Implementation of Efros and Freeman's Image Quilting for Texture Synthesis

Kenton McHenry
University of Illinois Urbana-Champain
mchenry@uiuc.edu

March 20, 2003

Figure 1. Sample texture.

Figure 2. Synthesized texture from tiled windows.

1 Goals

The goal of this project is to implement the method described in [1] which when given a sample of a texture is able to synthesize more of the texture in a computationaly efficient manner and with reasonably good results.

2 Texture Synthesis

The texture synthesis portion of the code was done in Matlab. To start the user must provide a sample texture, a window width, height, and overlap size.

In addition the user must specify the size of the output texture. A seed window is then randomaly chosen from the sample of the texture. Further windows from the sample are searched for under the constraint that they match well with the overlap of the tiles already placed in the output texture. The output texture is generated top row first, each new tile for this row only considers overlapping with the previous tile to the left. Next the entire left column is generated, each new tile here only consideres overlapping with the tile on top. All other tiles are now simular in that if we generated them in order, from left to right and from top to bottom, they need only consider overlap with the tile above and the tile and to the left.

At this point, though the new texture isn't bad, there are still clearly visible seams where the tiles meet (Figure-2). To eleminate this artifact an optimal cut is made along the overlap between neigboring tiles. Defining the error between a tile T_1 and tile T_2 as:

$$E = (T_1 - T_2)^2$$

the optimal path is easily found with dynamic programming.

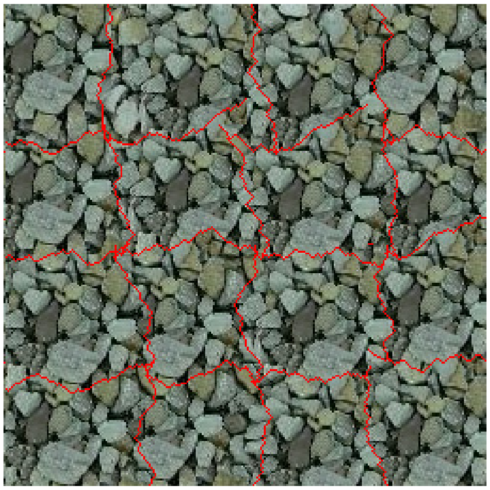

Figure 3. Synthesized texture with optimal cuts.

All such optimal cuts are found between overlapping tiles as shown above to produce the texture below.

Figure 4. Synthesized texture.

The running time for this example on a 750MHz AMD Athlon was roughly two minutes.

3 Texture Mapping

This portion of the implementation was written in C++ with OpenGL. We can map a sphere with vertices (x,y,z) and radius r to a planer texture containing 2D coordinates (u,v) by using spherical coordinates to map the 3D coordinates into a plane [2]:

$$v = \frac{acos(z/r)}{\pi}$$

$$u = \frac{acos(x/(rsin(v\pi)))}{2\pi}$$

With these equations we can assign a texture coordinate to every vertex of the sphere.

Figure 5. Textured sphere.

The resulting texture map will have distortion at the poles, where the texture pinches off. This can be taken care of by distorting the texture before it is mapped [2] so that the pixels at the poles are spread out. The texture is warped by assigning every pixel (u,v) to take its value from location (u',v) where u' is defined as:

$$u' = 2\pi \frac{u - \frac{WIDTH}{2}}{WIDTH} \times \cos\left(\pi \frac{v - \frac{HEIGHT-1}{2}}{HEIGHT-1}\right) \times \frac{WIDTH}{2\pi} + \frac{WIDTH}{2}$$

The resulting texture now looks like Figure-6. When it is now mapped to the sphere the pinching at the poles no longer occurs and the texture appears evenly sized over the sphere (Figure-7).

Figure 6. Warped texture.

Figure 7. Textured sphere, with warped texture to reduce distortion at poles.

Another distortion involves the seam where the textures left side meets its right side (Figure-8). This kind of distortion is easily handled by the image quilting technique. Instead of generating the texture of the exact size we desire we generate it to be a bit larger. The extra parts of the image will serve as overlap to elminate the seam. We can now split the texture in the middle and create two windows (each half the width of our desired texture). The extra to each side of these windows are overlapping regions we can use to find a cut that minimizes the error between the two windows (Figure-9). In this way a texture can be generated that tiles better, thus making the seam less apparent (Figure-10).

Figure 8. Seem in textured sphere.

4 Texture Transfer

The described technique for texture synthesis provides a convenient means of transfering a texture from one image to another. We do this by iterating over a specified sized window over the destination image. For each of these windows we look for a simular window in the source image based on some criteria (e.g. intensity). In addition each tile must also match well with the tiles already placed on the destination image. We control the preference between these two criteria with a parameter α such that the total

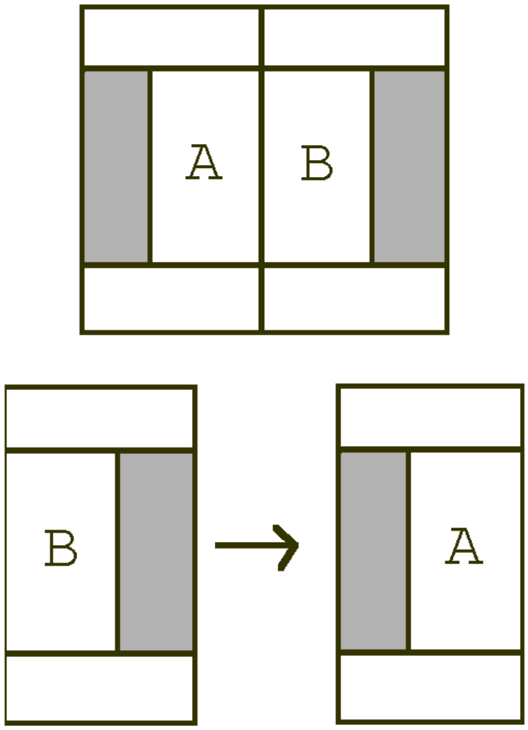

Figure 9. Split image.

error for a proposed tile is α times the block overlap error plus $(1-\alpha)$ times our correspondance measure. The process turned out to be quite slow so small images were used to test it (Figure-11).

References

[1] Efros and Freeman, *Image Quilting for Texture Synthesis and Transfer*, SIGGRAPH 2001.

[2] Paul Bourke, *Parametric Equation of a Sphere and Texture Mapping* http://astronomy.swin.edu.au/ pbourke

Figure 10. Textured sphere with less apparent seam.

Figure 11. Texture transfer