The Visual Computer manuscript No.
(will be inserted by the editor)

Jean-Michel Dischler · Florence Zara

Real-Time Structured Texture Synthesis and Editing Using Image - Mesh Analogies

Abstract We present a novel texture synthesis technique designed to reproduce at real-time frame-rates example texture images, with a special focus on patterns characterized by structural arrangements. Unlike current pixel-, patch- or texton-based schemes, that operate in image-space, our approach is structural. We propose to assimilate texture images to corresponding 2D geometric meshes (called texture-meshes). Our analysis mainly consists in generating automatically these meshes, while synthesis is then based on the creation of new vertex/polygon distributions matching some arrangement map. The output texture-image is obtained by rasterizing the previously generated polygons using graphics hardware capabilities, which guarantees high speed performance. By operating in geometry space instead of image / pixel-space, the proposed structural approach has a major advantage over current techniques: beyond pure texture reproduction, it permits us defining various tools, which allow users to further modify locally or globally and in real-time structural components of textures. By controlling the arrangement map, users can substitute new meshes in order to completely modify the structural appearance of input textures, yet maintaining a certain visual resemblance with the initial example image.

Keywords Texture · Synthesis · Editing

1 Introduction

Texture synthesis has proved to be a powerful tool for reproducing automatically and faithfully example texture images, and has thus been extensively studied during the past years. It has now reached an advanced degree of maturity. Beyond reproduction, new techniques now furthermore attempt to grant users more and more control over the synthesis process. These methods essentially focus on the control of feature positions and size, or on techniques to create consistent transitions among different textures (including texture mixing). However, semantic-related structural texture compositions have not been paid much attention yet, though this represents an important visual characteristic of many natural or artificial textures.

J.-M. Dischler
LSIIT-IGG, UMR CNRS-ULP 7005, Illkirch, France
Tel.: +33-3-90-244559 - Fax: +33-3-90-244455
E-mail: dischler@lsiit.u-strasbg.fr

F. Zara
LIRIS, UMR CNRS-UCBL 5205, Villeurbanne, France
E-mail: florence.zara@liris.cnrs.fr

In this paper, we propose a structural method designed to address this issue. Our motivation is to allow users a fast, faithful and automatic texture reproduction, but with an interactive control of structural texture compositions: not only shape and position of texture features / elements (called textons according to [11]), but also the way the textons are arranged with respect to each other. With our approach, users may, for instance, modify the structural arrangement of an input texture, while maintaining some visual resemblance with the corresponding example image, such as exchanging brick arrangements, but not the individual brick patterns. The structural appearance is controlled using an arrangement map, that can be extracted from images or freely designed by users.

Figure 1 depicts an example showing that our method covers well structural texture aspects. The top row shows on the left the input texture and on the right the corresponding texture-mesh extracted using image processing techniques. The second row illustrates synthesis: on the left, we show the used arrangement map and on the right the resulting texture synthesis (the synthesis uses the input example, the texture mesh and the arrangement map). Since on the second row, the arrangement map matches the initial arrangement of the example texture, we obtain a straight reproduction. The last row illustrates structural control: another arrangement has been substituted using a different arrangement map, thus modifying the structural composition of the texture. Although the arrangement is different, we maintain a certain visual resemblance with the original model.

To be able to provide efficient interaction tools, our synthesis technique must satisfy a strong constrain: it must run at interactive frame-rates. Our method guarantees such frame-rates by using graphics cards to accelerate the image generation process. More specifically, our technique consists in decomposing textures into sets of connected polygons, which are bounding individual textons. Once the textures have been expressed as 2D texton matching meshes, colors can be ignored. The synthesis operates entirely in geometric space: it consists in reproducing visually similar meshes coarsen by the supplied arrangement map. Once a new mesh has been synthesized (requires a few milliseconds), the corresponding texture image is finally generated in real-time by polygon rasterization.

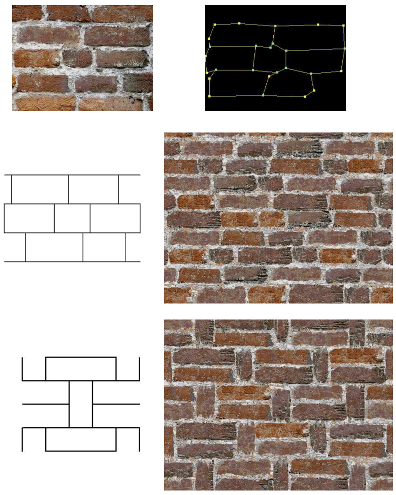

Fig. 1 Our synthesis is based on mesh extraction (top right) from input images (top left), in combination with arrangement maps (left) to control structural arrangements of resulting textures (right). In both cases, the textures were synthesized in a few milliseconds. Note how the second texture keeps a certain visual resemblance with the example image, though its structural arrangement is different.

The paper is organized as follows: next section briefly presents some related works. Section 3 then explains the pre-processing stage: the automatic generation of texture-meshes. Section 4 presents the synthesis of meshes using an arrangement map to control structural aspects. We propose a polygon fitting technique. Section 5 describes how to reconstruct texture images from the previously generated meshes. Finally, before concluding, we present some results, as well as a comparative study with existing synthesis techniques. As will be shown, our method, though not focusing on quality, compares well to current methods, but at a fraction of computational requirements. In addition, it considerably increases user control concerning structural composition. We further show that our technique not only applies to highly structured textures like brick-walls but also to random patterns such as lawns. The only condition that must be met, is that individual textons can be well identified.

2 Related Works and Motivation

Seminal texture analysis and synthesis methods were mainly based on histogram analogies using multi-scale or spectral approaches [9,3]. But such methods are strongly limited by the fact that they cannot deal with structured patterns. Alternate techniques, based on markovian processes, have then been proposed. Such techniques generate patterns pixel by pixel, by selecting at each step a color that minimizes an error according to a given neighborhood [6]. But the related "best pixel match search" may require some noticeable time in spite of proposed hierarchical data structures [24]. In addition, semantic-related structures are not well addresses with these methods. Another solution therefore consists in using complete texture pieces [17] instead. These can be randomly repeated and blended as in [22]. The quality of patch-based techniques depends on the types of overlap management. Blending, for instance, introduces some new frequencies (over-blurring), thus deteriorating visual aspects. Better results are usually obtained using a clever "cutting trajectory" along the overlap, computed according to an error-minimizing factor [5,13]. Recent improvements in the field of texture synthesis focus on recovering even better some feature-related aspects [26] by using additionally Laplacian filters. Other techniques focus on synthesis speed by separating analysis (pre-computation step) and synthesis [28] or by using the GPU [15].

Fast and faithful texture reproduction remained for a long time a major goal of texture synthesis, but lots of more recent computer graphics-oriented techniques, more and more include besides "random" / uncontrolled high quality reproduction also the possibility for users to change and constrain some visual aspects. In [2,23] different textures can be mixed. In [1] feature distributions can be controlled using a user-drawn feature probability map. In [4,29] feature sizes, orientations and so forth can be modified. In [19, 30], the difficult problem of smooth transitions between different types of textures is addressed. As for texture particles [4], [30] considers elementary texture components (textons) by using texton masks. We therefore call such techniques *texton-based*. With [30], users may also control other feature properties like size and orientation (using an underlying vector-field). More recently, a complete system has been proposed to design new textures from example texture databases [20]. In [15], a GPU implementation is proposed to produce controlled textures at very fast rates, which allows one to drag-and-drop individual textons at real-time rates. However, this technique uses a pixel-based approach thus failing to capture structural aspects. In [12] an optimization technique is used, which allows one to control the synthesis by using underlying flow fields. All of these techniques considerably increase the scope of texture synthesis, especially for computer graphics applications. They provide a wide range of tools, allowing users to design various effects beyond pure texture reproduction. However, none of these techniques ever considered semantic-related structural manipulations. Therefore, there remains an important limitation with respect to user control and free texture design from example images.

The motivation of our technique is to fill this gap. Unlike techniques operating in image-space, structural approaches have not been much studied yet, because they are known to be restrictive and / or technically more complex. In [14], for instance, the proposed structural method has been limited to specific types of textures such as brick wall patterns and woods. Our technique instead performs a full texture-mesh analogy, thus remaining generic. It is only based on the ability to segment and to identify textons in texture images. By using an arrangement map, the user straightforwardly controls the structural aspect of textures for synthesis.

2D meshes have been used before in the field of texture synthesis, but to our knowledge not for structural analysis. In [20], for instance, simplicial complexes are used to maintain sharpness along interpolations. In [18], meshes are used to evaluate the distortions of near-regular textures. Here, we extend this concept to characterize the actual structural composition of any type of texture, including irregular ones. In our case, we do not start from rectangular grids, but use image analysis to create 2D texture matching meshes. These are then coarsen to fit an arrangement map.

Our approach mainly extends the texture particles and texton masks approaches of respectively [4] and [30] by further bounding individual textons with polygons. Since our polygons may be considered to some extent as cells, our approach comes also close to cellular texture synthesis approaches, which have already been used successfully long before in the field of pure texture synthesis (that is, without analysis), for example for generating brick-wall patterns [21, 16] or noise functions [25]. Here, we apply this kind of structural classification to the field of texture reproduction and design from example image analysis.

3 Automatic Texture-mesh Generation

Our first objective is to generate automatically a 2D mesh well matching the structural composition of the input texture. This mesh represents a kind of geometric dual counterpart to the texture image. By viewing only the mesh, one should be able to recover the global structural appearance of the corresponding texture. This mesh will be later used, in conjunction with the arrangement map, to produce controlled structural arrangements.

Mesh reconstruction, for example by analyzing edges in images, is a widely studied area in the field of computer vision and digital image processing [7], therefore, we do not discuss all related topics in details here. Indeed, mesh extraction does not represent our core problematic. It rather represents a necessary pre-process.

We note that there exists a huge amount of work concerning more generally the creation of triangular, structured, unstructured, hexagonal, etc. meshes from image data (2D or even 3D). We found however that existing methods do not well adapt to the texture analysis and synthesis problem at hand. Therefore, we nevertheless briefly present the major aspects of the method we implemented for generating automatically texture matching meshes. To avoid too much details, we will assume that the reader is familiar with morphological operators, such as erosion, dilatation, thinning, and so forth.

As for texton-based techniques [4,30], the first step consists of texture segmentation, which means that we need to identify textons by creating a binary image $I^s(i,j)$ from the input texture image $I(i,j)$. In [30], such an image is called texton-mask. Figure 2 illustrates our segmentation: (a) represents the input texture $I(i,j)$ and (b) the segmented image $I^s(i,j)$.

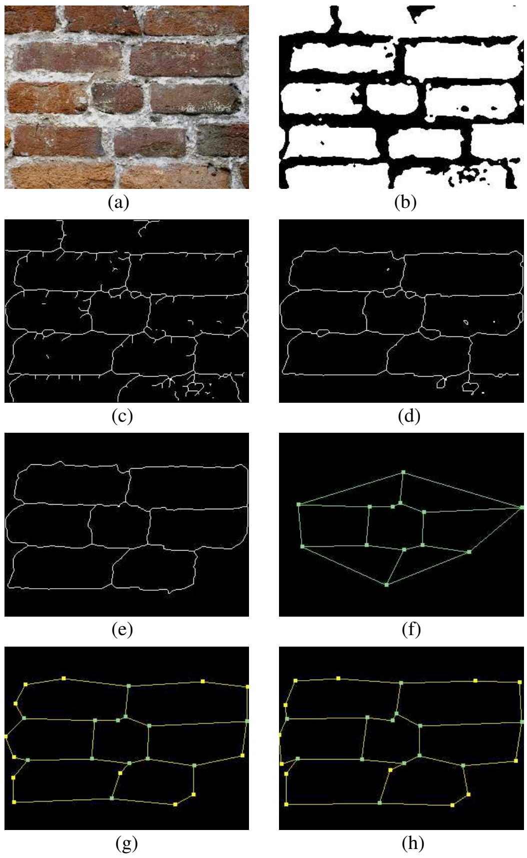

Fig. 2 Texture segmentation and automatic mesh reconstruction using basic image processing tools.

Gabor wavelets and windowed Fourrier transforms [8, 10] have had a wide success in the field of texture segmentation because they unify frequency and spatial analysis and have found to be well matching the human psycho-physical perception mechanism. In our case, we apply such filters to the input image, followed by a quantization.

In [30], the segmented images (texton masks) are straightforwardly used to control arrangements and to consistently mix couples of textures. In our case, we propose to use this mask to further build a texture-mesh (set of polygons). Such a mesh can be automatically and straightforwardly derived from $I^s(i,j)$ as described below. Figure 2 from (c) to (h) illustrate the different steps of the procedure. Firstly, we apply a thinning algorithm to the negative of $I^s(i,j)$, which consequently enlarges the textons, in such a way that these are separated by no more than one line of pixels (Figure 2.(c)). Indeed, thinning is a well-known morphological operation that reduces components in binary images to single pixel-wide branches, while preserving some properties [7]: it does not remove endpoints, it preserves connectedness, and it avoids excessive erosion of regions. Since segmentation often includes some noise, the next step consist in "cleaning" the image resulting from thinning by removing pending branches due for example to concave textons and by joining very close endpoints. This is again done by iteratively applying specific morphological erosion and dilatation operators. Figure 2.(d) illustrates the result of our cleaning technique. Using this result, one can identify individual cells that are matching some texton distributions. This image often (but not always) needs again to be processed to remove some remaining small residual features. Removal is performed by filling out very small cells, and by re-applying the same procedure. Figure 2.(e) shows the obtained result: we obtain a set of texton-matching cells defined by connected pixel-branches. Note that textons on borders (that is, textons which are incomplete) have been removed in our implementation (if necessary one could keep them). All of this process can be quite easily implemented and we found it to work very well. In fact, we experienced that the main difficulty was not cell generation, but rather to provide a good initial segmentation.

We call the image resulting from thinning and cleaning $I^c(i,j)$ because it identifies a set of texton-matching cells. Using the previously computed image $I^c(i,j)$, it is now possible to straightforwardly build a corresponding polygonal mesh. We first pick out branching cross-points, which represent the primary vertices of the texture mesh. In figure 2.(f) these are represented by green dots. We join these vertices by straight edges according to the branches of image $I^c(i,j)$. That is, two vertices are joint only if the corresponding cross-points are also linked together by one branch of pixels. Then, we introduce some new vertices by splitting some edges according to the shape of the corresponding branch. That is, if the straight segment is too distant according to a user selected threshold, we subdivide it to better fit the branch's curvature and shape. This is performed iteratively by introducing new vertices at positions that minimize the average distance of the resulting new edges from the corresponding branch. These new, secondary, vertices are depicted as yellow dots in figure 2.(g). They mainly appear on the borders of the outermost polygons.

Finally, this mesh is again processed to make it better fit the individual textons of $I^s(i,j)$. Indeed, some mesh edges may cross over the textons, which then results in apparent discontinuities during synthesis (see next sections). We therefore have to ensure that mesh edges do not cross over textons or at least to minimize such crossings. We apply an iterative procedure that progressively displaces vertices in order to minimize the amount of edges that cross over textons. The final resulting texture-mesh is shown on figure 2.(h). It is well matching the texton distribution of the input texture image. We note that resulting 2D-meshes are not regular, often also non-conformal and may contain polygons that have an arbitrary number of vertices (not necessarily the same number for each polygon).

4 Synthesis using Arrangement Maps

In the previous section, we described a technique to generate sets of connected polygons from given input texture images using segmentation and digital image processing. These polygons bound individual textons, so we call them texton-polygons. In this section, we show that meshes can be randomly reproduced to fit a given arrangement map. Our core problem is to be able to generate a new mesh that globally "matches" (from a visual point of view) this arrangement map, yet including some elements of the previously generated texture-mesh.

We propose a method taking into account two statistical elements: positions and shapes of polygons. We do so by applying consecutively two iterative procedures.

The first procedure creates a random arrangement map from a given periodic input arrangement map (figure 3), either extracted from example images or designed by users. The arrangement map is, as for the texture-mesh, composed of polygons. Each of these map polygons is composed of vertices, which are conformal or not. A non-conformal vertex is a vertex belonging to an edge of another polygon. Conformal vertices are exclusively edge extremities. The principle for producing random arrangement maps is straightforward: we first randomly displace vertices. Non-conformal vertices are only displaced along the corresponding edge. Then, we apply an iterative relaxation procedure, aiming at minimizing angular errors to respect initial angles of the input arrangement map. That is, vertices are again progressively displaced in order to match initial edge angles. The user can select the magnitude of randomness by providing a given magnitude coefficient. To keep a perfectly repetitive structure this coefficient may be set to zero.

Figure 3 (right) shows an input arrangement map (same as for figure 1), and shows a random perturbation (middle). Note that since all vertices, in this example, are non-conformal, we displaced them only along the corresponding edges, which explains that we keep a sort of stacked linear structure. The last image illustrates the result of relaxation after 50 iterations.

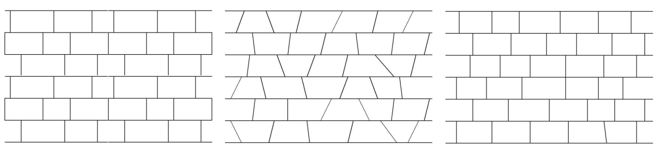

Fig. 3 Creating random arrangement maps from input arrangement maps using an iterative procedure.

The second procedure (figure 4) consists in fitting the texton-polygons into the previously generated arrangement map, which is followed by a second iterative procedure consisting in relaxing the resulting mesh to more or less well respect initial texton-polygon shapes (edge angles). We do this in two steps. Firstly, we randomly select for each map-polygon a given texton-polygon. The randomly selected texton-polygon vertices are placed onto the edges and vertices of the map-polygon by following a clock-wise cycle and by optimizing distance ratios with respect to the polygon perimeters (ratios with respect to the global distance around the outside of the polygons). Secondly, the resulting polygon, which is now totally matching the map-polygon is relaxed by an iterative procedure to better fit texton-polygon edge angles. We note that this procedure allows us to make any texton-polygon fit any map-polygon. Even if the texton-polygon contains less vertices than the map-polygon it is possible to duplicate some vertices (considering there was a null distance edge). Shapes can be also very different. This is illustrated in figure 4. The top shows a texton-polygon (extracted from figure 1). The two next rows then illustrate the fitting procedure for two different map-polygons (on the left): a rectangle and a triangle. The final result of fitting is shown on the right after 10 relaxation steps.

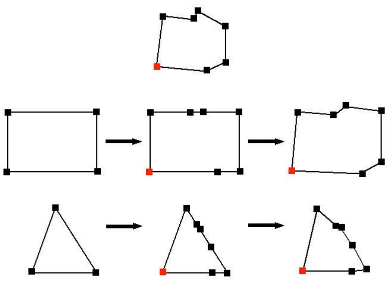

Fig. 4 Texton polygon (top) fitting technique: on the two bottom rows, the left shows a map-polygon, the middle the clock-wise vertex placement and the right the final result after relaxation.

The same procedure is applied to all map-polygons of the random arrangement map, thus obtaining a new arrangement matching texture-mesh. We call this new final mesh the synthesis-mesh. We now describe how the latter mesh can be used to create final texture images.

5 Texture Image Rasterization

The synthesis-mesh resulting from the previously described fitting technique could be, at first glance, straightforwardly used to create texture images. Indeed, the mesh is composed of polygons, which represent basic graphical primitives supported by all current graphics cards. Hence, one may directly associate to each polygon a 2D-texture map with texture coordinates that match the initial input texture image $I(i, j)$, thereby letting the final image straightforwardly be generated by fragment rasterization. 2D texture mapping-based mesh-manipulation tools are commonly and broadly used in nearly all interactive painting and photo-editing systems (for example to do image morphing). However, such a straightforward mesh rasterization approach does not apply well to texture synthesis. Indeed, there are at least two undesirable visual effects resulting from 2D texture mapping: 1) there are visible seams on the borders of the polygons since two adjacent polygons in the synthesis-mesh might not have been adjacent in the original texture-mesh (thus resulting in discontinuities), and 2) patterns related to textons appear stretched or shrieked, which is due to resampling during rasterization.

To avoid these two undesirable effects, we propose a technique that still consists in using polygonal 2D-texture mapping, as supported by graphics cards, but adapted to consider both textons and sub-textures together. We now describe what we mean by sub-textures and how to define these.

To generate texture-meshes (section 3), we have used a binary segmentation based on color quantization. However, it is possible to segment any input texture into more than just two zones (black / white). Therefore, after filtering by Gabor wavelets or Fourrier masks, we select a quantization number n_q higher than just two. Such a quantization is equivalent to performing a pixel classification: each pixel is assigned a *class* by means of a number (an integer value between 1 and n_q).

We call $I^q(i, j)$, the image resulting from filtering and quantization. $I^q(i, j)$ is composed of sets of connected pixel components C^q_k (that is, each class $k \in [1, n_q]$ corresponds to one or multiple connected pixel sets). Intuitively, these components are clustering pixels that are belonging to patterns, which have similar filter responses. In other words, it represents a partitioning of I into visually similar zones, that we call sub-textures according to [27]. Each texton may now be composed of one or multiple sub-textures.

Figure 5 shows an example of quantization for the brick-wall example of figure 1. As visible on this figure, individual textons may be composed of multiple sub-textures (each color represents another sub-texture on this figure).

We can now generate large texture fields visually matching given sub-textures. On figure 5, we show an example of sub-texture field corresponding to the concrete between the bricks. The arrows show some of the connected components C^q_k that have been used to generate this sub-texture field (left of figure 5). The sub-texture fields are synthesized in a pre-process by using any existing texture synthesis technique. We used a quilting-like approach. For applying the latter, we straightforwardly use the input image $I(i, j)$ cropped by the corresponding connected component C^q_k.

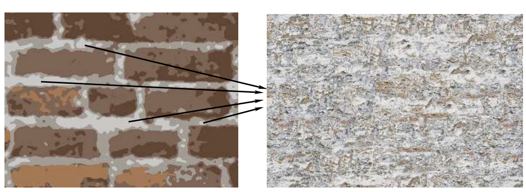

Fig. 5 Identifying sub-textures (left) and generating corresponding texture fields (right). Here the sub-texture corresponds to the concrete between the bricks of the brick wall texture of figure 1).

Once all fields have been generated (as for mesh generation, this needs to be done only once in a pre-process for a given input texture $I(i, j)$), they can be used in combination with $I^q(i, j)$ and with the texton map $I^s(i, j)$ to avoid the problems of seams and distortions appearing during rasterization.

The synthesis-mesh allows us to create texton distributions by using "traditional" texture mapping. But, instead of mapping straightforwardly the input image $I(i, j)$ as for usual painting systems we map the quantized image $I^q(i, j)$, further cropped by the texton-mask $I^s(i, j)$. Figure 6 illustrates the principles of our technique. The top shows traditional texture mapping on a brick texton cropped by its texton-mask. On the top right, we further show the same brick stretched horizontally, thus distorting underlying patterns. The two buttom rows illustrate our technique for an increasing amount of subtextures.

The quantized image $I^q(i, j)$ (left part of figure 6) is used for indexing the corresponding subtextures (it is used as indirection), see middle part of figure 6. That is, each polygon is actually rasterized with three 2D texture maps: the texton mask $I^s(i, j)$ used to extract only the pixels belonging to textons, the index mask $I^q(i, j)$ and the corresponding subtexture field. The texton mask and the index mask produce the shape of the texton (it is resampled according to the shape of the polygon) and the subtexture field the actual colors (small scale patterns). The result is given on the right part of figure 6. Note how global texton shapes can be deformed, without deforming subtextures.

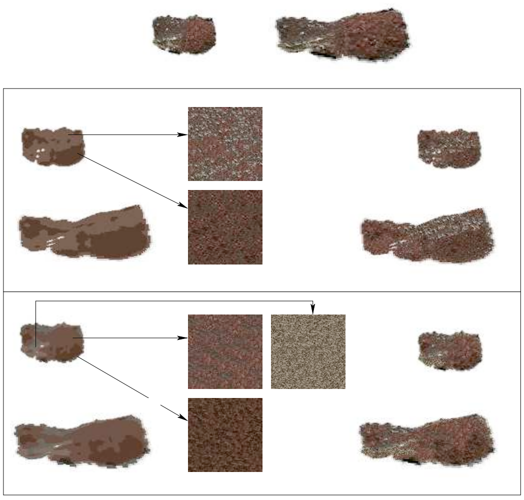

Fig. 6 Using multiple sub-textures (middle) for individual texture elements instead of traditionnal 2D texture mapping (top). We also show the effect of deforming the polygon and subsequently its texton (in this case, we stretched the brick horizontally).

Figure 7 illustrates a more complete example for the brick-wall. On this figure, we show, on the top, traditional texture mapping resulting in visible seams, since we have made polygons adjacent that where nor adjacent in the initial texture-mesh. Below, we show the corresponding texton mask (left part). And at right, we finally show the result obtained by applying our technique combining texton distributions with the corresponding sub-textures indexed by $I^q(i, j)$. A single rendering pass is necessary: firstly, we initialize the frame-buffer by copying the subtexture field corresponding to the background subtexture (in the case of the brick-wall, this is the concrete subtexture of figure 5). Then, each polygon is rasterized with its own index mask (a 2D texture map) used to access the corresponding subtexture fields as shown for one single brick in figure 6. For practical reasons all subtexture fields are fetched into texture memory once in the form of a 3D texture (the fields are simply stacked). The index value of the index mask then matches the Z coordinate in this 3D texture. In fact, using a 3D texture allows us to bind all sub-textures at once in texture memory.

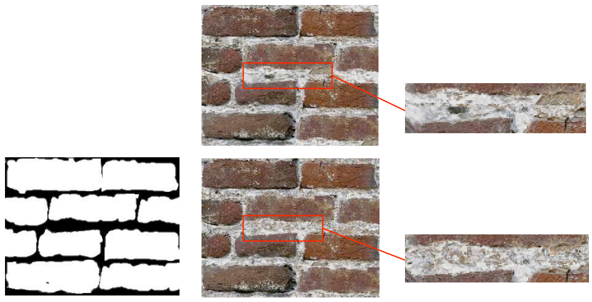

Fig. 7 Using texton masks and index masks to avoid seams: the top shows traditional 2D-texture mapping with seams and subtexture distortions. The bottom shows our result.

The interesting point with this approach is that texture distortions (according to the shape of the polygon) are only applied to the index and texton-mask $I^q(i, j)$ and $I^s(i, j)$ to modify the shape of the texton accordingly, but not to the sub-textures (see the stretched brick of figure 6). The reason is that we use two different texture coordinates, *e. g.* one for the masks and another for the subtexture. This allows us preserving subtexture frequencies. Note that instead of using a specific pixel-shader program and 3D texture, it is also possible to use the more simple multi-texturing functionality for rasterizing the polygons. However, in this case, multiple passes become necessary, especially if one texton is composed of multiple sub-textures (we need one pass for each subtexture).

We note that we generate sub-textures only if the size of the corresponding connected component C_k^q is large enough (we set a minimal size to 200 pixels). Indeed, we experienced that, if we use too small components, this results in very noisy subtexture fields, also producing final noisy results. When no subtexture field has been generated for a given texton, the previous procedure then simply indexes the original image $I(i, j)$ as for traditional 2D-texture mapping (yet, still using the texton-mask to avoid seams). Figure 6 (last row) illustrates this. On the left quantized image, one can actually see at least 5 classes. Hence, there should be also 5 subtextures. Yet, only 3 were computed as visible in the middle part of the figure. This is because the corresponding connected components were found to be too small.

For smoothly varying or non-stationay texton contents, the use of subtextures can cause incorrect results (we show this in the results section). In such cases, one must use a large number of classes, resulting in very small connected components. This, in turn, causes traditional texture mapping to be implicitly used (we do not compute subtexture fields if the components are smaller than a given number of pixels), which might result in visible pattern distortions.

To generate the final texture image from the previously generated synthesis-mesh, one simply has to rasterize each polygon using the texture mapping procedure that we just described. This is usually extremely fast (real-time) since graphics cards now reach high rasterization performances. Since the synthesis-mesh technique is also very fast (few iterations are usually sufficient) textures can be synthesized at interactive frame-rates. Users may also interactively edit the synthesis mesh by displacing vertices or by dragging some specific textons on some specific map-polygons.

6 Results

In this section, we present some results obtained with the previously described texture synthesis and editing technique.

Figure 9 illustrates an example of synthesis result. The top row shows from left to right : the input, the resulting cells and the corresponding mesh. The second row shows the arrangement map and the resulting synthesized mesh as well as the corresponding texton-mask. The last row is the resulting texture for a low (left) and high (right) amount of classes. Using a high amount of classes causes connected components to be very small, which results in nearly no computed subtextures.

Figure 10 illustrates a comparative study. The left column represents the model, the middle shows existing techniques (from top to bottom: texture quilting [5], the feature matching synthesis technique of [26], the parallel technique of [15] and the per-pixel jump map technique of [28]), the right column shows our result. Our objective was to compare both quality and speed. So, we took two good quality techniques and two high speed techniques. In our case, the synthesis took respectively from top to bottom 16ms, 123ms, 78ms and 57ms on a laptop with Pentium M processor 2.00 GHz and 1Gb RAM. The graphics hardware is a NVidia Quadro FX Go 1400. These timings include both the synthesis of the mesh and the rasterization. In all cases, the pre-processing time (required only once for a given texture) was below 5mn (this includes segmentation, texture-mesh generation and subtexture field synthesis). The number of computed subtexture fields was kept low (an average of 2 fields per texton, except for the second row where we used about 5 fields per texton, which also explains the somewhat increased noise). As demonstrated by this figure, our technique provides sufficiently good results at timings comparable to both [15] and [28]. Note that since both of these methods are based on per-pixel procedures they fail to capture well semantic-related structural aspects. Our method, on the contrary, is design to address structural textures, so it provides better results in these cases.

Figure 11 illustrates the effect of different arrangement maps on two different textures. The first row shows the input, the second one the reconstructed texture-meshes and the third one reproduction. The last two rows illustrate the influence of arrangement maps (depicted on the left most part). Note that arrangement maps can contain arbitrary polygons that do not necessarily need to be connected. For the three arrangements, the synthesis time for the lawn was between 16ms and 47ms, and for the panther texture between 14ms and 45ms.

Figure 12 shows some more examples. The right column shows the input, the second one reproduction and finally a new user designed arrangement.

The major limitation of our method is illustrated in figure 8. This figure shows textures that cannot be segmented into individual textons. In such cases, we cannot build texture-meshes and so the method simply cannot be used.

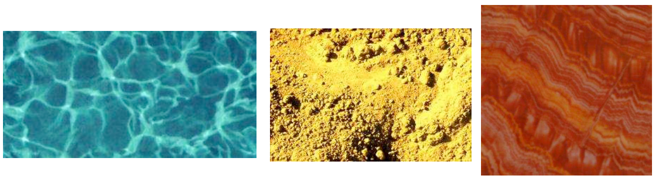

Fig. 8 Texture examples that cannot be processed with our technique, since textons cannot be segmented.

7 Conclusions and Future Work

We have presented a new structural approach for texture synthesis and editing. The method is based on a texture-mesh analogy, by associating to textures, sets of polygons bounding individual textons. It is adapted to textures that are characterized by strong structural components such as brick walls, tiles or lawns with individual flowers. The approach increases the manipulation possibilities while maintaining a certain visual consistency with the original texture. The technique furthermore processes textures at real-time rates as it uses standard polygon rasterization. We have shown examples of synthesis that compare in quality with other recent approaches.

Currently, the approach is not suitable for textures that are not characterized by an underlying texton-related structure. In our future works, we aim at addressing this issue. We believe that texture reproduction has now reached an advanced degree of maturity, and that efforts should be focused on increasing user manipulations, including the design of new structural aspects. We also intend to extend this method, in order to be able to edit and manipulate textures at interactive rates directly on arbitrary surfaces.

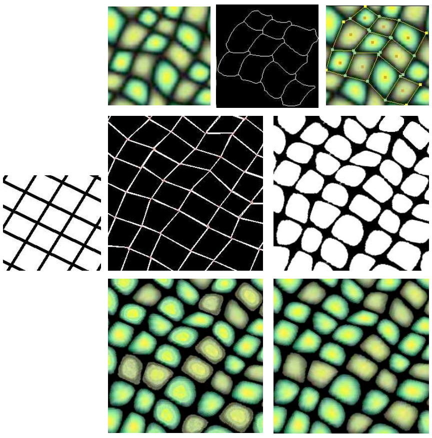

Fig. 9 An example of synthesis for different amounts of subtextures.

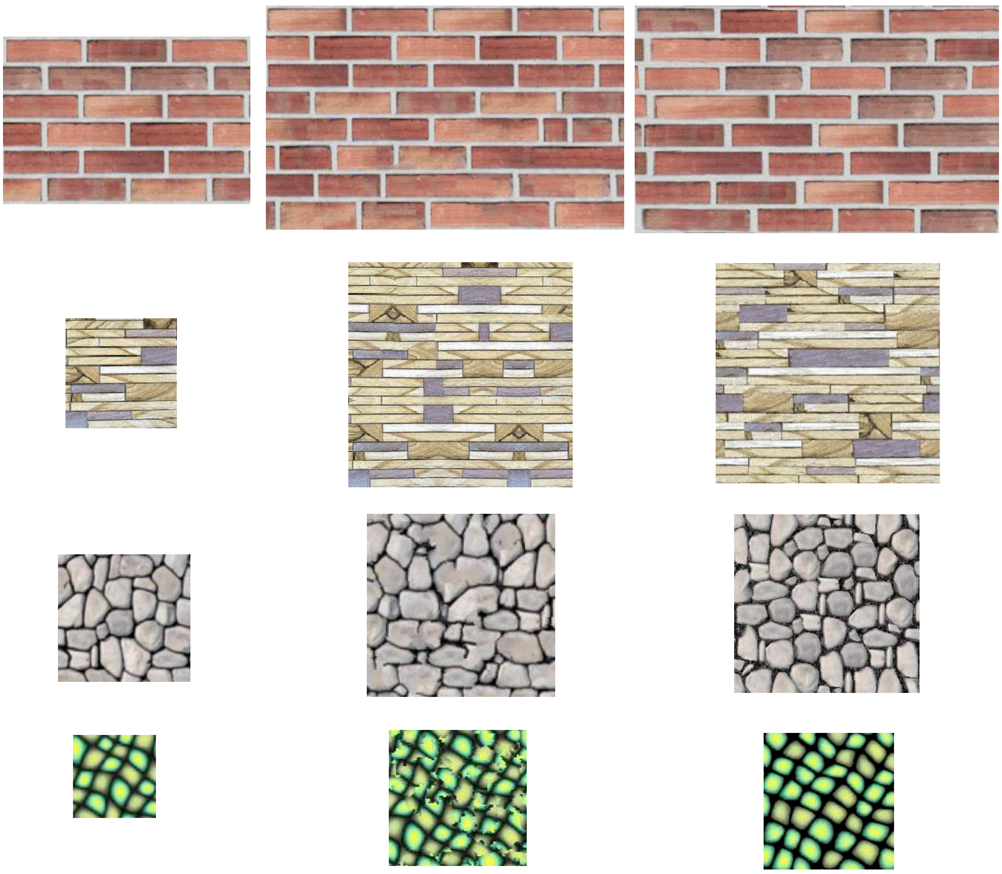

Fig. 10 Comparison with texture quilting [5], the feature matching synthesis technique of [26], the parallel technique of [15] and the per-pixel jump map technique of [28] (middle). Left is input, right is our result.

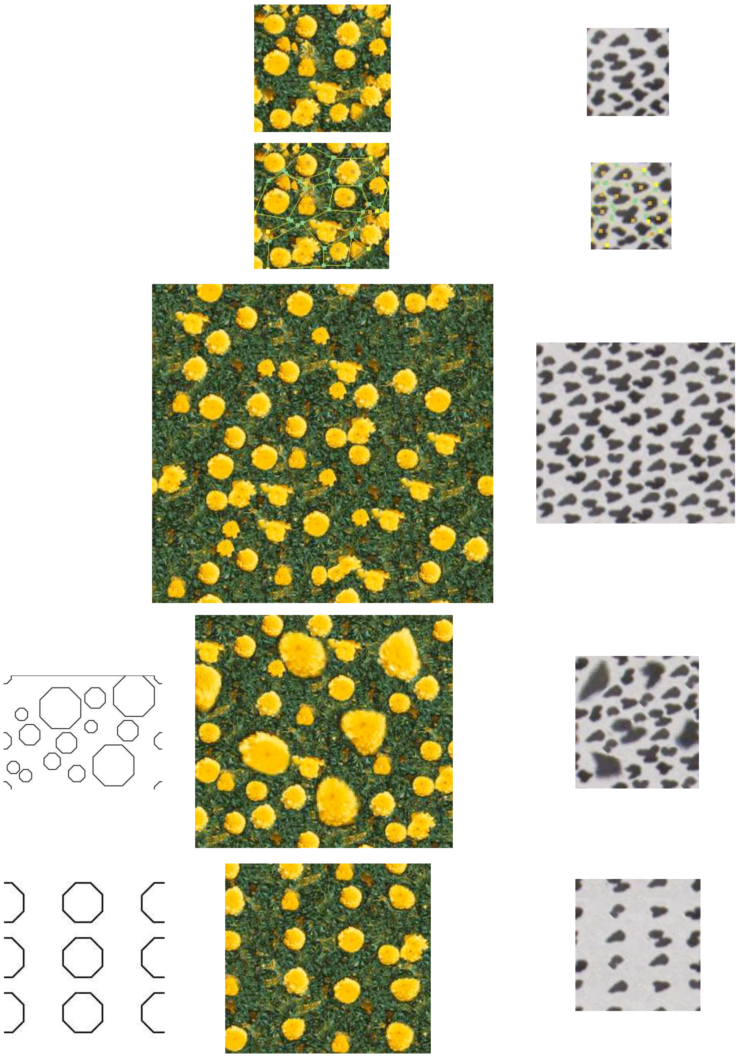

Fig. 11 Controlling arrangements using the arrangement map (left most part).

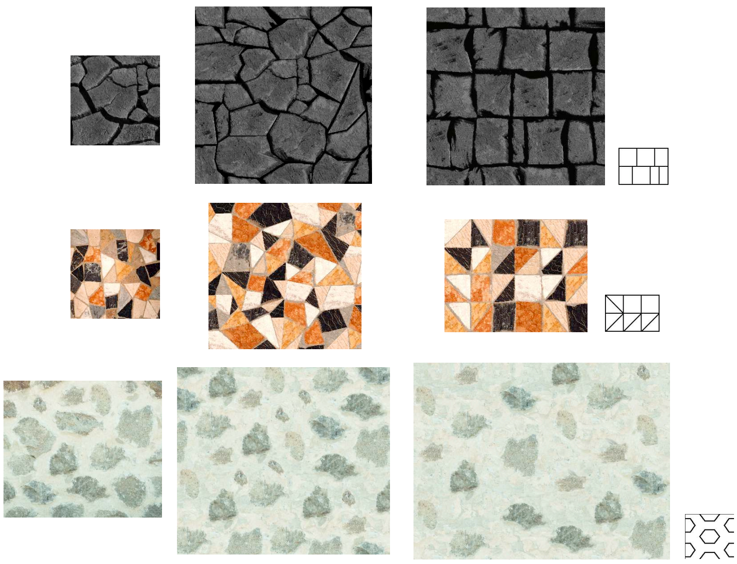

Fig. 12 More examples of synthesis and controlled arrangements.

References

1. Ashikhmin, M.: Synthesizing natural textures. In: SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 217–226. ACM Press (2001). DOI http://doi.acm.org/10.1145/364338.364405
2. Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., Werman, M.: Texture mixing and texture movie synthesis using statistical learning. IEEE Transactions on Visualization and Computer Graphics **7**(2), 120–135 (2001). DOI http://dx.doi.org/10.1109/2945.928165
3. Dischler, J.M., Ghazanfarpour, D., Freydier, R.: Anisotropic solid texture synthesis using orthogonal 2D views. Computer Graphics Forum **17**(3), 87–96 (1998)
4. Dischler, J.M., Maritaud, K., Levy, B., Chazanfarpour, D.: Texture particles. In: Eurographics 2002 - EG 2002, Saarbrcken, Germany (2002). URL http://www.loria.fr/publications/2002/A02-R-269/A02-R-269.ps
5. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 341–346. ACM Press (2001). DOI http://doi.acm.org/10.1145/383259.383296
6. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2, pp. 1033–1038. IEEE Computer Society (1999)
7. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1992)
8. Grigorescu, S., Petkov, N., Kruizinga, P.: Comparison of texture features based on gabor filters. IEEE Trans. on Image Processing **11**(10), 1160–1167 (2002)
9. Heeger, D.J., Bergen, J.R.: Pyramid-based texture analysis/synthesis. In: SIGGRAPH, pp. 229–238 (1995)
10. Idrissa, M., Acheroy, M.: Texture classification using gabor filters. Pattern Recognition Letters **23**(9), 1095–1102 (2002)
11. Julesz, B.: Texton, the elements of texture perception, and their interactions. Nature **290**(5802), 91–97 (1981)
12. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. ACM Transactions on Graphics, SIGGRAPH 2005 **4**(3) (2005)
13. Kwatra, V., Schdl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. ACM Transactions on Graphics **22**(3), 277–286 (2003). DOI http://doi.acm.org/10.1145/882262.882264
14. Lefebvre, L., Poulin, P.: Analysis and synthesis of structural textures. In: Graphics Interface, pp. 77–86 (2000)
15. Lefebvre, S., Hoppe, H.: Parallel controllable texture synthesis. ACM Trans. Graph. **24**(3), 777–786 (2005)
16. Legakis, J., Dorsey, J., Gortler, S.: Feature-based cellular texturing for architectural models. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 309–316. ACM Press (2001). DOI http://doi.acm.org/10.1145/383259.383293
17. Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. ACM Trans. Graph. **20**(3), 127–150 (2001). DOI http://doi.acm.org/10.1145/501786.501787
18. Liu, Y., Lin, W.C., Hays, J.: Near-regular texture analysis and manipulation. ACM Trans. Graph. **23**(3), 368–376 (2004). DOI http://doi.acm.org/10.1145/1015706.1015731
19. Liu, Z., Liu, C., Shum, H.Y., Yu, Y.: Pattern-based texture metamorphosis. In: 10th Pacific Conference on Computer Graphics and Applications (PG 2002), pp. 184–193. IEEE Computer Society (2002)
20. Matusik, W., Zwicker, M., Durand, F.: Texture design using a simplicial complex of morphable textures. ACM Trans. Graph. **24**(3), 787–794 (2005)
21. Miyata, K.: A method of generating stone wall patterns. In: SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques, pp. 387–394. ACM Press (1990). DOI http://doi.acm.org/10.1145/97879.97921
22. Praun, E., Finkelstein, A., Hoppe, H.: Lapped textures. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 465–470. ACM Press/Addison-Wesley Publishing Co. (2000). DOI http://doi.acm.org/10.1145/344779.344987
23. Wei, L.Y.: Texture synthesis from multiple sources. In: GRAPH '03: Proceedings of the SIGGRAPH 2003 conference on Sketches & applications, pp. 1–1. ACM Press (2003). DOI http://doi.acm.org/10.1145/965400.965507
24. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 479–488. ACM Press/Addison-Wesley Publishing Co. (2000). DOI http://doi.acm.org/10.1145/344779.345009
25. Worley, S.: A cellular texture basis function. In: SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 291–294. ACM Press (1996). DOI http://doi.acm.org/10.1145/237170.237267
26. Wu, Q., Yu, Y.: Feature matching and deformation for texture synthesis. ACM Transactions on Graphics **23**(3), 364–367 (2004). DOI http://doi.acm.org/10.1145/1015706.1015730
27. Zalesny, A., Ferrari, V., Caenen, G., Gool, L.V.: Composite texture synthesis. Intern. Journal of Computer Vision **62**(1,2), 161–176 (2004)
28. Zelinka, S., Garland, M.: Jump map-based interactive texture synthesis. ACM Trans. on Graph. **23**(4), 930–962 (2004). DOI http://doi.acm.org/10.1145/1027411.1027413
29. Zhang, J., Zhou, K., Velho, L., Guo, B., Shum, H.Y.: Synthesis of progressively-variant textures on arbitrary surfaces. ACM Trans. Graph. **22**(3), 295–302 (2003). DOI http://doi.acm.org/10.1145/882262.882266
30. Zhang, J., Zhou, K., Velho, L., Guo, B., Shum, H.Y.: Synthesis of progressively-variant textures on arbitrary surfaces. ACM Trans. on Graphics (Siggraph) **22**(3), 295–302 (2003)

J.-M. Dischler is an associate professor at the University Louis Pasteur, Strasbourg (France) since 2001. He belongs to the 3D Computer Graphics Group, IGG (http://lsiit.u-strasbg.fr/sites/igg/), where he is supervizing the realistic rendering, simulation and scientific visualization research activities. He is also member of the recently created INRIA project CALVI (http://math.u-strasbg.fr/calvi/). His research interests include texturing, texture synthesis, natural phenomena, real-time rendering and volume rendering.

F. Zara is an assistant professor in computer science department at the University Claude Bernard, Lyon (France) since 2005. She received her Ph.D. degree in computer science in 2003 at the Grenoble National Institut of Polytechnic (INPG). In 2005, she joined the 3D Computer Graphics Group of the LIRIS Lab. (http://liris.cnrs.fr/). Her research interests lie in parallel physically based animation of complex scenes on cluster with medical applications like hadronthepary and learning simulator for medical gesture.