Image Replacement through Texture Synthesis

Homan Igehy   Lucas Pereira
Computer Science Department
Stanford University

Abstract

Photographs and images often have regions which are in some sense flawed. Often, there may be a stain or an undesired feature covering a significant portion of the image, and an algorithm that can "fix" the image is desired. We propose a technique based on Heeger and Bergen's texture synthesis algorithm [4]. By integrating a composition step into the aforementioned algorithm, one portion of the image can be substituted with a synthetic texture derived from another portion of the image. The goal of this synthetic texture is to create a plausible patch without visible seams or repetitive features. We compare this technique, which works on areas that are "stochastic" in nature, with other variations of image replacement which utilize a combination of image composition and texture synthesis.

1 Introduction

Photographs and images often have regions which are in some sense flawed. At times, the flaw is due to a minute artifact: there may be scratches on the image, or perhaps a visible special effects wire. Fast interactive techniques exist for removing this type of small-scale noise [5]. At other times, the flaw is present in a large region of the image: there may be a stain on the photograph, or some unsightly object may be present in the scene. One simplistic approach is to copy and blend in a similar region from somewhere else in the image. This approach has two major drawbacks. First, visible repetition may appear when a small region is used to fill a large stain. Second, the compositing of the replacement region with the areas surrounding the stain may lead to visible boundaries even with the use of multi-resolution compositing techniques.

We propose an approach that utilizes synthetic textures for image replacement. This approach, which was demonstrated by Malzbender and Spach [7] in a different algorithm, has several advantages. Since synthetic textures may be created with arbitrary size, repetition artifacts are avoided. Furthermore, by integrating a composition step into our texture synthesis algorithm, boundary problems are reduced.

2 Algorithm

Heeger and Bergen [4] have developed an algorithm which can synthesize a texture that matches the appearance of a given "target" texture. While the algorithm does not work well on "deterministic" textures (e.g., a tile floor, a brick wall), it is effective on "stochastic" textures (e.g., granite, stucco). By matching first order statistics on a bank of (orientation and spatial-frequency selective) linear filters which roughly approximate the texture discrimination capabilities of human perception, a plausible synthetic texture can be produced from a variety of stochastic target textures. We extend this algorithm with a composition step in order to allow seamless integration of synthetic and real textures.

Our image replacement algorithm, *Texture-Replace*, is illustrated in Figure 1. The bold text denotes additions made to the original texture synthesis algorithm described in [4], but we will first briefly describe the original algorithm. The inputs into the original algorithm are a *target* texture and a *noise* image (e.g., white noise, fractal noise). By the end of the algorithm, *noise* will have been converted into a synthetic texture. First, *Match-Histogram* is used to force the intensity distribution of *noise* to match the intensity distribution of *target*. Then an analysis pyramid is constructed from the *target* texture. The pyramid representation can be chosen to capture features of various sizes (by using a Laplacian pyramid [1, 2, 6]) as well as features of various orientations (by using a steerable pyramid [3, 8]). Then, the *noise* is modified in an iterative manner. At each iteration, a synthesis pyramid is constructed from *noise*, and *Match-Histogram* is done on each of the sub-bands of the synthesis and analysis pyramids. The modified synthesis pyramid is then collapsed back into *noise*, and a *Match-Histogram* is done

```
Texture-Replace (noise, original, mask, target)
  original = Blend(Mean(target), original, mask)
  noise = Match-Histogram (noise, target)
  noise = Composite(noise, original, mask)
  analysis-pyr = Make-Pyramid (target)
  Loop for several iterations
     synthesis-pyr = Make-Pyramid (noise)
     Loop for a-band in sub-bands of analysis-pyr
          for s-band in sub-bands of synthesis-pyr
       s-band = Match-Histogram (s-band, a-band)
     noise = Collapse-Pyramid (synthesis-pyr)
     noise = Match-Histogram (noise, texture)
     noise = Composite(noise, original, mask)
```

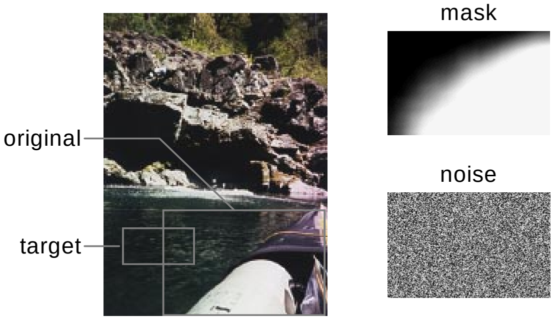

Figure 1: The Texture Replacement Algorithm. Some region of an image, *original*, is to be replaced by a new image according to a *mask*. The intensity of the *mask* image determines the extent to which the texture grain of the *original* image is replaced by the synthetic texture grain created from *noise*. The synthetic texture is made to match characteristics of the *target* texture.

between it and the *target* texture. After several iterations, the *noise* texture converges to have the same distribution of features as the *target* image; thus, a synthetic texture is created. Color textures may be synthesized by first performing a decorrelating color transform on the n-channel color texture, then synthesizing each of the resulting n grayscale images, and then performing a recorrelation. It is noteworthy to mention that since the filters of the pyramid structure do not exactly match the human perceptual system, and matching is done using first order statistics, differences between the synthetic and target textures are visible, especially when viewed side-by-side.

Our image replacement algorithm extends the original algorithm with the following goal: we want *noise* to turn into an image which is a combination of an *original* image and a synthetic texture derived from *target*; the combination is controlled by a *mask*. The algorithm remains the same as before, except that at each iteration of the algorithm, the *original* is composited back into the *noise* image according to the *mask* using a multi-resolution compositing technique that avoids blurring and aliasing [2]. Also, as a pre-process, we use the *mask* to replace the flawed region of the *original* with the average color of the *target* image to avoid low-frequency color bleeding from the flawed region. Matching is then done on the newly composited *noise* image. The resulting image becomes synthetic texture at pixels where the *mask* is 1, and *original* texture at pixels where the *mask* is 0. Using fractional values in the *mask* yields a smooth transition from real to synthetic texture from the point of view of texture discrimination. Since the non-masked regions of the *original* image are chosen to already match the *target* texture, the sub-band matching does not modify these regions too much. Thus, the *mask* values of a region determine how much of the *original* texture's characteristics are incorporated into the sub-band matching, allowing for smooth texture transition.

3 Results

Texture-Replace was used on the images of the left side of Figure 2 to produce the images of the right side of Figure 2. For each of the images, a smooth transition mask was created by hand to mark the undesired portions; automated image segmentation techniques could also be used to accomplish this task. Figure 3 demonstrates the shortcomings of other techniques for replacing part of an image with synthetic textures. Figure 3a uses our *Texture-Replace* algorithm with a smooth mask. In Figure 3b, a binary mask is used and a sharp boundary results between the synthetic and real textures. In Figure 3c, a pure synthetic texture was first created using the original algorithm of [4], and then was composited onto the original image using a binary mask. Though the border is alias-free through the use of multi-resolution composition, there is a sharp transition in the texture grain. In Figure 3d, a pure synthetic texture was composited onto the original image using a smooth mask. The resulting image replaces the sharp transition in texture grain with a less objectionable transitional artifact: a blurred blend of the two textures that washes out high-frequency features.

Texture synthesis can be useful in a variety of images which need the replacement of large areas with stochastic textures. This technique, however, is useless for images that need the replacement of areas with structured texture. Furthermore, as with all image replacement techniques, global cues can hint at the "doctoring" of a local region of an image. For example, in middle image pair in Figure 2, a portion of the removed person's shadow remains. In Figure 3, the synthetic reflection of the sun in the synthetic water doesn't match the real reflection of the sun. But despite these limitations, the resulting images look plausible if not examined in detail.

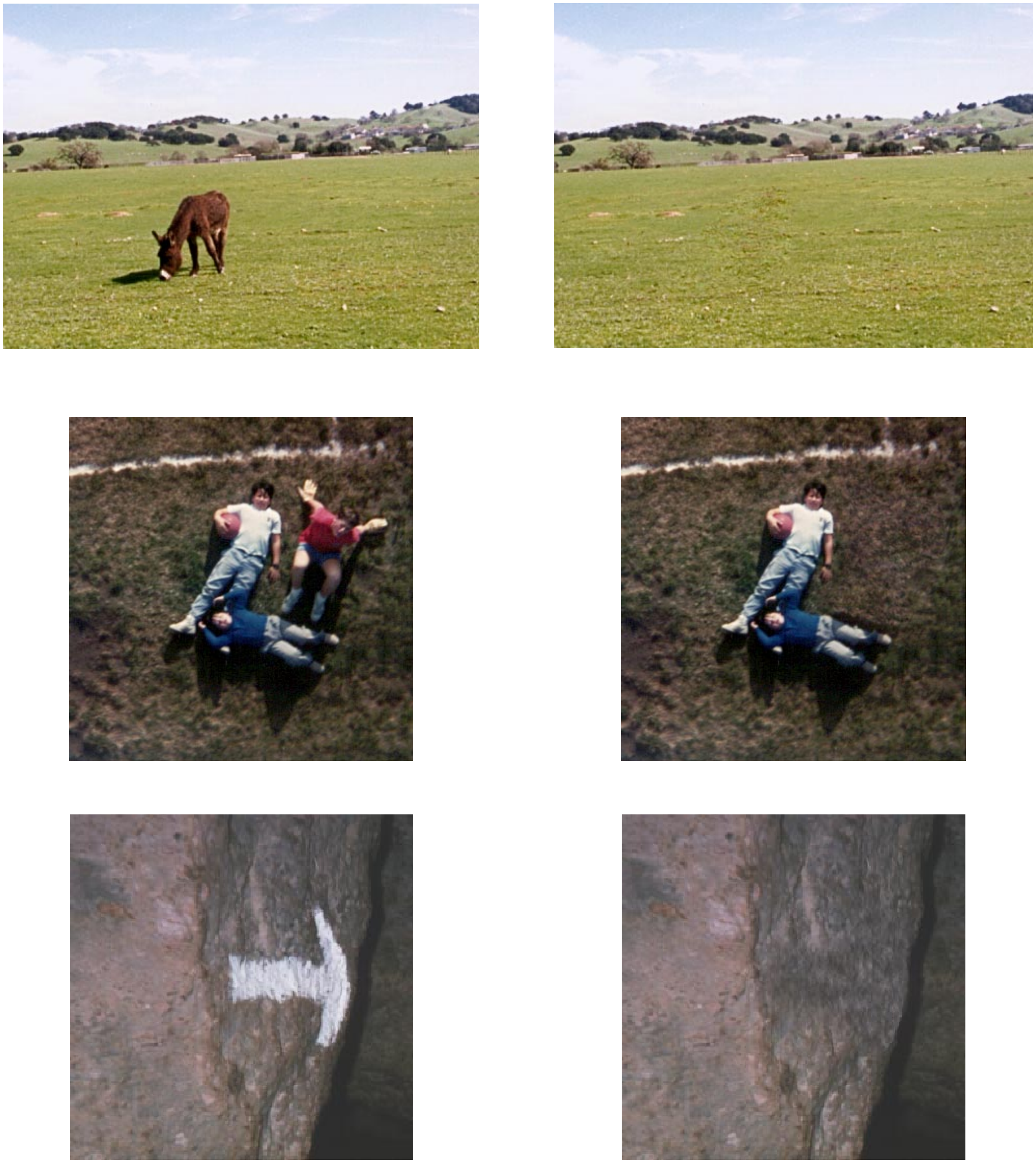

Figure 2: Examples of the *Texture-Replace* Algorithm.

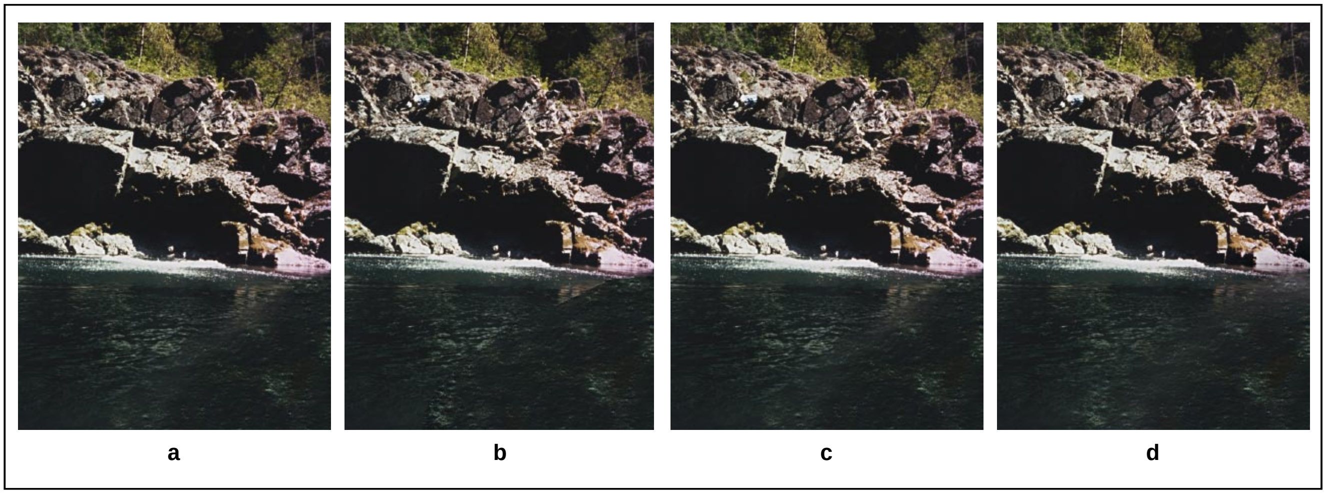

Figure 3: Variations of Texture Synthesis and Image Composition. Using the image of Figure 1, these images demonstrate the advantage of the *Texture-Replace* algorithm (used for image (a)) over variations which do not provide a smooth cross-over of features between the real and synthetic textures. Image (b) used the *Texture-Replace* algorithm with a binary mask instead of a smooth mask. In image (c), a regular synthetic texture was first generated and then composited as a post-process using a binary mask. In image (d), the same technique was applied using a smooth mask.

4 Conclusion

Image replacement through texture synthesis provides a new solution to the image replacement problem for a certain class of images. Direct composition of a pure synthetic texture into an image often works well for stochastic textures, but may occasionally result in undesirable artifacts due to differences in the grain of the real texture and the imperfect synthetic texture. By integrating composition into the texture synthesis algorithm, a smooth transition between real and synthetic texture grain may be realized.

References

[1] P. Burt., "Fast Filter Transforms for Image Processing". Computer Graphics and Image Processing, volume 16, pages 20-51, 1981.

[2] P. Burt and E. Adelson, "A Multiresolution Spline with Application to Image Mosaics". ACM Transactions on Graphics, volume 2, pages 217-236, 1983.

[3] W. Freeman and E. Adelson, "The Design and Use of Steerable Filters". IEEE Transactions on Pattern Analysis an Machine Intelligence, volume 13, pages 891-906, 1991.

[4] D. Heeger and J. Bergen, "Pyramid-Based Texture Analysis/Synthesis". In Computer Graphics (SIGGRAPH '95 Proceedings), volume 29, pages 229-238, 1995.

[5] A. Hirani and T. Totsuka, "Combining Frequency and Spatial Domain Information for Fast Interactive Noise Removal". In Computer Graphics (SIGGRAPH '96 Proceedings), volume 30, pages 269-276, 1996.

[6] J. Ogden, E. Adelson, J. Bergen, and P. Burt, "Pyramid-Based Computer Graphics". RCA Engineer, volume 30, pages 4-15, 1985.

[7] T. Malzbender and S. Spach, "A Context Sensitive Texture Nib". Communicating with Virtual Worlds, N.M. Thalmann and D. Thalmann, Eds. Springer-Verlag, New York, pages 151-163, 1993.

[8] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger, "Shiftable Multi-Scale Transforms". IEEE Transactions on Information Theory, Special Issue on Wavelets, volume 38, pages 587-607, 1992.