

# Texture Synthesis: Image Quilting

Allison Serio

## Motivation

The goal of texture synthesis is to create an arbitrarily large image from a sample image that is perceived to be the same texture as the sample. The main challenges for synthesizing algorithms are 1) creating believable textures for a wide range of texture types, and 2) being fast and efficient. Real-world textures usually fall in between being regular and stochastic in type, meaning some of its qualities are partially repeated and partially noisy, and so being able to handle these mixed textures are what make the algorithm useful. However, being able to successfully produce a high quality texture from any type of image comes at the cost of the algorithm's efficiency. This becomes a significant problem for when a large sample is given or large output texture is expected, and so finding a method that can overcome both these challenges will especially be beneficial for texture mapping in high-production entertainment.

## Previous Work

Some of the very early methods of texture synthesis could only produce textures of specific types, and/or produced unrealistic textures. The first papers that were influential in the field were on pixel-based approaches[2][3], and achieve high-quality texture results. However, these methods are very slow, mostly due to the computational cost in selecting the best pixel from the sample image for each pixel in the output texture. In the paper "Image Quilting For Texture Synthesis and Texture Transfer" by Efros and Freeman[1], a patch-based approach is used to generate textures, taking advantage of the fact that most pixels' values have already been determined by what's already been synthesized. This approach uses a larger unit of synthesis, and creates results comparable to the previous approaches at faster speed. For my project, I decided to utilize this method in generating textures.

## Algorithm

The image quilting algorithm works by selecting a patch sample from the sample image and laying the selected patch onto the output texture. The algorithm can be defined in 3 major steps: patches are laid onto the output image in raster scan order, patches are selected by some overlap constraint, and the boundary of the new patch is determined by the minimum cost paths of the overlap regions.

Below is a more detailed view of how I implemented the algorithm, such as how I determine overlap constraint, how I find the minimum cost path, etc.





1. Precompute all patches that can be made from sample image. I create patches at every location of the sample image, and these patches will be used later in selecting the best patch according to the overlap constraint later.
2. Begin going through the output image in raster scan order (left to right, top to bottom), in steps of patch size.
3. Select best patch based on overlap error constraint.
  - a. Overlap size is set to  $\frac{1}{2}$  of patch size.
  - b. My constraint is the L2 norm of the pixel colors in the overlap region of the existing patches and the candidate patch. I calculate the error for all patches created in step one.
  - c. Randomly select a patch that is within 10% of the minimum error --this will allow different textures to generate from a single sample image.
  - d. There 3 cases of patch overlap to consider: only overlap to the left of the selected patch, only overlap above the patch, or both
4. Calculate error surface of overlap regions and find minimum cost path.
  - a. Use squared difference between pixel color values to get error surface of overlap.
  - b. Dynamic programming to find minimum cost path on error surface--calculate total minimum path cost for reaching any pixel, and search last row or column for smallest minimum path cost.

User input include sample image, patch size, and output height and width.

## Results

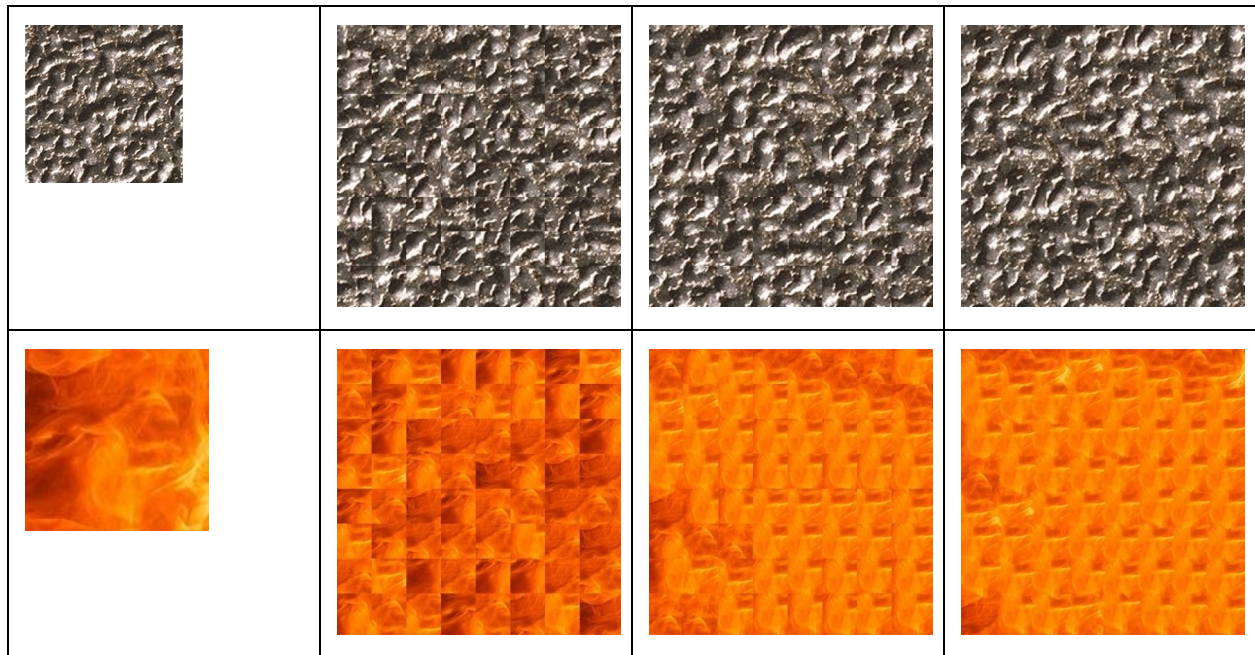
My results include regular, irregular, stochastic, and mixed textures. The algorithm performs very well for highly regular and highly stochastic texture, as seen by the brick and cement textures. However the minimum error boundary cut only mildly improved some of these textures, and so would be more time efficient to just use the overlap constraint implementation. The results for near-regular and near-stochastic textures(cells and flowers) are also impressive, however the seams could be more well hidden. Irregular textures(fire and magma) suffered the most, as the textures look repetitive and unnatural. It is also important to note that there is some parameter tuning that can be done to improve the output texture, such as adjusting the block size or increasing the overlap size.

I also created animations of the output image being synthesized, with the random, overlap constraint, and minimum path methods. These created as the textures were being generated. I think this animation is good at showcasing how the texture is built up, but it would be more effective if it also displayed the patches from the sample image it was taking from.

Sample	Random	Overlap Constraint	Min cost Path
			







## Conclusion

In my project I implemented the image quilting algorithm and analyzed its texture results and efficiency. This method renders a large-scale texture by selecting best matching patches from the sample image and redefining seam boundaries before patches are stitched together in the output. Speed depends on the desired size of image to create, and also the size of the sample image given. There are many ways to improve this algorithm to make the texture more believable and the program more efficient. Inputting or generating more data for the output texture to sample from, such as including different orientations of the patches sampled from the input image, would allow better patch matching and make the texture more realistic. A way to better hide the seams would be image blending at the seams, or to use patches of optimal shape[4]. This algorithm can also be extended to texture transfer-- the problem of re-rendering an image into another image's texture.

## References

- [1] A. A. Efros and W. T. Freeman, "Image Quilting for Texture Synthesis and Transfer", *SIGGRAPH 01*.
- [2] A. A. Efros and T. K. Leung, "Texture Synthesis by Non-parametric Sampling", *ICCV 99*.
- [3] Li-Yi Wei and Marc Levoy. "Fast Texture Synthesis Using Tree-Structured Vector Quantization.", *SIGGRAPH '00*.
- [4] Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, and Aaron Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts", *SIGGRAPH 03*