Parallel patch-based texture synthesis

Steven Diviney 08462267, Trinity College Dublin

Put stuff here later

1. INTRODUCTION

Texture synthesis algorithms have been the focus of a relativly large body of research over the past ten years. There are two main synthesis approaches; pixel based and patch based[Wei et al. 2009]. A third approach not discussed in this paper uses a parametric model to describe a variety of textures. While they can be used for texture generation they perform better for analysis of textures. [Saisan et al. 2001]. Although there is a variety of pixel and patch based algorithms they all share similar characteristics. Pixel-based approaches exhibit a high degree of parallelism and map well to GPUs but the results can seem quite lacking compared to patch-based approaches. This becomes more pronounced as the amount of structure in a texture increases.

Patch-based approaches require relatively slow algorithms to layout patches and stitch them together. However, they are better at preserving structure. Pixel based approaches can be improved by the addition of a feature distance map but obtaining such a map is another complex problem. [Lefebvre and Hoppe 2006]

The reason for the differences in output between the two approaches come down assumptions made by the algorithms. Pixel based texture synthesis typically fill in the result image by finding and copying pixels with the most similar local neighborhood as the synthetic texture. [Paget and Longstaff 1998] The neighborhoods are quite small so strong features become deformed or get completely destroyed. This approach works well when applied to the more traditional notion of a texture, that is, an infinite pattern that can be modeled by a stationary stochastic process.

Patch based approaches attempt to create a new texture by copying and stitching together textures at various offsets. The patches are aligned so as to preserve features and reduce the number of visual artifacts such as seams.

The paper discussed presents a fast patch-based texture synthesizer. While still not as fast as pixel-based approaches it is significantly faster than previous patch-based algorithms. This is accomplished through four main contributions;

— A fast, approximate, algorithm to optimize the patch boundaries. This greatly improves performance with little impact on quality.
— An algorithm to deform the patches after boundary optimization and improve feature alignment.
— A scheme for using new patches to hide existing errors. Patches with string visible seams are rejected. Patches can be processed independently.
— A full GPU implementation.

2. RELATED WORK

Patch-based texture synthesis is typically broken down into four problems; patch synthesis, patch placement, patch stitching and feature alignment. The paper being discussed does not stray too far from this model but gives more attention to the contributions presented. These four areas are used to categorize related work.

2.1. Patch-based texture synthesis

Early schemes select patches at random and feather the edges to form a new texture. [Guo et al. 2000] Feathering is a relatively simplistic approach to merging patches. Every pixel along a seam is assigned a weighted, typically a Gaussian kernel, sum of neighboring pixel values. This destroys any sharp features and amount to blurring the seams.

Image quilting adds patches in scan-line order to a grid. The first block is copied at random and the subsequent blocks are placed such that they partly overlap with previously place blocks. The boundary between two images is optimized by means of a minimum cost path. This ensures a seam introduces minimal colour difference. [Efros and Freeman 2001]

Graph-cut improves this process by creating patches of arbitrary shape. It also introduces the ability to hide existing error using new patches. [Kwatra et al. 2003] The implementation of the cycle cuts presented in this paper differs in its approach from the algorithm presented by V Kwatra, et al. but the idea is quite similar. It is worth examining Graph-cut in some detail to aid the comprehension of the ideas presented in this paper.

2.1.1. Graph-cut. At its core, Graph-cut is a classical graph problem called minimum cut. The wish is to find a low-cost path through the overlap region between patches. The measure used is the difference of colour between the pairs of pixels in this region. This is further improved by incorporating old seam costs into new cuts, potentially hiding visible seams under new patches. Lastly the frequency of the region is taken into account. Seam boundaries are prominent in low frequency regions so the cost function is adjusted to account for the gradient of the image.

The approach presented in this paper also attempts to find the cut of minimum cost, although it is approximate, not optimal.

2.2. Patch placement

The method outlined in this paper attempts to align patches based on seam cost optimization and feature alignment. Each new patch added is constrained to a small part of the output image. The patches are large compared to other approaches. The patch sampling strategy is a simple uniform random search. The large patch sizes allow for the synthesis of structured patterns. The iterative nature of the solution, the repeated overlay of patches, hides most of the error. As such, patch placement does not receive a great level of attention directly.

Patchmatch uses a more sophisticated sampling strategy for smaller patches. The search is alternated between a random and coherent method[Barnes et al. 2009].

2.3. Patch stitching

Graph-cut achieves good stitching results through use optimal seam cuts [Boykov et al. 2001]. This is further improved by its patch placement and matching strategy [Kwatra et al. 2003].

3. PARALLEL PATCH BASED SYNTHESIZER

4. FAST APPROXIMATE CYCLE CUTS

5. FEATURE ALIGNMENT

6. PATCH REJECTION

7. IMPLEMENTATION DETAILS

8. RESULTS

REFERENCES

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28.*

Boykov, Y., Veksler, O., and Zabih, R. 2001. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 23,* 11, 1222–1239.

Efros, A. and Freeman, W. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 341–346.

Guo, B., Shum, H., and Xu, Y. 2000. Chaos mosaic: Fast and memory efficient texture synthesis. *Microsoft research paper MSR-TR-2000-32*.

Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. 2003. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (TOG)*. Vol. 22. ACM, 277–286.

Lefebvre, S. and Hoppe, H. 2006. Appearance-space texture synthesis. In *ACM Transactions on Graphics (TOG)*. Vol. 25. ACM, 541–548.

Paget, R. and Longstaff, I. 1998. Texture synthesis via a noncausal nonparametric multiscale markov random field. *Image Processing, IEEE Transactions on 7,* 6, 925–931.

Saisan, P., Doretto, G., Wu, Y., and Soatto, S. 2001. Dynamic texture recognition. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.* Vol. 2. IEEE, II–58.

Wei, L., Lefebvre, S., Kwatra, V., Turk, G., et al. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. 93–117.