gamma^\perp(x)^T).
\]

where the eigenvalues are \( \lambda(x) \geq \lambda^\perp(x) \geq 0 \) 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 \( \Gamma \) 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 \( \Gamma \) by optimizing a sign field \( \epsilon \)

\[
\tilde{\Gamma}(x) = \epsilon(x)\Gamma(x) \quad \text{with} \quad \epsilon(x) \in \{-1, +1\},
\]

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 structural singularities which are the set of separatrices \( S(\Gamma) \). Each separatrix is a flow line \( \gamma \) parallel to \( \Gamma \)

\[
\forall t, \quad |(\gamma'(t), \Gamma(\gamma(t)))| = |\gamma'(t)|
\]

that connects two singular points of \( \Gamma \). 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 \( S(\Gamma) \).

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

\[
\min_{\epsilon} \sum_x \|\epsilon(x)\Gamma(x) - \frac{1}{|V_x|} \sum_{y \in V_x} \epsilon(y)\Gamma(y)\|^2.
\]

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 \( \epsilon \in \{-1, +1\} \) to \( \epsilon \in \mathbb{R} \), one solves the following linear problem

\[
\forall x, \quad \epsilon(x) = \frac{1}{|V_x|} \sum_{y \in V_x} \epsilon(y) \Gamma(y, \Gamma(y)).
\]

For the system to be uniquely solvable, one needs to set additional constraints \( \{\epsilon(z_i) = 1\}_{i=1}^N \) 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

\[
\forall x, \quad \tilde{\Gamma}(x) = \text{sign}(\epsilon(x))\Gamma(x).
\]
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 \( \Gamma \) 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 grouplet transform.

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 \( \Gamma \). Each field \( A_\ell \) is then used to compute the grouplet decomposition at a scale \( 2^\ell \). Each association field \( A_\ell \) groups together two pixels \( x \rightarrow y = A_\ell(x) \) 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 \( A_\ell \) 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 \( \Gamma \) computed in section II-B from \( \Gamma \) is the seed for a linear differential flow

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

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 \( A_0^\ell \) at various scales \( \ell \) is computed by sampling this integral curve at dyadic locations

\[
\forall x, \quad A_0^\ell(x) = [\varphi_x(2^\ell)]_{n \times n} \in \{0, \ldots, n-1\}^2,
\]

where \( \lfloor \cdot \rfloor_{n \times n} \) 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 \( A_0^\ell \).
Output: pruned association field \( A_\ell \) and coherent ordering \( V_\ell \).

1) Initialization: set \( A_\ell = A_0^\ell \) 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

\[
x \leftarrow A_\ell(x), \quad W(x) \leftarrow 0.
\]

4) Set up the ordering: initialize the pool \( P = \{ x \} \).

While \( P \neq \emptyset \), do

- extract \( y \in P \), remove \( P \leftarrow P \setminus y \),
- set \( V_\ell(y) \leftarrow k, \quad k \leftarrow k + 1 \),
- add fathers \( P \leftarrow P \cup (\Phi(y) \cap \{ z \mid V_\ell(z) < 0 \}) \).

5) Stop: if \( \{ z \mid V_\ell(z) < 0 \} \neq \emptyset \), 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 \( 2^\ell \), the pixels are visited in an order \( \{m_0, m_1, \ldots, m_{n^2-1}\} \) such that \( V_\ell(m_i) = i \).

For each \( \ell \), this ordering \( V_\ell \) is computed to be maximally coherent with the association field \( A_0^\ell \). This means that a pruned association field \( A_\ell \) close to the raw field \( A_0^\ell \) is computed together with an ordering \( V_\ell \) that satisfies

\[
\forall x, \quad A_\ell(x) \neq x \implies V_\ell(A_\ell(x)) < V_\ell(x).
\]

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

The computation of such a couple \( (A_\ell, V_\ell) \) is performed using a greedy algorithm detailed in Table 1. This algorithm
removes connections in the field $A_t$ that cause loops in the graph $x \rightarrow A_t(x)$.

**B. Grouplet Transform**

A grouplet tight frame

$$
B(\Gamma) = \{ b^\Gamma_{\ell,m} \mid m \in \{0, \ldots, n-1\}^2, \ell = 0, \ldots, L \}
$$

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

$$
\forall \ell, \forall m, \quad d^\ell_m = \langle f, b^\Gamma_{\ell,m} \rangle. \tag{5}
$$

The grouplet vectors $b^\Gamma_{\ell,m}$ are defined implicitly through a decomposition algorithm that directly computes 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 $2^\ell$. At a given scale $2^\ell$, 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 $\tilde{m} = A_t(m)$

$$
d^\ell_m = (\tilde{f}(m) - \tilde{f}(\tilde{m})) \sqrt{s(m)s(\tilde{m})} / \sqrt{s(\tilde{m})} + s(m), \tag{6}
$$

$$
\tilde{f}(\tilde{m}) = s(m)\tilde{f}(m) + s(\tilde{m})\tilde{f}(\tilde{m}) / s(\tilde{m}) + s(m), \tag{7}
$$

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

The weight $s(m)$ balances the contribution of each pixel to ensure conservation of energy, even in the case where two flows $A_t(m) = A_t(m')$ 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 $f$ 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

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

see [1]. The values of $\{ d^\ell_m \}_{\ell,m}$ are the coefficients of $f$ in a tight frame $B(\Gamma)$, 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 $\ell = 0, \ldots, L - 1$,

for each $m$ with increasing $V_\ell(m)$, define $\tilde{m} = A_t(m)$, compute $d^\ell_m$ using (6),

update $\tilde{f}(\tilde{m})$ and $s(\tilde{m})$ according to (7), (8).

**Normalize coarse scale:** $\forall m, d_L(m) = f(m) / \sqrt{s(m)}$.

**Table 2:** Fast forward grouplet transform algorithm.

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

$$
v_1 = \tilde{f}(\tilde{m}) - \frac{d_L(m)\sqrt{s(m)}}{\sqrt{Ss(\tilde{m})}}, \quad v_2 = \tilde{f}(\tilde{m}) + \frac{d_L(m)\sqrt{S}}{\sqrt{s(\tilde{m})s(m)}},
$$

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

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

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

$$
\tilde{f} = \sum_{0 \leq \ell \leq L} \tau_\ell \sum_m d^\ell_m b^\Gamma_{\ell,m},
$$

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 $\{ A_\ell, V_\ell \}_\ell$.

**Output:** image $f = \tilde{f}$.

**Initialization:** set $\tilde{f}(m) = d_L(m) / \sqrt{s(m)}$.

**Iterations:** For each $\ell = L - 1, \ldots, 0$,

For each $m$ with decreasing $V_\ell(m)$, define $\tilde{m} = A_t(m)$.

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 $b^\Gamma_{\ell,m}$ is computed by applying the backward transform to a set of coefficients $d^\ell_m(m') = 0$ 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 $B(\Gamma)$ 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.
such a tight frame reads

\[ f = \sum_{j,\ell} \tau_{j,\ell} \sum_m d_{j,\ell}(m)b_{j,\ell,m}^\Gamma \]  

(11)

where \( \tau_{j,\ell} = \kappa_j \tau_{j} \).

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 \( B^\Gamma(\Gamma) \). 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

\[ y = \Phi f + w \]  

(12)

where \( y \) is the image with missing pixels, \( w \) is an additive noise, and \( \Phi \) is defined as

\[ (\Phi f)(x) = \begin{cases} 0 & \text{if } x \in \Omega, \\ f(x) & \text{if } x \notin \Omega. \end{cases} \]  

(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

\[ f^* = \arg\min_{g \in \mathbb{R}^n} \frac{1}{2} ||y - \Phi g||^2 + \lambda \sum_k |\langle g, \psi_k \rangle|, \]  

(14)

where \( \lambda \) 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 \( B^\Gamma(\Gamma) = \{b_{j,\ell,m}^\Gamma\}_k \), the sparse regularization (14) reads

\[ f^* = \arg\min_{g \in \mathbb{R}^n} \frac{1}{2} ||y - \Phi g||^2 + \lambda \sum_{j,\ell,m} |\langle g, b_{j,\ell,m}^\Gamma \rangle|. \]  

(15)

The difficulty is that the geometric layer \( \Gamma \) is only partly known outside \( \Omega \) and in the vicinity of the boundary \( \partial \Omega \). The flowing section details an iterative algorithm that computes an optimized geometric layer over large missing regions.
where $S$ iteratively thresholding algorithm to solve analysis [21]. For orthogonal bases, it is equivalent to the image on a set of atoms, see [62]. Such an iterative scheme where minimizes (15) by updating an estimate $f$ sparsity of the solution as the iterations solve an analysis prior that measures the noiseless case. During the iterations, see [21].

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

**B. Inpainting Algorithm**

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

$$f^{(i+1)} = S^\Gamma_{\lambda}(\bar{f}^{(i)})$$  

where $S^\Gamma_{\lambda}$ soft thresholds the decomposition (11) in the tight frame $B^\Gamma(\Gamma)$

$$S^\Gamma_{\lambda}(g) = \sum_{j,\ell} \tau_{j,\ell} \sum_m s_\lambda(\langle g, b^\Gamma_{j,\ell,m} \rangle) b^\Gamma_{j,\ell,m}$$  

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

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

These iterations solve an analysis prior that measures the sparsity of the solution as the $\ell^1$ 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 $\ell^1$ regularization, see [61].

Similarly to the best basis optimization proposed by Peyré [63], the geometric layer $\Gamma$ is updated iteratively during the iterative thresholding algorithm to improve the sparsity of the solution. Table 4 details the corresponding algorithm. The tolerance parameter $\eta$ controls the number of iterations. If there is no noise, $w = 0$, the threshold $\lambda$ 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 $\Omega$ 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].

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

**Table 4: Grouplet inpainting algorithm.**

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

![Fig. 7. Examples of inpainting with grouplets.](image)
the image, while the wavelets take 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-parametric 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 \( \Gamma_0 \) 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 \( B^w(\Gamma_0) \)

\[ \forall (j, \ell), \quad D_{j,\ell}(f_0, \Gamma_0) = \{ \langle f_0, b_{j,\ell,m}^\Gamma \rangle \}_{m} , \]

together with the set of pixel values \( D^{sp}(f_0) = \{ f_0(x) \}_{x} \). 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 \( (j, \ell) \) 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 \( \Gamma \) and a new set of coefficients. This section assumes that the geometric layer \( \Gamma \) is known. The synthesized image is drawn at random from a texture ensemble parameterized by \( \Gamma \). 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 \( \Gamma_0 \) and the coefficient layers \( D_0 \) have been learned, a grouplet texture ensemble \( T(\Gamma, D_0) \) is defined for any orientation field \( \Gamma \). It is the set of textures having the same wavelet-grouplet and pixels marginals as \( f_0 \)

\[ T(\Gamma, D_0) = \{ f \in \mathbb{R}^N \setminus \{ f \in \mathbb{R}^N \setminus D^{sp}(f) = D^{sp}(f_0) \} , \]

This texture ensemble is thus parameterized by the geometric layer \( \Gamma \) of the new texture and by the coefficient layer \( D_0 \) of the exemplar. The geometry \( \Gamma \) that defines the model might be different from the learned geometry \( \Gamma_0 \). 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 \( P(d_{j,\ell}, D_{j,\ell,0}) \)
of wavelet-grouplet coefficients \(d_{j,\ell} \in \mathbb{R}^N\) on
\[
\{d \in \mathbb{R}^N \setminus \{d(m)\}_m = D_{j,\ell,0}\}.
\]

The same projection algorithm applies to the pixel values constraints generated by \(D^p(f_0)\).

This projection corresponds to an histogram equalization of \(d_{j,\ell}\) with the values of \(D_{j,\ell},0\), see [65]. We denote by \(d'_{j,\ell}(i)\) 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
\[
P(d_{j,\ell}, D_{j,\ell},0) = \tilde{d}_{j,\ell} \quad \text{with} \quad \tilde{d}_{j,\ell}(r(i)) = d'_{j,\ell}(i).
\]

(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 \(T(\Gamma,D_0)\) has maximal entropy, thus leading to the least synthesis bias. Performing an unbiased 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 \(\mathcal{S}(f^{(0)}; \Gamma,D_0) \in T(\Gamma,D_0)\). The procedure iteratively decomposes the image into the wavelet-grouplet frame \(B^w(\Gamma)\), 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 \(D^p(f)\) 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.

**C. Geometric Layer Synthesis**

**Texture synthesis with a user-defined geometry** Figure 10 shows examples of synthesis with the same exemplar \(f_0\), meaning \(\Gamma = \Gamma_0\). The turbulent geometry of \(f_0\) is transferred to the synthesized texture \(f\).

![Figure 10. Texture synthesis using the same geometry \(\Gamma = \Gamma_0\) as the exemplar texture. Top: original \(f_0\); bottom: synthesized \(f\).](image1)

Figure 11 shows examples of synthesis with the same exemplar \(f_0\) and several user-defined flows \(\Gamma\) with an increasing complexity. 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 \(\Gamma\). This orientation field is turned into a smooth vector field using the following angle-doubling mapping
\[
\Gamma = (\cos(\theta), \sin(\theta)) \mapsto \beta(\Gamma) = (\cos(2\theta), \sin(2\theta)).
\]

(20)

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

2The corrals are 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].

3The texture on the top row is obtained from [40].
The resulting field $\beta(\Gamma)$ is smooth except 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 constrains 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 $\beta$. The corresponding geometry synthesis algorithm is detailed in Table 6.

**Table 6: Geometry synthesis algorithm.**

<table>
<thead>
<tr>
<th>Input: exemplar orientation field $\Gamma_0$.</th>
<th>Output: synthesized orientation field $\Gamma$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Initialization: set $\Gamma^{(0)} \leftarrow$ random, $i \leftarrow 0$, compute the marginals $\beta_i = { \langle \beta(\Gamma_0), \psi_{j,p} \rangle }_p$ of the wavelet coefficients of $\beta(\Gamma_0)$.</td>
<td></td>
</tr>
<tr>
<td>2) Wavelet transform: compute the wavelet coefficients of $\beta(\Gamma^{(i)}): \beta_j^{(i)}[p] = \langle \beta(\Gamma^{(i)}), \psi_{j,p} \rangle$.</td>
<td></td>
</tr>
<tr>
<td>3) Impose wavelet marginal: for each $j$, perform the histogram equalization (19): $\beta_j^{(i)} \leftarrow P(\beta_j^{(i)}; \beta_j)$.</td>
<td></td>
</tr>
<tr>
<td>(each of the two coordinates of $\beta_j^{(i)}$ and $\beta_j$ are equalized separately).</td>
<td></td>
</tr>
<tr>
<td>4) Backward transform: compute $\beta(\Gamma^{(i)})$ and thus $\Gamma^{(i)}$ from the wavelet coefficients $\beta_j^{(i)}$.</td>
<td></td>
</tr>
<tr>
<td>Set $\Gamma^{(i+1)} = \Gamma^{(i)}/|\Gamma^{(i)}|$.</td>
<td></td>
</tr>
<tr>
<td>5) Stop: while not converged, set $i \leftarrow i + 1$ and go back to 2.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 12 shows examples\(^4\) of synthesis that combine the algorithm of Table 6 for the synthesis of the geometry $\Gamma$ and the algorithm of Table 5 for the synthesis of the texture $f \in T(\Gamma, D_0)$. 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 $32 \times 32$, 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 $\sigma$ in (1) should be small, which creates instabilities in the synthesis process. The geometric layer $\Gamma$ captured with a small $\sigma$ 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.

\(^4\)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].

**VI. DYNAMIC TEXTURE SYNTHESIS**

A time dependent texture $f_t$ is obtained by computing a time evolution of both the geometric layer $\Gamma_t$ 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 $\{f_t\}_{t \geq 0}$ 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

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

with Neumann or periodic boundary conditions, where $\nu$ is the viscosity of the fluid and $\Delta$ 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 $\Gamma_t$ that is not oriented.

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

$$\tilde{v} = v - \nabla V \quad \text{where} \quad \Delta V = \nabla \cdot v.$$  (21)
The synthesized texture $\tilde{f}_t$ for $t > 0$ is obtained by solving an advection equation projected on the grouplet texture model $T(\Gamma_t, D_0)$ 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 $\tau$

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

where $\tilde{\nu}$ is the viscosity of the texture, that might be different from $\nu$. The projection $S(\cdot; \Gamma, D_0)$ 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

$$W_\Gamma(f) = \tilde{f} \quad \text{where} \quad \tilde{f}(x) = f(x - \tilde{\Gamma}(x)).$$

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

Texture mixing computes a dynamic texture $f_\tau$ for $t \in [0,1]$, where $(f_0, f_1)$ are visually similar to a pair of exemplars $(f_0, f_1)$. This is achieved by interpolating between the geometric layers $(\Gamma_0, \Gamma_1)$ and the coefficient layers $(D_0, D_1)$ to obtain new layers $\Gamma_t$ and $D_t$ for each $t$. Each coefficient layer is decomposed as $D_t = \{D_{j,\ell,k}(\Gamma_t^0) \}$ and

$$f_{t+1} = S(\tilde{f}_{t+1}^2 ; \Gamma_{t+1} / \left| \Gamma_{t+1} \right|, D_0)$$

Stop: while $\tau t < T$, go back to 2.

**Table 7:** Grouplet fluid texture synthesis.

synthesis is possible by coupling the synthesis with a fluid simulation or by interpolating between two grouplet 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**


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

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


