Layered Shape Synthesis: Automatic Generation of Control Maps for Non-Stationary Textures

Amir Rosenberger
Tel Aviv University

Daniel Cohen-Or
Tel Aviv University

Dani Lischinski
The Hebrew University

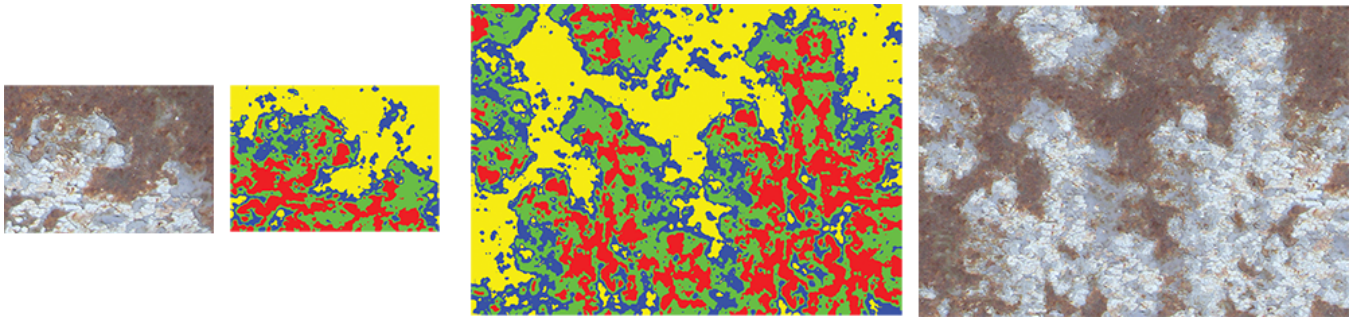


Figure 1: *An inhomogeneous texture, exhibiting a non-uniform mixture of peeling paint, bare metal, and rust. From left to right: input texture exemplar, control map extracted from the exemplar, a larger control map synthesized by our approach, and the resulting new texture.*

Abstract

Many inhomogeneous real-world textures are non-stationary and exhibit various large scale patterns that are easily perceived by a human observer. Such textures violate the assumptions underlying most state-of-the-art example-based synthesis methods. Consequently, they cannot be properly reproduced by these methods, unless a suitable control map is provided to guide the synthesis process. Such control maps are typically either user specified or generated by a simulation. In this paper, we present an alternative: a method for automatic example-based generation of control maps, geared at synthesis of natural, highly inhomogeneous textures, such as those resulting from natural aging or weathering processes. Our method is based on the observation that an appropriate control map for many of these textures may be modeled as a superposition of several layers, where the visible parts of each layer are occupied by a more homogeneous texture. Thus, given a decomposition of a texture exemplar into a small number of such layers, we employ a novel example-based shape synthesis algorithm to automatically generate a new set of layers. Our shape synthesis algorithm is designed to preserve both local and global characteristics of the exemplar's layer map. This process results in a new control map, which then may be used to guide the subsequent texture synthesis process.

Keywords: control maps, example-based texture synthesis, non-stationary textures, shape synthesis

1 Introduction

Computer generated imagery relies heavily on textures to achieve realism. One easy way to acquire realistic textures is by scanning or taking photographs of surfaces and materials that surround us in the real world. Therefore, a large number of methods have been proposed for synthesizing textures from examples, in the last decade [Wei et al. 2009]. Many of these methods are able to produce impressive results when applied to homogeneous textures that may be described by stationary Markov random field (MRF) models. Yet many real world textures are highly inhomogeneous, and are not modeled well by a stationary stochastic process.

Consider, for example, the rusty metal surface shown on the left in Figure 1. The texture on this surface is clearly non-stationary, and it may be seen as a highly non-uniform mixture, or superposition, of several different textures: peeling paint, bare metal, and rust. While each of these three textures is roughly homogeneous, the texture as a whole is not. This is a typical situation for many real world surfaces, whose texture often results from natural processes, such as weathering, corrosion, color cracking and peeling, growth of moss, etc. [Dorsey and Hanrahan 1996; Dorsey et al. 1999; Bosch et al. 2004; Desbenoit et al. 2004; Dorsey et al. 2008].

A common remedy to cope with such textures is to guide the synthesis process by a control map that encodes the large scale variations and the non-local features of the desired output texture (e.g., [Ashikhmin 2001; Hertzmann et al. 2001; Zhang et al. 2003; Wang et al. 2006; Wei et al. 2008]). However, such control maps are typically either user-specified or produced by a custom tailored simulation (e.g., biological or physically-based).

In this work we propose a new method for automatically generating control maps from examples, geared at natural textures such as the one in Figure 1. As observed above, such textures often look like a superposition of several layers, where each visible regions of each layer are occupied by a more homogeneous texture. The shape of the texture-occupied regions in each layer is far from arbitrary. Rather, it is the consequence of the specific natural process that produced this texture, as well as the shape of the layer underneath. Neither global statistics, nor small neighborhoods are capable of faithfully capturing such higher level structures. Such appearances may be generated by specialized shaders or by physically-based simulations. However, we are not aware of any general fully automatic way for generating such a shader from a specific example.

Our approach begins by decomposing the input exemplar into a number of layers, which we order bottom to top. A novel example-based *shape synthesis* algorithm is then used to generate a new set

ACM Reference Format
Rosenberger, A., Cohen-Or, D., Lischinski, D. 2009. Layered Shape Synthesis: Automatic Generation of Control Maps for Non-Stationary Textures. *ACM Trans. Graph. 28*, 5, Article 107 (December 2009), 9 pages. DOI = 10.1145/1618452.1618453 http://doi.acm.org/10.1145/1618452.1618453.

of layers, whose local and global characteristics visually resemble those of the exemplar's layers. This algorithm makes use of a bidirectional measure of similarity between the shapes of the layers, which is based on the shapes' boundaries. Starting from some initial output shape, we iteratively optimize the shape with respect to this similarity measure. Once the new layers are available, a texture transfer process based on "texture-by-numbers" [Hertzmann et al. 2001] is invoked, resulting in the final output texture, such as the result shown in Figure 1.

In summary, the main novelty in our approach lies in example-based synthesis of a suitable control map, rather than working directly on the texture, or on some associated appearance space [Lefebvre and Hoppe 2006]. To our knowledge, such an approach has not been explored before.

2 Related Work

Example-based texture synthesis has enjoyed considerable research attention in recent years. Most of the relevant previous methods may be roughly classified to parametric methods [Heeger and Bergen 1995], and non-parametric methods, which include pixel-based methods [Efros and Leung 1999; Wei and Levoy 2000], patch-based methods [Efros and Freeman 2001; Kwatra et al. 2003], optimization-based methods [Kwatra et al. 2005], and appearance-space texture synthesis [Lefebvre and Hoppe 2006]. Parametric methods attempt to construct a parametric model of the texture based on the input sample, which has proven to be a challenging task, and are mostly successful with structureless stationary textures. Non-parametric methods have demonstrated the ability to handle a wider variety of textures, by growing the texture one pixel/patch at a time. Optimization-based methods evolve the texture as a whole, further improving the quality of the results and making the synthesis more controllable. We refer the reader to [Wei et al. 2009] for a more comprehensive overview of example-based texture synthesis.

While non-parametric methods are typically able to reproduce small scale structure, they have a difficulty coping with highly inhomogeneous textures, since such textures cannot be modeled by a stationary Markov Random Field (MRF) model, which provides the theoretical basis for most of these methods. In order to handle such textures and control large scale structure, Ashikhmin [2001] proposed to guide the synthesis process by a user-provided target image, which specifies the local average colors across the target texture. *Texture-by-Numbers* [Hertzmann et al. 2001] extends this idea further by augmenting the input exemplar with a label map, where regions with distinct texture are distinguished by different labels. A suitable label map may be painted manually by the user, or created automatically using unsupervised image segmentation. To synthesize a new image, a target label map is provided, which indicates how the different textures should be arranged in the resulting image. However, that work addressed neither the issue of automatically generating a label map for natural inhomogeneous textures, nor the automatic synthesis of the target label map, as we do in our work.

Many other works since made use of control maps when synthesizing non-stationary textures, for example [Zhang et al. 2003; Wang et al. 2006; Gu et al. 2006; Lu et al. 2007; Wei et al. 2008]. However, in all of these works the control map for the target texture is either provided by the user, or derived from a specific model of texture formation across a 3D surface (e.g., [Lu et al. 2007]), and we are not aware of any previous attempts of example-based control map generation.

Our shape synthesis approach is related to texture optimization techniques [Wexler et al. 2004; Kwatra et al. 2005], which synthesize textures by minimizing a *texture energy* function. This function consists of a sum of local terms measuring how close each synthesized texture patch is to an exemplar patch. However, this formulation does not account for the possibility that there may be many other patches in the exemplar that are not represented at all in the synthesized result. While this may be adequate for homogeneous textures, where most patches are similar to each other, the quality of the results for inhomogeneous textures is often compromised. While it is possible to inject some global statistics into the optimization [Kopf et al. 2007], the resulting process still fails to capture the large scale appearance of highly inhomogeneous natural textures that are the target of this work. In contrast, we perform shape synthesis with a bidirectional similarity measure (inspired by Simakov et al. [2008] and Wei et al. [Wei et al. 2008]), and demonstrate more faithful reproduction of appearance in the comparisons we present in Section 4.

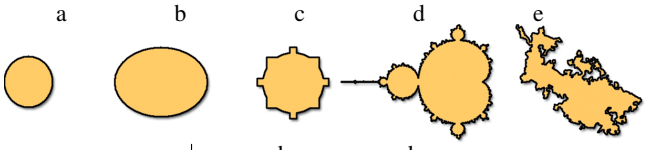
Appearance-space texture synthesis [Lefebvre and Hoppe 2006] is another optimization method that operates in a feature space, rather than using the values of pixels or small patches directly. A point corresponding to a pixel in a more general feature space may encode more information, allowing structure to be reproduced better. The layer map that we associate with the input exemplar in our approach could be viewed as a feature space custom-tailored for synthesis of layered inhomogeneous textures.

A variety of methods generate textures of weathered surfaces by assuming and simulating a physical model [Dorsey and Hanrahan 1996; Dorsey et al. 1999; Merillou et al. 2001; Bosch et al. 2004; Desbenoit et al. 2004; Dorsey et al. 2008]. While such methods have produced some highly realistic results, they are not geared towards matching a particular appearance given by an example. Also, controlling the results of the synthesis typically involves specifying a large number of parameters, which are not always intuitive. In contrast, our approach is example-based, rather than physically-based.

Our approach synthesizes the boundaries of the layer shapes by example. Thus, it is related to the Curve Analogies work of Hertzmann et al. [2002], where a similar framework was applied to reproduce the style of curved shapes. However, our work uses a different similarity measure and operates on a discrete patch-based representation of a shape's boundary, rather than a vector-based representation. Also related is the work of Baht et al. [2004], which uses binary voxel grids in order to synthesize geometric details on volume surfaces. These voxel grids are similar to the binary neighborhoods that we use to optimize the shape boundaries. However, their goal is to add smaller-scale detail to an existing global shape, while we focus on synthesizing the entire shape from scratch.

3 Layered Shape Synthesis

This work deals with example-based generation of control maps represented as *layer maps*. A layer map is an image where different pixel values indicate to different layers. Let $v_1 < v_2 < \ldots < v_K$ be the values of layer map pixels, sorted in ascending order. Then, a layer L_i is defined as the set of all pixels whose value is greater than or equal to v_i. Note that a pixel with value v_j actually belongs to all layers $L_1, \ldots, L_j$. One can think of the layers as stacked on top of each other, with layers higher in the stack partially "concealing" lower layers. Each layer has an associated foreground shape S_i, which we encode as a binary image of the same dimensions as the layer map. Note that the shape S_{i+1} is always contained in S_i. As may be seen in Figures 1 and 6, the boundaries of these nested shapes are highly correlated, but not aligned. In the figures in this paper, we display values corresponding to different layers using unique colors.



	a	b	c	d	e
a	0.0	1.3	3.0	5.3	6.0
b		0.0	2.7	5.0	5.4
c			0.0	5.0	5.3
d				0.0	4.6
e					0.0

Figure 2: *Similarity measure between pairs of five different shapes.*

Given a set of such shapes, our goal is to synthesize a new set of shapes, while maintaining both global and local similarity to the original ones. For this purpose, it is important that each shape occupies the same relative amount of pixels in the synthesized map as it did in the exemplar, and that the boundaries of the synthesized shapes locally resemble those in the exemplar. We found that representing the shape by means of its boundary curve fails to capture all of the relevant information. Such a representation cannot predict the spatial relationship between disconnected components of the shape, and does not prevent self-intersections. Instead, we represent a binary shape by a collection of patches centered on the shape's boundary pixels (at multiple resolutions) in order to captures the necessary shape properties.

Our shape synthesis approach employs optimization similarly to [Wexler et al. 2004; Kwatra et al. 2005], where the synthesized result is iteratively optimized with respect to some measure of its similarity with the exemplar. We begin by deriving a suitable bidirectional shape similarity measure, similarly to Simakov et al. [2008] and Wei et al. [2008]. Next, we describe a novel greedy optimization scheme that iteratively modifies an initial shape, so as to increase its similarity to a given exemplar shape. Finally, we discuss how this mechanism is used to create an entire new layer map, which is a sequence of nested shapes, from the layer map produced in the layer decomposition phase described in the previous section.

3.1 Shape similarity measure

Let B_1 and B_2 be the sets of boundary pixels of shapes S_1 and S_2, respectfully. A boundary pixel is a pixel inside a shape with at least one of its 4-neighbors outside the shape. Let $x_1 \in B_1$ and $x_2 \in B_2$ be two boundary pixels, and let $N_\theta(x_1)$ and $N_\eta(x_2)$ be the neighborhoods centered around them and rotated by $\theta, \eta \in \{0°, 90°, 180°, 270°\}$. We refer to such neighborhoods as *boundary patches*. We define the similarity, D, between two boundary pixels as the L^2 distance between their neighborhoods (rotated such that the distance is minimized). Formally,

$$D(x_1, x_2) = \min_{\eta} \left\| N_0(x_1) - N_\eta(x_2) \right\|_2 \qquad (1)$$

Since we deal with binary images, the L^2 norm above is simply the number of different pixels between two patches. Next, we define the local similarity between a boundary pixel $x_1 \in B_1$ and the boundary of (another) shape S_2 as the similarity between x_1 and the pixel most similar to it on the boundary B_2:

$$D(x_1, S_2) = \min_{x_2 \in B_2} D(x_1, x_2). \qquad (2)$$

Note that this similarity measure is not symmetric. While it ensures that every boundary patch of S_1 is similar to a boundary patch in S_2, there may be boundary patches in S_2 that are not well represented in S_1. For example, a simple shape may be deemed similar to a more complex one that also happens to contain some simple features. Thus, we require a bidirectional similarity measure, defined as

$$D(S_1, S_2) = \frac{\sum_{x_1 \in B_1} D(x_1, S_2) + \sum_{x_2 \in B_2} D(x_2, S_1)}{|B_1 \cup B_2|}, \qquad (3)$$

which is the average number of different pixels between a boundary patch of one shape to its nearest neighbor on the other. Figure 2 shows several different shapes and reports their pairwise bidirectional similarities.

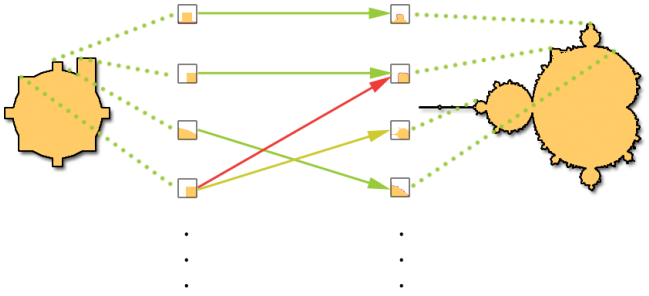


Figure 3: *Iterative assignment of boundary patches. The exemplar boundary patches (left) are assigned to the synthesized boundary patches (right). In cases where two patches are assigned to the same one, the assignment with the larger L^2 difference (red arrow) is discarded and will be assigned to another patch in a future iteration (yellow arrow).*

3.2 Shape optimization

Armed with the similarity measure above, we use an optimization procedure that iteratively modifies the boundary of a synthesized shape S to make it more similar to that of the exemplar shape E. The optimization proceeds from coarse to fine resolution. At each resolution we alternate between two main steps: (i) matching each boundary patch of S to a boundary patch of E, and (ii) modifying S by adding or removing pixels based on the results of the matching achieved in the previous step. This iterative optimization procedure resembles that of Kwatra et al. [2005], but each of the two main steps differs significantly from its counterpart, because we minimize a different (bidirectional) energy function, and work with binary images, rather than textures. These two steps are discussed in more detail below.

Boundary patch matching. As pointed out earlier, we would like every boundary patch of S to resemble one of E, but we would also like every boundary patch of E to be represented in S. Thus, assuming we have an equal number of boundary patches in E and S, we seek a minimum cost assignment, a fundamental combinatorial optimization problem [Schrijver 2003]. Solving this problem exactly is too expensive for our purposes ($O(n^3)$, where n is the number of patches), so we resort to an approximate solution using the iterative greedy approach described below.

Let B_E and B_S denote the sets of boundary patches of E and S, respectively, and assume for now that the two sets have the same size. Each patch in B_E is initially assigned to its nearest neighbor in B_S. As a result, some patches in B_S may have more than one exemplar patch assigned to them, while others may have none (see Figure 3). In the former case, we keep only the assignment with the smallest L^2 difference, and discard the rest. All of the pairs of

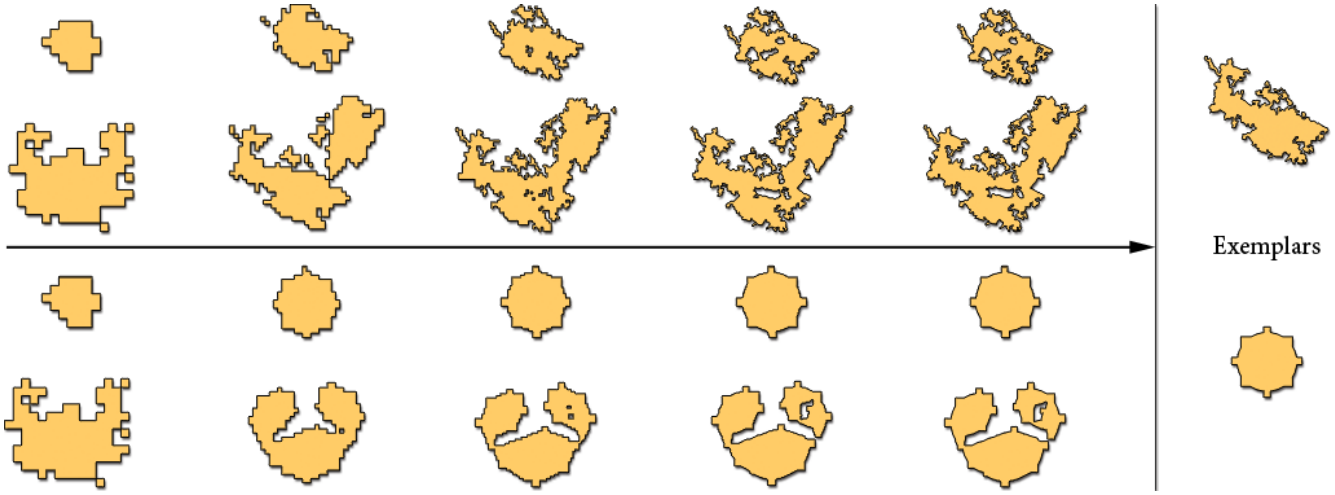


Figure 5: *Refining shape boundaries with our multi-resolution optimization. Two initial shapes (left) are evolved using two different exemplars (right).*

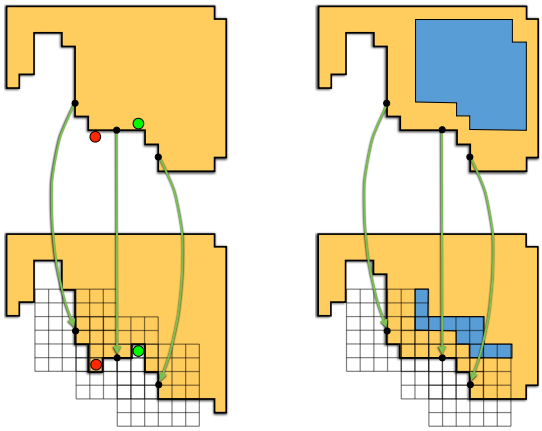


Figure 4: *Left: Shape adjustment. Boundary exemplar patches are superimposed over their assigned positions. Pixels in regions of overlap between these superimposed patches may be added to the shape (green dot), or removed from it (red dot), making the new boundary more similar to that of the exemplar. Right: Matching patches from the previous shape (gold) are superimposed again to seed the new shape (blue).*

patches which have been assigned are then removed from further consideration, and the process is repeated until every patch in B_S has been assigned.

In general, B_E and B_S differ in size. Typically, the synthesized shape is larger than the exemplar. Thus, assuming that $|B_S| = Q|B_E| + R$, we construct a set of exemplar patches of size $|B_S|$ by including each exemplar patch Q times, and randomly selecting R additional patches from B_E. In this way we ensure that all the boundary features in the exemplar shape get an equal chance to be represented in the synthesized shape.

Shape adjustment. After finding the assignment as described above, our goal is to modify the boundary of S so as to increase the similarity to E (by reducing $D(S,E)$). To achieve this, we superimpose each exemplar patch over its counterpart in B_S. Consider a pixel x outside S, which is covered by several overlapping superimposed patches from B_E. Informally, if these patches agree that x should be part of the shape, it is added to S. Similarly, a pixel inside S might be removed if the overlapping patches agree that it should not belong to the shape. This is illustrated in Figure 4 (left).

More specifically, consider a pixel x in the vicinity of the boundary of S. It is covered by two groups of overlapping superimposed exemplar patches: one group predicts that x belongs to the shape, while the other one predicts that x is outside the shape. For each of these two predictions we compute a score by summing up the weights of the corresponding group's patches at x. Let x_S be a boundary pixel of S and x_E the exemplar pixel assigned to it. Then the entire exemplar patch is assigned the following weight

$$\frac{1}{1+D(x_S,x_E)}, \tag{4}$$

which is further multiplied by a Gaussian falloff function (thus the weight decreases away from the center of the patch). The sigma value for this function was chosen to be half of the patches size. The group with the highest score at x determines whether x should be included or excluded from the shape.

When the sum of weights accumulated at each pixel x is below a threshold, its value remains unchanged. This is because patch weights reflect a degree of certainty, so areas of low weight are more sensitive to randomness generated by our approximated nearest neighbor search and the greedy assignment, such that using the new values may produce noise. This threshold also determines the final amount of pixels in the shape after the iteration is done. Therefore, it is set dynamically so that the (relative) amount of the pixels inside the shape is the same as in the exemplar. Candidates from the interval $[10^{-2}, 10^{-7}]$ are tested and the one which results in the nearest amount is chosen. After the update is complete, the optimization procedure is repeated until convergence. Convergence is reached when the number of changed pixels falls below a threshold.

As mentioned earlier, the optimization proceeds from coarse to fine resolution. The result computed at each resolution level is upsampled to serve as a starting point for the next (finer) level. At coarser resolutions the global structures are formed, while fine resolutions fill in the fine details along the shape's boundary. In our examples, we use 5 to 6 resolution levels. Figure 5 shows how different initializations lead to different global shapes. However, in all of the examples the synthesized shape contains boundary features that

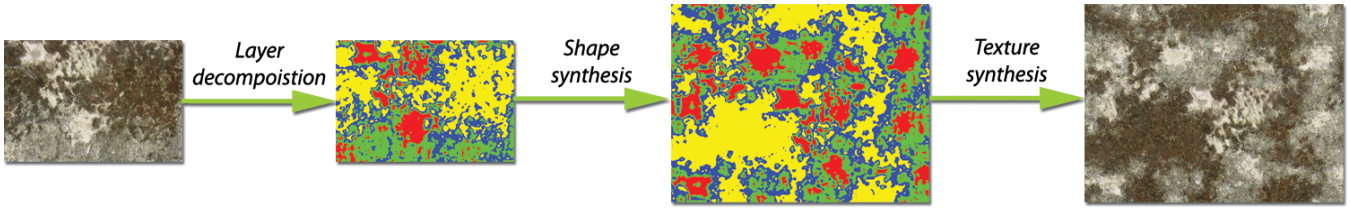


Figure 6: *Our inhomogeneous texture synthesis approach.*

are very similar to those present in the exemplar, resulting in close overall resemblance.

3.3 Layer map synthesis

The shape optimization procedure presented in the previous section may be used directly to synthesize the first (bottom) layer. A randomly generated shape, with the number of foreground pixels matching that of the corresponding exemplar layer may be used for initialization. In order to generate the following layers, however, we must introduce a number of modifications. First, the shape of each layer is nested inside the shape of the layer beneath it. Second, the boundaries of two successive layers are typically highly correlated. Preserving this correlation is important, as it is instrumental for faithfully reproducing the appearance of the exemplar in the synthesized result. It is not obvious how to initialize the shape of the next layers under these conditions.

To address these requirements we begin the synthesis of each layer L_{i+1} by creating a mask that defines the area (contained inside the shape of the previous layer S_i), where the current shape is allowed to evolve. Initially, this mask is set to the entire shape S_i, but we use the most recent boundary patch assignments (from the last shape optimization iteration) to shrink this mask down to a better initial guess for the region containing S_{i+1}. More specifically, we again superimpose boundary patches from the exemplar over their assigned locations on the boundary of S_i, but this time we try to predict which of the pixels inside S_i should belong to the mask of S_{i+1}, as illustrated in Figure 4 (right). Thus, the interior of S_i is seeded with pixels which are predicted to belong to S_{i+1} with a sufficiently large weight. The mask is then shrunk to include only these seeded pixels. Seeded pixels with high weights form the initial guess for S_{i+1}, while those with somewhat smaller weights define the remaining region of the mask, within which the shape is allowed to evolve in the course of the optimization.

The initialization of each new layer is done via this seeding mechanism in the coarsest resolution, where boundary patches are large enough to fully cover the interior of the previous shape. A similar step is repeated at the beginning of each resolution level to recreate an accurate mask for the current level at the new resolution, and to refine the shape boundary. After this step, shape optimization proceeds as described before.

Continuous control maps. In our experiments we found that the subsequent texture synthesis process can sometimes be improved by switching from a discrete layer map to a continuous control map. Specifically, for each pixel x inside the shape S_i its continuous map value is set to

$$i + \frac{d_{i-1}(x)}{d_{i-1}(x) + d_{i+1}(x)}, \tag{5}$$

where $d_{i-1}(x)$ and $d_{i+1}(x)$ are the distances from x to S_{i-1} and S_{i+1}, respectively (see Figure 7). The distances are obtained by performing distance transforms over S_i. Distance transforms were also used to create control maps by Lefebvre and Hoppe [2006].

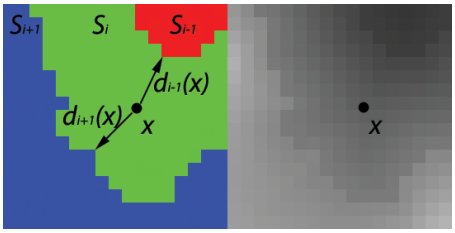


Figure 7: *The distances from a point inside a shape to the neighboring shapes are used to convert a discrete label map (left) to a continuous one (right).*

4 Applications and Results

We found the layered shape synthesis approach described in the previous texture to be effective for synthesis of inhomogeneous textures, such as those resulting from natural aging or weathering processes. The synthesis process for such textures consists of three successive phases depicted in Figure 6: layer decomposition, shape synthesis, and texture synthesis.

The layer decomposition phase takes an inhomogeneous texture sample as input, and generates a *layer map* which encodes the distinct homogeneous texture regions (layers) present in the input, by assigning a unique label to all of the pixels belonging to the same layer. Following the texture classification approach advocated by Varma and Zisserman [2003], we first segment the exemplar's pixels by performing K-Means clustering on the N^2 dimensional feature vectors formed by concatenating the values of each pixel's $N \times N$ neighborhood. We currently rely on the user to specify K as the number of distinct textures visible in the exemplar, typically between 3 and 5. N is set to 15 in all of our examples. The resulting clusters should roughly correspond to the distinct textures present in the exemplar. Points closer to the cluster centers are due to pixels that are the more typical representatives of the corresponding textures, while points far away from the center come from areas of transition between textures. Let $C_1, C_2, \ldots, C_K$ be the resulting clusters, ordered by the user such that C_1 is the bottom layer (original "clean" surface), and C_K is the top layer (most "weathered" surface). For the layer map, we set the foreground pixels of L_i to be $C_i \cup C_{i+1} \cup \ldots C_K$. If the clusters have been ordered properly, the sequence $L_1, L_2, \ldots, L_K$ expresses a possible natural evolution and spread over time of the weathering phenomenon captured by the exemplar.

In the last phase we use the new layer map obtained in the shape synthesis phase (Section 3) to synthesize a new inhomogeneous texture. This is done by applying the "texture-by-numbers" framework [Hertzmann et al. 2001]. While applying texture-by-numbers directly on the layer map often produces satisfactory results, their visual quality may be further improved by switching from a discrete layer map to a continuous one, as described in the previous section. The continuous map may be seen as a heuristic analogue for the

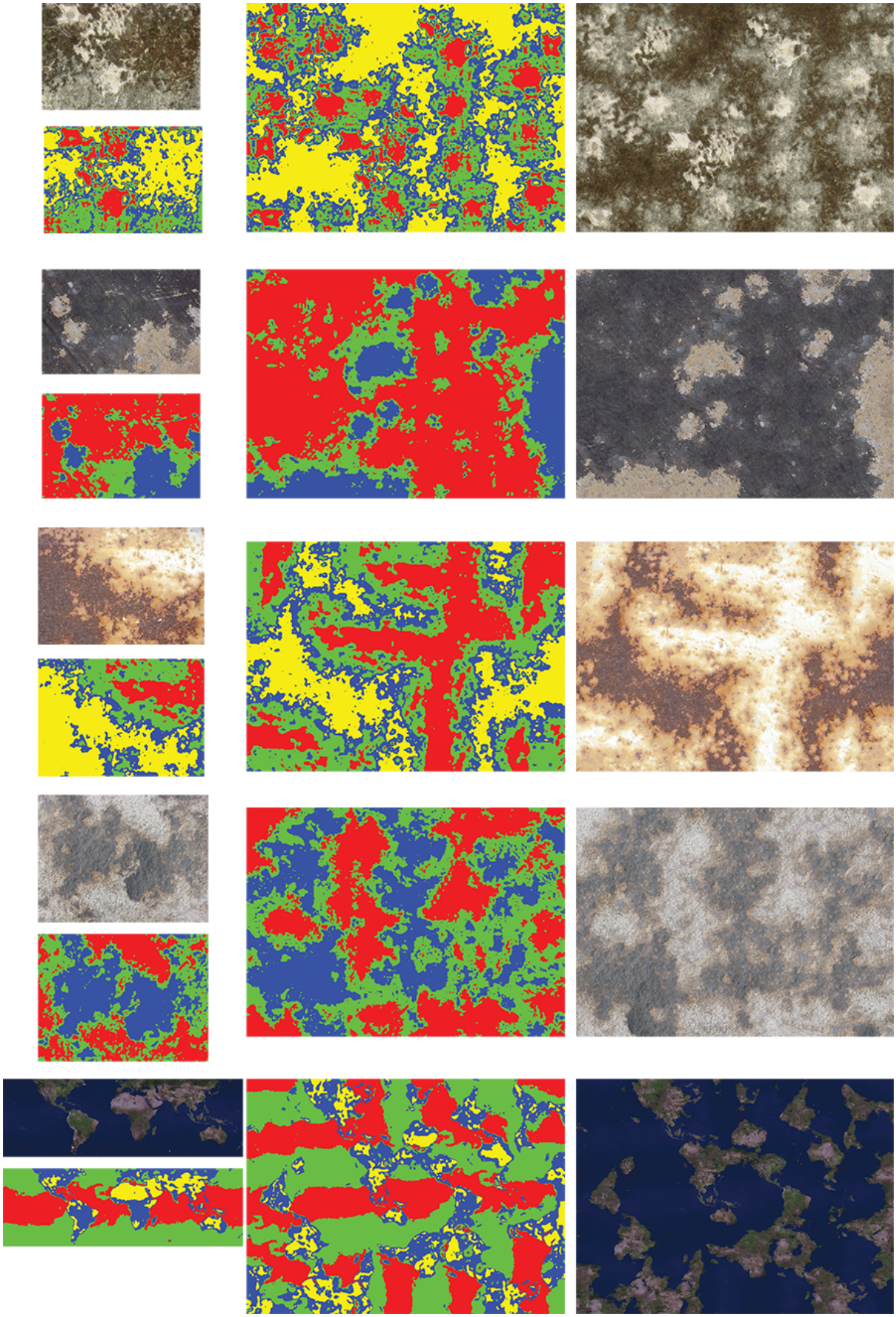


Figure 8: *A variety of results produced by our method. Left: input exemplars and their decompositions to layers; Middle: synthesized layer map; Right: final synthesized result.*

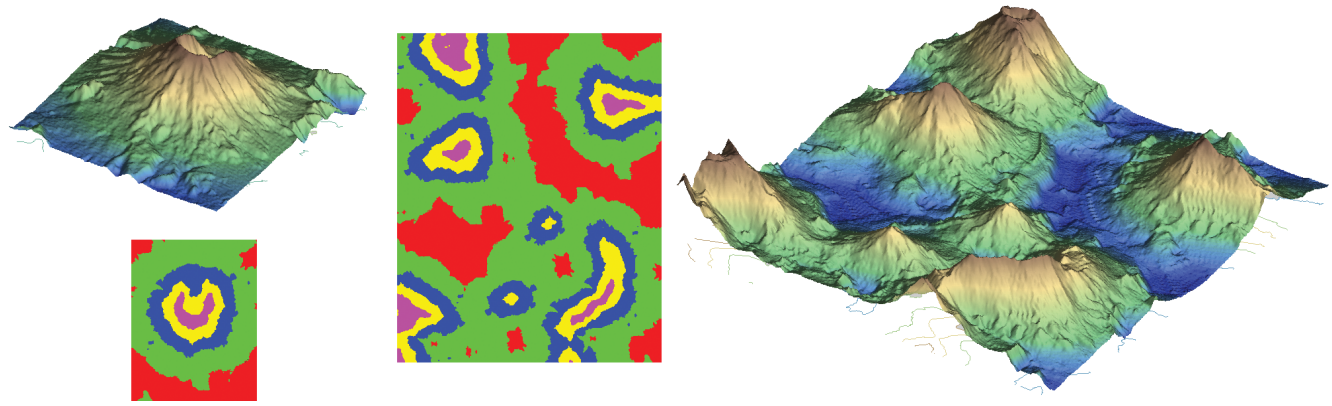


Figure 9: *Terrain generation by height map synthesis using our method. Left : input height map and its decomposition to layers; Middle: the synthesized layer map; Right: final synthesized result.*

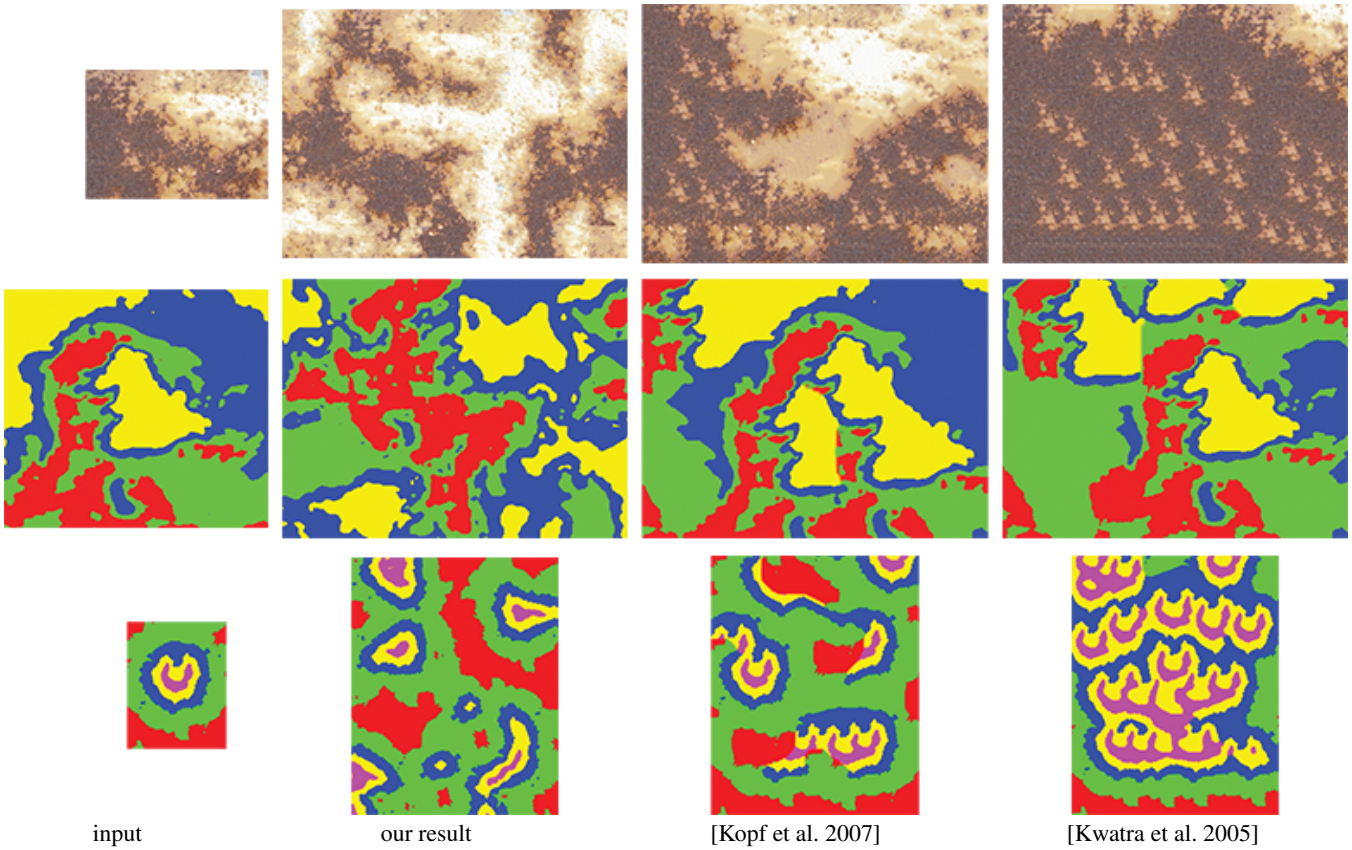


Figure 10: *A comparison of our approach to texture optimization with and without histogram matching. In the top row both methods perform synthesis directly from the exemplar. In the bottom two rows, we attempt to use texture optimization to synthesize a new layer map.*

weathering degree map of [Wang et al. 2006].

We experimented with our method on a variety of natural inhomogeneous textures. Some results are shown in Figures 1, 6, and 8. The examples are quite varied, showcasing phenomena such as corrosion, rust, lichen, and peeling paint. They differ significantly not only in their appearance, but also in their underlying layer structure, as may be seen from the layer maps extracted by our method. Our method successfully reproduces the global layer structures, the local fine details of the shape boundaries, and the final appearance of these textures. Our method is not limited to such textures, however. Other inhomogeneous textures that exhibits a layered structure with nested shapes may be synthesized as well. For example, we have synthesized a plausible fictional satellite image from one of Earth (bottom row in Figure 8). Since we use patch-based shape synthesis, some repetitions do occur, but they are mostly difficult to spot, as they are explicitly limited in our approach by our fair boundary sampling and assignment mechanisms.

Another application of our approach is example-based terrain synthesis, as demonstrated in Figure 9. Height maps used to represent terrains may also be considered as non-stationary textures for

Figure 11: *Inpainting. Left: original, right: our result.*

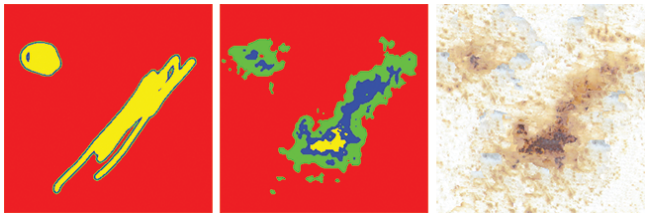


Figure 12: *Example of user control via a painting interface.*

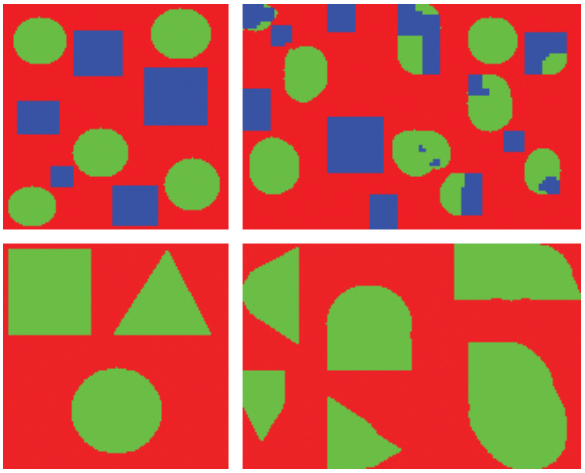


Figure 13: *Examples of failure cases. Top: violation of the layer model. Bottom: failure to reproduce specific shapes.*

which our layered shape synthesis approach fits perfectly. For the generation of the layer map, a simple quantization of the height map is used. The boundaries of layers resemble contour lines in a topographic map. In this application, which is similar to texture synthesis described before, the shape synthesis phase generates a new topographic layout for the synthesized terrain and the texture synthesis phase adds the fine details.

The computation time of our method is dominated by the "texture by numbers" phase, which takes up to five hours for a 800×600 image using 5×5 neighborhoods. The time it takes to synthesize a new layer map depends on the total length of the shape boundaries. The complexity of the optimization step is linear in the number of nearest neighbor calls for each boundary patch. We use approximate nearest neighbor search via locally sensitive hashing [Datar and Indyk 2004]. It takes our unoptimized code 20–30 seconds on average to complete one optimization iteration for an 800×600 image. Typically, 5–10 iterations are used at each resolution level, so the execution time per layer is up to 30 minutes.

We compare our method to two previous example-based texture synthesis methods: texture optimization [Kwatra et al. 2005] and texture optimization with color histogram matching [Kopf et al. 2007]. Figure 10 shows the results of the three methods side-by-side. The top row shows the final synthesis result on a texture of a rusting surface. Kwatra's method is suited for stationary textures, and exhibits multiple obvious repetitions making the result quite dissimilar from the exemplar. Kopf's result matches the global color statistics of the exemplar, and produces a better result, but some repetitions are still apparent, and some regions of the synthesized texture do not have a similar counterpart in the exemplar (such as the large region of lighter rust near the center). It is also interesting to examine whether these previous methods are able to synthesize the layer map, rather than synthesizing the texture directly, and this is done on two examples shown in the two bottom rows of Figure 10. The middle row is a layer map extracted from a cloudy sky texture, while in the bottom row the layer map is from the terrain in Figure 9. In both of these examples, the previous methods generate more repetitions of entire regions of the layer map, and in several places there are direct transitions between non-adjacent layers, which are not present in the input map.

Figure 11 shows an inpainting example, where a hole is filled inside an inhomogeneous texture. While the result is obviously not identical to the original image, it is quite plausible, and the filled region blends well with the original parts. The layer map inside the hole is initialized randomly. Since our method modifies the entire layer map, after each optimization step we reset the layer map outside the hole back to the original one.

Figure 12 demonstrates the feasibility of controlling the result of the synthesis via a painting interface. The exemplar and its layer map are shown in the third row of Figure 8. The user sketches in yellow the approximate position where rust should be, and the resulting sketch serves as the initialization for the shape synthesis phase. Thin strips of blue and green pixels are automatically added by the system, since the yellow layer shape is supposed to be nested inside the blue and the green layer shapes. In order to avoid changing the user sketched shape too much, fewer resolution levels are used by the shape synthesis method, resulting in the middle image in Figure 12, while the rightmost image is the synthesized texture.

Limitations: Our approach makes two basic assumptions: (1) the control map consists of an ordered set layers, nested within each other; (2) the proposed shape similarity measure captures all the shape characteristics that one aims to reproduce. Violation of either of these assumptions may lead to a failure, as discussed below.

(1) The first type of failure is demonstrated by the synthetic example in the top row of Figure 13. Here we green and blue regions that are independent of each other in the input (e.g., a natural texture whose appearance results from two independent processes). Our approach assumes a layered model and generates the green layer first, followed by the blue layer. As a result, the relations between the green and blue regions are not preserved, and several blue regions are synthesized inside green ones.

(2) The proposed similarity measure characterizes shapes by the local appearance of their boundaries, without considering the shape as a whole. Thus, it is better suited for fine-scale unstructured shapes and fractal-like boundaries. More structured elements might be better handled by other models. For example, the locations of the deserts in the bottom row of Figure 8 might have been reproduced better using the context-aware textures framework [Lu et al. 2007]. The bottom row of Figure 13 shows another synthetic example where the easily recognizable distinct shapes in the input map appear mixed in the map generated by our method.

5 Conclusion

We have presented a novel example-based method for synthesis of control maps suitable for non stationary textures, such as those resulting from weathering. To that end, we have developed a new

powerful example-based shape synthesis algorithm that represents shapes as a collection of boundary patches at multiple resolution, and synthesizes a new shape from an example by optimizing a bidirectional similarity function. Applications of our method include synthesis of natural textures and terrain generation.

In future work we hope to extend the method of shape synthesis to a larger set of textures, for example, textures that do not exhibit a clear hierarchy of layers, and textures with larger structures. Our current measure emphasizes boundary similarity over other properties, such as area to boundary length ratio, which is maintained only implicitly. We would like to gain a better understanding of the relations between such properties, and experiment with various extensions of our similarity measure.

We would also like to discover additional applications of our shape synthesis approach. In particular, it would be interesting to explore the applicability of such an approach to the synthesis of 3D shapes.

Acknowledgments: This work was supported in part by grants from the Israel Ministry of Science, and from the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities. The authors would also like to thank the anonymous reviewers whose suggestions were greatly helpful.

References

ASHIKHMIN, M. 2001. Synthesizing natural textures. In *Proc. Symp. Interactive 3D Graphics*, 217–226.

BHAT, P., INGRAM, S., AND TURK, G. 2004. Geometric texture synthesis by example. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, ACM, New York, NY, USA, 41–44.

BOSCH, C., PUEYO, X., MÉRILLOU, S., AND GHAZANFARPOUR, D. 2004. A physically-based model for rendering realistic scratches. *Computer Graphics Forum 23*, 3 (Sept.), 361–370.

DATAR, M., AND INDYK, P. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. SCG '04*, ACM Press, 253–262.

DESBENOIT, B., GALIN, E., AND AKKOUCHE, S. 2004. Simulating and modeling lichen growth. *Computer Graphics Forum 23*, 3 (Sept.), 341–350.

DORSEY, J., AND HANRAHAN, P. 1996. Modeling and rendering of metallic patinas. In *Proc. SIGGRAPH '96*, Addison Wesley, 387–396.

DORSEY, J., EDELMAN, A., JENSEN, H. W., LEGAKIS, J., AND PEDERSEN, H. K. 1999. Modeling and rendering of weathered stone. In *Proc. SIGGRAPH '99*, ACM Press, 225–234.

DORSEY, J., RUSHMEIER, H., AND SILLION, F. 2008. *Digital Modeling of Material Appearance*. Computer Graphics. Morgan Kaufmann / Elsevier, Dec. . 336 pages.

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

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. *Proc. ICCV '99 2*, 1033–1038.

GU, J., TU, C.-I., RAMAMOORTHI, R., BELHUMEUR, P., MATUSIK, W., AND NAYAR, S. 2006. Time-varying surface appearance: acquisition, modeling and rendering. *ACM Transactions on Graphics 25*, 3 (Proc. SIGGRAPH 2006), 762–771.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. *Proc. SIGGRAPH '95*, 229–238.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. *Proc. SIGGRAPH 2001*, 327–340.

HERTZMANN, A., OLIVER, N., CURLESS, B., AND SEITZ, S. M. 2002. Curve analogies. In *Proc. 13th Eurographics Workshop on Rendering*, Eurographics Association, 233–246.

KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. 2007. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics 26*, 3 (Proc. SIGGRAPH 2007), 2.

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

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

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics 25*, 3 (Proc. SIGGRAPH 2006), 541–548.

LU, J., GEORGHIADES, A. S., GLASER, A., WU, H., WEI, L.-Y., GUO, B., DORSEY, J., AND RUSHMEIER, H. 2007. Context-aware textures. *ACM Trans. Graph. 26*, 1, 3.

MERILLOU, S., DISCHLER, J.-M., AND GHAZANFARPOUR, D. 2001. Corrosion: simulating and rendering. In *Proc. Graphics Interface 2001*, Canadian Information Processing Society, 167–174.

SCHRIJVER, A. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*, vol. A. Springer-Verlag, Berlin Heidelberg.

SIMAKOV, D., CASPI, Y., SHECHTMAN, E., AND IRANI, M. 2008. Summarizing visual data using bidirectional similarity. In *Proc. CVPR 2008*, IEEE Computer Society.

VARMA, M., AND ZISSERMAN, A. 2003. Texture classification: Are filter banks necessary. In *Proc. CVPR 2003*, IEEE, 691–698.

WANG, J., TONG, X., LIN, S., PAN, M., WANG, C., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Appearance manifolds for modeling time-variant appearance of materials. *ACM Transactions on Graphics 25*, 3 (Proc. SIGGRAPH 2006), 754–761.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. *Proc. SIGGRAPH 2000*, 479–488.

WEI, L.-Y., HAN, J., ZHOU, K., BAO, H., GUO, B., AND SHUM, H.-Y. 2008. Inverse texture synthesis. *ACM Trans. Graph. 27*, 3, 1–9.

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G., 2009. State of the art in example-based texture synthesis. Eurographics 2009 State of The Art Report, April.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2004. Space-time video completion. In *Proc. CVPR 2004*, vol. 1, 120–127.

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 (Proc. SIGGRAPH 2003), 295–302.