Texture Synthesis

A Literature Synthesis by *Ryan Mazzolini*

Contents

1. Abstract

Texture synthesis is a method of computationally creating synthetic textures. It is important for applications in computer graphics, computer vision and image processing. This Literature Synthesis aims to explore existing techniques for texture synthesis and its applications highlighting the strengths and weaknesses of each. Texture synthesis is a large field containing a large variety of solutions and techniques. This makes finding a synthesis technique to fit a desired application a daunting task.

2. Introduction

Texture synthesis is a large field with many competing techniques and a variety of applications. This makes it an important and active field of study. As with most research areas around computing, texture synthesis is a young field with most of its research being done during and after the late 1990's. This makes it an interesting field of study with room for improvement and application. In this literature synthesis we define what texture synthesis is, investigate the applications of texture synthesis and explore the existing techniques of each technique.

2.1. Defining texture synthesis

Texture synthesis is the creation and placement of textures through computation and procedural generation. Textures can be created manually by texture artists however this is labour intensive, taking long periods of time. The larger the texture needs to be the more time it takes to make a realistic and consistent looking texture. Some types of textures are simply too complicated to create by hand. This is why the field of creating textures computationally has been investigated so thoroughly. Creating realistic textures that are perceived by humans can be a very hard problem to solve. Textures need to be consistent, for example not contain visual artefacts such as having straight lines skewed. Textures also need to be perceived naturally random and not just reoccurring sections. While synthesising realistic looking textures might be possible having them synthesised in a realistic time period is also essential. Users do not want to have to wait for textures to be synthesised over an impractical amount of time. Some applications even require textures to be synthesised in real-time which puts large constraints on the computational time available.

2.2. The uses of texture synthesis

Texture synthesis is used in many different areas and for different applications. It is predominantly important in computer science applications like computer graphics, computer vision and image processing. The uses however include many applications outside of the computer science.

Texture synthesis is important for many applications that are reliant on computer generated environments. These include animated films, games and construction or designing environments. The environments often need objects that are wrapped in seamless textures which appear realistic. How realistic these textures are can often be reliant upon how randomised the texture looks and if there are obvious visual artefacts or patterns. This can however be conflicting in different texture types as some need to be more randomised for example in natural looking textures like bark and some need to be more regular for example material textures like cotton. Texture synthesis is also important for areas outside on computer generated environment for example in the medical and other scientific fields that apply computer graphics, computer vision and image processing.

3. Techniques in texture synthesis

The area of texture synthesis has created various techniques in the creation and distribution of textures. These predominantly include texture placement, procedurally generated textures and example-based texture synthesis.

3.1 Procedural Texture Synthesis

Procedural texture synthesis is the method of creating synthesised textures via computation and procedural means. Examples of these techniques are the Perlin[1] and Worley[2] three dimensional noise techniques. Procedural texture synthesis is constrained by limitations in its approach this is mainly due to the reliance of the programmer. Each texture relies on a programmer writing and testing the procedural code until the resulting texture looks correct [3]. For these reasons this literature synthesis will not focus on procedural texture synthesis but rather focus on texture synthesis from samples.

3.2 Texture Synthesis from Samples

As we have seen from section 5.2 procedural texture synthesis techniques have a strong reliance on the programmer creating code for each type of texture desired. This is where the example-based texture synthesis can prove to be a useful technique. Instead of procedurally generating a texture a small sample image is used to synthesize larger texture areas. Texture synthesis using example-based methods are filtered into three categories; Pixel based methods, Patch-based methods and tile based methods [4].

3.2.1 Pixel-based methods

Pixel based methods of texture synthesis create the synthesized image on a per-pixel basis. Pixel based methods that process each pixel based on its neighbourhood of surrounding pixels. The example image is searched to find the closest matching area of pixels to the neighbourhood of a particular pixel. The resulting area is then used to formulate the pixel. The search will then continue for the next pixel. To make the synthetic image original the methods will often include an image of noise in the synthesis process. The main advantage of pixel-based methods is the ability to control textures at the pixel level. These methods are however very sensitive to the size of the window in order to produce adequate results. [4]

Efros and Leung [5] developed a system which used non-parametric sampling that would grow a texture out from a centre pixel in an image. Neighbourhoods of similar pixels are found which match the square window of pixels already grown around the pixel being analyzed. These neighbourhoods are found to be similar by modelling it as a Markov Random Field (MRF) measuring the probability brightness values of pixels given the brightness values of its neighbourhood. Similar neighbourhoods are chosen at random and are used to create each new pixel being grown from the centre. The technique employed by Efros and Leung was adequate for most textures but was computationally expensive and would often grow portions of the texture that were undesired. Other neighbourhood variations are used in methods such as the proposed solution by Wei and Levoy [6] and Turk [3]. Wei and Levoy used a raster scan line method and a smaller neighbourhood of only pixels made recently. They also sped up Efros and Leung's method by using tree-structured vector quantization. Turk used both the square neighbourhoods used by Efros and Leung, and a half square neighbourhood.

For increased quality Wei and Levoy's as well as Turks methods use a multi-resolution technique. This technique is a multi-resolution pyramid based on a hierarchically statistical method. Another method

introduced by Ashikhmin [6] used only the pixels around the targeted pixel for its neighbourhood to reduce the blur effect found in Wei and Levoy's method. This technique was also much faster than Wei and Levoy's even without the tree-structured vector quantization. Ashikhmin's method produces good results for natural or similar samples but can also contain obvious artefacts from some samples.

3.2.2 Patch-based methods

With patch-based algorithms a texture is generated patch by patch. These patches are chosen through the use of cutting paths. These cutting paths are created in the attempt to minimise errors due to overlapping.[4] A method used by Efros and Freeman quilts together random overlapping blocks according to a minimum cost path across the overlapped area. This obtains good results at low computational cost. Another method by Liang et al. [7] uses sampling patches according to Markov Random Field density function. This is fast and real time however errors and artefacts are created regularly. By using whole chunks of image data only a limited amount of variation is possible by these techniques.

Kwatra et al.[8] introduced a graph-cut approach that computes seams of patches. This technique is useful for partially structured textures and can create good quality images from quilting the patches. A problem with the above patch based techniques is that the resulting textures are all regular and not near-regular textures. Liu, Lin and Hays [9] put forward their deformation field solution where they view near-regular textures as statistical departures from a regular texture along different dimensions. While this is a novel and good approach their method can only create near-regular textures. A Directional Empirical Mode Decomposition-based texture synthesis algorithm has been proposed by Y. Zhang et al. [10] which decomposes the sample texture into a series of IMF images in the inherent direction of the texture image. This technique manages to maintain the features of the sample image while synthesizing at a high level. It is also generates textures in a short amount of time.

Patch-based methods manage to preserve the global structure of the sample texture often much better than pixel-based techniques. However they sometimes produce local artifacts on the overlapping regions and do not keep the local consistency as well as pixel-based methods. The search for good patches and cutting paths is not trivial and will often require iterative improvement [4].

3.2.3 Tiling-based methods

In tiling-based methods tiles are pre-computed and placed seamlessly together to create a texture. This makes synthesizing a large texture very fast.[4] However tile based methods often fall victim to similar artefacts as patch-based methods.

The first of these techniques is the Wang Tiles by Cohen et al. [11] method which uses a set of tiles to synthesize on the fly. This was proven to be highly efficient for real-time applications with memory constraints. This technique however suffered from artefacts created from the corner, joint and sampling problems. Recently ω-tiles having been introduced by Ng et al. [12] with a higher quality of pre-computed tiles and smaller tile sets than Wang tiles. This has been shown to produce tiles of reasonable quality while only using 16 tiles as opposed to 64 tiles generated for Wang Tiles. This means that it is even more efficient than Wang tiles with respect to computation and memory. A method introduced by Zhang and Kim [4] aimed at the reducing the artefacts produced by Wang Tiles. Their removes the key artefacts using graph cut and sampling techniques. The tile set is reduced from that of Wang Tiles however the method requires greater pre-computation in developing the tiles.

4. Tabulated Comparison of texture synthesis methods

Method	Synthesis method	Synthesis scale	Synthesis quality	Synthesis speed	Synthesis type
Wei and Levoy [6]	Pixel-based	Multi-scale	Medium	Medium	General textures
Ashikhmin M[13]	Pixel-based	Single-scale	Medium	Fast	A wide variety of textures
Liang L, Liu C, Xu Y, et al. [7]	Patch-based	Single-scale	Good	Real-time	Natural textures
Efors AA, Freeman WT [14]	Patch-based	Single-scale	Medium	Medium	General textures
Kwatra V, Schödl A, Essa I, et al.[8]	Patch-based	Single-scale	High	Fast	Partial structural textures
Liu YX, Lin WC, Hays J.[9]	Patch-based	Single-scale	Good	Fast	A wide variety of textures
Lefebvre S, Hoppe H. [15]	Pixel-based	Multi-scale	High	Real-time	Near-regular textures
Kwatra V, Essa I, Bobick A, et al. [16]	Pixel- and patch-based	Multi-scale	High	Fast	A wide variety of textures
Zhang Yet al.[10]	Patch-based	Multi-scale	High	Fast	A wide variety of textures
Cohen et al. [11]	Tile-based	Single-scale	Good	Real-time	A wide variety of textures
Ng et al. [12]	Tile-based	Single-scale	Good	Real-time	A wide variety of textures
Zhang and Kim [4]	Tile-based	Single-scale	Medium	Real-time	A wide variety of textures

Fig 1. Method comparison for texture synthesis techniques [10].

5. Conclusion

From this literature synthesis we have shown the existing techniques and highlighted the applications for texture synthesis. We have given motivation as to why texture synthesis is important and a popular research area. The three main techniques for texture synthesis were explained with greater emphasis into the analysis of example-based techniques. The emphasis on example based techniques was due to procedural methods being so reliant on the programmer creating the procedural method for each new type of texture.

We found that there are three main types of example based techniques. These are pixel-based methods, patch-based methods and tile based methods. Pixel-based techniques were found to be efficient good at synthesising textures at a local level however sometimes lost parts of the global structure. Pixel-based techniques are reliant on the choice of window size which can often eliminate the growth of garbage at the cost of more computation. Patch-based methods were found to be more computationally efficient than pixel-based methods. They are also good at maintaining global structures however often produced artefacts over the seams of the patches. Using whole chunks of image data also limit the amount of variation possible by patch techniques. The last group of methods were the tile-based methods which were even more efficient than patch based methods and capable in memory constrained environments. This makes them ideal for real time texture synthesising however they are prone to the same problems as patch based methods as well as new problems specific to the tile-based methods.

6. References

[1] K. Perlin, “S I G G R A P H ’ 85,” in *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 95)*, 1985, vol. 19, no. 3, pp. 291–300.

[2] S. Worley, “A Cellular Texture Basis Function,” in *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 96)*, 1996, pp. 291–294.

[3] G. Turk, “Texture Synthesis on Surfaces,” in *SIGGRAPH ’01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, no. August, pp. 347-354.

[4] X. Zhang and Y. J. Kim, “Efficient texture synthesis using strict Wang Tiles,” *Graphical Models*, vol. 70, no. 3, pp. 43-56, May 2008.

[5] A. A. Efros and T. K. Leung, “Texture Synthesis by Non-parametric Sampling,” in *Computer Vision. The Proceedings of the Seventh IEEE International Conference*, 1999, vol. 2, pp. 1033 - 1038 vol.2.

[6] L.-yi Wei and M. Levoy, “Fast Texture Synthesis using Tree-structured Vector Quantization,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques SIGGRAPH 00*, 2000, pp. 479-488.

[7] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, “Real-time texture synthesis by patch-based sampling,” *ACM Transactions on Graphics*, vol. 20, no. 3, pp. 127-150, Jul. 2001.

[8] V. Kwatra, I. Essa, and A. Bobick, “Graphcut Textures : Image and Video Synthesis Using Graph Cuts,” *ACM transactions on graphics*, vol. 22, no. 3, pp. 277–286, 2003.

[9] Y. Liu, W.-chieh Lin, and J. Hays, “Near-Regular Texture Analysis and Manipulation,” in *SIGGRAPH ’04 ACM SIGGRAPH 2004 Papers*, 2004, vol. 23, no. 3, pp. 368-376.

[10] Y. Zhang, Z. Sun, and W. Li, “Texture synthesis based on Direction Empirical Mode Decomposition,” *Computers & Graphics*, vol. 32, no. 2, pp. 175-186, Apr. 2008.

[11] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen, “Wang Tiles for Image and Texture Generation,” in *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 2003, pp. 287–294.

[12] T.-young Ngt, C. Wen, T.-seng Tan, X. Zhang, and Y. J. Kim, “Generating an ω -Tile Set for Texture Synthesis,” in *Proceed- ings of the 23rd Computer Graphics International 2005 (CGI’05)*, 2005, vol. 2005, pp. 177–184.

[13] M. Ashikhmin, “Synthesizing natural textures,” in *Proceedings of the 2001 symposium on Interactive 3D graphics - SI3D ’01*, 2001, pp. 217-226.

[14] A. A. Efros and W. T. Freeman, “Image Quilting for Texture Synthesis and Transfer,” in *Proceedings of ACM SIGGRAPH. Los Angeles: ACM Press*, 2001, pp. 341–7.

[15] S. Lefebvre and H. Hoppe, “Parallel Controllable Texture Synthesis,” in *SIGGRAPH ’05 ACM SIGGRAPH 2005 Papers*, 2005, pp. 777-786.

[16] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, “Texture optimization for example-based synthesis,” *ACM SIGGRAPH 2005 Papers on - SIGGRAPH ’05*, p. 795, 2005.