ISSN:0975-9646

Aarthi.K/ (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (5) , 2015, 4465-4468

Texture Synthesis – A Comparative Study

Aarthi.K
PG Scholar,M.E - II year
Department of Computer Science and Engineering
Jansons Institute of Technology
Coimbatore, Tamilnadu

Abstract **- Texture synthesis has a variety of applications in computer vision, graphics, and image processing. An important motivation for texture synthesis derived from texture mapping. Texture images usually come from scanned photographs, and the available photographs may be too small to cover the object surface. However, it remains difficult to design an algorithm that is both efficient and capable of generating results in high quality. This paper presents the concept of non-parametric method for texture synthesis and TSVQ method. Non-parametric method aims to preserve a local structure and TSVQ method which avoid exhaustively searching the input image pixels and comparisons are made between two methods.**

Keywords **- Texture synthesis, Texture mapping, TSVQ, Non-parametric**

I. INTRODUCTION

Texture synthesis has been an active research topic in computer vision both as a way to verify texture analysis methods, as well as in its own right [1]. Texture synthesis is a method that takes in small patch of texture and produces a new texture. Texture is one of the important characteristics of a digital image. Although textures have been widely studied over the past two decades, a precise definition of texture still does not exist. We define a texture as an image containing no explicit objects. There are two categories of textures: structural textures and statistical textures. A structural texture consists of primitives with their relocations based on some replacement rules, such as scaling, rotation, translation, and reflection. Structural textures are frequently used to investigate the relationships among similarity, homogeneity and symmetry in the responses of the human visual system whereas structured approach sees an image texture as a set of primitive Texel in some regular or repeated pattern. This works well when analyzing artificial textures [4].

In the general definition of this problem, an input sample of a texture is given, and the goal is to produce more of that texture. The simplest solution is to tile the texture sample on a rectangular grid of desired size. However, even if the sample can be tiled seamlessly, the resulting larger grid structure is easily noticeable and it distorts the perception of the actual texture. More sophisticated techniques are required for reproducing the actual texture with all its features and nothing more. The most eminent property of a texture is its regularness. A regular (also called deterministic, structured, periodic) texture is characterized by a primitive element that is regularly placed on a grid or a lattice [5].

There are several methods are available for texture synthesis. They are tilling, stochastic texture synthesis, single purpose structured texture synthesis, chaos mosaic, pixel based texture synthesis, patch based texture synthesis and chemistry based texture synthesis. Pixel based texture synthesis typically synthesize a texture in scan-line order by finding and copying pixels with the most similar local neighborhood as the synthetic texture. These methods are very useful for image completion [7]. Texture synthesis by non-parametric sampling and TSVQ methods are based on pixel based texture synthesis.

The remainder of this paper is organized as follows: In Section 2, we review the some texture synthesis techniques. In section 3, we detail the concept of non-parametric and TSVQ method and section 4 describes the comparisons between two methods followed by our conclusions presented in the final section.

II. RELATED WORK

Texture synthesis has received a lot of attention recently in computer vision and computer graphics. The most recent work has focused on texture synthesis by example, in which a source texture image is re-sampled using either pixel-based or patch-based algorithms to produce a new synthesized texture image with similar local appearance and arbitrary size.

Alexei A. Efros and Thomas K. Leung [1] says that the texture synthesis process grows a new image outward from an initial seed, one pixel at a time. Texture synthesis by non-parametric sampling is based on pixel-based algorithm generate the synthesized image pixel by pixel and use spatial neighborhood comparisons to choose the most similar pixel in a sample texture as the output pixel.

Li-Yi Wei and Marc Levoy [6] introduced the Fast Texture Synthesis using Tree-structured Vector Quantization. It is a very simple algorithm that can efficiently synthesize a wide variety of textures. More over implementation includes the process of Edge handling and initialization. However is capable of generating high quality results.

Anteneh Addis Anteneh [2] tells that, second round of synthesis is conducted on the first synthesized image. The motivation for doing this is to optimize the synthesized image so that any irregularities caused during the first run of optimization will be minimized during multiple runs as the synthesized texture becomes more refined since its current pixel neighborhoods are more similar to the input image.

Otori and Kuriyama [4] pioneered the work of combining data coding with pixel-based texture synthesis. Secret messages to be concealed are encoded into colored dotted patterns and they are directly painted on a blank image. The capacity provided by the method of Otori and Kuriyama depends on the number of the dotted patterns. However, their method had a small error rate of the message extraction.

III. PIXEL BASED TEXTURE SYNTHESIS

A. Non-Parametric Sampling

In this paper, the algorithm grows texture pixel by pixel, they choose a single pixel (P) as a unit of synthesis so that model could capture as much high frequency information as possible. Already synthesized pixels in a square window are used as the context. The non-parametric sampling [1] technique although simple, is very powerful at capturing statistical processes.

Markov Random Field model is used to model the texture. To synthesize a pixel (P), first we have to search the sample image for pixels with similar neighborhood to P, then construct a histogram for the distribution of these pixels, finally sample that distribution. Similarity is based on the Gaussian weighted sum of difference. Sample image is finite so an exact neighborhood match might not be present. So we find the best match using SSD error and take all samples within some distance from that match.

Synthesizing one pixel is applicable, when its neighborhood pixels are already known, but it cannot be used for synthesizing the entire texture. So we go for heuristic method i.e., texture is grown in layers taken randomly from the sample image. So the algorithm has been modified to handle unknown neighborhood pixel values. This can be done by normalizing the error by the total number of known pixels when computing PDF for P. This algorithm produces good results for a wide range of textures. The only parameter set by the user is the width of the context window. Window size corresponds to the degree of randomness.

The following figure 1 has been synthesized several times while increasing window size. In (a) the context window is not big enough to capture the structure of the ring so only the notion of curved segment is preserved. In (b) context captures whole ring, but does not know anything about inter ring distances. In (c) we can see rings getting away from each other finally In (d) the inter ring structure within the reach of window.

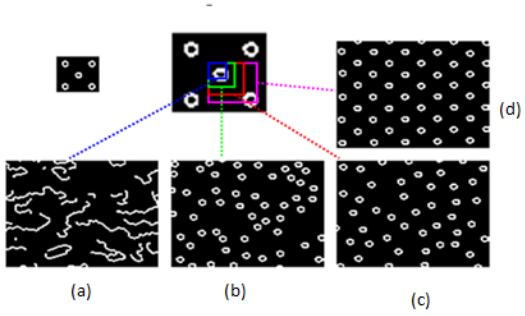

Fig.1 Window size corresponds to the degree of randomness

1) Merits

This approach is well-suited for constrained synthesis problem i.e., hole filling. Black regions in each image are filled in sampling from that sample image it can be easily applied to motion synthesis such as ocean waves, rolling clouds and there is no visual discontinuities between original hole outline and the newly synthesized patch.

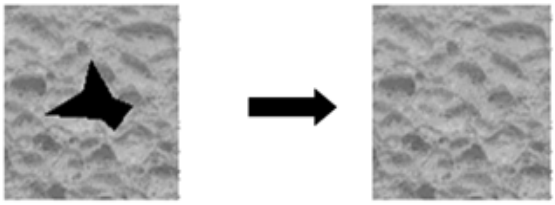

Fig.2 Hole-filling Examples

2) Demerits

Its textures slip into a wrong part of the search space and start growing garbage or get stuck at a particular place and produce verbatim copies of the original.

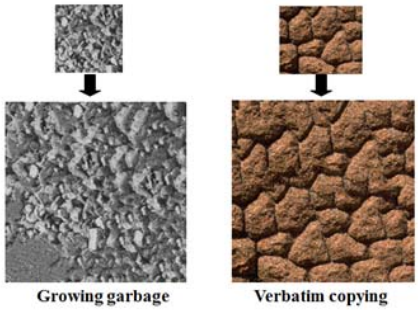

Fig.3 Failure Examples

B. TSVQ

Wei and Levoy published "Fast Texture Synthesis Using Tree-Structured Vector Quantization". In this paper, they describe a texture synthesis algorithm based on Markov Random Fields. The algorithm produces high-quality stochastic textures by computing the value of each pixel in the synthesized image through a deterministic searching process in the input image. In the second half of the paper, they explain how Tree-structured vector quantization (TSVQ) [6] can be used in a search method to help speed up their algorithm. While they claim that the tree-structured vector quantization searching technique greatly improves the processing time, the overall texture synthesis algorithm does not change if using the simple exhaustive search method described in the first half of their paper. As my purpose is to describe the basic texture synthesis algorithm.

The great strength of Wei and Levoy's algorithm is that it can be run in a single resolution or at multiple resolutions using image pyramids which can improve the results of the final texture as well as speed up the texture synthesis generation time.

1) Wei and Levoy's Algori'thm -- Single Resolution

Step 1 **:** Produce a random white noise of the desired size.

Step 2 **:** Match the random image's histogram to the input image's histogram.

Step 3 **:** Replace each pixel in the random image in raster scan order with a pixel from the input image that best matches it. To compute the best match, use the SSD of the neighborhood around the pixel.

First, the algorithm creates a random white noise image of the size m*n, where m and n are the dimensions of the texture patch to be generated. Since a true random white noise will not have the same color distribution as the input image, the random image will not have the same color distribution as the input image must be modified so that its histogram matches that of the input image. This is done by using pyramid based texture synthesis. Finally, each pixel in the random image is replaced with a pixel from the input image which best matches it. The pixel in the input image is replaced with the lowest sum of squared differences (SSD) in an S*S neighborhood when compared with the pixel, where S is specified by user.

There are two things to pay special attention to when computing the SSD of the pixels' neighborhoods. First, when computing the value for any particular pixel in the random image, only some of the pixels in its neighborhood have been given definite values. In other words, when computing the value for the pixel at (x, y) in the random image, only pixels in raster scan order before the one at (x, y) have been computed already. The part of a pixel's neighborhood that has already been computed is called its casual neighborhood. Only the casual neighborhood should be used to compute the amount of match for a pair of pixels.

Second, the neighborhood around a pixel on the edge of an image must be computed in a slightly different manner. For the random image, the edges are treated toroidally (i.e. a pixel on the left edge of the image is a neighbor of the pixel in the same row of the image but on the right edge; the corresponding fact is true of the top and bottom edges). By treating the edges in the random image toroidally, you guarantee that the generated texture will be tillable. When the toroid rule is used to find the value for a neighbor, the casual neighborhood rule is ignored. In this way, the random pixels on the right and bottom edges of the image are used to compute the final synthesized image thus adding a bit of randomness to the result.

For the input image, edges must be treated differently. An input image is not guaranteed to be tillable. So, if the image where to be treated toroidally, discontinuities in the generated texture could arise. To solve this problem, only pixels with a completely interior neighborhood are considered for placement into the generated image.

2) Wei and Levoy's Algorithm -- Multiple Resolutions

Step 1 **:** Do steps 1 and 2 of the single resolution algorithm.

Step 2 **:** Compute the Gaussian pyramid of the input image and the random image.

Step 3 **:** Starting at the lowest resolution level of the pyramids, do step 3 of the single resolution algorithm on each level of the pyramids using multi-resolutional neighborhoods to compute a pixels match.

Wei and Levoy [6] make only slight modifications to their single resolution algorithm in order to make it work in multiple resolutions. First, the random image for the multiple resolution versions is computed in the same way as it was in the single resolution version. The next step is to compute the Gaussian pyramid for each of the images. Finally, for each level of the pyramid starting at the lowest resolution, the new level is computed by matching neighborhoods as in the single resolution algorithm. The one difference is that neighborhoods are computed in a slightly different manner. The neighborhood for pixel (x,y) at level L of the random image pyramid is its casual, toroid S x S neighborhood as well as the non-casual, toroid N x N neighborhood of pixel (x/2,y/2) at level L+1, where N < S. The same is true for the input image's pyramid except that its neighborhoods must again be completely interior.

IV. COMPARISION

Texture synthesis by non-parametric sampling produces a better result for constrained synthesis. This method can easily apply to motion synthesis like ocean waves. Compared to TSVQ method, this algorithm is slow because sample image contains too much different type of texels. TSVQ [6] method is used for efficiently synthesizing a broad range of textures. A proper acceleration using TSVQ, typical textures can be generated within seconds. TSVQ method is easy to use and produces efficient results. TSVQ method is used for direct synthesis over meshes, Texture compression or decompression, constrained synthesis and also to model the geometric details whereas non-parametric [1] is used only for constrained synthesis. Compared to non-parametric method TSVQ is fast. Hence TSVQ method is better than Non-parametric sampling.

V. CONCLUSION

Synthesis by example is one of the most promising ideas for providing end-users with powerful content creation tools. Non-parametric method aims to preserve a local structure and TSVQ method which avoid exhaustively searching the input image pixels. Compared to non-parametric method TSVQ is fast and efficient. Hence TSVQ method is better than Non-parametric sampling. Almost anyone can learn to create new results by combining examples. This is an intuitive approach, requiring minimal experience and technical skills. I hope this survey will help to spread these ideas and that it will encourage research efforts directed towards these goals.

ACKNOWLEDGMENT

A special thanks to Vijay Kumar SD for useful discussions on many texture-related issues and his valuable comments. All work done in this paper will surely help to the researchers for future work.

REFERENCES

[1] Alexei A.Efros and Thomas K.Leung *"Texture Synthesis by Non-Parametric Sampling"*, IEEE International Conference on Computer Vision, Corfu, Greece, September 1999

[2] Anteneh Addis *"Texture synthesis using TSVQ and Target Re-synthesis"*, CMSC 635 – Project

[3] Chaur-Chin Chen and Chien-Chang Chen *"Texture Synthesis: A Review and Experiments"* National Tsing Hua University, Journal of Information Science and Engineering 19, 371-380 (2003)

[4] Hirofunni otori and Shigeru Kuriyama *"Data-Embeddable Texture Synthesis"* Toyohashi University of technology

[5] Isk Bars¸ Fidaner, *"A Survey on Variational Image Inpainting, Texture Synthesis and Image Completion"* Bogazici University

[6] Li-Yi Wei and Marc Levoy *"Fast Texture Synthesis using Tree-Structured Vector Quantization"* Stanford University

[7] https://en.wikipedia.org/ wiki/Texturesynthesis

AUTHOR PROFILE

Aarthi K is currently studying the M.E degree with the Department of Computer Science and Engineering, Jansons Institute of Technology, Coimbatore where she received the B.E. degree from the Department of Computer Science and Engineering, in 2014.