To appear in SIGGRAPH 2005.

Texture Design Using a Simplicial Complex of Morphable Textures

Wojciech Matusik
Mitsubishi Electric Research Laboratories
(MERL)

Matthias Zwicker Frédo Durand
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory

Figure 1: Continuous interpolation along a path connecting four samples in the space spanned by our database. Morphable textures with sharpness preservation lead to an artifact free interpolation.

Abstract

We present a system for designing novel textures in the space of textures induced by an input database. We capture the structure of the induced space by a simplicial complex where vertices of the simplices represent input textures. A user can generate new textures by interpolating within individual simplices. We propose a morphable interpolation for textures, which also defines a metric used to build the simplicial complex. To guarantee sharpness in interpolated textures, we enforce histograms of high-frequency content using a novel method for histogram interpolation. We allow users to continuously navigate in the simplicial complex and design new textures using a simple and efficient user interface. We demonstrate the usefulness of our system by integrating it with a 3D texture painting application, where the user interactively designs desired textures.

Keywords: Texture Synthesis, Data-driven Models, Image Warping, Morphable Models

1 Introduction

Textures play a fundamental role in enhancing the complexity and realism of 3D models. They are usually defined using procedural modeling or input photographs. Procedural textures, e.g., [Perlin 1985; Ebert et al. 1994], provide great flexibility and allow for fine tuning of parameters to control the visual pattern. Unfortunately, they require programming skills that are out of the reach of most users. Furthermore, it is challenging to reproduce a realistic natural pattern. The use of photographs ensures realism but requires finding exactly the desired texture in the real world or performing tedious image retouching. The user might be able to find a number of textures that are similar to the wanted result but do not quite match. It is then appropriate to combine these textures to produce the desired output. In addition, textures often vary spatially due to natural processes [Walter et al. 2001; Zhang et al. 2003] such as weathering [Dorsey et al. 1996; Dorsey et al. 1999]. These effects are hard to achieve with photo editing software and are not trivial to implement as procedural textures.

We propose an alternative technique for modeling and designing a wide variety of natural textures. Our approach is data-driven, building upon a collection of photographic textures. We allow the user to combine these textures and continuously explore the space of textures induced by the dataset.

The structure of the space of all textures, however, is extremely complex as shown by research in human vision, e.g., [Richards and Koenderink 1995; Heaps and Handel 1999]. Quoting Heaps and Handel [1999], "there is no fixed set of dimensions that characterize natural textures." Traditional linear analysis and manifold embedding cannot be used to achieve our goal. The space of textures induced by our database is likely to be non-manifold; that is, various neighborhoods have different dimensionality. Hence, we apply a learning technique based on adaptive distance graphs [Giesen and Wagner 2003] to construct a simplicial complex [Ngo et al. 2000] of textures.

Our technique involves two main components. The first component is a morphable model that facilitates the interpolation of textures using a warp deformation. Our interpolation scheme also includes a technique to preserve high frequency content, or sharpness, in interpolated textures. The second component consists of a simplicial complex that represents the space spanned by the textures in our database. For this, we define a texture distance metric and construct the simplicial complex using a neighborhood graph based on texture distances. We describe a simple and efficient user interface to navigate in this space and produce new, natural textures as illustrated in Figure 1. We have also integrated our system with a 3D texture painting application, allowing the user to design desired textures interactively.

In summary, this paper makes the following contributions:

Data-driven texture model. We present a model for the space of textures induced by an input database. New textures are generated by the combination of input textures.

Simplicial complex. The structure of the induced space is captured by a simplicial complex where vertices represent input textures. We

allow interpolating input textures inside each simplex.

Morphable textures and distance metric. We propose a morphable interpolation for textures, which also defines a metric used to build the simplicial complex.

Sharpness preservation. We present a general method that guarantees sharpness in image morphing by enforcing histograms of high-frequency content. Appropriate histogram interpolation is achieved by interpolating the inverse cumulative histograms rather than the histograms themselves.

Navigation. We allow users to continuously navigate in the simplicial complex. Navigation relies on barycentric coordinates inside simplices. The user can move from simplex to simplex by removing and adding textures from a set of neighbors.

1.1 Previous Work

Our work draws from two research areas: texture synthesis and data-driven models.

Texture synthesis Texture synthesis seeks to generate textures of arbitrary size given a small texture sample. Several algorithms characterize textures according to distributions of multi-scale image properties [Heeger and Bergen 1995; Bonet 1997; Zhu et al. 1997; Zhu et al. 1998; Bar-Joseph et al. 2001; Portilla and Simoncelli 2000]. In non-parametric texture synthesis [Efros and Leung 1999; Hertzmann et al. 2001; Efros and Freeman 2001; Wei and Levoy 2000; Zelinka and Garland 2002], texture is synthesized one pixel (or one patch) at a time by finding pixels in the source sample similar to the already synthesized pixels. Brooks and Dodgson [2002] presented a related technique that uses similarity between texture pixels to facilitate editing. Wu and Yu [2004] improved patch-based texture synthesis using feature matching and patch deformation to reduce artifacts at patch boundaries. Their feature matching and texture deformation approach is similar to our warp computation. Epitomic representations [Jojic et al. 2003] encode images using a set of representative patches and mappings of those patches to the original image. Hence, patch based texture synthesis can be seen as the inverse process of constructing an epitome.

A number of authors have tackled the challenge of combining and mixing textures. Heeger and Bergen [1995] and Bar-Joseph et al. [2001] create novel textures that combine the multi-scale properties of different input textures. Wei [2001], Hertzman et al. [2001], Efros and Freeman [2001], and Kwatra et al. [2003] synthesize a non-uniform texture composed of homogeneous patches. Similar to our algorithm, the texture metamorphosis approach by Liu et al. [2002] is based on warp functions, but it requires the user to specify feature correspondences manually. Zhang et al. [2003] generate spatially-varying textures from two input textures. However, these approaches rely on a user to choose suitable textures that can be combined in a meaningful way. In contrast, we automatically construct a texture space that spans the range of textures induced by a database of natural images.

Liu et al. [2004] describe a system to analyze and manipulate photographic textures that allows a user to design novel textures. However, they focus on near-regular textures, whereas we strive to build a comprehensive texture model. Although they enable the user to modify structure and color of individual textures, they do not provide ways to compute sets of similar textures and interpolate their properties as we do.

Data-Driven Models Data-driven approaches offer powerful means to generalize the information present in input datasets. Linear approaches such as Principal Component Analysis are most popular, but they cannot account for more complex phenomena. Two main strategies have been proposed to introduce non-linearities in the analysis.

Morphable models were developed for face analysis and synthesis [Jones and Poggio 1998; Blanz and Vetter 1999]. A non-linear warp registers face images or 3D models to a reference face. Linear analysis is then performed on the shape described by the warp vector field and the registered images. Interpolation is carried out both on the shape and texture components.

In contrast to linear techniques, *non-linear dimensionality reduction* [Tenenbaum et al. 2000; Roweis and Saul 2000] permits the capture of spaces that are curved. However, it is usually assumed that the local dimensionality is constant (manifold assumption) and the topology is equivalent to a disk. Gomes and Mojsilovic [2002] alleviate these restrictions, but their variational approach is limited to low-dimensional data by high computational costs. Ngo et al. [2000] define non-manifold configuration spaces for animation using simplicial complexes. However, the topology of the space needs to be defined by the user. We use a similar representation but propose an automatic construction technique for arbitrary datasets.

Similarly to our work, some methods combine morphable and non-linear approaches. In Schödl et al.'s video textures [2000], optical flow permits a morph between pairs of data-samples (video frames). The graph connecting the frames can be seen as a simplicial complex with simplices of dimension one. Peters [2003] constructs morphable models for specific image domains (e.g., fish, dogs) using non-linear dimensionality reduction. His image correspondences are, however, defined manually. In our work, we build on these techniques to deal with non-manifold data sets and arbitrary dimensionality.

1.2 Paper Overview

In Section 2, we discuss the structure of texture space and point out the difficulties involved in building a data-driven model. In Section 3, we explain how textures can be interpolated using morphing and present our method for sharpness preservation. We also introduce a new distance metric which is used to define similarity between pairs of textures. In Section 4, we present our simplicial complex based on the pairwise texture distances and describe our navigation method. Section 5 presents results obtained using our model. Finally, we summarize the paper and describe possible directions for future work in Section 6.

2 Data-Driven Texture Model

To make textures amenable to mathematical analysis, we interpret discrete texture images as high-dimensional vectors. We unroll the pixels of each texture and concatenate red, green, and blue color channels to form a vector. However, vectors corresponding to natural textures occupy only a small portion of this high-dimensional space. It is our goal to model the subspace of natural textures such that a user can easily explore it and generate novel, valid textures.

Unfortunately, the space of natural textures does not form a linear subspace of the original high-dimensional space: Linear blending between two textures might result in an image that is blurry, which prevents the use of standard linear analysis such as PCA. In addition, the topology of the neighborhood of a texture varies from texture to texture. Some textures can be naturally blended with a large number of textures, while others have only a small number of natural neighbors. We must emphasize that the notion of perceptual texture similarity and the structure of texture space is challenging at best [Richards and Koenderink 1995; Heaps and Handel 1999]; it is highly dependent on the task or context. Perceptual studies have shown mixed results when it comes to parameterizing this space, and Heaps and Handel [1999] argue that "a dimensional model is inappropriate" for natural textures.

We propose to model the space of natural textures as a union of low dimensional convex sets (linear spaces) of different dimensions. This construction allows us to capture the topological and non-linear complexity of texture space and accurately represent the range of valid textures. Our work should be taken in the context of computer graphics: It is designed for the task of texture generation. The navigation and texture adjacencies we define allow a user to interpolate smoothly between textures and create the texture they need. While our data-driven model captures important aspects of the structure of texture space and texture similarity, we realize that perception and machine vision might involve additional concerns.

Our data-driven texture model is built from a database of about 1500 natural textures acquired from photographs. The textures in our database include a wide variety of phenomena such as building materials (e.g., bricks, stone), organic materials (e.g., wood, grass), and others. Furthermore, the textures were manually rotated and scaled to have approximately the same orientation and feature size. In our prototype, we represent each texture by a *texture sample* defined as a 128×128 texture patch. We make the patches tileable by using dynamic programming to find two pairs of optimal, non-straight patch boundaries in a vertical and horizontal overlap area around the patch [Efros and Freeman 2001]. Then we solve a Poisson equation with periodic boundary conditions to eliminate color seams [Pérez et al. 2003].

3 Morphable Textures

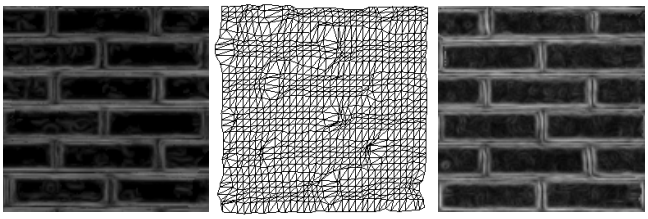
Our morphable texture model is designed to facilitate seamless interpolation of similar textures. In particular, we want to avoid ghosting artifacts that occur when strong features of the interpolated textures are misaligned. To address this, we use a warp function that optimizes feature alignment. The residual feature misalignment error after warping indicates the amount of artifacts during interpolation. We use this error as a texture similarity metric, which is central to the construction of our simplicial complex of textures.

In the following, we explain this approach in more detail: We describe feature extraction in Section 3.1 and computation of the warp function in Section 3.2. We show how to perform morphable interpolation between multiple textures in Section 3.3. Then, we explain how we use the warp functions and the morphing procedure to compute a pairwise symmetric similarity metric in Section 3.4. Finally, we introduce our technique for sharpness preservation in Section 3.5.

3.1 Feature Extraction

From a wealth of feature detectors, we chose the compass operator introduced by Ruzon and Tomasi [2001] to extract oriented edge features from the texture samples. The compass operator extends the notion of edges from discontinuities in color value to discontinuities in color distribution (texture edges). It is based on a circular filter window split into two half-windows along a line. This window is rotated, and for each orientation, the filter response is the difference in color distribution between the half-windows. The compass operator reports the orientation and strength of the maximum response at each pixel.

The compass operator has three parameters: the standard deviation σ of a Gaussian that is used to weight pixels in the circular window, the number of discrete orientations k at which the operator is evaluated at each pixel, and the maximum number of clusters c that are used to represent color distributions. For optimal results, the value of σ should be adapted to the noise level of the input image. Hence, we manually choose a value of either 1, 1.5, or 2 for each texture. The other parameters are fixed at $k = 30$ and $c = 10$ for all textures. In our approach, we use the strength of the maximum response for each pixel as a scalar feature map.



Feature map F_0 Warp W_{01} Feature map F_1

Figure 2: Visualization of the dense warp field W_{01} between two feature maps F_0 and F_1.

3.2 Warp Computation

Based on the per-pixel feature maps obtained with the compass operator, we determine warp functions that minimize the feature misalignment error between pairs of textures. Our approach to compute the warp functions is based on a coarse-to-fine search, with a regularization to ensure smoothness.

Let us denote the feature maps of a pair of textures (i, j) by F_i, F_j. For each point (x,y) in F_i, the warp function W_{ij} defines a corresponding point $(x,y) + W_{ij}(x,y)$ in F_j. To find W_{ij}, we perform a coarse to fine search as follows:

Starting at the coarsest scale, we overlay both feature maps F_i and F_j with a regular triangulation based on a square grid of vertices (each vertex has six neighbors). At the coarsest scale, the feature maps are subsampled to a resolution of 16×16 pixels and we use a triangulation of 8×8 vertices. Now, we exhaustively search for the optimal position of each vertex in the triangulation of F_j while keeping the vertices in F_i fixed. The cost of a vertex position is determined by warping each triangle of its one-ring neighborhood from F_j to F_i and computing the L_2-error of the per-pixel scalar feature strength. To ensure smoothness of the warp field, we also compute the 2×2 Jacobian of the affine mapping that relates a triangle in F_j to its corresponding triangle in F_i. As a measure of the induced space deformation, we compute the L_2 difference between the elements of this Jacobian and the identity matrix (i.e., the Frobenius norm). The deformation penalty for a vertex is given by the sum of these values for all triangles in its one-ring neighborhood. Finally, we use a weighted average of the L_2 feature error and the deformation measure as the cost for the vertex position. We have determined heuristically a suitable weighting factor that was used for all textures in the database.

At each scale, we iterate once over all vertices of feature map F_j and apply the above procedure to find their optimal positions. Then, we increase the resolution of the feature maps by a factor of two and subdivide the mesh by splitting each triangle into four. We repeat the search procedure and continue until the original resolution is reached.

Note that, because we work with tileable patches, we construct tileable warp fields by using modulo arithmetic at the patch boundaries. A visualization of the dense warp field between two brick textures is shown in Figure 2. Furthermore, it is straightforward to obtain the inverse mapping $W_{ji} = W_{ij}^{-1}$, which we will need in the next section.

3.3 Morphable Interpolation

Our morphable texture interpolation is based on a barycentric formulation of the morphing algorithm described by Jones and Poggio [1998]. In contrast to their approach, we do not work with a single reference texture. Warps are defined between pairs of images rather than with respect to a global reference image.

Each texture in the morphable model is represented by a *color* and a *shape component*. Color is given by the map $I : \mathbb{R}^2 \rightarrow \mathbb{R}^3$

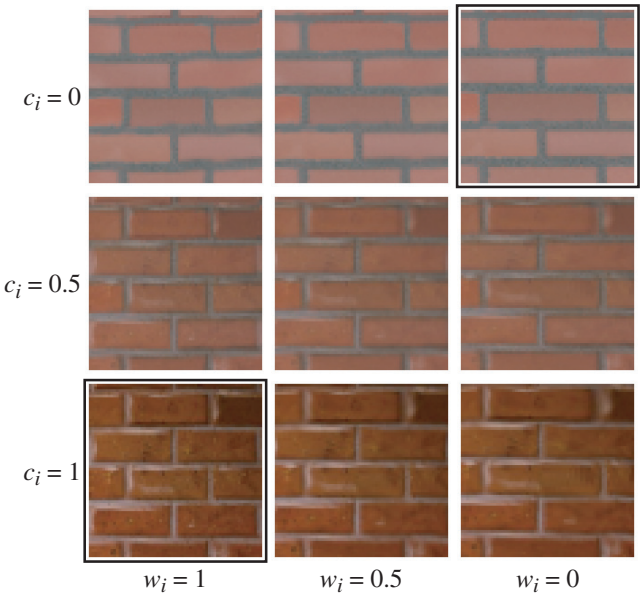


Figure 3: Interpolating between two textures I_i (bottom left) and I_j (top right) with independent weights for shape and texture.

that assigns a color value to each point. Shape is represented by the warp vector fields $W : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ establishing dense correspondences between the pixels of pairs of textures. Barycentric morphing between n textures I_i, $0 \leq i < n$, involves two steps: First, we linearly interpolate the warp vectors to produce intermediate warped versions of the input textures. Then, we linearly blend between these intermediate textures to get the output texture $\hat{I}$. This can be combined into a single expression

$$\hat{I}(x,y) = \sum_{i=0}^{n-1} c_i I_i \left((x,y) + \sum_{j \neq i} \left(w_j W_{ij}(x,y) \right)^{-1} \right), \qquad (1)$$

where c_i and w_j are separate weights for color and shape interpolation, respectively. We need to obey the constraint $\sum_j w_j = 1$ to guarantee that all the textures are aligned. Morphable interpolation between two textures is illustrated in Figure 3. The textures in the bottom left and top right corners are samples from the database, while the others are generated using different weights to interpolate shape and color. Results of interpolation between multiple textures are given in Section 5.

3.4 Warp-Based Texture Similarity Metric

Our texture similarity metric is based on the dense correspondences given by the warp field; it measures the residual error in the feature registration achieved by the warp. This metric is a good indicator for the amount of artifacts due to feature misalignment occurring in morphable interpolation (Section 3.3).

To define a symmetric distance, we warp the *feature maps* of a pair of textures (i, j) half-way. That is, we compute a pair of warped feature maps $\hat{F}_i$ and $\hat{F}_j$ by using the feature maps F_i, F_j instead of the corresponding texture images I_i, I_j in Equation 1. We choose interpolation coefficients $c_i = 1$, $c_j = 0$, $w_i = w_j = 0.5$, and $c_i = 0$, $c_j = 1$, $w_i = w_j = 0.5$, respectively. We then compute the similarity between textures i and j as the sum of squared differences between the warped feature maps $\hat{F}_i$ and $\hat{F}_j$. As in Section 3.2, we only take into account the difference between scalar feature strength and ignore feature orientation.

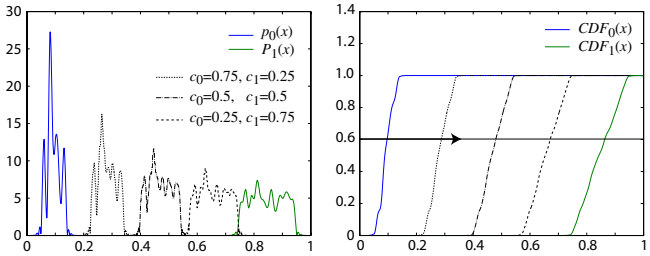


Figure 4: (Left) Interpolation of density functions $p_0(x)$ and $p_1(x)$ for different weights c_0 and c_1. (Right) We interpolate the inverse *CDF*s (i.e., we interpolate *CDF* values horizontally as indicated by the horizontal arrow); the interpolated density is obtained by differentiation.

3.5 Sharpness Preservation by Histogram Interpolation and Matching

The above warp succeeds in aligning major features, but blending still has an averaging effect that leads to a small amount of blurriness. In order to match the sharpness level of the input textures, we extract their high frequency statistics and enforce these statistics on the interpolated textures.

We capture high-frequency content using histograms of steerable pyramid coefficients [Simoncelli and Freeman 1995], and we achieve proper histogram interpolation using a novel technique based on interpolating inverse cumulative histograms. We preserve high frequency content in interpolated textures by enforcing interpolated histograms following the approach of Heeger and Bergen [1995]. Below, we focus on our novel technique for histogram interpolation. Please refer to the original work by Simoncelli and Freeman [1995] and Heeger and Bergen [1995] for more details on steerable pyramids and histogram matching,.

Histogram matching is based on the cumulative distribution functions (*CDF*) of the histograms. A desired histogram can be enforced by substituting all pyramid coefficients with value v with new coefficients v' given by

$$v' = CDF_{new}^{-1}(CDF_{old}(v)), \qquad (2)$$

where CDF_{new} is the *CDF* of the desired histogram, and CDF_{old} is the original *CDF*, which is obtained from the interpolated texture.

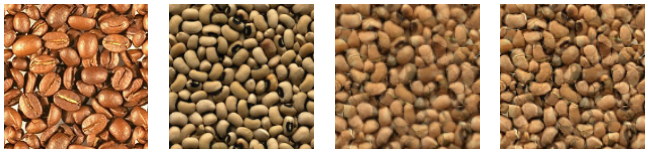
To preserve sharpness in the interpolated texture, we generalize this approach by setting the desired histogram to be an interpolation of the histograms of the input textures. This is achieved by linearly interpolating the inverse input *CDF*s:

$$CDF_{new}^{-1} = \sum_j w_j CDF_j^{-1}, \qquad (3)$$

where w_j are the shape interpolation weights and CDF_j are the *CDF*s of the input textures. This procedure is illustrated for a synthetic example in Figure 4. It leads to a more natural interpolation than directly averaging the histograms, smoothly morphing one histogram into the other.

Because we are interested in enhancing high frequencies in the interpolated texture, we only match the residual high-pass and the highest-frequency pass-band histograms of the pyramid decomposition. The enhanced interpolated texture is then reconstructed by collapsing the pyramid with the modified high-frequency bands. This is in contrast to Heeger and Bergen's texture synthesis algorithm, where histogram matching and image reconstruction is performed iteratively over all pyramid subbands and, in addition, the pixel color histograms.

Figure 5 illustrates our technique with a result obtained by enhancing the interpolation of two textures. In Figure 6, we point



(a) (b) (c) (d)

Figure 5: Morphable interpolation with sharpness preservation: (a), (b) input textures I_0, I_1, (c) morphable interpolation without sharpness preservation, (d) with sharpness preservation.

(a) (b) (c) (d)

Figure 6: Comparison of our technique to linear blending: (a), (b) input textures I_0, I_1, (c) linear blending, (d) our technique. Interpolation weights are $w_i = 0.5$, $c_i = 0.5$, $i \in \{0,1\}$.

out the improvements achieved over linear blending without morphing and sharpness preservation. Note that the procedure described above is general and can be used with any image morphing application.

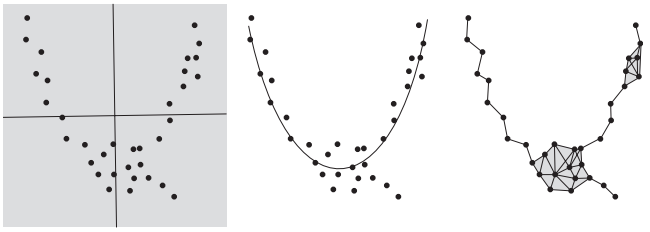
4 Simplicial Complex Modeling

The goal of our technique is to facilitate the exploration of the space induced by the input database and the design of novel textures. Since only similar textures can be combined in a compelling way, we use the distance metric described above to capture the topological structure of this space and guide morphable interpolation. As discussed in Section 1.1, we expect natural textures to have a complex structure in a high dimensional space as illustrated in Figure 7. However, standard linear and non-linear dimensionality reduction algorithms are not flexible enough to describe such datasets with non-manifold structure (Figures 7a and 7b). Therefore, we apply an alternative method for constructing a non-linear representation, which handles closed manifolds, manifolds with holes, and non-manifolds (see Figure 7c). The method we propose is general and can be applied to a wide range of data.

We model the space spanned by our dataset using an adaptive neighborhood graph, where each texture corresponds to a vertex in the graph. Two vertices are linked by an edge only if the distance between them obeys an adaptive distance criterion [Giesen and Wagner 2003]. Finally, we define the space that represents valid textures as the space inside all cliques of the graph. Our representation is a union of convex sets, or a simplicial complex of the graph.

To construct the graph, we start by computing a pairwise distance matrix between all texture samples. For each texture i, we introduce undirected edges to other nodes j if the distance $d_{i,j}$ obeys the local threshold $d_{i,j} \leq c \cdot d_{i,min}$, where $c > 1$ is a global constant and $d_{i,min}$ is the distance from node i to its closest neighbor. The advantage of this graph construction over other neighborhood graphs is that the connectivity parameter c is independent of the local dimensionality of the data points and adapts to variations in sample density [Giesen and Wagner 2003].

Our model for generating new textures is based on two assumptions: First, if there is an edge between two nodes in the graph, then morphable interpolation leads to a valid texture. Second, if n nodes in the graph form a *clique* (each of these nodes has an edge to all other nodes), then all convex combinations of these nodes produce valid model points. That is, we interpret an n-clique as an $(n-1)$-dimensional *simplex*. Each point in the interior of the simplex can



(a) Linear model (b) Non-linear model (c) Piecewise linear model

Figure 7: Comparison between data-driven models. In this example linear models fit a line or the whole 2D to the 2D data points (left). Non-linear manifold learning methods approximate these data points with locally linear spaces of a fixed dimension (center). Our method constructs a piece-wise linear model with variable dimensionality (right).

be specified using barycentric coordinates of the n vertices. A subset of m nodes that are shared by two cliques can be interpreted as an m-dimensional boundary face between the two simplices.

More formally, our simplicial complex model can be described as follows: In the case of textures represented by our morphable model from Section 3, a data point is defined by a shape weight vector $\mathbf{w}$ and a color weight vector $\mathbf{c}$. Both these vectors are in $\mathbb{R}^N$, where N is the number of samples in the dataset. The necessary condition for a data point to be in the simplicial complex model is therefore

$$\sum_{i=1}^{N} w_i = 1, \quad w_i \geq 0 \quad , \quad \sum_{i=1}^{N} c_i = 1, \quad c_i \geq 0, \tag{4}$$

and the set of indices $\{i \mid w_i > 0 \;\; or \;\; c_i > 0\}$ must form a clique.

4.1 Navigation

Given the above construction, we now describe how a user navigates in this space to design novel textures. To make our interface intuitive, we abstract from the underlying topological representation and facilitate smooth navigation in the space spanned by the model.

Our interface is centered around a set of *active textures* and a set of *neighbor textures* that are presented to the user visually. The active set always corresponds to a clique in the neighborhood graph. A texture belongs to the set of neighbor textures if it can be added to the active set without breaking the clique constraint.

To design new textures, the user interactively changes the shape and color weights of the active textures. The weights are normalized to obey the barycentric constraints of Equation (4) at any time. To move towards one specific sample in the active set, the user gradually increases both its color and shape weight. When the weights reach one, the sample is reproduced exactly.

The user can modify the active set to include desired source textures by adding and removing individual textures. To add a texture, one selects any texture from the neighbors of the active set. Textures are added one at a time, as the set of neighbor textures is affected by this operation and needs to be updated before the next texture can be selected. The user can continuously move from the currently interpolated texture towards the newly added texture by increasing its weights. On the other hand, one can remove active textures with zero shape and color weights, as these do not contribute to the currently interpolated texture. This will enlarge the set of neighbor textures and offer new textures to be added to the active set (Figure 8, left).

In the process of adding and removing textures from the active set, the user effectively hops from one clique in the neighborhood

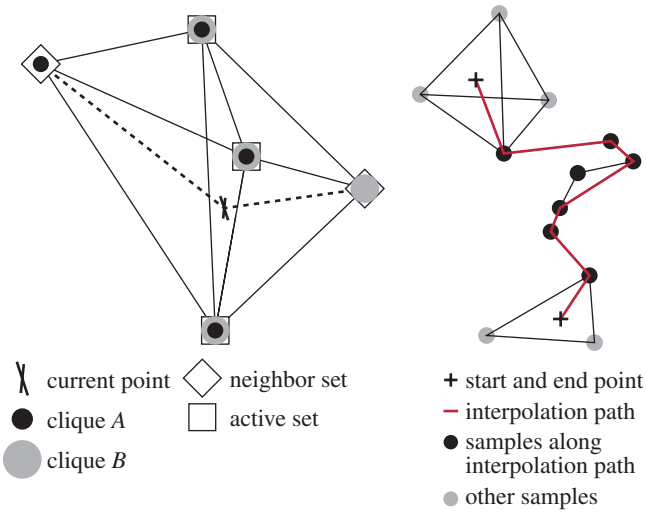


Figure 8: Given a current interpolation point, the user can move into a neighboring clique A or B by adding a texture from the neighbor set to the active set (left). Piecewise linear interpolation along the shortest path between two points (right).

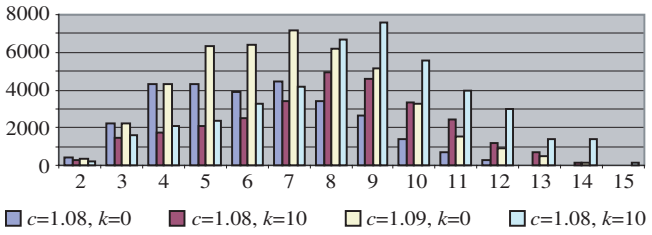


Figure 9: Histograms of clique sizes illustrating the connectivity of the simplicial complex.

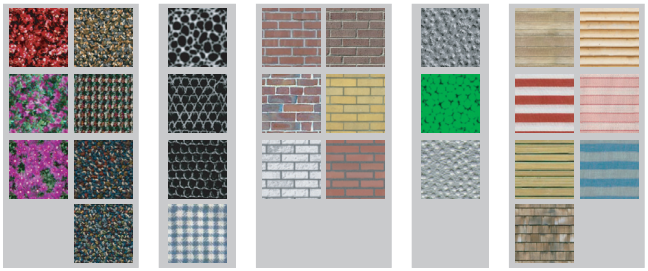


Figure 10: Examples of maximal cliques in the simplicial complex model with $c = 1.08$ and $k = 10$.

graph to the next. If the active set is a subset with m nodes of a maximal clique with n nodes, then the user has moved to a $(m-1)$-dimensional boundary of the $(n-1)$-dimensional simplex spanned by the maximal clique. Transparently to the user, adding another node moves the active set into any of the neighboring maximal cliques.

We can also generate new textures by continuously interpolating along a path between two arbitrary points in the model. In this scenario, the user picks a start and an end point, specified by their color and shape weights. We first determine the closest sample in the cliques containing each of the points and then compute the shortest path in the neighborhood graph that connects these samples, as shown in Figure 8 on the right. Textures along the path are then generated using Equation 1.

5 Results

We constructed a simplicial complex of morphable textures from roughly 1500 images. We first extracted texture samples (i.e., 128×128 pixel patches) from these images and added them one-by-one to the database. We manually picked one representative subwindow in each original photograph such that the orientation, scale, and translation matched with perceptually similar input samples already processed. This manual process took about one man-day for the whole database.

We then processed the samples as described in the previous sections, computing pairwise warps and the corresponding distance matrix. For each pair of textures, the warp computation (Section 3.2) and the evaluation of the similarity metric (Section 3.4) together take about five seconds. Computing the full 1500^2 distance matrix is not practical. Therefore, we adopted a two-stage procedure. In the first stage, we computed an approximation of the similarity metric for all 1500^2 pairs of textures, which is obtained by finding only the coarsest scale warps (Section 3.2) and evaluating the distance metric based on these. This took about eight hours on a single PC. In a neighborhood graph with reasonable connectivity, it is safe to assume that the number of neighbors for each texture is limited. We found that with our database it is not useful to connect each texture to more than 200 neighbors (see below). In the second stage, for each texture we calculated the high resolution warps only for the 200 most similar textures (according to the approximate similarity). We then updated the similarities based on the high resolution warps and constructed a sparse distance matrix with 200 entries for each texture. This stage took several days to compute on a single PC.

Figure 9 illustrates the connectivity of the simplicial complex model through histograms of the size of maximal cliques. We document results with an adaptive distance threshold of $c = 1.08$ and $c = 1.09$. In both cases, the model consists of a single connected component, and the maximum number of neighbors for any texture is below 200. We can also add edges to the k nearest neighbors of each texture to ensure that the user has a minimum number of neighbors to choose from. The disadvantage of increasing the graph connectivity is that the navigation interface becomes cluttered with textures that lead to artifacts when combined with morphable interpolation. We found that values $c = 1.08$ and $k = 10$ work best with our database. A number of representative maximal cliques for these parameters is shown in Figure 10.

Although our database consists of texture patches with 128×128 resolution, this does not limit the size and quality of images that we can produce with our system. We apply the following two techniques to alleviate this seeming limitation. First, we use tileable patches for the texture samples. Second, if the original input texture was of higher resolution than the database patch size, we retain the original image. During texture interpolation, we can use the original data and simply upsample the warp fields to the original resolution to obtain high quality results. Alternatively, any of the texture synthesis algorithms discussed in Section 1.1 could be applied to the interpolated sample.

We have implemented an interactive 3D texture painting prototype and integrated it with our user interface. Figure 11 shows a screen-shot of a painting session. We can create realistic effects such as wear and tear of cloth using the wide variety of textures in the database.

Figure 1 shows an example of a continuous interpolation along a path in texture space. Four samples are pairwise connected and piecewise linear interpolation is performed in each pair. A similar example with a set of 26 different textures connected to a path is given in Figure 12. This example highlights how a wide variety of textures can be connected and smoothly interpolated using simplicial complex modeling (Section 4). For all the results presented in this paper, we used tileable texture patches that were constructed as described in Section 2. We achieve spatially varying interpolation

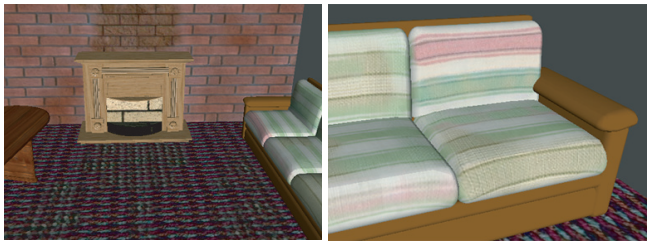


Figure 11: Interactive texture design using a prototype 3D painting application.

using manually specified weight maps as illustrated in Figure 13. Here, we set the color weights to be equal to the warp weights. The source texture samples are shown at the bottom.

6 Conclusions and Future Work

We have presented a novel approach for designing realistic textures based on a database of input textures. The main advantage of our model is a high degree of realism since the synthesized textures are combinations of natural textures acquired from photographs. Our system allows for both easy navigation in the space of textures spanned by the database and continuous interpolation between multiple textures. This is achieved by combining simplicial complex modeling, morphable models, and sharpness enhancement for interpolated textures. We also believe that our sharpness preservation technique and simplicial complex modeling could be used for generating models in other domains.

Our morphable texture interpolation is based on a single one-to-one warp deformation between pairs of texture samples, which might be too restrictive for textures with highly irregular structures. In the future, we will investigate epitomic image representations, which are based on collections of small patches and discontinuous mappings of these patches to the original image. However, to deal with these discontinuous mappings in the context of image interpolation seems challenging. Finally, it would be desirable to extend the texture model to incorporate reflectance as well (e.g., using bidirectional texture functions [Dana et al. 1999; Tong et al. 2002]).

Acknowledgments

We wish to thank Gabriel Lopez-Betanzos and Yang Ruan for their help with data collection and processing. Special thanks go to Kevin Der for implementing the painting application. We would also like to thank the internal reviewers at MIT and the anonymous referees for their valuable comments. This work was supported by an NSF CAREER award 0447561 “Transient Signal Processing for Realistic Imagery”, an NSF CISE Research Infrastructure Award (EIA9802220), a grant from the Shell, and a stipend by the Swiss National Science Foundation.

References

BAR-JOSEPH, Z., EL-YANIV, R., LISCHINSKI, D., AND WERMAN, M. 2001. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics 7*, 2, 120–135.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proceedings of SIGGRAPH 99*, Annual Conference Series, 187–194.

BONET, J. S. D. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of SIGGRAPH 97*, Annual Conference Series, 361–368.

BROOKS, S., AND DODGSON, N. 2002. Self-similarity based texture editing. In *Proceedings of SIGGRAPH 2002,*, Annual Conference Series, 653–656.

DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., AND KOENDERINK, J. J. 1999. Reflectance and texture of real world surfaces. *ACM Transactions on Graphics 1*, 18, 1–34.

DORSEY, J., PEDERSEN, H. K., AND HANRAHAN, P. M. 1996. Flow and changes in appearance. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, 411–420.

DORSEY, J., EDELMAN, A., LEGAKIS, J., JENSEN, H. W., AND PEDERSEN, H. K. 1999. Modeling and rendering of weathered stone. In *Proceedings of SIGGRAPH 99*, Annual Conference Series, 225–234.

EBERT, D., MUSGRAVE, K., PEACHEY, D., PERLIN, K., AND WORLEY. 1994. *Texturing and Modeling: A Procedural Approach*. Academic Press, Oct. ISBN 0-12-228760-6.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, Annual Conference Series, 341–346.

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, 1033–1038.

GIESEN, J., AND WAGNER, U. 2003. Shape dimension and intrinsic metric from samples of manifolds with high co-dimension. In *Symposium on Computational Geometry*, 329–337.

GOMES, J., AND MOJSILOVIC, A. 2002. A variational approach to recovering a manifold from sample points. *Lecture Notes in Computer Science 2351*, 3–17.

HEAPS, C., AND HANDEL, S. 1999. Similarity and features of natural textures. *Journal of Experimental Psychology: Human Perception and Performance 25*, 1–24.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH 95*, Annual Conference Series, 229–238.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *Proceedings of SIGGRAPH 2001*, Annual Conference Series, 327–340.

JOJIC, N., FREY, B., AND KANNAN, A. 2003. Epitomic analysis of appearance and shape. In *ICCV*, 34–41.

JONES, M. J., AND POGGIO, T. 1998. Multidimensional morphable models. In *ICCV*, 683–688.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3, 277–286.

LIU, Z., LIU, C., SHUM, H.-Y., AND YU, Y. 2002. Pattern-based texture metamorphosis. In *PG ’02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, 184–191.

LIU, Y., LIN, W.-C., AND HAYS, J. H. 2004. Near regular texture analysis and manipulation. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3 (August), 368 – 376.

NGO, T., CUTRELL, D., DANA, J., DONALD, B., LOEB, L., AND ZHU, S. 2000. Accessible animation and customizable graphics via simplicial configuration modeling. In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, 403–410.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3, 313–318.

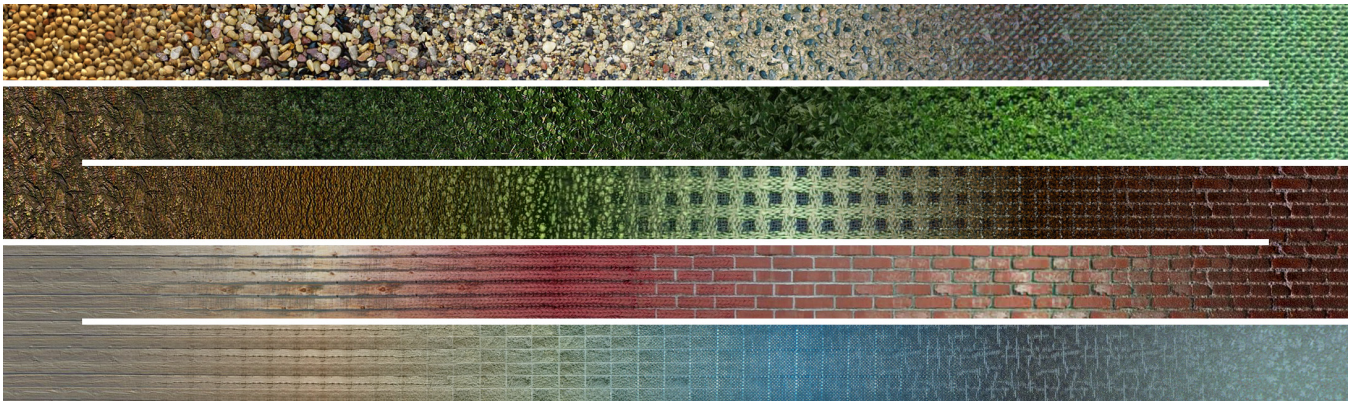


Figure 12: A path connecting 26 samples in the simplicial complex model.

Figure 13: Examples of spatially varying texture interpolation using manually painted weight maps.

PERLIN, K. 1985. An image synthesizer. In *Proceedings of SIGGRAPH 85*, Annual Conference Series, 287–296.

PETERS, M. R. 2003. *Multidimensional image morphs: construction and user interface*. Master's thesis, Massachusetts Institute of Technology.

PORTILLA, J., AND SIMONCELLI, E. P. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. Journal of Computer Vision 40*, 1 (Oct.), 49–70.

RICHARDS, W., AND KOENDERINK, J. J. 1995. Trajectory mapping ("TM"): A new non-metric scaling technique. *Perception 24*, 1315–1331.

ROWEIS, S., AND SAUL, L. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (December), 2323–2326.

RUZON, M., AND TOMASI, C. 2001. Edge, junction, and corner detection using color distributions. *IEEE Trans. on Pattern Analysis and Machine Intelligence 23*, 11 (November), 1281–1295.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of SIGGRAPH 2000*, Annual Conference Series, 489–498.

SIMONCELLI, E. P., AND FREEMAN, W. T. 1995. The steerable pyramid: a flexible architecture for multi-scale derivative computation. In *Proceedings of the 1995 IEEE International Conference on Image Processing*, 3444.

TENENBAUM, J., DE SILVA, V., AND LANGFORD, J. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (December), 2319–2323.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics 21*, 3 (July), 665–672.

WALTER, M., FOURNIER, A., AND MENEVAUX, D. 2001. Integrating shape and pattern in mammalian models. In *Proceedings of ACM SIGGRAPH 2001*, Annual Conference Series, 317–326.

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

WEI, L.-Y. 2001. *Texture Synthesis by Fixed Neighborhood Searching*. PhD thesis, Stanford University.

WU, Q., AND YU, Y. 2004. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3 (August), 362–365.

ZELINKA, S., AND GARLAND, M. 2002. Towards real-time texture synthesis with the jump map. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, 99–104.

ZHANG, J., ZHOU, K., VELHO, L., GUO, B., AND SHUM, H.-Y. 2003. Synthesis of progressively variant textures on arbitrary surfaces. *ACM Transactions on Graphics 22*, 3 (July), 295–302.

ZHU, C. S., WU, Y., AND MUMFORD, D. 1997. Minimax entropy principle and its application to texture modeling. *Neural Computation 9*, 8.

ZHU, C. S., WU, Y., AND MUMFORD, D. 1998. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision 27*, 2.