Synthesis of Complex Image Appearance from Limited Exemplars

OLGA DIAMANTI
ETH Zurich
CONNELLY BARNES
Adobe Research and University of Virginia
SYLVAIN PARIS and ELI SHECHTMAN
Adobe Research
and
OLGA SORKINE-HORNUNG
ETH Zurich

Editing materials in photos opens up numerous opportunities like turning an unappealing dirt ground into luscious grass and creating a comfortable wool sweater in place of a cheap t-shirt. However, such edits are challenging. Approaches such as 3D rendering and BTF rendering can represent virtually everything, but they are also data intensive and computationally expensive, which makes user interaction difficult. Leaner methods such as texture synthesis are more easily controllable by artists, but also more limited in the range of materials that they handle, for example, grass and wool are typically problematic because of their non-Lambertian reflectance and numerous self-occlusions. We propose a new approach for editing of complex materials in photographs. We extend the texture-by-numbers approach with ideas from texture interpolation. The inputs to our method are coarse user annotation maps that specify the desired output, such as the local scale of the material and the illumination direction. Our algorithm then synthesizes the output from a discrete set of annotated exemplars. A key component of our method is that it can cope with missing data, interpolating information from the available exemplars when needed. This enables production of satisfying results involving materials with complex appearance variations such as foliage, carpet, and fabric from only one or a couple of exemplar photographs.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Image Manipulation—*Texturing*

General Terms: Algorithms, Design

This project was funded in part by the ERC Starting Grant iModel (StG-2012-306877).

Authors' addresses: O. Diamanti (corresponding author), Department of Computer Science, ETH Zurich, CNB G 82.2, Universitaetstrasse 6, 8092 Zurich, Switzerland; email: olga.diamanti@inf.ethz.ch; C. Barnes, University of Virginia, Charlottesville, VA; S. Paris and E. Shechtman, Adobe Research; O. Sorkine-Hornung, ETH Zurich, Universitaetstrasse 6, 8092 Zurich, Switzerland.

DOI: http://dx.doi.org/10.1145/2699641

Additional Key Words and Phrases: Texture synthesis, material editing, image-based editing, texture interpolation, natural material synthesis, missing data

ACM Reference Format:

Olga Diamanti, Connelly Barnes, Sylvain Paris, Eli Shechtman, and Olga Sorkine-Hornung. 2015. Synthesis of complex image appearance from limited exemplars. ACM Trans. Graph. 34, 2, Article 22 (February 2015), 14 pages.
DOI: http://dx.doi.org/10.1145/2699641

1. INTRODUCTION

Editing materials in a photograph is a powerful operation that can boost the users' creativity, as it allows to create more visually appealing images. In this article, we are interested in manipulating the appearance of common materials such as grass, foliage, moss, wool, carpet, or stone. We enable a fine-grain control of the appearance of these materials for a variety of applications, including weathering, interior and exterior design, landscaping, and cloth design. For instance, with our approach, one can create a flower topiary with plausible shading cues so that it can be inserted into a photograph of a garden. Or, in another scenario, users can edit a picture of a t-shirt to render it with several different types of fabric to preview various options.

Material editing is, however, a delicate task; to produce good results, one has to preserve the complex interplay between material properties, illumination, and viewing direction. As an example, due to self-occlusions, the appearance of grass depends heavily on the viewing direction and lighting environment; not modeling these variations yields unnatural looking grass. In addition, natural materials often vary continuously, such as between various colors and/or weathering levels, further adding to the complexity of the task. Enabling artist control in these conditions becomes particularly challenging: an overly simplified appearance model like a repetitive texture may be easy to manipulate, but fails to capture the richness of most materials; too-detailed representations like 3D geometry and BRDFs are difficult to author. In this work, we seek to address this tension by building upon texture synthesis and texture interpolation.

Existing texture synthesis tools can handle many regular and semiregular materials such as brick walls and straw [Efros and Leung 1999; Wei and Levoy 2000] and reproduce their appearance

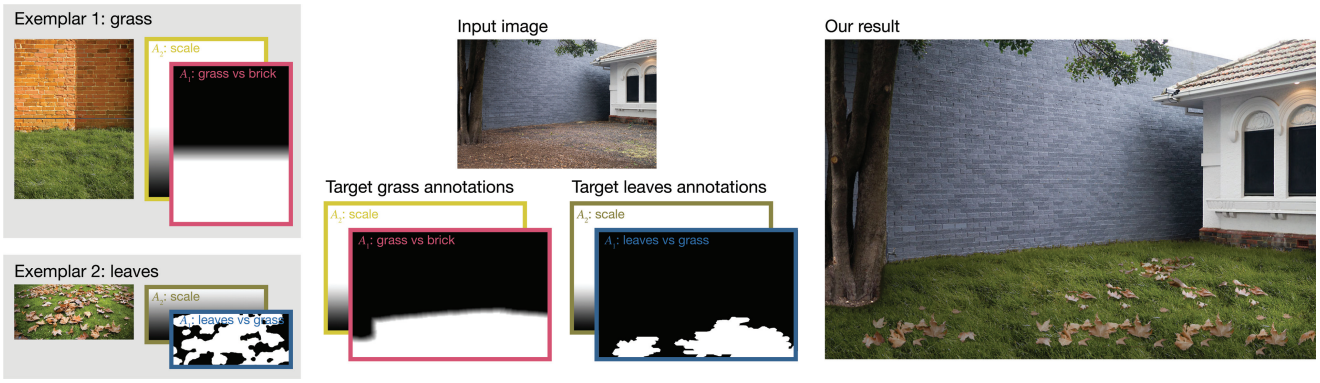


Fig. 1. Replacing the dirt in an image by a lawn covered with leaves. The grass and leaves exemplars are annotated to indicate the grass region and the scale of the grass. The user specifies the desired annotation values for the target image, and our algorithm synthesizes the result. Standard texture representation would fail to handle the intricate occlusions in such an example, or would introduce unsightly repetitions, while capturing a BTF of a size on the order of a lawn is infeasible with existing techniques. Our approach produces plausible results for materials with complex appearance using only images downloaded from the Internet and minimal user input.

under varying lighting conditions [Fang and Hart 2004], using small images as exemplars. The space of possible appearances for stochastic and complex materials such as grass and foliage cannot be represented by a single, small exemplar, which poses a challenge to these methods. At the other end of the spectrum, one could resort to more complex representations such as bidirectional texture functions (BTFs) [Dana et al. 1999], which can represent a much wider range of appearances. However, this dense sampling of the appearance space comes at the cost of practicality: the associated capture systems are cumbersome compared to casual image capture [Ngan and Durand 2006], and user control is only available through involved, dedicated interfaces [Kautz et al. 2007].

Our key insight is that many complex materials can be reproduced well by a few large-enough (i.e., larger than regular 2D texture patches) exemplars that sparsely sample the appearance space, provided these exemplars capture at least the extremes of the materials' appearances, and that we provide the ability to interpolate between them. This strikes a balance between simple but limited standard texture synthesis and rich but impractical BTF rendering. This model can be controlled by a simple and expressive annotation metaphor; we demonstrate standard aspects such as scale and orientation, as well as more sophisticated ones including weathering level, material transition, and viewing angle.

Algorithm summary. Figure 2 illustrates our pipeline. The input to our algorithm is a set of exemplar images representing the materials of interest; they can be user photos of the material or selected from an online photo collection. We opt for larger image exemplars rather than small texture patches; as we shall see in the results section, this is crucial in avoiding repetitiveness in the output and capturing the variability of the material. This variability is indicated by user-provided annotations that allow for control over given parameters of the material appearance. The user also provides an image to be edited as well as a set of annotations that describe the desired appearance in this target image using the same representation (Section 3).

From these data, our algorithm renders an image with the desired appearance variations. A key property of our approach is that we do not assume that all desired appearance parameter values are present in the exemplars, that is, the input data may not cover the entire annotation space, and the target annotation combinations need not have exact matches in the source annotation data. To cope with this situation, we first segment the input data into a few clusters that represent a consistent appearance (Section 4). Before the synthesis process, for each desired annotation value, we identify the relevant candidates from all clusters (Section 4.2) and assign them weights (Section 4.3). The final step is a coarse-to-fine synthesis process that generates a candidate patch for each selected cluster (Section 5.1) and produces the result by merging the candidates using the cluster weights (Section 5.2). As we shall see in the discussion section, the combination of interpolation with clustering and weighting is necessary for producing plausible results; skipping any of these steps significantly reduces the output quality.

Contributions. We demonstrate that complex and heterogeneous materials can be synthesized according to specified annotations. Our main contribution is twofold: first, we generate previously unseen appearance by using interpolation. This differentiates us from most current texture synthesis approaches (e.g., appearance manifolds [Wang et al. 2006]). Our technique's ability to generate plausible results even in the presence of large amounts of missing data is verified experimentally. The second part of our contribution is taking the clustering approach and introducing the interpolation weights as a more effective way to handle annotations. We found that all components of our clustering and sampling scheme are crucial in order to effectively combine interpolation with larger exemplars and continuous annotations as the ones we are using.

2. RELATED WORK

Texture synthesis. Many approaches exist to generate new image textures from exemplars, such as Efros and Leung [1999], Wei and Levoy [2000], Lefebvre and Hoppe [2005, 2006], Barnes et al. [2009], Kwatra et al. [2005], and Han et al. [2008]. They work well for homogeneous, flat 2D textures with scale and orientation changes, but do not handle the other variations in which we are interested, for instance, in illumination or view angle. Our work is closer to the texture-by-numbers technique that enables variations guided by annotations [Hertzmann et al. 2001]. Many recent methods follow this paradigm; the control maps of Rosenberger et al. [2009] and the texton masks of Zhang et al. [2003] are examples of guiding annotations. However, Rosenberger et al. [2009] assume a

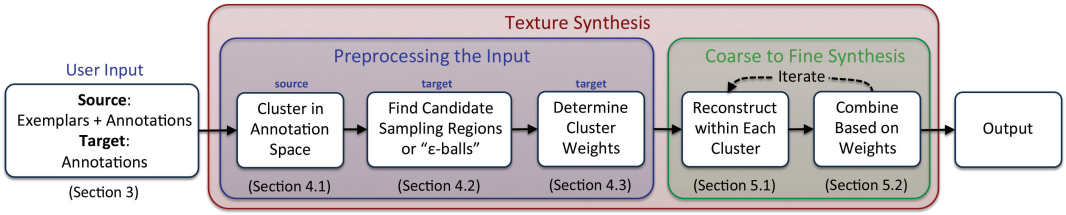


Fig. 2. Our pipeline: user input—source and target (Section 3), preprocessing of source and target data to generate constraints for the synthesis (Section 4) and coarse-to-fine synthesis (Section 5).

layered model for the material, with hard boundaries between the intermediate layers, and is thus unsuitable for rendering smooth material transitions. The texton masks of Zhang et al. [2003] only serve to ensure that prominent texture features are preserved during synthesis; they do not allow for the kind of control over material appearance that we introduce here, particularly for more complex materials such as grass or tree foliage. Moreover, methods based on image analogies only utilize the annotations inside the similarity metric and thus often fail to adhere to the user's intent (see Section 6). Finally, they require an exemplar for each possible annotation combination, which may be an unrealistic requirement in practice. We propose a more robust way to exploit annotations, and additionally deal with missing annotation data by texture interpolation that is not otherwise possible using current approaches.

Our interpolation scheme is inspired by previous work for texture interpolation [Matusik et al. 2005; Risser et al. 2010; Darabi et al. 2012; Park et al. 2013]; more specifically, we build upon image melding [Darabi et al. 2012], which uses gradient energies to allow for smooth texture interpolation. However, in all these works, the exemplars between which interpolation happens are known and identified beforehand, and separated into small discrete images that are typical of the material. This is not true in our case, where we are interpolating between large exemplars of complex, heterogeneous materials, with large variability present in each exemplar. Since this variability is captured by the annotations, the two extremal appearances between which interpolation is applied are determined differently for each pixel of the target image, depending on the user's specification for the target appearance. Precomputing interpolated appearances between all available pairs of extremal material appearances to satisfy any possible user request would be infeasible. Instead, we devised the use of clustering of the input exemplars as a first way to handle the variability, and a sampling scheme for handling the annotations as an easier way to automatically infer which exemplars to use for filling in the missing data and how much importance to assign to each cluster for a particular user-specified target appearance. This sets us apart from other texture interpolation techniques, even multimaterial ones [Ruiters et al. 2013].

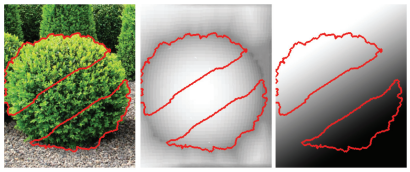
To model lighting and foreshortening effects, some techniques apply texture synthesis on a proxy geometry. In Bonneel et al. [2010], users generate a coarse 3D model of a scene and synthesize texture to augment it with details and natural materials. Johnson et al. [2011] apply texture synthesis to refine 3D renderings. Both works only let users control the final result through the 3D geometry of the scene; we work in image space and offer direct control over the material appearance using annotations. Eisenacher et al. [2008] use Bézier patches to gather and apply texture under arbitrary geometric transformations, but still focus on regular textures. Textureshop [Fang and Hart 2004] is also based on image manipulations and provides tools to model a 2.5D proxy that is later textured. Appearance variations are created by occlusions and shading stemming from a synthesized displacement map. The assumption is, however, that the base material is regular, its geometry representable as a displacement map, and all its appearance changes due to shading. While we follow a similar motivation, we aim to handle complex materials that cannot be modeled with a displacement map (e.g., grass, foliage); for this we rely on sophisticated texture synthesis and interpolation. Our generic annotations can represent a variety of phenomena, including (but not limited to) shading, weathering, and transitions between materials.

Some recent methods focus on the particular problem of simulating weathering effects [Lu et al. 2007; Wang et al. 2006]. They might yield more physically accurate weathering results, but are limited in scope. Our approach handles a variety of editing scenarios including weathering and, contrary to Lu et al. [2007], only utilizes standard photographs without the need for specific acquisition procedures. It also adds the ability to extrapolate missing data from the available exemplars.

BTF synthesis. BTFs enable very high-fidelity reproduction of texture [Dana et al. 1999], but their capture requires lab conditions and thus can be prohibitively expensive and time consuming as opposed to, for instance, texture acquisition from the Web or casual photography. Ngan and Durand [2006] simplify BTF acquisition, but still require a large number of photographs and a lab-capture setup. In Kautz et al. [2007], editing BTFs is achieved via an out-of-core architecture; LePage and Lawrence [2011] introduce an interface for breaking down a spatially varying BRDF into a foreground and background layer, for subsequent separate editing of the materials. Ruiters et al. [2013] demonstrate interpolation of BTF materials by separation into a height-map and a parallax-compensated BTF, however, binary feature masks must be provided by the user to preserve features. All these works show high-quality results, but BTFs still must be captured in a laborious process prior to their processing. An initial approach to bridge the gap between BTFs and exemplar-based textures can be found in Liu et al. [2001], where a small number of image inputs are used to synthesize a BTF. However, the assumption of a height-field type of surface geometry practically limits the range of materials that can be handled; materials such as grass and straw remain out of reach for this approach.

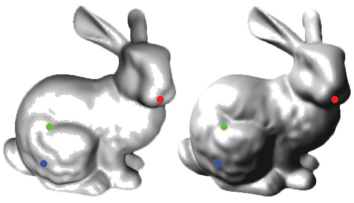
3. USER INPUT AND CREATION OF THE EXEMPLAR SPACE

The required user input consists of source input and target input data. The source input data comprises a set of images capturing the various appearances of the materials in question, and corresponding annotations for these images (Figure 3). The target data comprises the target image and target annotations, similar in spirit to the ones



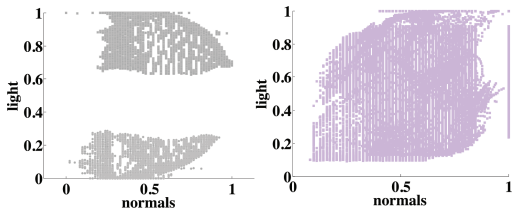
(a) source image (b) annotation: normals (c) annotation: light

Fig. 3. Step 1: Collecting the source input data. The users supply a source image, and optionally mask the portion of it to be used during synthesis (a). For illustration purposes, we choose to discard the middle portion of the source topiary. Also provided is a set of numerical annotations for the source, in this case for the normal (b) and the light direction (c).



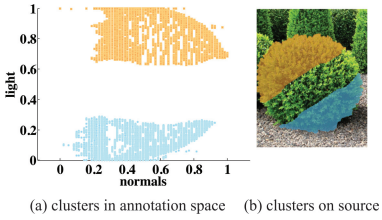
(a) target annotation: normals (b) target annotation: light

Fig. 4. Step 2: Collecting the target input data. In this case, the user wants to synthesize a bunny-shaped topiary onto a blank target image. For this, the user provides normal (a) and lighting (b) annotations corresponding to those in Figure 3 (here, they were created by rendering a bunny mesh). The marked red, green, and blue points will be used in the upcoming illustrations.



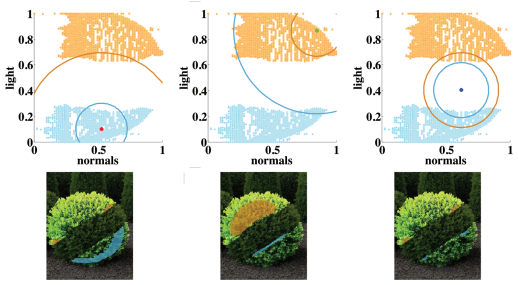
(a) source annotation space (b) target annotation data

Fig. 5. Step 3: Structuring the annotation space. By concatenating the source annotations for each pixel, we get the annotation space (a). In (b), we visualize the target annotation data in the same space. The points inside regions with limited availability of source data (the middle portion of the figure) will be interpolated using these annotations.



(a) clusters in annotation space (b) clusters on source

Fig. 6. Step 4: Clustering the source annotation data. Each cluster will be used to provide a candidate for the target image, and the candidate images will then be interpolated using appropriate weights. Given the annotation space of Figure 5(a), the clustering algorithm produces two clusters (a). These, when plotted on the source image, indicate the top and bottom portion of the source topiary, as expected (b).



(a) ε-balls, red point (b) ε-balls, green point (c) ε-balls, blue point

Fig. 7. Step 5: Assigning sampling regions (ϵ-balls) inside each source cluster for each target pixel. The figure shows the ϵ-balls for the three marked points on the target bunny annotations of Figure 4. Each point has two ϵ-balls, one per cluster; the clusters are as shown in Figure 6. The top row in (a), (b), (c) shows the ϵ-balls in the annotation space of Figure 5, colored according to the cluster. The second row shows the ϵ-balls overlaid onto the source image. The requirement for a minimum amount of candidate samples makes for unequally sized ϵ-balls; the further away the target data point from the the cluster points, the larger the ball radius.

Fig. 8. Step 6: Computing cluster weights. The appearance for the portions of the target image with target annotations located in the "holes" in the annotation space missing data (Figure 5(b)) will be interpolated from appearances in the source clusters, using cluster weights. In this visualisation of the cluster weights (clusters are as in Figure 6), the closer the color to orange, the higher the weight for the top topiary cluster, similarly for blue and the bottom cluster. The highlighted points are the same as in Figure 7; note the weight values as compared to the ϵ-ball radii. The result of the synthesis is in Figure 14.

of the source (Figure 4). Each annotation is a per-pixel scalar map with all values in the [0, 1] range; it associates the image appearance of the neighborhood around each pixel in the source images with a numerical value for a particular characteristic that is relevant for the appearance variations of the material. Typical examples of such characteristics are the viewing angle, illumination intensity, degree of weathering, local structure orientation and scale, etc. In the case where more than one material is to be synthesized, annotations can specify which type of material is contained in each source image and, in the case of transitions between two materials, possibly also quantify how far between the two materials a given appearance.

Our algorithm is not constrained to any particular way of acquiring the annotations. They can be generated via painting with an image authoring software, possibly with the assistance of more advanced tools such as diffusion curves [Orzan et al. 2008]. They can also be the output of dedicated methods that extract a particular characteristic from a source image (e.g., shape-from-shading for the surface normal and/or scene lighting); for characterizing the local shape, one can also use synthetic renderings of simple geometric primitives as annotations.

Optionally, the user can provide alpha matted (RGBA) source images as input. We do not consider transparency as a factor affecting material appearance, however, it can prove very helpful in synthesizing some types of material transitions, as we shall see in the results section.

By collecting all the values for the various characteristics for each pixel, we get a tuple that fully describes the local appearance around this pixel; the exemplar (or annotation) space for the particular synthesis task in question then is the set of all tuples for all pixels in

the source images (Figure 5(a)). Holes in the annotations indicate this data is missing for this part of the annotation space.

The source images do not need to capture the full spectrum of appearance variations. Our algorithm is able to synthesize appearance for characteristic tuples not present in the input data. However, we assume that the source images at least span the spectrum of appearance that the user would like present in the synthesis result; that is, we can synthesize appearance for any annotation point inside the convex hull of the source annotation data, but not outside it.

The dimensionality of the exemplar space depends on what the user considers important features of the material appearance. We have found that two or three dimensions are mostly sufficient to account for common variations in material appearances; however, other than an increase in runtime, our algorithm is not inherently constrained to low-dimensional exemplar spaces.

4. STRUCTURING THE EXEMPLAR SPACE

We now consider a material represented by an exemplar image E (created by concatenating all source images) and a corresponding n-dimensional annotation image A. The annotation data populate the n-dimensional annotation space.

4.1 Clustering in Annotation Space

The first step of the preprocessing is clustering the source data into clusters of similar appearance; each cluster will then generate independent synthesized results, and these results will be merged together. Regions of the annotation space well covered by source annotation data that are close to each other in value are assigned to one cluster; separation between the regions indicates the need for a different cluster, as shown in Figure 6. The intuition here is as follows (see also Figure 5): when it comes to synthesizing appearance associated with a particular target annotation combination, we need to check whether there are enough source appearance candidates to ensure a plausible result. If the target data point is in a coherent region of the annotation space well covered by source annotation data, then we can safely use samples from this region only to synthesize. When an appearance is requested with an annotation value in a "hole" in the annotation space, then we need to interpolate from source data at the boundary of the hole. Clustering provides a way of identifying these "holes". It also ensures that all extremes of material appearance variations are represented with candidates for a particular target appearance specification. As shall be displayed in the results, this is of particular importance when synthesizing appearance for target annotations not present in the input data.

Clustering in the annotation space is done using the mean shift algorithm [Comaniciu and Meer 2002] with bandwidth of 0.5 for all examples (recall that all annotation images are normalized to [0, 1]). In order to scale to high resolutions we first merge all source data points with identical annotation values into a single point. If the number of remaining points still exceeds a large threshold $c_{\max}$ (we used $c_{\max} = 10^8$), we perform quantization by binning before proceeding with the clustering. All points within the same bin are assigned to the same point, and bins are equally spaced along all dimensions so that the total number of bins is equal to $c_{\max}$.

4.2 Candidate Sampling Regions

During synthesis for each cluster, we will have a number of target n-dimensional annotation data points to synthesize. In order to allow for better control over the output appearance, we further restrict the set of candidate apperances that will be considered for a given target annotation to a neighborhood around this target annotation point.

Fig. 9. In this example, we experiment with different ways of annotating. We use the top source in (a), and annotate with 3D rendering of a sphere (source normals), shape-from-shading (t-shirt normals), and manual painting (source and t-shirt lighting) to get the result in (d). Sketching two more annotation images via diffusion curves (t-shirt orientations and scales, not shown) changes the woven pattern appearance (e). Switching the source changes the fabric of the t-shirt (f).

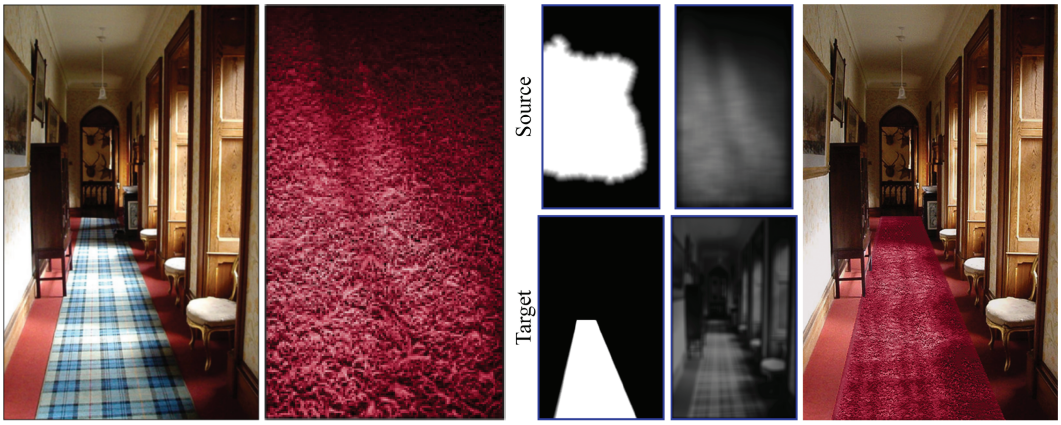
We now define this notion of neighborhood in the annotation space. For each such annotation point a, and for each cluster i, we need to collect all candidate exemplar data points belonging to i that lie in the neighborhood of a. We call these neighborhoods ϵ-*balls* and denote them by $B_i(a)$.

The ϵ-balls $B_i(a)$ are axis-aligned *ellipsoids* centered on the target annotation point a. The user defines the scaling parameters ϵ_k for each annotation dimension k. Selecting a small ϵ_k results in a strict sampling ball with candidates that follow the k-th annotation more closely, but may lead to more repetitive results. A larger (more permissive) sampling ball contains candidates that deviate more from the annotation, but also allows for more variation.

To avoid excessive repetition, we ensure that a minimum number of candidates N_i are always present in the ϵ-balls. Thus the radius of $B_i(a)$ along annotation dimension k is in fact:

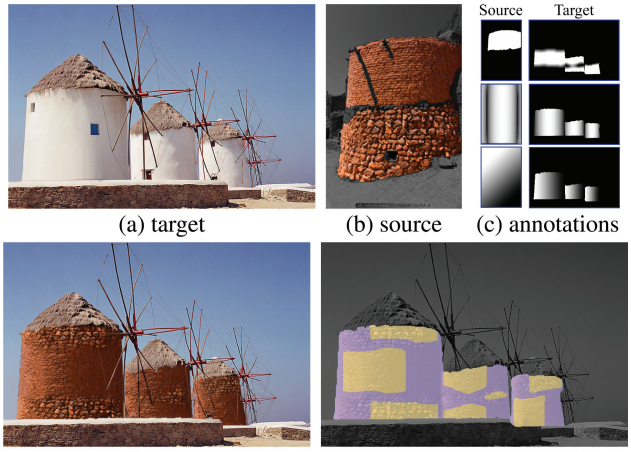
$$R_{ik}(a) = \max\{\epsilon_k, R^0_{ik}(a)\}. \quad (1)$$

where the minimal radius $R^0_{ik}(a)$ is the result of scaling an ellipsoid that initially has radii $(\epsilon_1, \ldots, \epsilon_n)$ by the smallest isotropic scale $R(a)$ so that it contains at least N_i candidate points. The scale $R(a)$ can be determined by a kd-tree that finds the nearest N_i points in a Euclidean metric [Mount and Arya 1998]; we only need to scale all source annotations by $1/\epsilon_k$ before inserting them into the tree, and then reverse the scaling by setting $R^0_{ik}(a) = \epsilon_k R(a)$. We set $N_i = \max\{1000,\ N_{i,\text{total}}/30\}$, where $N_{i,\text{total}}$ is the total number of points in cluster i, to ensure that we have at least 1000 neighbors in the ϵ-ball from which to choose. We visualize some ϵ-balls in Figure 7.



(a) target (b) source (c) annotations (d) result

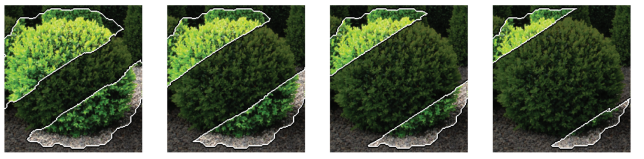
Fig. 10. A carpet example. A background image (a) and a source texture image (b) are chosen. The source and target are annotated according to lighting amount and fine-scale carpet fiber transitions (c) as well as scale (not shown), resulting in a new carpet (d).



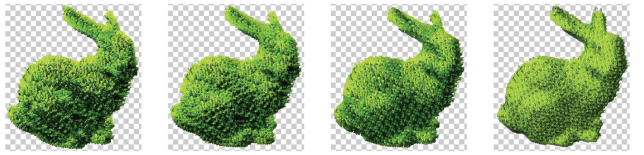
(a) target (b) source (c) annotations

(d) result (e) missing data (52%)

Fig. 11. Using one cluster per type of brick, and annotations for transitions between the two types of brick, normals, and light direction, we successfully handle the brick material and generate clean transitions between its two variants. Here 52% of the synthesized regions corresponds to previously unseen target annotation combinations (shown in purple).



(a) 35% missing (b) 55% missing (c) 70% missing (d) 80% missing



(e) result for (a) (f) result for (b) (g) result for (c) (h) result for (d)

Fig. 12. Our algorithm can synthesize visually plausible results even in the presence of missing data, with the quality of the result degrading gracefully as less data becomes available. Repetitiveness and failure to match the desired appearance only occur in the extreme case with 80% of missing data. See Figure 14(c) for the result without missing data.

4.3 Cluster Weights

At several points during synthesis, the appearances suggested by all clusters are weighted by appropriate weights to generate the final interpolated appearance. The interpolation weights represent how well the overall appearance of a given cluster matches the target specification, and thus how useful this cluster is when synthesizing appearance for that specification. We would like our weights to have three properties: (i) if a target annotation is far from all clusters, the weighting should roughly be in inverse proportion to the distance to each cluster; (ii) inside a cluster, the weights should be sparse with just that cluster's weight near 1 and the other clusters' weights near 0; (iii) on the boundary of two clusters, the weights should be high for these two clusters and low for the others.

The weights are themselves images of size equal to the size of the target image, whose value at a given target pixel depends on the annotation vector, a, of that pixel. For cluster i, the value of the weight image $W_i(a)$ is defined as

$$W_i(a) = \frac{1}{Z} \sum_{k=1\ldots n} \frac{\epsilon_k}{g(R_{ik}(a) - \epsilon_k)}. \quad (2)$$

Here $k = 1 \ldots n$ are the dimensions of the annotation space, Z is a normalization factor so the weights sum to 1 for each target annotated point a, and g is a clipping function that returns its argument if it is positive, otherwise returns a small clipping value that is $0.8 \min_a\{R_{ik}(a) - \epsilon_k | R_{ik}(a) - \epsilon_k > 0\}$, to make sure that target points whose ϵ-balls are fully inside a cluster still get higher weights for this cluster than points with ϵ-balls partially inside the cluster.

Our weights satisfy the three properties: (i) is satisfied by design. Inside a cluster $R_{ik}(a) - \epsilon_k$ tends to be small and so the given cluster's weight dominates, thus satisfying (ii). On the boundary between two clusters, the radii to these two clusters tend to be only a moderate factor larger than ϵ_k while all other radii are quite large, making these two clusters dominate in the weights, satisfying

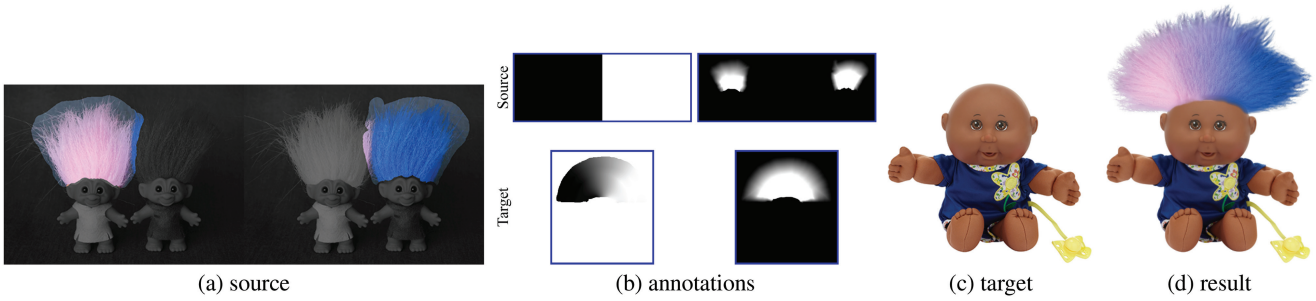


Fig. 13. Synthesizing colorful hair. The annotations (b) account for the color and the transitions of the hair in the source (a). Using these, we can add gradient colored hair to the target image (c).

property (iii). We show the weights for the target shown in Figure 4 and the clusters shown in Figure 6 in Figure 8.

5. COARSE-TO-FINE ITERATIVE SYNTHESIS

In this section we describe the actual synthesis process. Given our appearance clusters with their corresponding weights, we proceed to synthesize the output image by synthesizing for each cluster separately and combining the synthesized images. We take the same general patch-based texture synthesis approach as in multisource image melding [Darabi et al. 2012], for its smooth interpolation of both color and texture. Image melding starts with an initial guess for the target image at a low resolution; it then proceeds to iteratively synthesize candidate images from each of the sources one at a time, and merges the candidates using appropriate weights. This is done in a coarse-to-fine multiscale fashion: the merged result of the previous (coarser) scale is used as a common starting point for the synthesis iterations for all sources at the next (finer) scale. The "inner" synthesis iterations are done by means of the generalized PatchMatch [Barnes et al. 2010] algorithm; after a certain number of matching iterations, a color "voting" from overlapping patches takes place to generate a single color per pixel. The result of this process is a candidate image reconstructed from each source independently. A combination of the candidates using the weights produces an improved output image to be used as input for the next scale.

In our case, we use image melding to synthesize and combine images generated from each cluster separately, that is, our "sources" for melding are the various clusters; a number of improvements were necessary in order for this process to be applicable in our scenario. First, we generalize image melding to allow for multiple sources and annotations in the melding step. Additionally, the per-cluster "inner" synthesis step was modified to enable the constrained synthesis necessary for handling missing data, namely to disallow candidate patches that are outside the current cluster or the ϵ-balls for a particular target pixel. Finally, our enhanced melding process is able to handle matting by synthesizing using RGBA source and target images with alpha premultiplied.

For each cluster, we generate a different initial guess by copying to the target image a source patch that has most similar annotation and averaging overlapping patches for each pixel into a single image (a process called *voting* in Darabi et al. [2012]). We store the locations of the copied source patches in the initial Nearest-Neighbor Field (NNF [Barnes et al. 2010]) for each cluster.

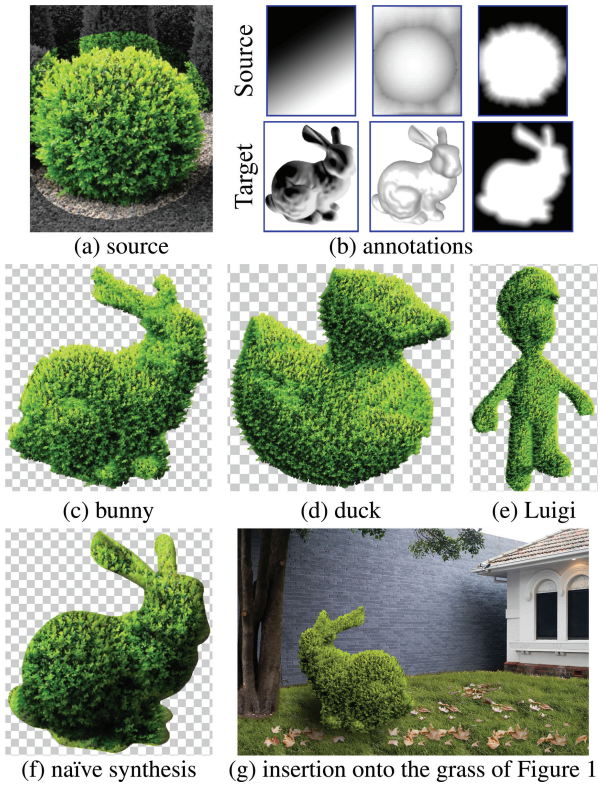
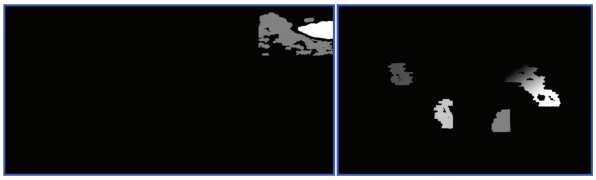


Fig. 14. We synthesize topiaries (c), (d), (e) by rendering annotation maps using 3D models. We added an alpha channel to the exemplar, which enables us to generate finely detailed boundaries not present in the 3D models. Our approach outperforms naïve synthesis (f) that uses only the center of the exemplar and simply modulates the result to convey lighting effects; this result appears artificially smooth, does not match the source in highlight and shadow colors, and lacks realistic boundaries. Our synthesized topiaries and grass can be creatively composited together to render a more complex scene (g) with little effort (here, we manually adjusted brightness and contrast of the synthesized topiary after its creation so as to match the colors of the synthesized grass). We also added a shadow to the composited result.

5.1 Reconstruction within Clusters

For each cluster i, we use generalized PatchMatch [Barnes et al. 2010] to select for each patch p_j in the target image the best matching source patch among the set of acceptable candidate source patches. This set consists of all source patches whose annotation point at the center pixel of the patch is inside the ϵ-ball $B_i(p_j)$ of the center pixel of the target patch; during PatchMatch, we reject

(a) source, with the weathered part magnified



(b) annotations

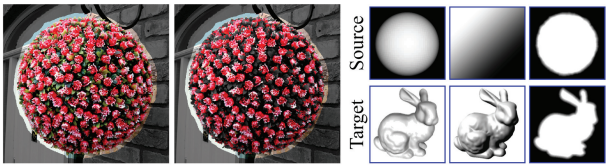
(c) our method, with missing data as an inset

(d) image analogies

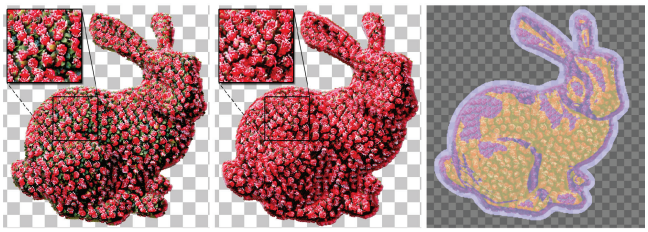
Fig. 15. Transfer of three differently weathered materials onto a brick wall. The brick exemplars were scaled to similar sizes during preprocessing. The source annotations are discrete weathering levels, and the target ones continuous (33% missing data, shown in purple in (c)). Compared to image analogies [Hertzmann et al. 2001] (d), our approach better handles smooth transitions and better matches the specified annotations.

all candidates that are outside these ϵ-balls. We use a weighted Euclidean patch distance on the channels, RGBA as well as the gradients of RGB as a similarity measure, with a high weight on the alpha channel to avoid mixing transparent and opaque areas. Once we have correspondence NNFs for each cluster, we create reconstructed RGBA images V_i for each cluster i by voting.

Rotation and scale control. We optionally allow patch scales and rotations to be constrained by auxiliary scale and rotation annotation images S, Θ that are specified on the sources and the target,



(a) full source (b) source with no leaves (c) annotations



(d) result for (a) (e) result for (b) (f) missing data for (e)

Fig. 16. In this example, missing data points are isolated and spread over the entire exemplar (annotations as in Figure 14). Our algorithm nonetheless succeeds in generating a bunny with the correct apparent shape by exploiting nearest neighbors available in the ϵ-balls. The images are better viewed in higher resolution. (e) has 52% of missing data, shown in purple.

but are otherwise exempt from the preprocessing stage of Section 4. In the typical PatchMatch scenario, the matching process searches for the rotation and scaling transformations for the source patch that best fit the target patch. We also employ such search, however, we first transform the source patch according to the difference between the annotated scales and orientations in source and target, prior to applying the rotation/scaling generated by the search. In the case of rotations, for example, we constrain the minimum and maximum angle for each (transformed) source patch to be within $[\theta_1 - \theta_2 - \delta_\theta, \theta_1 - \theta_2 + \delta_\theta]$, where θ_1 and θ_2 are source and target angles, δ_θ is a global tolerance, and target patches remain upright and are not transformed. This enables control over the generated orientations and increases the available number of candidate patches, since we do not need to restrict ourselves to patches of a particular orientation. Scale is constrained likewise in the log domain.

5.2 Combination Based on Weights

We now combine the reconstructed versions $\{V_i\}$ of the target, each independently created from cluster i, into an improved target for the next iteration in a way that smoothly interpolates texture and color. Even though the per-cluster images are synthesized independently, their textures at this stage are approximately registered, thanks to the PatchMatch-type synthesis of Section 5.1 (which includes image gradients) and the coarse-to-fine nature of the synthesis (ensuring that the synthesis for all clusters starts from the same image at each iteration). However, we still need smooth color interpolation.

We do this interpolation first by alpha-compositing each reconstructed image V_i on the background. We also retain the alpha as a separate component. We now convert to $L^*a^*b^*$ color space so that we can perform smooth interpolation between any discrete material colors by means of Poisson reconstruction on L^*. We additionally augment each reconstructed image with the gradient channels ∇L^*, so that the augmented images $\widetilde{V_i}$ have 6 channels: $(L^*, a^*, b^*, \alpha, \partial L^*/\partial x, \partial L^*/\partial y)$.

We wish to reduce our collection of 6 channel images to a single image based on our per-cluster weights from Section 4. For the nongradient components, we take a weighted average based on

the per-cluster weights W_i, while for the gradient components we choose, per pixel, that gradient with maximum norm $\|W_i \nabla L^*\|$. This generalizes the two-source blending algorithm of Darabi et al. [2012] to multiple sources. The result is a single image with 6 channels. The 3 channels $(L^*, \partial L^*/\partial x, \partial L^*/\partial y)$ are consolidated into a single channel L^* by performing screened Poisson reconstruction [Bhat et al. 2008] (to avoid blur, a high weight of 5 for the gradient channel gives best results). Finally, we convert back to RGBA space and multiply by the alpha to gain an improved target image that is alpha-premultiplied.

6. RESULTS

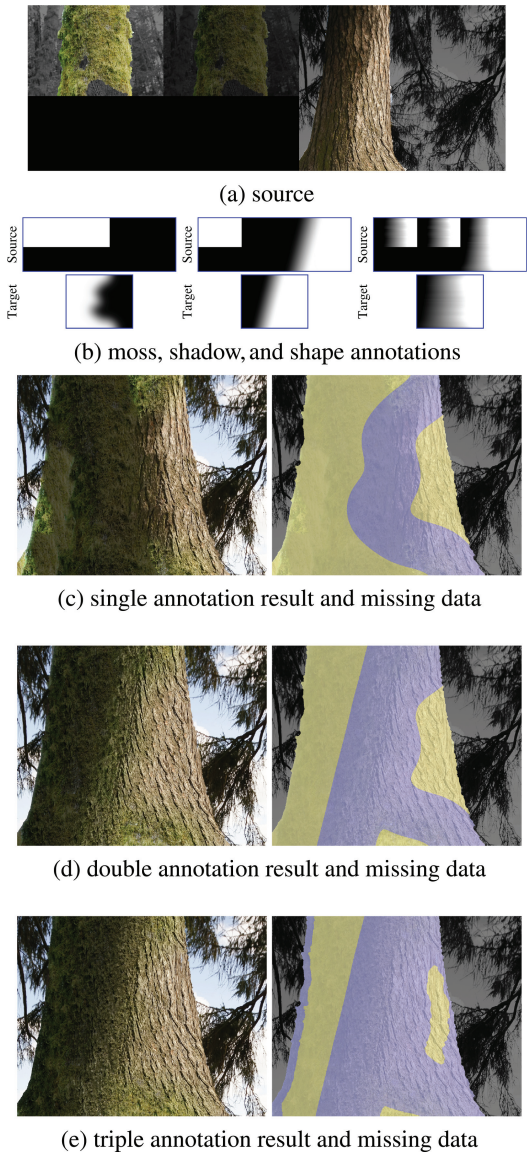
Our experiments demonstrate that our approach can synthesize a variety of complex materials. In most figures, in addition to the exemplars, annotations, and final result, we also show which regions were synthesized from existing data (in yellow) and from missing data (in purple). We consider that data is missing for a target annotation combination when there is insufficient source data in the neighborhood of this annotation, that is, we had to enlarge the ϵ-ball by increasing the radius along any dimension using the max in Eq. (1), Section 4.2. All images referenced in this section are best viewed in high resolution.

Structure of the annotation space. We first illustrate our approach on a scenario with a single cluster and negligible amount of missing data (Figure 10). In these conditions, our method is conceptually similar to previous texture synthesis work such as Zhang et al. [2003], with the difference that we only allow candidates from within an ϵ-ball, thus forcing the result to follow the annotations closely. Figure 9 shows another single-cluster example. This example also illustrates the flexibility of our annotations as they were acquired in four different ways, and shows that, given the annotations, one can easily change the exemplar to get a new appearance.

In Figure 11, we manipulate two different brick materials with two clusters, and manage to transfer this complex and spatially varying texture onto the windmills while also making the brick materials follow a new user-specified labeling pattern. Figure 17 illustrates the influence of the number of annotation maps on a mossy tree example. While too few annotations might be insufficient, after a certain point adding more annotation images becomes redundant and has only a minor effect. The existing annotation images already capture most of the useful information; adding better annotations improves how these data is used but we are limited in terms of appearance by the variation available in the exemplars, which remained unchanged.

Complex boundaries and transitions. Figures 11, 15, 20, 13, and 17 show that our approach can synthesize complex transitions between materials, which is key to adding effects such as weathering and rendering natural scenes, like moss on a tree trunk or tree foliage. The topiary of Figure 14 shows that we can reproduce the complex boundaries using alpha matting and a transition annotation image. We generated the annotations by rendering a 3D model that had smooth boundary and our technique was able to synthesize the finely detailed contour typical of foliage. This allows for complex compositing: in Figure 14 we show a result of inserting this topiary into the yard of Figure 1. For this case, we manually adjusted brightness and contrast to match the background and added a shadow.

Missing data. Most of our results are generated from incomplete exemplars, that is, some data is missing. Figure 12 systematically studies the robustness of our approach to such missing data.



(a) source

(b) moss, shadow, and shape annotations

(c) single annotation result and missing data

(d) double annotation result and missing data

(e) triple annotation result and missing data

Fig. 17. Varying the number of annotations. The user provides mossy and moss-free tree exemplars (a). The user also creates a dark moss exemplar by adjusting the brightness of the initial bright moss exemplar to simulate the appearance of moss in the shade. The available annotations (b) indicate the mossy regions, the lighting variation, and a horizontal annotation image that encodes the cylindrical shape. Using only the moss annotation image to indicate where moss should be yields a poor result (c), while adding the illumination annotation image greatly improves the result (d). Adding the third annotation image yields a more modest improvement (e). Missing data regions are shown in purple (42%, 57%, and 70%, respectively).

The main artifact that appears is an increased level of repetition when most data is missing (80%), where the result fails to match the specified annotations.

Missing data regions are usually concentrated in a large portion of the annotation space that is not represented in the exemplars. Figure 16 shows another, less common case where missing data points are isolated and spread over the entire space. In this example, users seek to generate a flower topiary without leaves, and to

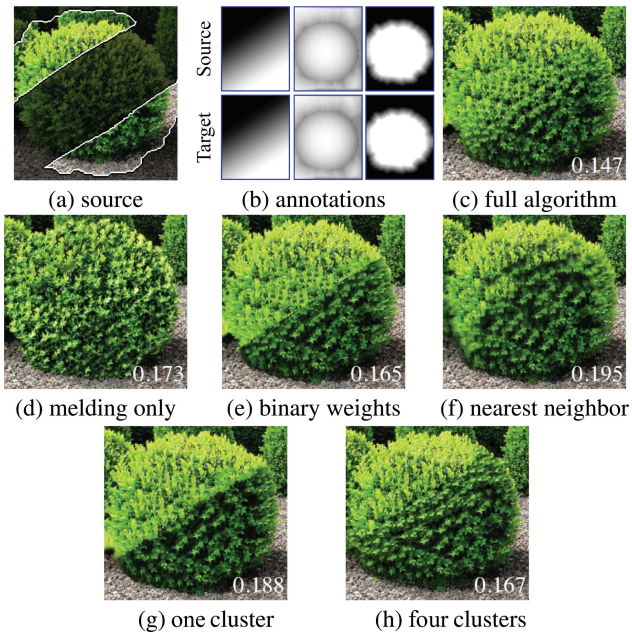


Fig. 18. Evaluation of the different components of our algorithm. Using melding only, without any annotations, yields a poor result exhibiting random and uncontrollable appearance variations. This is due to using a large topiary source with a lot of appearance variation without annotation control (d). If we use binary weights, such as weights quantized to 0 or 1 (e), the transition is not smooth. For (f), we select a single nearest neighbor in image space out of all patches from all clusters, instead of reconstructing inside each cluster and melding. The result deviates from the annotations, since the synthesis is biased against a portion of the valid data. Without clustering or with too few clusters (g), transitions are not smooth. Once there are sufficient numbers of clusters, results are not as sensitive to the number of clusters (c) and (h). See Figure 19 for explanation of the inset numbers.

do so they exclude the leaves from the source exemplar, leaving many holes spread over the entire exemplar. Our approach handles this case with a single cluster and approximates the appearance of missing data parts by looking at nearest neighbors available in the ϵ-balls.

Comparison to image melding. We already demonstrated in Section 2 the conceptual differences between standard texture interpolation techniques (e.g., image melding [Darabi et al. 2012]) in terms of determining the data being interpolated. There exists a second difference from image melding, namely the continuous annotation images that are crucial for producing the desired appearance. Figure 18(d) shows a result of only using melding without any annotations. For this example, we only used as sources the two discretely labeled image regions with the two extremal appearances (i.e., "top and lit", "bottom and in the shade") without any annotations, and hoped for image melding interpolation to generate the smooth geometry and lighting transitions the user asked for. Instead we get a relatively flat result exhibiting random and uncontrollable appearance variations, instead of a smooth dark-to-bright gradient. This is due to using a large topiary source with a lot of appearance variation without annotation control. Naturally, one could eliminate these variations by making the source image regions smaller but, as shown in Figure 12, restricting the source exemplars to small

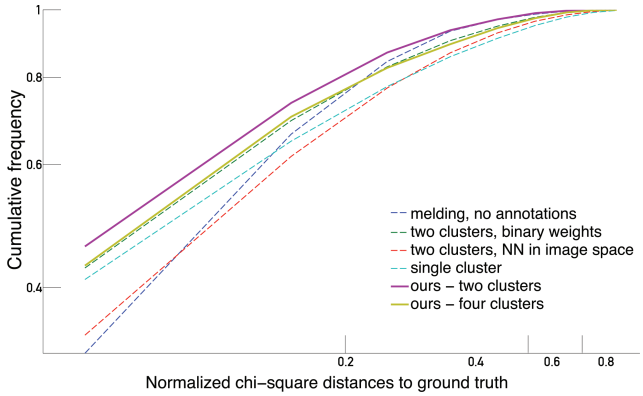
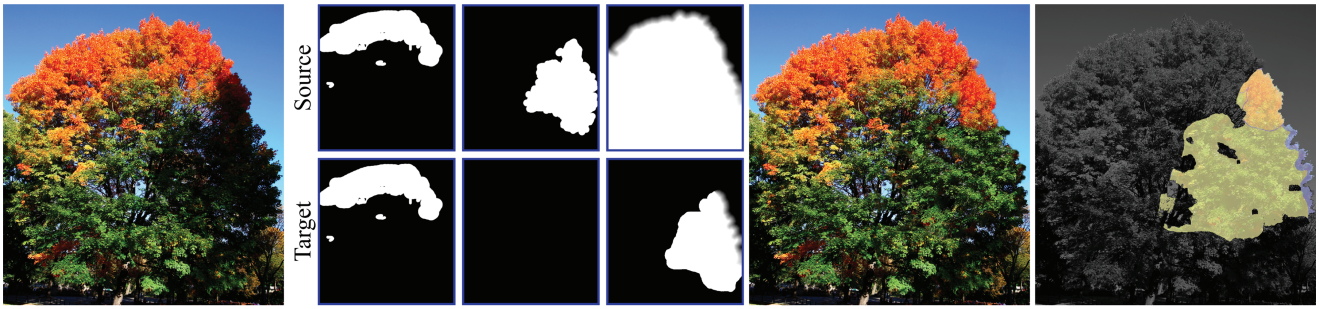


Fig. 19. Quantitative comparison of the figures in Figure 18. The graphs show the cumulative frequency of chi-square distances of the local (windowed) luminance histograms between the various results in Figure 18(d)–(i) and the ground truth in Figure 18(a). See the text for details on the computation of these distances; as a general rule, graphs that are more concentrated towards the top-left side of the figure are preferable, as they are more concentrated around smaller distance values and converge faster. The full method with two clusters outperforms the other results, with the four-cluster result coming second. The numbers shown in Figure 18 are the mean values of the above shown distances.

material chunks yields repetitive results that do not exhibit the rich variations present in most natural materials. Thus, texture interpolation is insufficient in the presence of larger exemplars; our annotations and ϵ-ball sampling scheme are still necessary to control the output appearance.

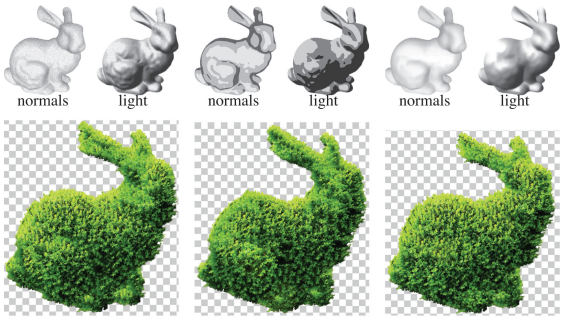
Necessity of clusters. We already demonstrated in Section 2 the necessity of clustering in the context of missing data, where melding is required, that is, clustering is a practical way of specifying what is being interpolated. Clusters are, however, also necessary in the single-material scenario in order to generate smooth transitions of appearance and allow for control in the presence of larger image exemplars. Figure 18(g) studies exactly this case: here, all available source patches are considered in a single cluster. In such a case, patch-based synthesis approaches will often try to only use patches from the "bottom and in the shade" part of the source to fill in the lower part of the rectangular hole, and those patches from the "top and lit" part of the source for the upper part of the hole. This is because such selection leads to good coherence in most of the final result, in particular around the edges of the hole. However, around the transition zone we get an abrupt, discontinuous switch in appearance. Instead, we want to make sure that all extremals of material appearance (in this case, patches from both the bottom and the top of the source) are represented and used as candidates until the last possible moment, when they are interpolated to give a smooth appearance transition. Hence, there is a need for two or more different clusters in the presence of missing data.

Quantitative evaluation. Figure 18 also studies the effect of using incorrect weights during the melding. A quantitative comparison of the different results in Figure 18 is shown in Figure 19. The comparison metric is obtained by calculating local histograms for sliding 11×11 windows on the reconstructed topiary result, and calculating an 8-bin histogram of the luminance channel inside each window. We then compare these histograms to those calculated on the equivalent windows in the ground-truth image, using



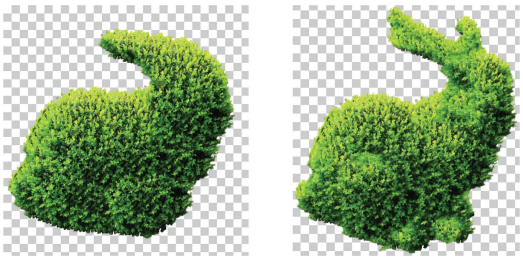
(a) original tree (b) annotations (c) result tree (d) missing data

Fig. 20. Modifying the lighting in a tree. The original tree image (a) has an unsightly shadow on the right. For the exemplar, we loosely matte the foliage regions. The annotations (b) indicate red foliage, shadow, and transition regions, with source annotations in the top row and target annotations in the bottom row. The resulting tree (c) has the shadow removed and matches the target colors. Some amount (d) of previously unseen annotations (about 5% of the total target region) is present in this example as well.



(a) noisy annotations (b) quantized annotations (c) blurry annotations

Fig. 21. Our method is robust to a reasonable amount of error in the annotations, which suggests it could handle different ways of generating the annotations. In this example, we test with different versions of the normal and light annotations of the bunny topiary in Figure 14(c) while keeping the source annotations and the transition annotation unmodified. In (a) we add 20% of noise, in (b) we use an annotation of lower detail (quantized), while in (c) we use blurred versions of the target annotations.



(a) trivial use of annotations (b) our result

Fig. 22. Simply incorporating the annotations in the patch distance without our clustering/sampling scheme can be hard to control in practise, especially in the presence of multiple annotations and an alpha channel. Here, we attempt to create a topiary bunny using the 3 annotations of Figure 14 as well as an alpha mask for the boundary. Adding these 4 channels to the color and gradient channels produces an augmented image of 13 channels, to be used by PatchMatch during synthesis. Appropriately weighting all these channels can be a hard task; in this case, the synthesis completely misses one ear (a) due to incorrect synthesis of the alpha channel; it also does not respect the annotations as well as our result (b).

the chi-square histogram distance as a difference measure. Figure 19 shows the cumulative histogram of these distances collected over all the overlapping windows. An ideal result would have most of the windows near distance zero, and thus would be more concentrated around the top-left part of the plot area. The figure shows our method outperforms the melding result as well as the other variations that don't make use of the full pipeline.

Figure 23 shows a quantitative evaluation on a rendered image. A scene, complete with surface normal information, was rendered with Mitsuba [Jakob 2010] and used as a source, with the three coordinates of the normals as annotations. We then synthesize for part of it and compare the synthesized normals to the ground truth.

Comparison to naïve texture synthesis. In Figure 14(d) we compared our method to naïve texture synthesis, where a small texture sample of homogeneous appearance is provided. We generated this result by synthesizing from a small texture square chosen from the center of the exemplar bush, using our pipeline without annotations. This produces an intermediate result with constant lighting that we multiply by the lighting annotation map. Simply using modulation to render the lighting effects in their entirety does not well reproduce the shape and light variations of the appearance of the foliage, and the synthesized boundary is implausible and exhibits a "wrapping-paper" kind of appearance. This being said, once the details of the variation of the material appearance have been captured with our method (e.g., the local behavior of foliage around shape cusps and curves), our method can still benefit from modulation by adding low-frequency lighting variations. For instance, we rendered the ambient occlusion on the grass in Figure 1 using modulation.

Comparison to image analogies. We also compared our approach to image analogies [Hertzmann et al. 2001] on a weathering example (Figure 15). In all fairness, Hertzmann et al. [2001] do not aim at generating smooth transition boundaries, hence the abrupt switch around those regions where the user requested modifications; our method avoids such problems due to the presence of gradient channels and Poisson reconstruction. Ignoring these boundary artifacts, Hertzmann et al. [2001] also do not follow the user's specifications as closely. This behavior was noted also in Zhang et al. [2003]: only relying on the L2-norm of an image

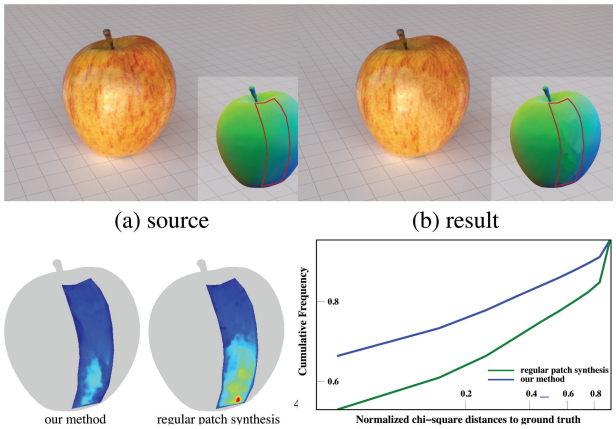


Fig. 23. Synthesizing on a synthetic scene. A textured 3D model of an apple is rendered into a source image (a). The renderer also provides exact information about the (x, y, z)-coordinates of the surface normals, visualized as (r,g,b) in color in (a, inset). In this experiment, we use these coordinates as annotation channels for both source and target and try to reconstruct the original apple; we exclude from the source image and annotations the portion outlined in red. The result is shown in (b). This result is derived by looking up, for each patch of the target image, the patch in the source image specified by the optimal map (i.e., NNF) found by PatchMatch during synthesis, and subsequently performing a voting step for overlapping patches. Using this same map to look up patches on the source annotation (a, inset) instead of the source image (a), and using the same voting scheme, we can generate an image that approximates the normals of our synthesized image. We can then compare to the ground-truth source annotation. These synthesized normals are shown in (b, inset), and the angle deviation between the two insets is shown in (c, left). While the ϵ-balls guarantee the annotations are closely followed, the annotations are not directly used in the patch distance; hence the focus during synthesis is on synthesizing coherent appearance for the apple. In (c, right), the angle error is shown for a result obtained via regular image-analogies-style patch synthesis, without our sampling scheme but with the annotations taking part in the patch distance, as in Figure 22. Our method respects the prescribed annotations better. A quantitative comparison of the synthesized texture to the ground truth (with the same metric as in Figure 19) is shown in (d).

augmented with annotation channels can produce results arbitrarily different from the desired ones, particularly if there are multiple such channels. Weighting the channels can help, but we found that tuning these weights is tedious and may not achieve results similar to ours even after several tries, particularly given the randomized character of the synthesis, and the presence of an alpha channel that also needs to be weighted. Figure 22 shows a typical result of simply using an L2-distance on the annotations while trying to reproduce the results of Figure 14; the boundaries are incorrectly synthesized and the annotations not closely followed. Figure 23(c, right) also shows a quantitative comparison of the quality of the reconstructed annotations. In Figure 15, our automatic clustering and interpolation combined with hard constraints provided by the ϵ-balls successfully renders the intermediate degrees of weathering requested, while Hertzmann et al. [2001] copy appearance from the closest matching degree of weathering. We observed this behavior regardless of the settings that control the patch distance metric; we demonstrate the best analogies result we obtained after fine tuning.

Fig. 24. Failure of the automatic annotation.

Discussion. We implemented our prototype in Matlab and C++. Our code is not optimized and runs offline; the results shown in the article took in the order of a few hours to compute, the bottleneck being the inefficient implementation of image melding. Since PatchMatch can be made very fast using multithreading and vectorization, and the number of iterations can be fine tuned for speed, we are confident that after optimization we can reach reasonably fast runtimes for interactivity. We leave this as future work.

The only significant parameter in our system remains ϵ_k, for which we used the default value 0.1 in most cases. The number of clusters is chosen automatically by the mean shift algorithm. When we had to tune ϵ_k, the set value was either 0.05 (better match of the annotation) or 0.2 (less repetition). Despite this, repetition remains visible in a few examples, as is common in texture synthesis and patch-based optimization methods. As future work, we plan to add spatial and nearest-neighbor "jittering" [LeFebvre and Hoppe 2006; Risser et al. 2010], which should help alleviate this issue.

In terms of exemplars, while our approach fully supports more complex exemplars, we also found it easier to work with simple shapes like spheres and planar surfaces, when possible, in order to keep the annotation time at a minimum. In terms of annotations, we found that no special expertise is required to create them; it suffices for users to understand the (mostly straightforward) effects that they want to capture, such as light changes or the shape variations. With the exception of Figure 9, the remaining annotations were either binary or gradient images (painted within a few minutes) or prerendered images of simple primitives (spheres, cylinders) adapted to the size of the sources. We found that keeping annotations simple can be advantageous for our method; it is helpful to give annotations that provide a coarse guidance of the high-level effects while the texture synthesis process renders the fine details necessary to produce a natural looking image. This being said, our method is reasonably robust to different ways of capturing the same desired effect, as well as to some amount of noise. We showcase this in Figure 21 by adding noise, blurring, or quantizing the target annotations of Figure 14 to simulate, for instance, manual painting/labeling of the annotations instead of using an automatically rendered image. On the other hand, if automatic annotations are used and the annotation generation procedure completely fails to capture the desired effect, our method might be unsuccessful, too. An example is shown in the inset of Figure 24, the automatic shape-from-shading method used [Barron and Malik 2012] did not manage to capture the symmetry of the shirt, and the reconstructed shirt looks dirty and implausible. For this example, manual correction of the annotation was nessecary.

Our method can currently only handle opaque, nonreflective materials; the appearance of transparent or reflective objects may heavily depend on distant parts of the scene and is left as future work. Our materials have stochastic textures with small- to middle-scale

structure details, thus covering a broad range of common natural materials such as grass, foliage, stone walls, fabric, etc. Most problematic cases included materials with large-scale structures of different appearance that are hard to place in correspondence (e.g., walls with large, differently shaped bricks, closeups of plants with ample empty space).

Our approach offers two ways to deal with transitions, either with an alpha channel or by providing an exemplar that exhibits a sample transition. When the backgrounds of the input and the exemplar match, we recommend the latter; this makes the process simpler and faster by avoiding creation and handling of an alpha channel.

7. CONCLUSION

We have demonstrated a new method for synthesizing images containing complex and/or heterogeneous materials by generalizing from only limited exemplars. Our approach is able to process arbitrary images including photos downloaded from the Internet and enables user control through simple tools, such as brushes and diffusion curves. It produces results richer than standard texture synthesis without adding the complexity of BTF capture. The variety of examples that we have shown attests to the versatility of our technique.

ACKNOWLEDGMENTS

We are grateful to Jim McCann for various discussions and idea suggestions on interactivity. We are also grateful to Wenzel Jakob for the introduction to Mitsuba. We thank our image sources: Flickr users Anup Jaiswal (`Anup_Nikon D40`), Glen Scott (`hey mr glen`), Mark Walsh (`4mtr`), Lindsay Buckley (`lindsaypunk`), Robert Moore (`Brron`), and the websites `www.istockphoto.com`, `www.molon.de`, `graphicleftovers.com`, and `www.publicdomainpictures.net` .

REFERENCES

C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.* 28, 3.

C. Barnes, E. Shechtman, D. Goldman, and A. Finkelstein. 2010. The generalized patchmatch correspondence algorithm. In *Proceedings of the European Conference on Computer Vision (ECCV'10).* 29–43.

J. T. Barron and J. Malik. 2012. Color constancy, intrinsic images, and shape estimation. In *Proceedings of the European Conference on Computer Vision (ECCV'12)*. A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., Lecture Notes in Computer Science, vol. 7575, Springer, 57–70.

P. Bhat, B. Curless, M. Cohen, and C. L. Zitnick. 2008. Fourier analysis of the 2D screened poisson equation for gradient domain problems. In *Proceedings of the European Conference on Computer Vision (ECCV'08).* 114–128.

N. Bonneel, M. Van De Panne, S. Lefebvre, and G. Drettakis. 2010. Proxy-guided texture synthesis for rendering natural scenes. In *Proceedings of the International Workshop on Vision, Modeling and Visualization (VMV'10).* 87–95.

D. Comaniciu and P. Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Patt. Anal. Mach. Intell.* 24, 5, 603–619.

K. J. Dana, B. Van Ginneken, S. K. Nayar, and J. J. Koenderink. 1999. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.* 18, 1, 1–34.

S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. 2012. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.* 31, 4.

A. A. Efros and T. K. Leung. 1999. Texture synthesis by nonparametric sampling. In *Proceeding of the International Conference on Computer Vision (ICCV'99).* Vol. 2. 1033–1038.

C. Eisenacher, S. Lefebvre, and M. Stamminger. 2008. Texture synthesis from photographs. *Comput. Graph. Forum* 27, 2, 419–428.

H. Fang and J. C. Hart. 2004. Textureshop: Texture synthesis as a photograph editing tool. *ACM Trans. Graph.* 23, 3, 354–359.

C. Han, E. Risser, R. Ramamoorthi, and E. Grinspun. 2008. Multiscale texture synthesis. *ACM Trans. Graph.* 27, 3.

A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. 2001. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'01).* 327–340.

W. Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

M. K. Johnson, K. Dale, S. Avidan, H. Pfister, W. T. Freeman, and W. Matusik. 2011. CG2Real: Improving the realism of computer generated images using a large collection of photographs. *IEEE Trans. Visual. Comput. Graph.* 17, 9, 1273–1285.

J. Kautz, S. Boulos, and F. Durand. 2007. Interactive editing and modeling of bidirectional texture functions. *ACM Trans. Graph.* 26, 3.

V. Kwatra, I. A. Essa, A. F. Bobick, and N. Kwatra. 2005. Texture optimization for example-based synthesis. *ACM Trans. Graph.* 24, 3, 795–802.

S. Lefebvre and H. Hoppe. 2005. Parallel controllable texture synthesis. *ACM Trans. Graph.* 24, 3, 777–786.

S. Lefebvre and H. Hoppe. 2006. Appearance-space texture synthesis. *ACM Trans. Graph.* 25, 3, 541–548.

D. Lepage and J. Lawrence. 2011. Material matting. *ACM Trans. Graph.* 30, 6, 144.

X. Liu, Y. Yu, and H.-Y. Shum. 2001. Synthesizing bidirectional texture functions for real-world surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'01).* ACM Press, New York, 97–106.

J. Lu, A. S. Georghiades, A. Glaser, H. Wu, L.-Y. Wei, B. Guo, J. Dorsey, and H. E. Rushmeier. 2007. Context-aware textures. *ACM Trans. Graph.* 26, 1.

W. Matusik, M. Zwicker, and F. Durand. 2005. Texture design using a simplicial complex of morphable textures. *ACM Trans. Graph.* 24, 3, 787–794.

D. M. Mount and S. Arya. 1998. ANN: Library for approximate nearest neighbour searching. http://www.cs.umd.edu/-mount/ANN.

A. Ngan and F. Durand. 2006. Statistical acquisition of texture appearance. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques (EGSR'06).* 31–40.

A. Orzan, A. Bousseau, H. Winnemoller, P. Barla, J. Thollot, and D. Salesin. 2008. Diffusion curves: A vector representation for smooth-shaded images. *ACM Trans. Graph.* 27, 3.

H. Park, H. Byun, and C.-H. Kim. 2013. Multi-exemplar inhomogeneous texture synthesis. *Comput. Graph.* 37, 1–2, 54–64.

E. Risser, C. Han, R. Dahyot, and E. Grinspun. 2010. Synthesizing structured image hybrids. *ACM Trans. Graph.* 29, 4.

A. Rosenberger, D. Cohen-Or, and D. Lischinski. 2009. Layered shape synthesis: Automatic generation of control maps for non-stationary textures. *ACM Trans. Graph.* 28, 5.

R. Ruiters, C. Schwartz, and R. Klein. 2013. Example-based interpolation and synthesis of bidirectional texture functions. *Comput. Graph. Forum* 32, 2.

J. Wang, X. Tong, S. Lin, M. Pan, C. Wang, H. Bao, B. Guo, and H.-Y. Shum. 2006. Appearance manifolds for modeling time-variant appearance of materials. *ACM Trans. Graph.* 25, 3, 754–761.

L.-Y. Wei and M. Levoy. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'00)*. 479–488.

J. Zhang, K. Zhou, L. Velho, B. Guo, and H.-Y. Shum. 2003. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Trans. Graph.* 22, 3, 295–302.

Received February 2014; revised August 2014; accepted October 2014