Feature Matching and Deformation for Texture Synthesis

Qing Wu **Yizhou Yu**
University of Illinois at Urbana-Champaign
{qingwu1,yyz}@uiuc.edu

Abstract

One significant problem in patch-based texture synthesis is the presence of broken features at the boundary of adjacent patches. The reason is that optimization schemes for patch merging may fail when neighborhood search cannot find satisfactory candidates in the sample texture because of an inaccurate similarity measure. In this paper, we consider both curvilinear features and their deformation. We develop a novel algorithm to perform feature matching and alignment by measuring structural similarity. Our technique extracts a feature map from the sample texture, and produces both a new feature map and texture map. Texture synthesis guided by feature maps can significantly reduce the number of feature discontinuities and related artifacts, and gives rise to satisfactory results.

CR Categories: I.3.7 [**Computer Graphics**]: Three-dimensional Graphics and Realism—color, shading, shadowing, and texture I.4.3 [**Image Processing**]: Enhancement—filtering, registration I.4.6 [**Image Processing**]: Segmentation—Edge and feature detection

Keywords: Image Registration, Oriented Features, Texture Warping, Distance Transforms

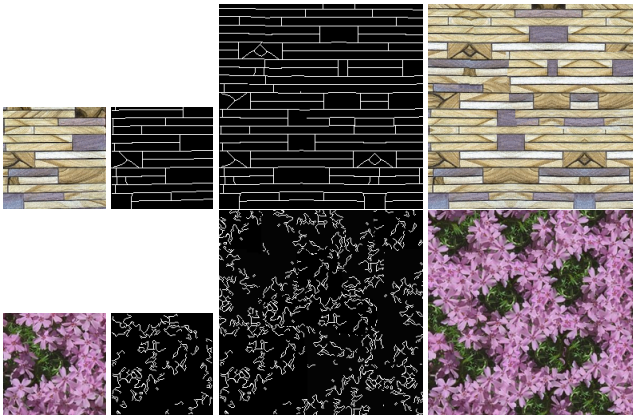


Figure 1: Texture synthesis with feature maps. From left to right: sample textures (128x128), feature maps of the sample textures, synthesized feature maps, output textures (256x256). Shown are FLOOR and FLOWERS©1995 MIT VisTex.

1 Introduction

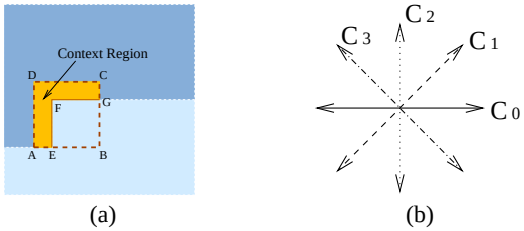
Texture synthesis has been widely recognized as an important research topic in computer graphics. Recently, neighborhood-based synthesis methods [Efros and Leung 1999; Wei and Levoy 2000; Ashikhmin 2001; Liang et al. 2001; Hertzmann et al. 2001; Efros and Freeman 2001; Zhang et al. 2003; Kwatra et al. 2003], especially patch-based techniques [Liang et al. 2001; Efros and Freeman 2001; Kwatra et al. 2003], have achieved significant progress. Nevertheless, the presence of broken features at the boundary of two adjacent patches is still a serious problem, though attempts have been made to alleviate it [Efros and Freeman 2001; Kwatra et al. 2003].

These neighborhood-based algorithms have two common stages: 1) search in a sample texture for neighborhoods most similar to a context region; 2) merge a patch or a pixel with the (partially) synthesized output texture. Dynamic programming [Efros and Freeman 2001] and graph cuts [Kwatra et al. 2003] have been used to optimize the patch merging stage. One problem with these algorithms is that the optimization schemes for the second stage may fail to find a smooth transition when the first stage cannot find satisfactory neighborhoods because of an inaccurate similarity measure.

We propose to perform texture synthesis using both salient features and their deformation. Note that not every pixel is equally important when we measure perceptual similarity. A good perceptual measure should account for the fact that the human visual system is most sensitive to edges, corners, and other high-level features in textures. We call these high-level features structural information. Structural similarity should be an important factor during neighborhood search. So far, summed squared differences (SSD) of colors is the most commonly used similarity measure between texture patches. It is not very good at capturing structural information. A desirable metric for structural information should estimate the minimal accumulated distance between corresponding features.

On the other hand, a small amount of deformation is less noticeable than visual discontinuities. Neighborhood search in texture synthesis remarkably resembles image registration [Zitova and Flusser 2003] in computer vision. Rigid template matching is the simplest of these registration methods. Deformable templates and elastic matching [Zitova and Flusser 2003] are not uncommon because object features and shapes may have different levels of distortion in different images. Deformation has also been considered for a subclass of textures with near-regular patterns [Liu and Lin 2003]. It is desirable to have deformable models for generic texture synthesis as well. The visual difference between two texture neighborhoods should reflect both color differences and shape deformations.

Among the large body of image registration techniques, of particular interest and relevance is chamfer matching [Barrow et al. 1977; Borgefors 1988] which was originally introduced to match features from two images by means of the minimization of the generalized distance between them. Inspired by chamfer matching, we develop a novel feature synthesis algorithm which considers structural similarity during feature matching and patch deformation during feature alignment. We also develop a hybrid method for texture synthesis by considering both features and colors simultaneously. In this method, features are used to guide and improve texture synthesis.



(a) (b)

Figure 2: (a) The L-shaped context region of a patch $ABCD$ being inserted into the output feature map. (b) The orientations of tangents are quantized into four intervals.

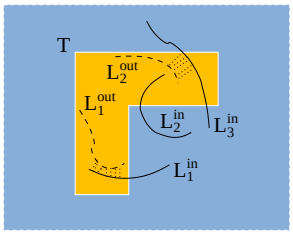


Figure 3: The edges, L_1^{out} and L_2^{out} (dashed), in the context region are being matched against the edges, L_1^{in}, L_2^{in} and L_3^{in} (solid), in the overlapping region of the input feature map. Pixels on edges L_1^{out} and L_2^{out} should be matched to those on L_1^{in} and L_3^{in}, respectively, because the tangents of L_2^{in} are not consistent with those of L_2^{out} even though it is closer to L_2^{out} than L_3^{in}.

2 Curvilinear Feature Matching and Synthesis

Curvilinear thin features, such as edges and ridges, provide the overall structural layout of textures. The set of curvilinear features of a texture can be represented as a binary image, which is called the *feature map* of the original texture. In this section, we introduce a simple but effective algorithm that synthesizes a new feature map from an existing one. Without loss of generality, we present this algorithm in the context of patch-based synthesis. Curvilinear feature detection will be addressed in Section 2.3. Examples of input and synthesized feature maps are given in Fig. 1.

2.1 Curvilinear Feature Matching

We consider two factors when performing feature matching: differences in both position and tangent orientation. Position is important since a smaller difference in position between matching features indicates a smaller gap between them. Consistent tangent orientation is critical to guarantee desirable geometric continuity and, therefore, visual smoothness between matching features.

Consider inserting a new patch in the output feature map. This patch has a causal L-shaped context region in the already synthesized region (Fig. 2(a)). The set of feature pixel locations in the context region is represented as $\{\mathbf{f}_i^{out}\}_{i=1}^m$. We simply translate the context region to all feasible locations in the input feature map when searching for a best match. The context region has an overlapping region in the input feature map. Its location is specified by the translation vector, $\mathbf{T} = (\Delta x, \Delta y)$, of the context region. The set of features in this overlapping region is represented as $\{\mathbf{f}_j^{in}\}_{j=1}^n$.

A matching cost between the two sets of aforementioned features typically requires the shortest distance between each feature, $\mathbf{f}_i^{out}$, in the first set and all the features in the second set. The feature in the second set that actually achieves this shortest distance can be defined as the corresponding feature of $\mathbf{f}_i^{out}$. We use a non-parametric mapping W_f to represent such a correspondence. Since our feature matching cost actually considers differences in tangent vectors in addition to Euclidean distance, directly computing W_f needs $O(mn)$ time. To avoid such an expensive computation, we only seek an approximate solution for W_f through a quantization of tangent orientation. The discrete quantization levels serve as "buckets", making it much faster to search for features with a desired orientation. A distance transform can be performed for features falling into the same quantization level. Such distance transforms make it possible to locate the nearest feature with a desired orientation in a constant amount of time.

The orientations of the tangents at $\{\mathbf{f}_i^{out}\}_{i=1}^m$ are uniformly quantized into four intervals (Fig. 2(b)). Each interval involves two opposite directions. Depending on which interval its tangent belongs to, we classify a feature pixel into one of four groups, $\{C_l^{out}\}_{l=0}^3$. We index the groups such that two adjacent groups have adjacent orientation intervals. C_0^{out} and C_3^{out} are considered adjacent. Similarly, the orientation-based classification for $\{\mathbf{f}_j^{in}\}_{j=1}^n$ is denoted as $\{C_l^{in}\}_{l=0}^3$.

Consider a feature $\mathbf{f}_i^{out}$ which has been classified into $C_{l_i}^{out}$. To maintain tangent consistency, only features in $C_{l_i}^{in}$, $C_{l_i-1}^{in}$ or $C_{l_i+1}^{in}$ are allowed as candidates for $W_f(\mathbf{f}_i^{out})$. $\mathbf{f}_i^{out}$ has a nearest feature in each of these three groups. These nearest features are denoted as $\mathbf{f}_{i_0}^{in}$, $\mathbf{f}_{i_{-1}}^{in}$ and $\mathbf{f}_{i_1}^{in}$, respectively, where $1 \le i_{-1}, i_0, i_1 \le n$. Our new distance metric between a pair of feature pixels is defined as

$$gdist(\mathbf{f}_i^{out}, \mathbf{f}_j^{in}) = \|\mathbf{f}_i^{out} - \mathbf{f}_j^{in}\|_2 + \tau\|\mathbf{v}_i^{out} - \mathbf{v}_j^{in}\|_2 \quad (1)$$

where $\mathbf{v}_i^{out}$ and $\mathbf{v}_i^{in}$ represent the tangents at $\mathbf{f}_i^{out}$ and $\mathbf{f}_i^{in}$, respectively, and τ indicates the importance of tangent consistency. In our approximate solution, $W_f(\mathbf{f}_i^{out})$ satisfies the following condition, $gdist(\mathbf{f}_i^{out}, W_f(\mathbf{f}_i^{out})) = \min_{k\in\{-1,0,1\}} gdist(\mathbf{f}_i^{out}, \mathbf{f}_{i_k}^{in})$. Fig. 3 illustrates the necessity of considering tangent consistency. For feature matching, we simply set $\tau = 0.1$ although further fine-tuning is possible.

To facilitate feature alignment, it is desirable to have a one-to-one mapping. Therefore, we define another quantity B_f over $\{\mathbf{f}_j^{in}\}_{j=1}^n$ to measure the bijectivity of the mapping W_f. $B_f(\mathbf{f}_j^{in})$ represents the number of different features in $\{\mathbf{f}_i^{out}\}_{i=1}^m$ that are mapped to the same feature $\mathbf{f}_j^{in}$. The matching cost between the two sets of features is defined to be dependent on the amount of distortion introduced by W_f and the bijectivity of W_f. It is formulated as

$$\frac{1}{m}\sum_i gdist(\mathbf{f}_i^{out}, W_f(\mathbf{f}_i^{out})) + \beta\frac{1}{n}\sum_j |B_f(\mathbf{f}_j^{in}) - 1| \quad (2)$$

where β is a positive weight designed to adjust the relative importance between the two terms. In all experiments, we use $\beta = 0.3$ if distance is measured in pixels. The translation vector $\mathbf{T}$ that minimizes the above matching cost indicates the optimal matching patch in the input feature map. At the end, we enforce a one-to-one mapping by removing extraneous corresponding features from the context region.

We precompute four distance transforms for the input feature map with a distinct transform for features in each of the four groups. Such distance transforms can be efficiently computed using the level set method [Sethian 1999]. At every pixel of the distance maps, we also store a pointer which points to the closest feature pixel. Meanwhile, every feature pixel in the input feature map has a counter indicating the number of features to which it corresponds in the current context region. With these data structures, evaluating the above matching cost between a pair of overlapping regions has a linear time complexity with respect to the number of feature pixels, which are often very sparse in the image plane.

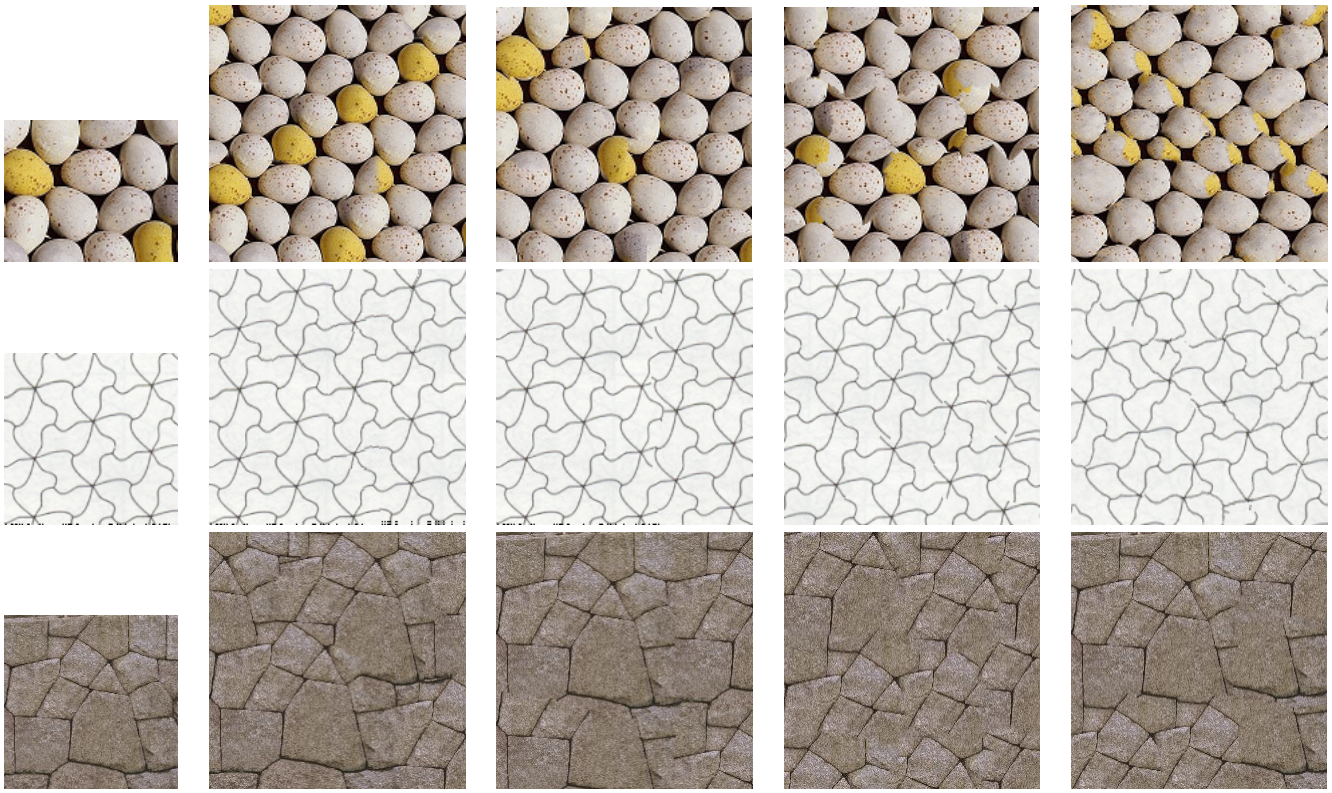


Figure 4: Comparison of our algorithm with Graphcut, Image Quilting and Texton Masks. The second column shows the results from our method. The graphcut result of the first example is courtesy of the authors of [Kwatra et al. 2003]; the results for the last two were generated using our own implementation. The quilting results for the first two examples were generated from our implementation of [Efros and Freeman 2001]; the result for the last one is courtesy of Efros and Freeman. The results using texton masks were from our implementation of [Zhang et al. 2003]. The samples shown are EGGS, PATTERN(©Branko Grünbaum and G.C. Shephard) and ROCK©1995 MIT VisTex. The PATTERN sample shown here is a slightly skewed version of an original periodic tiling. See additional comparisons in the DVD-ROM.

2.2 Feature Alignment Using Deformation

Even with the feature matching cost defined in the previous section, feature misalignments may remain especially when the input texture is aperiodic and contains no exact copies of the context region. Directly merging the optimal matching patch with the output feature map would produce discontinuities that may not be removed by the techniques in [Efros and Freeman 2001; Kwatra et al. 2003]. As an optional step, we explicitly remove feature misalignment by introducing a small amount of deformation in the image plane. This can be conveniently accomplished by deforming the matching patch using a smooth warping function.

We first compute a new feature mapping, W_f', to obtain a sparse feature correspondence between the context region and the optimal matching patch found in the previous section: $(\mathbf{x_i}, \mathbf{x}_i'), i = 1, 2, ..., m$. This mapping is in general different from the mapping, W_f, obtained during feature matching. In the current context, the mapping W_f' is computed using the same matching cost as in (2) but with τ in (1) set to 10 to emphasize tangents. A warping function should smoothly deform the optimal matching patch while moving each feature $\mathbf{x}_i'$ in this patch to the location of its corresponding feature $\mathbf{x_i}$ in the context region. Note that the optimal matching patch has four borders. To prevent the accumulation of deformations, we require that the pixels on the bottom and right borders be fixed during warping. These fixed pixels are also considered as constraints that the warping function should satisfy.

Obtaining the warping function is equivalent to scattered data interpolation. We apply two commonly used interpolation techniques: thin-plate splines (TPS) [Meinguet 1979; Turk and O'Brien 1999] and Shepard's method [Hoschek and Lasser 1993]. Thin-plate splines have the minimal bending energy among all interpolants satisfying the warp constraints. However, computing the thin-plate spline requires solving a linear system which may be occasionally ill-conditioned. Therefore, given a sparse feature correspondence, we first apply thin-plate spline interpolation. If the resulting warping function cannot satisfy the warp constraints, we switch to Shepard's method instead.

2.3 Feature Detection

In this paper, we only consider easy-to-detect features, such as edges and ridges. For edge detection, we first apply bilateral filtering [Tomasi and Manduchi 1998] to sharpen the edges. In the bilateral filter, the scale of the closeness function σ_d is always set to 2.0, and the scale of the similarity function σ_r is always set to 10 out of 256 greyscale levels. We then use finite differences along the two image axes as a simple gradient estimator to obtain an edge response at every pixel. This is followed by a two-pass classification. In the first pass, a global high threshold is used to detect strong edges which are usually broken into small pieces. In the second pass, a spatially varying lower threshold is used to detect weaker edges in the neighborhood of each strong edge. Unlike the lower threshold in the Canny detector [Canny 1986], it is locally dependent on the edge responses of the pixels detected in the first

pass. In practice, we choose a fixed ratio α. The lower threshold for a neighborhood surrounding a strong edge with response R is set to αR. The second pass can effectively connect broken strong edges. For ridge detection, we apply the Laplacian filter after bilateral filtering. Once there is a filter response at every pixel, the same two-stage classification for edges is also applied to ridges. For the results shown in this paper, α is always set to 0.3. The global high threshold is first estimated automatically using a fixed percentile of the highest filter response in the entire image. It can then be adjusted interactively by the user to improve the results.

Since the detected edges or ridges may have a multi-pixel width, we further apply a revised thinning algorithm which removes pixels with weak filter responses first while preserving the connectivity of the features. The resulting features always have one-pixel width. Detailed discussions on thinning algorithms can be found in [Pavlidis 1982]. At each detected feature pixel, we store its color and smoothed tangent as its attributes.

3 Feature-Guided Texture Synthesis

Our feature map synthesis is complementary to existing patch-based texture synthesis algorithms [Liang et al. 2001; Efros and Freeman 2001; Kwatra et al. 2003]. To incorporate feature maps into texture synthesis, we designed a hybrid method that generates a new texture and its feature map simultaneously given an input texture and its feature map. Every time we need to insert a new patch, we consider feature matching and color matching simultaneously. We apply the matching cost in (2) for features, and SSD for colors. The total matching cost is a weighted average between these two. Once a matching patch is chosen from the sample texture, it is deformed according to Section 2.2. Before merging the matching patch with the partial output texture, we apply graph cuts [Kwatra et al. 2003] to further improve the transition at the patch boundary.

Fig. 1 shows synthesis examples along with feature maps. As shown in the second row of Fig. 1, stochastic feature maps can also generate decent synthesis results. A comparison between our method and three other state-of-the-art algorithms is given in Fig. 4. It demonstrates that our method does very well at maintaining the continuity of structural features as well as the shapes of individual objects in the textures without merging them. More synthesis examples are given in Fig. 5, where the resolution of the output textures is 256x256. The running time for texture synthesis is less than two minutes on a 2GHz Pentium IV processor for sample and output textures at a resolution 128x128 and 256x256, respectively. The patch size is chosen between 32x32 and 64x64; the relative weight between feature and SSD color matching costs is always set to 0.5. For certain sample textures (FLOOR and FLOWERS in Fig. 1, LEAVES and WATER in Fig. 5), their rotated or reflected versions are also presented as input to our program to provide more texture variations.

4 Discussion

In this paper, we introduced a novel feature-based synthesis method that extends previous texture synthesis techniques through the use of local curvilinear feature matching and texture warping. When such oriented features are absent, our technique reverts back to the behavior of previously published color-based methods. Unlike texton mask extraction [Zhang et al. 2003] which needs manual intervention, it is easier to automatically obtain our feature maps. The feature map of a sample texture can also be improved with user interaction. Most importantly, new feature maps are synthesized using a novel matching criterion custom-designed for curvilinear features. This criterion is capable of guiding texture synthesis to produce better results.

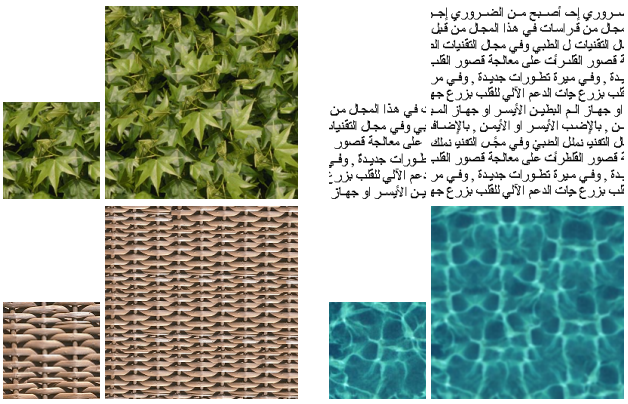


Figure 5: More texture synthesis results. The smaller images are the sample textures. Shown are LEAVES©Paul Bourke, ARABIC TEXT, BAMBOO©1995 MIT VisTex, and WATER.

Acknowledgments

We wish to thank the authors of [Kwatra et al. 2003] and [Efros and Freeman 2001] for sharing their results, Stephen Zelinka for proofreading, and the anonymous reviewers for their valuable comments. This work was funded by NSF (CCR-0132970).

References

ASHIKHMIN, M. 2001. Synthesizing natural textures. In *ACM Symposium on Interactive 3D Graphics*, 217–226.

BARROW, H., TENENBAUM, J., BOLLES, R., AND WOLF, H. 1977. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proc. 5th Intl. Joint Conf. on Art. Intell.*, 659–663.

BORGEFORS, G. 1988. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence 10*, 849–865.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Trans. Pat. Anal. Mach. Intell. 8*, 6, 679–698.

EFROS, A., AND FREEMAN, W. 2001. Image quilting for texture synthesis and transfer. In *SIGGRAPH'01*, 341–346.

EFROS, A., AND LEUNG, T. 1999. Texture synthesis by non-parametric sampling. In *Intl. Conf. Computer Vision*, 1033–1038.

HERTZMANN, A., JACOBS, C., OLIVER, N., CURLESS, B., AND SALESIN, D. 2001. Image analogies. In *SIGGRAPH'01*, 327–340.

HOSCHEK, J., AND LASSER, D. 1993. *Fundamentals of Computer Aided Geometric Design*. AK Peters, Ltd.

KWATRA, V., SCHODL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. In *SIGGRAPH'03*, 277–286.

LIANG, L., LIU, C., XU, Y., GUO, B., AND SHUM, H.-Y. 2001. Real-time texture synthesis using patch-based sampling. *ACM Trans. Graphics 20*, 3, 127–150.

LIU, Y., AND LIN, W.-C. 2003. Deformable texture: the irregular-regular-irregular cycle. In *The 3rd intl. workshop on texture analysis and synthesis*, 65–70.

MEINGUET, J. 1979. Multivariate interpolation at arbitrary points made simple. *J. Applied Math. Physics 5*, 439–468.

PAVLIDIS, T. 1982. *Algorithms for Graphics and Image Processing*. Computer Science Press.

SETHIAN, J. 1999. *Level Set Methods and Fast Marching Methods*. Cambridge University Press.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proc. Intl. Conf. on Computer Vision*, 836–846.

TURK, G., AND O'BRIEN, J. 1999. Shape transformation using variational implicit functions. In *SIGGRAPH 99 Conference Proceedings*, 335–342.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of Siggraph*, 479–488.

ZHANG, J., ZHOU, K., VELHO, L., GUO, B., AND SHUM, H.-Y. 2003. Synthesis of progressively-variant textures on arbitrary surfaces. In *SIGGRAPH'03*, 295–302.

ZITOVA, B., AND FLUSSER, J. 2003. Image registration methods: a survey. *Image and Vision Computing 21*, 977–1000.