Graph Matching with Subdivision Surfaces for Texture Synthesis on Surfaces

Shaun Bangay*
Department of Computer Science
Rhodes University
Grahamstown, South Africa

Chantelle Morkel †
Department of Computer Science
Rhodes University
Grahamstown, South Africa

Figure 1: Texture synthesis on surfaces.

Abstract

Existing texture synthesis-from-example strategies for polygon meshes typically make use of three components: a multi-resolution mesh hierarchy that allows the overall nature of the pattern to be reproduced before filling in detail; a matching strategy that extends the synthesized texture using the best fit from a texture sample; and a transfer mechanism that copies the selected portion of the texture sample to the target surface. We introduce novel alternatives for each of these components. Use of $\sqrt{2}$-subdivision surfaces provides the mesh hierarchy and allows fine control over the surface complexity. Adaptive subdivision is used to create an even vertex distribution over the surface. Use of the graph defined by a surface region for matching, rather than a regular texture neighbourhood, provides for flexible control over the scale of the texture and allows simultaneous matching against multiple levels of an image pyramid created from the texture sample. We use graph cuts for texture transfer, adapting this scheme to the context of surface synthesis. The resulting surface textures are realistic, tolerant of local mesh detail and are comparable to results produced by texture neighbourhood sampling approaches.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Color, shading, shadowing and texture

Keywords: texture synthesis, subdivision surfaces, vertex region matching, graph cut

*e-mail: s.bangay@ru.ac.za
†e-mail: g99m0761@campus.ru.ac.za

AFRIGRAPH 2006, Cape Town, South Africa, 25–27 January 2006.
© 2006 ACM 1-59593-288-7/06/0001 $5.00

1 Introduction

Texture synthesis techniques are used to reproduce the pattern contained in a sample image onto either a larger 2D image or a surface in 3D. A variety of techniques have been employed for 2D image texture synthesis. The repertoire used for synthesis onto polygonal meshes is more restricted. In this paper we investigate the use of alternative strategies to those commonly used for texture synthesis on surfaces.

Neighbourhood matching is a fundamental component of many texture synthesis algorithms. This relies on textures having the Markov Random Field property, whereby the value at any point in the texture is dependant on the values of the neighbouring points. Therefore finding matching regions about two points is sufficient to allow transfer of texture attributes from one point to the other. Square neighbourhoods within the example texture have become the defacto standard for neighbourhood matching. We show that a region based on local surface connectivity is a viable alternative for matching and has benefits with respect to controlling texture scale and anti-aliasing.

Considerable success has been achieved using an image pyramid for texture synthesis. Lower resolution versions of the sample texture are used first during synthesis, covering the target with a coarse outline of the desired pattern. This provides a guide for later iterations to use higher resolution versions of the sample image to fill in details. Surface meshes at corresponding levels of detail are also required, and we demonstrate the use of $\sqrt{2}$-subdivision surfaces for providing these.

Some synthesis strategies work one point at a time. Others copy larger patches of the texture sample at each step, which is faster. We apply the graph cut mechanism to allow transfer of large areas while minimising seams. This has previously been applied to texture synthesis on 2D images. We show how it can be adapted to surface synthesis.

The terminology used in the remainder of the paper is as follows. We use the term *colour* as an intuitive term to refer to the attribute

of a texture map. This attribute can represent other quantities, such as transparency, or surface displacement [Dong and Chantler 2005; Lefebvre and Hoppe 2005; Ying et al. 2001], as shown in Figure 10c and 10f. The desired process is to transfer the pattern of colours from a *example* texture image to the *vertices* on a surface represented as a polygonal mesh. In the process we match a *neighbourhood* consisting of a set of *texels* in the sample image to a *region* comprised of the *graph* resulting from a set of *vertices* linked by the polygon edges of the surface.

1.1 Overview

We review details of existing texture synthesis strategies in section 2, concentrating on those approaches relevant to our problem domain: synthesis-from-example and synthesis on mesh surfaces. Section 3 provides the details of the approach that we use, with examples of the results achieved being discussed in section 4. We summarize the contributions of this work in section 5.

2 Related Work

A number of approaches have succeeded in synthesizing various classes of texture. In the following discussion we concentrate on those related to texture-from-example approaches [Efros and Leung 1999; Lefebvre and Hoppe 2005; Matusik et al. 2005; Turk 2001; Wei and Levoy 2001], which is the strategy employed in this paper. Approaches using procedural and solid texturing strategies are outside the scope of this work, and have been adequately summarized by others [Heeger and Bergen 1995; Turk 2001].

Existing synthesis strategies for surfaces typically use the following steps [Turk 2001; Wei and Levoy 2001; Ying et al. 2001; Zhang et al. 2003]:

1. Generate a coordinate system over the surface to provide a mapping between surface and texture example.
2. Create multiple versions of the surface, at different resolutions.
3. Find the neighbourhood in the texture sample that is the best match to a region of the surface.
4. Use this neighbourhood to transfer texture to the surface.

2.1 Coordinate system creation

The order of synthesis for a 2D target image can be determined relatively easily, using a raster-scan sequence [Ashikhmin 2001; Dong and Chantler 2005; Liang et al. 2001; Wei and Levoy 2000] or spiralling outwards from the region already synthesized [Efros and Leung 1999].

An analogous synthesis order is required when working with surfaces. Ordering points on the surface relative to a starting point allows the surface to be coloured in a continuous fashion from this starting point. Sweep values used by Turk [2001] approximate the distance of each point from an anchor vertex along an *orientation* field, and determine an ordering for vertices during synthesis. Similar functions, such as the average geodesic distance measure [Zhang et al. 2005] could also be used. Others have found random ordering of vertices provides acceptable results [Praun et al. 2000; Wei and Levoy 2001]. Filling in the tricky regions first (such as areas of high curvature) assumes that it will be easier to remove flaws propagated from these areas when reaching areas with more regular properties [Praun et al. 2000].

Surface texture synthesis methods thus require a tangential vector or orientation field that covers the surface of the object [Lai et al. 2005; Praun et al. 2000; Turk 2001; Wei and Levoy 2001; Ying et al. 2001; Zhang et al. 2003]. Creation of these fields requires that the user specify texture orientation at points on the surface. These are then interpolated over the remainder of the surface using, for example, Gaussian radial basis functions [Praun et al. 2000; Zhang et al. 2003], mis-alignment minimization [Wei and Levoy 2001] or up-sampling, down-sampling and low-pass filtering on a multi-resolution mesh hierarchy [Turk 2001]. Other scalar fields may be created in the same way, such as transition values to control texture scale [Zhang et al. 2003; Lai et al. 2005]. Scale of the texture can also be controlled by the magnitude of the tangential vectors [Praun et al. 2000].

The orientation field at each point on the surface allows the creation of a 2D coordinate system in the tangent plane to the point. This can then be mapped to the 2D coordinate system of the sample texture image for matching surface texture to neighbourhoods on the texture sample.

2.2 Multi-resolution synthesis

A number of texture synthesis approaches produce impressive results by repeating the synthesis steps at increasing resolutions [Ashikhmin 2001; Bonet 1997; Efros and Freeman 2001; Kwatra et al. 2005; Wei and Levoy 2000]. This could be viewed as incrementally synthesizing features of differing sizes [Bonet 1997]. Image pyramid approaches [Heeger and Bergen 1995] achieve this by using a sequence of example textures, each twice the resolution of the previous. Various filters can be used when down-sampling pixels, such as Gaussians [Wei and Levoy 2000] or Laplacians [Bonet 1997]. The oriented filters used to create a *steerable* pyramid [Heeger and Bergen 1995; Portilla and Simoncelli 2000] allow elongated structures to be propagated for anisotropic textures. Image *stacks* avoid quantization errors resulting from the decreasing image sizes used in pyramids [Lefebvre and Hoppe 2005].

Surface synthesis strategies use Gaussian pyramids for representing the texture sample at different resolutions, and a multi-resolution mesh hierarchy to provide different versions of the target surface with a corresponding level of detail [Lai et al. 2005; Turk 2001; Wei and Levoy 2001; Ying et al. 2001]. The multi-resolution mesh hierarchy can be created in a number of ways. The strategy introduced by Turk [2001] creates levels of the mesh hierarchy by placing the desired number of points randomly over the surface of an object. Repulsion is used to spread these points evenly over the surface, and Delaunay triangulation is used to create a connected mesh structure. Alternative strategies include the use of subdivision [Zhou et al. 2005], and subdivision surfaces [Ying et al. 2001].

2.3 Texture matching

The texture to be applied to a region of the surface can be chosen based on the texture already present in neighbouring regions, assuming that the texture conforms to the Markov Random Field model [Efros and Leung 1999; Turk 2001; Wei and Levoy 2000; Ying et al. 2001]. Finding a neighbourhood in the texture sample that matches the region around a surface point determines the colour for that point. The size of the neighbourhood used is an important factor in determining how well the different frequency components in the texture sample are reproduced [Efros and Leung 1999].

Matching is performed by scanning a fixed sized window over the texture sample and comparing it to the region around the target point. Comparisons are typically done using a sum-of-square-of-differences metric, normalised over the matchable region [Ashikhmin 2001; Bonet 1997; Efros and Leung 1999; Lefebvre and Hoppe 2005; Liang et al. 2001; Turk 2001; Wei and Levoy 2000; Wu and Yu 2004; Ying et al. 2001]. While the minimum value of the metric indicates the best match, allowing random choice of candidates from a narrow range about the best value can break up unwanted regular patterns in the resulting texture [Efros and Leung 1999]. This metric is widely used but it is known that it does not identify texture features particularly well, and results in some blurring [Ashikhmin 2001]. Alternative metrics weight pixels according to the distance from the centre of the window [Efros and Leung 1999], minimize seams in low frequency areas [Kwatra et al. 2003], or measure both geometric distortion and texture mismatch [Zhou et al. 2005]. Alternatively features can be identified for explicit preservation using the compass operator [Matusik et al. 2005], a label map identifying subtextures [Zalesny et al. 2005] or with texton masks [Wu and Yu 2004; Zhang et al. 2003].

Search performance can be improved by restricting it to regions around points previously matched [Ashikhmin 2001; Dong and Chantler 2005; Ying et al. 2001]. This also results in a more consistent transfer of texture features as coherent groups of neighbouring pixels tend to be copied.

The vertices in a mesh usually do not have a regular arrangement which complicates matching against a regular window shape in the texture sample. A regular coordinate system can be defined locally about a vertex, using the orientation and normal vectors, scaled according to average distance between mesh vertices. Colours at surface points corresponding to the discrete texel points in the texture sample can be found by interpolating values from nearby vertices according to this coordinate system [Turk 2001]. Surface matching still involves comparing a rectangular neighbourhood in the texture sample against corresponding points on the surface [Lai et al. 2005; Turk 2001; Ying et al. 2001; Wei and Levoy 2001].

When image pyramids are available, matching can be extended to compare neighbours at different resolutions. Alternating matching against two successive levels of the image pyramid extrapolates the overall pattern, and provides detail to features [Turk 2001]. Wei and Levoy [2001] use neighbourhoods that contain pixels from two adjacent levels of the pyramid.

Matching issues can occur when the texture sample has too great a range of patterns or colours [Efros and Leung 1999]. The synthesis process then locks into a single portion of the sample region.

Alternative synthesis strategies exist that do not rely on explicit matching. Copying statistical characteristics of the sample to the target image [Bonet 1997; Heeger and Bergen 1995; Paget and Longstaff 1998; Portilla and Simoncelli 2000; Xu et al. 2000] has been used successfully. Tiled textures can be jittered to add variation to the pattern [Lefebvre and Hoppe 2005; Xu et al. 2000]. Global optimization techniques define a quality metric over the entire target, and use an optimization strategy to iteratively improve texture quality [Kwatra et al. 2005].

2.4 Texture Transfer

Colour information can be transferred once matching regions in the texture sample and the target have been identified. Some approaches build up the target texture one pixel at a time [Efros and Leung 1999; Turk 2001; Wei and Levoy 2000].

Patch based strategies copy large regions of the sample over to the target surface. Boundaries between overlapping patches can also be chosen to minimize visible seams using dynamic programming [Dong and Chantler 2005; Efros and Freeman 2001], feathering and blending [Liang et al. 2001; Praun et al. 2000], cross edge filtering [Xu et al. 2000] or graph cuts [Kwatra et al. 2003; Wu and Yu 2004]. Wang tiles can be combined to create surface textures [Fu and Leung 2005], based on existing 2D texture synthesis techniques for extending the texture sample and creating tilable boundaries.

Histogram matching of high frequency content ensures that the resulting texture remains sharp, overcoming any blurring from previous stages [Matusik et al. 2005].

Surface synthesis approaches can produce a 2D image which can be mapped onto the original surface to reproduce the synthesized texture. Such a texture atlas can be created by segmenting the surface into regions of similar orientation, flattening these and projecting them onto a 2D image [Zhang et al. 2005]. Arranging the various pieces on the surface is a difficult problem, and is solved with direct user intervention [Praun et al. 2000], or an optimization strategy [Zhang et al. 2005]. Mapping the individual triangles to corresponding triangles on the texture map provides an automated alternative [Turk 2001].

3 Texture Synthesis Strategy

We base our approach on existing surface texture synthesis strategies [Praun et al. 2000; Turk 2001; Wei and Levoy 2001; Ying et al. 2001; Zhang et al. 2005]. The process involves creating an orientation field and coordinate system for the surface, building a multi-resolution hierarchy from the mesh, and then setting the surface colour according to the best match from the texture sample. The last step is repeated for increasing resolutions of mesh and texture sample. Texture synthesis is performed directly on a polygon mesh by assigning a colour directly to each vertex. The greater the number of vertices, the higher the resolution of the texture that can be produced.

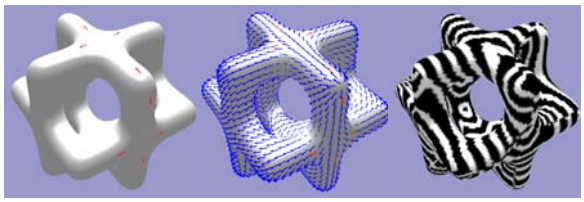
3.1 Initialization

3.1.1 Coordinate system creation

Orientation vectors at each vertex align the surface texture with the texture sample. This vector, in conjunction with the vertex normal, is used to create a local coordinate system for mapping surface coordinates to texture coordinates.

The orientation vector field over the surface mesh is created using the approach described by Turk [2001]. Orientation at key vertices is specified by the user (as shown in Figure 2a). We make repeated passes over the mesh, and during each pass the orientation vector at each vertex is set to the average orientation of its neighbours. The resultant vector is projected back onto the surface tangent plane at the vertex, and normalized. This process also has the effect of diffusing orientation values over the mesh, until every vertex has a defined value (see Figure 2b).

We also make use of the sweep field described by Turk [2001]. This is a scalar value, assigned to each vertex which increases in the direction of the orientation field. It can be used to order the vertices so that they can be visited in a continuous fashion. This reduces the number of passes needed for colour matching which requires that at least some neighbours have colours assigned.



(a) User input (b) Vector field (c) Sweep value contours

Figure 2: Vector field creation.

(a) Random start (b) Sweep field (c) Multi-resolution

Figure 3: Effects of starting conditions for synthesis on a grid.

The sweep field is not sufficient to guarantee a one pass traversal of points on the surface. Local minima may form in disconnected regions where the orientation field reverses. These areas will have no initialized adjacent vertices until the surface has been traversed at least once.

Both orientation vector and sweep field creation are performed in a hierarchical manner starting with the coarse mesh, propagating the values over the mesh, and diffusing them to the higher resolution meshes.

3.1.2 Order of synthesis

Initial colour values assigned to the mesh affect the nature of the synthesized result. Some approaches recommend starting by assigning random colours selected from the texture sample [Lai et al. 2005; Turk 2001; Wei and Levoy 2000; Wei and Levoy 2001]. This is well suited to stochastic textures, where no distinct overall pattern is required, since it creates matching histograms in the texture sample and surface texture. Texture synthesis with regular patterns is disadvantaged by lack of global coherence, as shown by the textured grid in Figure 3a.

An alternative approach is to grow regions from a starting point. In the surface texturing context, this can be done with a sweep field, as described in section 3.1.1. This provides a pattern with greater global coherence, although flaws still occur at interfaces where the growing region diverges and later rejoins (Figure 3b).

Multiple passes can overcome these issues to an extent, as can multi-resolution synthesis. The pattern laid down using a coarse version of the texture serves as a guide for later refinement of local features (Figure 3c). Turk [2001] notes that the quality of the initial colouring pass on the coarsest mesh is key to the quality of the final texture. We use the sweep field for the initial pattern on the coarse mesh, and random ordering thereafter.

3.2 Multi-resolution meshes

Existing multi-resolution texture synthesis schemes for polygon meshes rely on the mesh hierarchy having specific properties [Turk 2001]. The number of vertices from one level of the mesh to the next increases by a factor of four, ensuring that the complexity of the mesh increases at the same rate as the effective texture resolution in the image stack. Vertices are evenly spaced over the surface, to ensure regular texture samples when using vertex colouring to approximate texturing.

3.2.1 Subdivision for multi-resolution surfaces

Since texture is sampled only at vertices, a subdivision surface is used to increase the number of points and resolution of the surface texture. The regular nature of a subdivision surface also contributes to distributing these points evenly across the surface. Numerous subdivision schemes exist which satisfy the requirements for texture synthesis. We make use of the $\sqrt{2}$-subdivision strategy [Li et al. 2004], which has a number of advantages when combined with our other refinements:

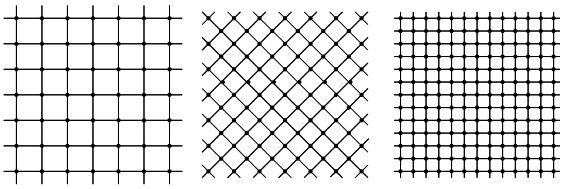
- The scheme is an approximatory subdivision scheme, so vertices are retained during the subdivision process. Attributes calculated for vertices at one level of the mesh hierarchy can be carried over directly to the next level.
- The mesh faces are all quadrilaterals improving correspondence when matching to the regular grid of a texture image, or when generating a texture map from the final coloured surface.
- In general, the number of vertices doubles with each subdivision step. A four fold increase can be achieved with two successive subdivisions. However our refinements to region matching are able to make use of the intermediate mesh.
- Adaptive subdivision is supported and is used to reduce variation in the size of polygons over the surface, and to produce an even distribution of vertices over the surface.

The refinement process for $\sqrt{2}$-subdivision is illustrated in Figure 4a-f. A center point is added to each face. Each edge then forms a new quadrilateral face by connecting the two ends, and the two adjacent face points. While this process works with meshes containing polygons with various numbers of sides, the meshes produced consist only of quadrilaterals (except for triangles which can occur on boundaries). Ordinary vertices are connected to exactly four others.

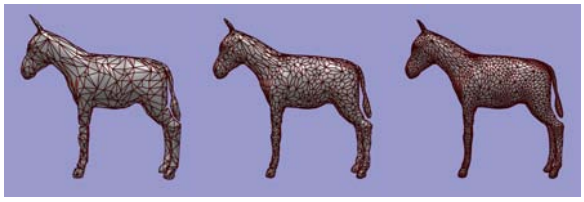
3.2.2 Regular vertex distributions

Subdivision on its own is not sufficient to produce an even distribution of vertices over the surface of the polygon mesh. Meshes with an uneven vertex distribution retain this after subdivision, although it is possible to keep subdividing until all edges fall below some threshold length. The $\sqrt{2}$-subdivision scheme can be modified to support adaptive subdivision, where the subdivision process is limited to a region within the mesh.

Care must be taken at the boundaries to ensure that boundary polygons do not become degenerate and that the mesh remains manifold and watertight. We use a variation on the scheme proposed by Li et al. [2004]. Triangular faces are used to ensure connectivity between the subdivided region and the remainder of the mesh, as shown in

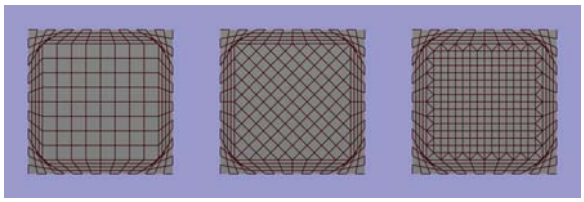


(a) Original (b) 1 refinement (c) 2 refinements



(d) Original (e) 1 refinement (f) 2 refinements

Figure 4: Refinement steps for $\sqrt{2}$-subdivision: (a-c) For a grid mesh (d-f) Applied to a polygonal object.



(a) Original (b) 1 refinement (c) 2 refinements

Figure 5: Adaptive subdivision applied to the center region of the grid.

Figure 5a-c. We relax the requirement that neighbouring faces in the boundary region differ by at most one level of subdivision.

We apply adaptive subdivision to the process of creating a regular distribution of vertices in the mesh by selecting all faces with an area that exceeds the mesh average by a given factor. Polygon P_i with area a_i is selected for subdivision if $a_i > f\overline{a}$, where $\overline{a}$ represents the average area of all polygons in the mesh. Adaptive subdivision is then applied to these selected faces.

Choice of the threshold factor, f, depends on the nature of the model. Improvements in vertex distributions are shown in Figure 6a-c for discrete values of f. The vertical axis represents the standard deviation in polygon area across the model, shown as a percentage of the average area. The change in height over the width of each bar (for a given threshold value) represents improvement in the distribution of polygon areas as a sequence of refinement steps are applied. The models referenced can be seen in Figures 10b, 10c and10e. Figure 6a shows an example of an object typical of the majority of objects tested. These give good results for $f \simeq 1.5$. Deviations for these objects tends to bottom out at 30%-40%. Objects with components at different scales behave as shown in Figure 6b where greatest improvement occurs for values of f that isolate the component with the coarsest resolution. Finally Figure 6c is an object with an even vertex distribution and provides an indication of the variance that can be expected even in such desirable cases. Some threshold factors have an adverse effect on vertex distribution; in other cases all polygons are within the threshold.

Models that have distinct sections at different resolutions do

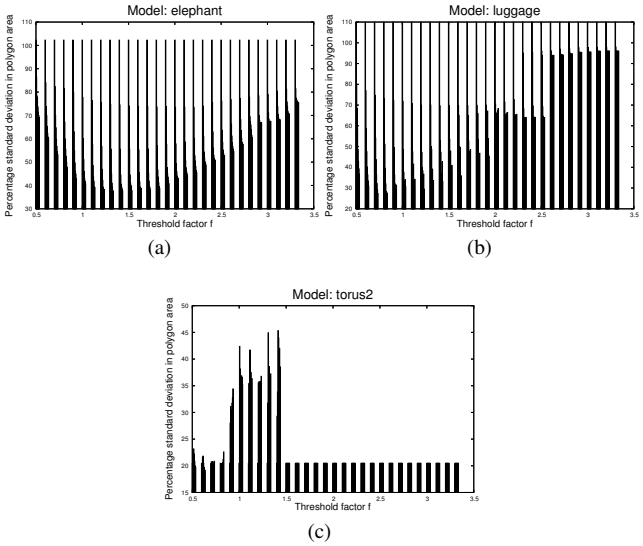


Figure 6: Effect of threshold factor on vertex distribution.

achieve a regular vertex distribution without problems. Degenerate polygons (in the sense that they have a high aspect ratio) are produced around boundaries in some cases, particularly when large isolated polygons are repeatedly selected for division. Later subdivision of the complete model during creation of the mesh hierarchy reverses this to some extent.

3.3 Strategies for texture matching

Our matching strategy involves matching regions on the surface to neighbourhoods in the texture sample. We allow the user to determine the scale at which the texture is synthesized, and cater for over- and under-sampling.

Texture matching strategies in 2D have typically compared rectangular windows in the source and target images. Since these texture samples are unlikely to correspond exactly to mesh vertices, techniques such as interpolating between adjacent vertices are required to produce corresponding colour values for comparison.

The purpose of neighbourhood matching is to provide a measurement of probabilities, assuming the texture has the Markov Random Field property. The rectangular shape of the neighbourhood is only an implementation convenience. We instead match regions defined by the local connectivity of the polygon mesh about the vertex under consideration, to a neighbourhood graph consisting of the corresponding texels, regardless of shape.

3.3.1 Matching strategy

Given a point P on the mesh which needs to be coloured, let k be a measure of the size of a region on the polygon mesh which is to be used for matching. We define the k-region about P as the set of vertices on the mesh that are connected to P by no more than k edges (as shown in Figure 7a-c).

We perform an exhaustive search over all potential candidate points, Q, in the texture sample, as outlined in Algorithm 1. The attributes of the best match, Q_{best}, are transferred to P.

Algorithm 1 Region/Neighbourhood matching algorithm.

```
for each P_i in the k-region about P
  calculate surface coordinates (u_i, v_i)
for each texel Q in the texture sample
  d ← 0
  for each P_i in the k-region about P
      d ← d + ||Colour(P_i) − Colour(Q + (u_i, v_i))||
  if d is better than d_best then
    Q_best ← Q
    d_best ← d
```

The surface coordinates of a neighbouring vertex are calculated by projecting them onto the tangent plane of the vertex under consideration. Coordinates for vertices that are not directly connected to P are calculated incrementally relative to each successive vertex on a path starting from P, and averaged over all such (shortest) paths. Similar but more complex unfolding schemes have been used elsewhere [Praun et al. 2000; Wei and Levoy 2001; Ying et al. 2001].

The metric used is the sum-of-squares-of-differences between corresponding colour values. Given vertex colours $\{v_1, v_2, \ldots, v_n\}$ with corresponding texel colours $\{w_1, w_2, \ldots, w_n\}$, a match occurs for the lowest value of $d = |v_1 - w_1|^2 + |v_2 - w_2|^2 + \ldots + |v_n - w_n|^2$.

We perform matching at multiple resolutions using a Gaussian image stack [Lefebvre and Hoppe 2005] constructed from the texture sample. The initial mesh is coloured with the texture from the top (lowest resolution) image from the pyramid. We then refine the mesh using $\sqrt{2}$-subdivision, and repeat the process using the next texture image from the stack. While it is perfectly feasible to have the complexity of the mesh and effective resolution of the texture increase at the same rate (by a factor of four), the refinements described below remove the need for strict synchronization between mesh and texture resolution.

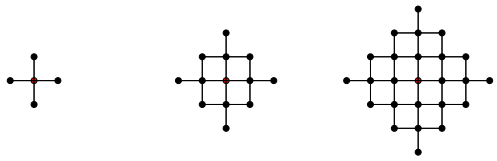
The matching process is equally applicable to image pyramids. The principle difference with the stack is that successive images are the same size; the same filtering is applied but without the downsampling. This reduces quantization effects at the cost of increased matching time.

3.3.2 Texture scale

The scale at which the texture is applied is a variable in our synthesis process, which is left under user control. Previous approaches have settled on a canonical scale where average edge length corresponds to distance between texels [Turk 2001; Wei and Levoy 2001], or where manual intervention is required to prepare a texture sample at a suitable scale [Matusik et al. 2005]. We define a scale ratio, r_{scale}, which represents the ratio between units in texture space (individual texels in the texture sample), and units in the 3D object space of the polygonal model.

Region and neighbourhood sizes are both specified in texture space, by giving the edge length of a square on the texture sample that contains the desired number of sample points. In this way the amount of texture covered during matching is independent of the scale of the polygon mesh.

A k-region in the mesh containing ordinary vertices (in terms of the $\sqrt{2}$-subdivision, these are 4-connected vertices) contains $(k+1)^2 + k^2$ vertices (illustrated in Figure 7a-c). Given a region size n corresponding to the edge length of a square in texture space relative to level m in the image stack, the corresponding value of k



(a) 1-region (b) 2-region (c) 3-region

Figure 7: k-regions for ordinary vertices.

can be found by solving:

$$(k+1)^2 + k^2 = \left(\frac{a^m n}{r_{scale}}\right)^2$$

Variable a has the value 1 for an image stack, and 2 for a pyramid. We use the same value as an approximation for regions containing extraordinary vertices.

3.3.3 Oversampling

In cases where the density of mesh vertices leads to multiple vertices being mapped onto the same texel, the matching process can be optimized to reduce the number of comparisons.

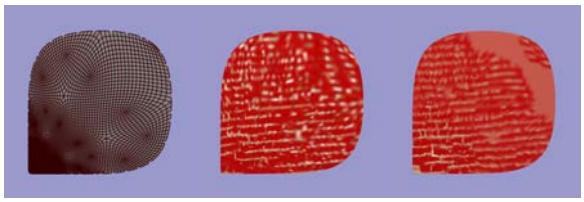
Consider the case where several vertices map to a single texel. Assume the original n vertices are grouped into k groups, where the m_i vertices in the i^{th} group all map to the same texel with colour w_i. The matching metric becomes:

$$\begin{aligned}
d &= |v_1 - w_1|^2 + |v_2 - w_2|^2 + \ldots + |v_n - w_n|^2 \\
&= \sum_{i=1}^{k} \sum_{j=1}^{m_i} |v_{ij} - w_i|^2 \\
&= \sum_{i=1}^{k} \sum_{j=1}^{m_i} \left[|v_{ij}|^2 - 2v_{ij} \cdot w_i + |w_i|^2 \right] \\
&= \sum_{i=1}^{k} \left[m_i |w_i|^2 + \sum_{j=1}^{m_i} |v_{ij}|^2 - 2w_i \cdot \sum_{j=1}^{m_i} v_{ij} \right] \\
&= \sum_{i=1}^{k} m_i \left(|w_i|^2 - \frac{2}{m_i} w_i \cdot \sum_{j=1}^{m_i} v_{ij} + \left| \frac{1}{m_i} \sum_{j=1}^{m_i} v_{ij} \right|^2 \right) + \\
&\quad \sum_{i=1}^{k} \left[\sum_{j=1}^{m_i} |v_{ij}|^2 - \frac{1}{m_i} \left| \sum_{j=1}^{m_i} v_{ij} \right|^2 \right] \\
&= \sum_{i=1}^{k} \left[m_i \left| w_i - \frac{1}{m_i} \sum_{j=1}^{m_i} v_{ij} \right|^2 + C \right]
\end{aligned}$$

The term C is dependant only on the values of m_i and v_{ij}. When matching the region around a vertex, these are independent of the neighbourhood on the texture sample. Thus all of the vertices that map to a single texel can be represented by their average colour. Since we are only interested in finding the neighbourhood with the minimum value of d, the same result would be found by dropping the additive constant C.

Object	r_{scale}	$\frac{1}{average\,edge\,length}$	Vertices	Vertex comparisons		Matching Time/[s]	
				Without averaging	With averaging	Without averaging	With averaging
donkey	30	110	65858	5.7×10^6	0.7×10^6	8876	752
donkey	100	110	65858	4.3×10^5	3.2×10^5	645	478
donkey	200	110	65858	2.04×10^5	2.02×10^5	334	326
bunny	40	154	240384	1.75×10^7	0.22×10^7	8397	1069
elephant	50	89	120193	2.50×10^6	0.98×10^6	1158	251

Table 1: Effects of vertex averaging during oversampling.



(a) Wireframe view of mesh (b) Without anti-aliasing (c) With anti-aliasing

Figure 8: Effects of anti-aliasing.

Examples of the improvement in matching performance in cases of oversampling are listed in Table 1. Matching is done using the equivalent of 3×3 texture neighbourhood; the region size in surface space depends on r_{scale}, and surface resolution. Results reported are for a single pass at fixed resolution for our prototype implementation on a 3GHz Pentium 4. The oversampling occurs when one texel extends over several polygons for small values of r_{scale}. It is still possible for multiple vertices to map onto a single texel for large values of r_{scale} due to variations in the shape of local region geometry.

3.3.4 Anti-aliasing

A potential problem with mapping the region around a vertex to a texture image is sparse sampling of the example texture. Previous approaches have relied on interpolation of vertex colours to provide a 1-1 mapping between texels and sample points around the vertex. When the vertices are widely separated in texture space, samples are taken at large intervals and aliasing issues can arise.

The potentially irregular sampling pattern around a vertex becomes an advantage when combined with the image stack. The average edge length around a given vertex gives an indication of the area over which that vertex has influence, and for which the colour of the vertex represents the combined colour of the texture for that area. Since the vertex can only represent a single colour, we match this vertex against a level of the image stack that has texels representative of a corresponding area.

For a vertex with an average edge length $\overline{e}$ to its neighbouring vertices, the area controlled by that vertex can be approximated by a square with diameter $\overline{e}$. Translated into texture space, the square has diameter $r_{scale}\overline{e}$. Matching for this vertex can be performed by comparing it to texels from the stack at level $\log_2(r_{scale}\overline{e})$. Colour transferred to the vertex being matched can also be selected from the appropriate level of the image stack in the same way.

Figure 8a-c shows the effect of the anti-aliasing when applied to a grid with irregular vertex densities. As would be expected, the narrow white stripes show greater continuity, and less variation in brightness over their length in the anti-aliased image.

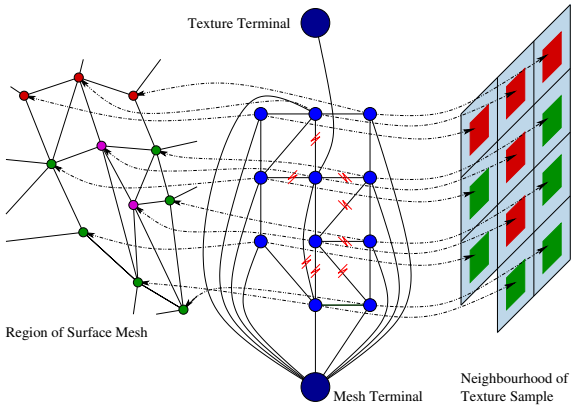


Figure 9: Graph configuration for graph cut.

Since the scale of the region matched for each vertex can vary on a per vertex basis, this allows synthesis to adapt to areas with non-uniform distributions of vertices.

3.4 Texture Transfer

The colour information from the texture sample is transferred to the surface once an appropriate match for the region about a vertex is found. While some surface synthesis strategies prefer to transfer colour only to the vertex under consideration [Efros and Leung 1999; Turk 2001; Wei and Levoy 2001], we support copying a portion of the texture neighbourhood to the region around the vertex.

Boundaries between overlapping texture fragments produce visible artifacts, and so we combine patches using the graph cut technique, which is used to smoothly blend textured regions for 2D texture synthesis. We employ the basic strategy described in Kwatra et al. [Kwatra et al. 2003], but adapt the graph formulation and matching metric to correspond to the single point matching approach used. The implementation uses the max-flow software written by Kolmogorov [Boykov and Kolmogorov 2004].

Matching no longer occurs between two regular texel neighbourhoods (as per [Kwatra et al. 2003]), but between a region of the surface mesh, and its corresponding imprint on the texture sample. A graph is created with nodes representing the coloured vertices in the surface. Each vertex has a corresponding texel in the texture sample once matching has been performed, where the correspondence is as shown in Figure 9. The central graph in this figure represents corresponding vertices/texels. Edge weights are assigned in

proportion to the severity of colour change that would result from replacing the current vertex colours with colours from the texture sample. A min-cut of this graph segments it along a path with the least colour discontinuity.

A pair of terminal nodes are also added, which represent the surface and texture sample respectively. The node representing the target vertex must receive its colour from the texture sample, and is connected to the texture terminal node with an edge weight of ∞. The border region around the graph (corresponding to vertices on the boundary of the surface region) is constrained to retain the current surface colour. Provided the boundary vertices have already been coloured, they are connected to the surface terminal node with an edge weight of ∞.

Weights between nodes A and B are assigned according to the differences in surface colour C_s and texture colour C_t for both nodes as given in (1). We include a weighting factor dependant on the distance from the target texel. This alleviates the bottleneck in graph flow into the single node connected to the texture terminal node.

$$W(A,B) \quad = \quad M(A,B)e^{-\left(\frac{x_A^2+x_B^2}{\sigma r^2}\right)} \tag{1}$$

where

$$M(A,B) = (|C_s(A) - C_t(A)| + |C_s(B) - C_t(B)|)$$

The variables x_A and x_B represent the distance of the two vertices concerned from the target vertex, while r is the diameter of the texture neighbourhood. Both are measured in texture space. The parameter σ controls the influence of the weighting term.

3.5 Summary

The matching process uses a surface independent scale for texture size (section 3.3.2), and can adapt to the density of the vertices (sections 3.3.3 and 3.3.4). There is no need to correlate the level of detail in the mesh with a particular level of the image stack. Each region can select appropriate levels automatically. Adaptive subdivision can be used at any point of the process without concern of breaking a relationship between mesh and texture sizes. We can also take advantage of the slow mesh refinement offered by $\sqrt{2}$-subdivision to control the complexity of the surface at a finer scale.

The surface colouring can be converted into a conventional texture map using strategies such as that used by Turk [2001].

4 Results

Figures 1 and 10a-f show the results of applying the complete texture synthesis process to a range of models. Synthesis is slow, as indicated in Table 1, since image quality has been the only consideration to date. Transfer of texture patches with graph cut improves performance to a degree dependent on the texture used. Results are comparable to other surface synthesis strategies [Turk 2001; Wei and Levoy 2001], where similar models and textures are shown.

Adaptive subdivision is used extensively on the model in Figure 10d. The wings are at a much lower resolution than the body, and the legs have even finer detail. Sixteen levels of adaptive subdivision reduce the percentage standard deviation in polygon area from 722.9% to 38.3% (see section 3.2.2). The apparent discontinuities on the wing result from a pair of extraordinary vertices creating degenerate polygons in that area (see section 3.2.2).

As mentioned previously (section 1), the texture values synthesized can represent values other than colour, such as a surface displacement value as used in Figure 10c and Figure 10f.

When region matching using sweeping, the area already coloured can be weighted more heavily than an adjacent region that is sparsely filled because more samples are taken from the region of dense colour. This can affect fine detail on textures which can become "locked out" by unfortunate choices in the initial colours. Missing entries must first be filled in, either by interpolation from known neighbours, or by employing a two pass process (as used elsewhere [Lefebvre and Hoppe 2005; Wei and Levoy 2001; Zhang et al. 2003]) which first fills in missing values, then refines the entries that were previously coloured.

We find that additional passes at each level serve to improve the sharpness of the synthesized texture. Each iteration is able to better match the region around the vertex under consideration when the colour values for the surrounding vertices have also been updated. Additional passes can also overwrite fine detail, as observed in other approaches [Ashikhmin 2001], so the number of passes used varies with the texture being synthesized.

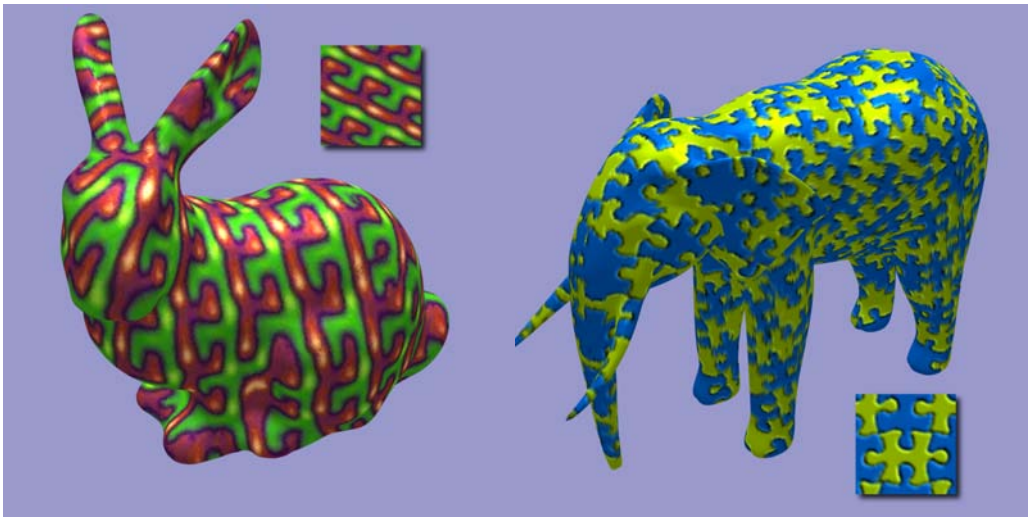
Not all synthesis attempts perform as well as those shown. While stochastic textures usually produce acceptable results, textures with definite structures sometimes fail to reproduce all features. Some textures perform badly and start growing garbage as described by several authors [Ashikhmin 2001; Efros and Leung 1999], a characteristic of synthesis using pixel transfer [Liang et al. 2001]. Others, such as the wood texture used in Figure 10c looses the grooves between planks if more than one pass is performed. The fine vertical bars of mortar in the brick texture used in Figure 8a-c are often lost during synthesis. These issues relate to the fundamental synthesis process, however, and not to the refinements presented in this paper.

5 Conclusion

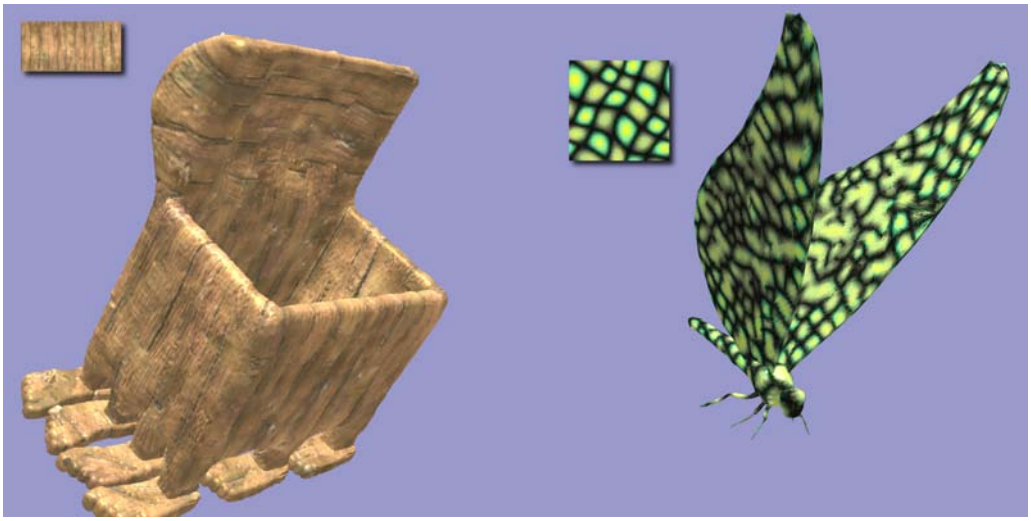
We successfully synthesize textures onto the surface of polygon meshes using the techniques described. Results obtained are comparable to those reported for regular neighbourhood matching strategies using alternative multi-resolution approaches.

The paper makes the following contributions:

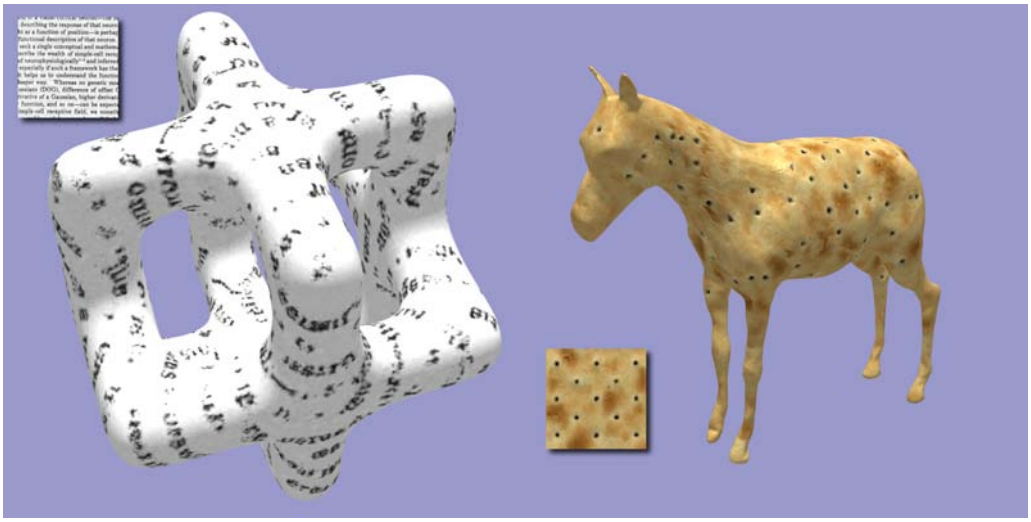
- Neighbourhood matching is performed using regions defined by the local connectivity of vertices, rather than relying on rectangular neighbourhoods. We demonstrate that regions based on local surface connectivity are a viable representation for matching. The region matching approach provides considerable flexibility when matching against the image stack. The matching process adapts to local vertex density on the surface, catering for situations in which the texture is over- or under-sampled, and is easily adapted to working with textures at different scales.
- The $\sqrt{2}$-subdivision strategy is applied to the problem of producing a multi-resolution mesh hierarchy suitable for region matching against image stacks. A technique for using adaptive subdivision is described which achieves an even vertex density over the polygon mesh.
- An adaptation of the graph cut technique for use with a surface synthesis strategy is described, introducing a graph configuration and edge weight metric specific to this scenario. The use of graph cuts allows multiple points to be coloured for each



(a) bunny (b) elephant



(c) luggage (with displacement mapping) (d) monarch



(e) torus2 (f) donkey (with displacement mapping)

Figure 10: Examples of the use of the texture synthesis process.

matching operation performed, reduces seams and improves the speed of the process.

5.1 Future Work

Performance issues have been ignored in this paper. Recent advances in texture synthesis strategies could be investigated with respect to their suitability for region matching.

5.2 Acknowledgements

Models used are from the Stanford 3D Scanning Repository, and 3dcafe.com. Textures are taken from web sites maintained by the authors of [Efros and Leung 1999; Kwatra et al. 2003; Turk 2001]. Anonymous reviewers provided several valuable suggestions.

References

ASHIKHMIN, M. 2001. Synthesizing natural textures. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM Press, 217–226.

BONET, J. S. D. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of ACM SIGGRAPH 1997*, ACM Press/Addison-Wesley Publishing Co., 361–368.

BOYKOV, Y., AND KOLMOGOROV, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9, 1124–1137.

DONG, J., AND CHANTLER, M. 2005. Capture and synthesis of 3D surface texture. *International Journal of Computer Vision 62*, 1-2, 177–194.

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

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *Proceedings of IEEE International Conference on Computer Vision (ICCV'99)*, 1033–1038.

FU, C.-W., AND LEUNG, M.-K. 2005. Texture tiling on arbitrary topological surfaces. In *Proceedings of Eurographics Symposium on Rendering*, 99–104.

HEEGER, D., AND BERGEN, J. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of ACM SIGGRAPH 1995*, ACM Press, 229–338.

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

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics 24*, 3, 795–802.

LAI, Y.-K., HU, S.-M., GU, D. X., AND MARTIN, R. R. 2005. Geometric texture synthesis and transfer via geometry images. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, ACM Press, New York, NY, USA, 15–26.

LEFEBVRE, S., AND HOPPE, H. 2005. Parallel controllable texture synthesis. *ACM Transactions on Graphics 24*, 3, 777–786.

LI, G., MA, W., AND BAO, H. 2004. $\sqrt{2}$ subdivision for quadrilateral meshes. *The Visual Computer 20*, 2 (May), 180–198.

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

MATUSIK, W., ZWICKER, M., AND DURAND, F. 2005. Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics 24*, 3, 787–794.

PAGET, R., AND LONGSTAFF, I. D. 1998. Texture synthesis via a noncausal nonparametric multiscale markov random field. *IEEE Transactions on Image Processing 7*, 925–931.

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

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. In *Proceedings of ACM SIGGRAPH 2000*, 465–470.

TURK, G. 2001. Texture synthesis on surfaces. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, 347–354.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH 2000*, ACM Press/Addison-Wesley Publishing Co., 479–488.

WEI, L.-Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, New York, NY, USA, 355–360.

WU, Q., AND YU, Y. 2004. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics 23*, 3, 364–367.

XU, Y.-Q., GUO, B., AND SHUM, H. 2000. Chaos mosaic: Fast and memory effecient texture synthesis. Tech. Rep. TR-2000-32, Microsoft Research, Redmond, WA.

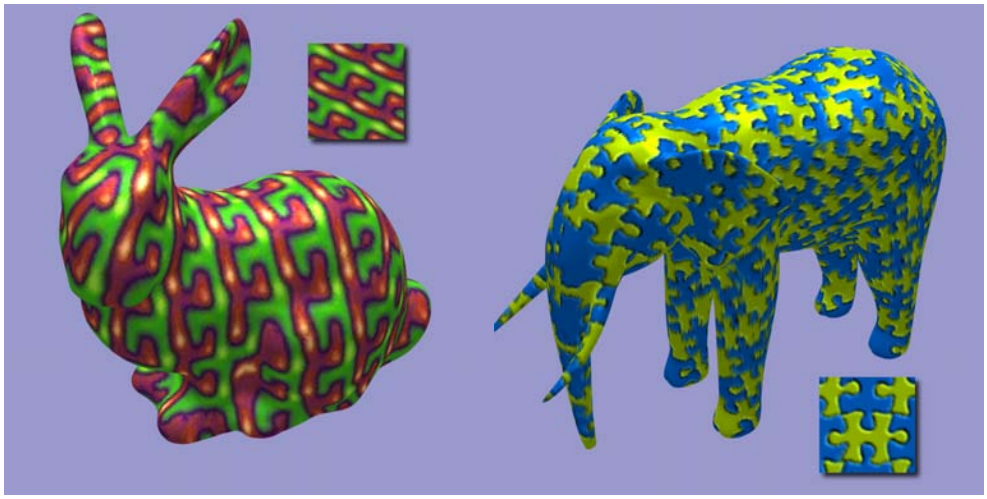
YING, L., HERTZMANN, A., BIERMANN, H., AND ZORIN, D. 2001. Texture and shape synthesis on surfaces. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, Springer-Verlag, 301–312.

ZALESNY, A., FERRARI, V., CAENEN, G., AND GOOL, L. V. 2005. Composite texture synthesis. *International Journal of Computer Vision 62*, 1-2, 161–176.

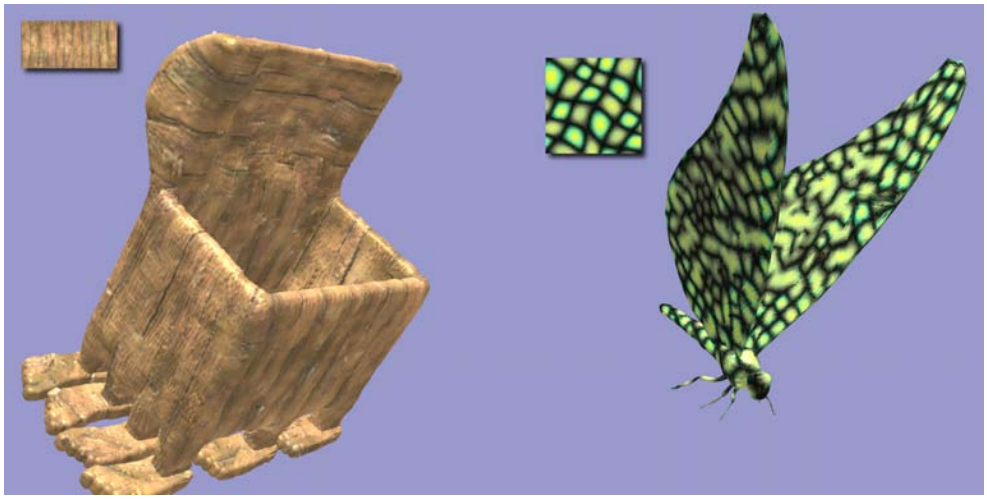
ZHANG, J., ZHOU, K., VELHO, L., GUO, B., AND SHUM, H.-Y. 2003. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics 22*, 3, 295–302.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2005. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics 24*, 1 (January), 1–27.

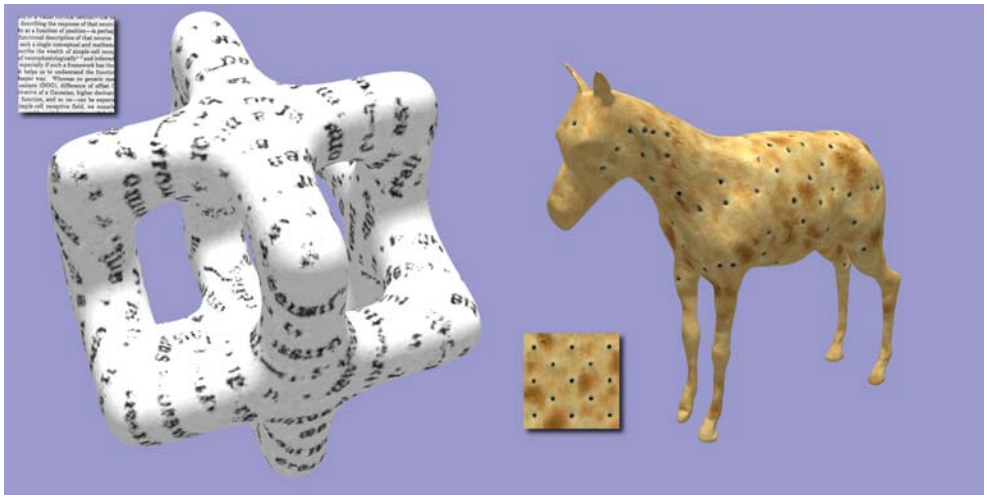
ZHOU, K., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2005. Texturemontage. *ACM Transactions on Graphics 24*, 3, 1148–1155.



(a) bunny (b) elephant



(c) luggage (with displacement mapping) (d) monarch



(e) torus2 (f) donkey (with displacement mapping)

Figure 10: Examples of the use of the texture synthesis process.