Transform domain texture synthesis on surfaces

TRANSFORM DOMAIN TEXTURE SYNTHESIS ON SURFACES

By

Rupesh Nathuram Shet

A doctoral thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Loughborough University

June 2008

Supervisors: Dr. Eran Edirisinghe, Dr. Helmut Bez

Director of Research: Dr. Chris Hinde

ABSTRACT

Transform Domain Texture Synthesis on Surfaces

Rupesh N. Shet, June 2008.

In the recent past application areas such as virtual reality experiences, digital cinema and computer gamings have resulted in a renewed interest in advanced research topics in computer graphics. Although many research challenges in computer graphics have been met due to worldwide efforts, many more are yet to be met. Two key challenges which still remain open research problems are, the lack of perfect realism in animated/virtually-created objects when represented in graphical format and the need for the transmission/storage/exchange of a massive amount of information in between remote locations, when 3D computer generated objects are used in remote visualisations. These challenges call for further research to be focused in the above directions. Though a significant amount of ideas have been proposed by the international research community in their effort to meet the above challenges, the ideas still suffer from excessive complexity related issues resulting in high processing times and their practical inapplicability when bandwidth constraint transmission mediums are used or when the storage space or computational power of the display device is limited.

In the proposed work we investigate the appropriate use of geometric representations of 3D structure (e.g. Bézier surface, NURBS, polygons) and multi-resolution, progressive representation of texture on such surfaces. This joint approach to texture synthesis has not been considered before and has significant potential in resolving current challenges in virtual realism, digital cinema and computer gaming industry. The main focus of the novel approaches that are proposed in this thesis is performing photo-realistic texture synthesis on surfaces. We have provided experimental results and detailed analysis to prove that the proposed algorithms allow fast, progressive building of texture on arbitrarily shaped 3D surfaces. In particular we investigate the above ideas in association with Bézier patch representation of 3D objects, an approach which has not been considered so far by any published world wide research effort, yet has flexibility of utmost practical importance. Further we have discussed the novel application domains that can be served by the inclusion of additional functionality within the proposed algorithms.

ACKNOWLEDGEMENT

While pursuing one's lifetime goals, many helpful and sympathetic hands come forward to help, knowingly as well as unknowingly. Often only when the goals are achieved, one becomes aware of that help.

I take this opportunity to express my deep sense of gratitude towards my supervisors Dr. Eran A. Edirisinghe and Dr. Helmut E. Bez for their sincere support, patience, and encouragement throughout my research and for their valuable guidance provided in writing and completing this thesis. It is not often that one finds a supervisors and colleagues that always find the time for listening to the little problems and roadblocks that unavoidably crop up in the course of performing research. Their technical and editorial advice was essential to the completion of this thesis and has taught me innumerable lessons and insights into the workings of academic research in general.

I am also very much thankful technical and clerical staff members of Department of Computer Science for their guidance towards the successful completion of this thesis. I would also specially like to thank all members of the Digital Imaging Research Lab, Loughborough University, and my friends Dhammike, Iffat, Sajeed, Amey for their support and valuable guidance while completing my research project.

I have a deep regard for my wife, Jagruti and Son Miheer, Krish who has always been supportive of me and provided me with immeasurable love and inspiration. Above all I must thank my wife her for all the sacrifices made on my behalf during my studies. I am profoundly thankful to my parents for creating all the opportunities for me and for their love, affection, and encouragement. I should express my sincere gratitude for my parents-in-law and the rest of the family for their extensive love, support and friendship.

Finally, and perhaps most importantly, I am forever indebted to my elder brother Sandeep Shet and his wife Deepa who encouraged me to consider research study. I wish to thank them for their support, understanding, endless patience and encouragement when it was most required.

Further I wish to thank Greg Turk for providing the source code of his texture synthesis algorithm on 3D surface. I wish to thank Vivek Kwatra, Muath Sabha and Dhammike Wickramanayake for their valuable support and discussions throughout my PhD research.

Rupesh Nathuram Shet ***30st June 2008***

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

FIGURES

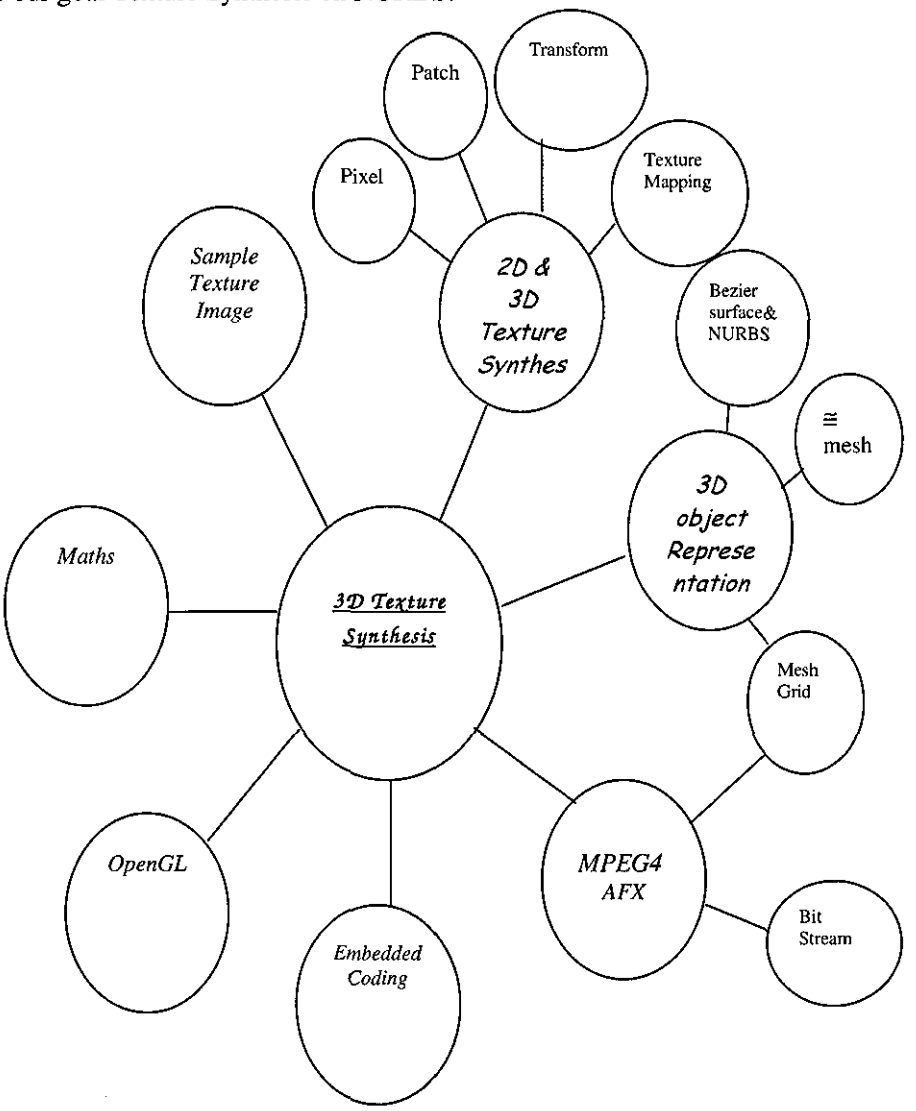
TABLES

ABBREVIATIONS AND NOTATIONS

1D	ONE DIMENSIONAL
2D	TWO DIMENSIONAL
3D	THREE DIMENSIONAL
ANN	APPROXIMATE NEAREST NEIGHBOURS
AFX	ANIMATION FRAMEWORK EXTENSION
DCT	DISCRETE COSINE TRANSFORM
STFT	SHORT TIME FOURIER TRANSFORM
WT	WAVELET TRANSFORM
CWT	CONTINUOUS WAVELET TRANSFORM
DWT	DISCRETE WAVELET TRANSFORM
EZW	EMBEDDED ZEROTREE WAVELETS
FFT	FAST FOURIER TRANSFORM
GFX	GRAPHICS FRAMEWORK EXTENSION
HH	HIGH-HIGH
HL	HIGH-LOW
LL	LOW-LOW
LH	LOW-HIGH
IDWT	INVERSE DISCRETE WAVELET TRANSFORM
IDCT	INVERSE DISCRETE COSINE TRANSFORM
MEBC	MINIMUM ERROR BOUNDARY CUT
MPEG	MOVING PICTURE EXPERTS GROUP
MRA	MULTI RESOLUTION ANALYSIS
MRF	MARKOV RANDOM FIELDS
SSD	SUM OF SQUARED DIFFERENCES
SSE	SUM OF SQUARED ERROR
TSVQ	TREE STRUCTURED VECTOR QUANTISATION
CCD	CHARGE –COUPLED DEVICE
DPCM	DIFFERENTIAL PULSE CODE MODULATION
SAT	SUMMED AREA TABLE

LINE OF INVESTIGATION

The flowing diagram illustrates the areas that we have studied briefly in order to achieve our goal Texture Synthesis on NURBS.



Chapter 1

An Overview

1.0 INTRODUCTION

Digital technology has already taken over the entertainment market with new developments in 3D modelling, animation, digital films (Finding Nemo, Cars, Shrek), console based games, internet games, and PC based games. gaming, movie, and television industries are widely relying on virtual multimedia production. Virtual production technology has been developed to a level where it can be used to replace some or all of the scenery in conventional television production. In using such 3D environments, the modelling and rendering of scenes becomes a key part, which is time consuming, costly and requires a large amount of computational power. Rendering natural textures on computer generated models gives realism to 3D objects in an application popularly known as 'Texture Mapping' within the research community. Therefore accurate and efficient texture mapping has become a vital part of object realism. The above has been an important motivation to the texture synthesis research presented in this thesis.

This chapter introduces the research problem, original contributions made by the research and the structure of the thesis. To this effect, section 1.1 presents various definitions of *texture* quoted from the literature. Section 1.2 presents a categorisation of texture and introduces the basic concepts of texture synthesis on 2D and 3D surfaces along with examples of possible practical applications. Section 1.3 provides an insight to applications of texture synthesis and section 1.4 details the motivational factors and the need for the research presented in this thesis. Section 1.5 introduces the specific objectives of the research and presents the original contributions made. Finally section 1.6 details the thesis structure.

1.1 DEFINITION OF TEXTURE

The word 'Texture' has been given various definitions in the computer graphics literature. Several key definitions can be quoted as follows:

- IEEE standard 610.4-1990 for image processing & pattern recognition [1]: *"Texture is an attribute representing the spatial arrangement of the grey levels of the pixels in a region."*

- R.C. Gonzalez & R.E.Woods: Digital Image Processing [2]: "We *intuitively view this descriptor as a measure of properties such as smoothness, coarseness and regularity."*

- A.K.Jain: Fundamentals of Image Processing [3]: *"The term texture generally refers to repetition of basic texture elements called texels. A texel contains several pixels, whose placement could be periodic, quasi-periodic or random. Texture may be coarse, fine, smooth, granulated, rippled regular, irregular or linear."*

- Julesz (1960s-1980s) : *Motivated by the question, "What features and the statistics are characteristics of a texture pattern, so that texture pairs that share the same features and statistics cannot be told apart by pre-attentive human visual perception [129] ", texture is defined as an equivalent class of images in 2D that share identical features.*

More widely in the literature, "texture" is defined as an image containing repeating patterns, with the possibility of random variations. This definition seems closer to A.K.Jain's [3] and holds well as the majority of natural surfaces consist of repeating elements. This definition, although limited, is sufficient to describe many surface properties.

1.2 TEXTURE AND TEXTURE SYNTHESIS

In figure 1.1, we present examples of a variety of texture patterns that can be used to describe a wide variety of surfaces such as, terrain, plants, minerals, walls, fur, skin and some natural phenomena like weathering and corrosion. These texture samples can be categorised in a texture spectrum ranging from stochastic to regular. In this thesis, for ease of texture classification and detailed performance analysis of texture synthesis algorithms, we define and use four texture categories namely, regular, near-regular, irregular and stochastic [4] (see figure 1.1). These form a texture spectrum in which the perceptual structural regularity varies continuously from regular to randomness.

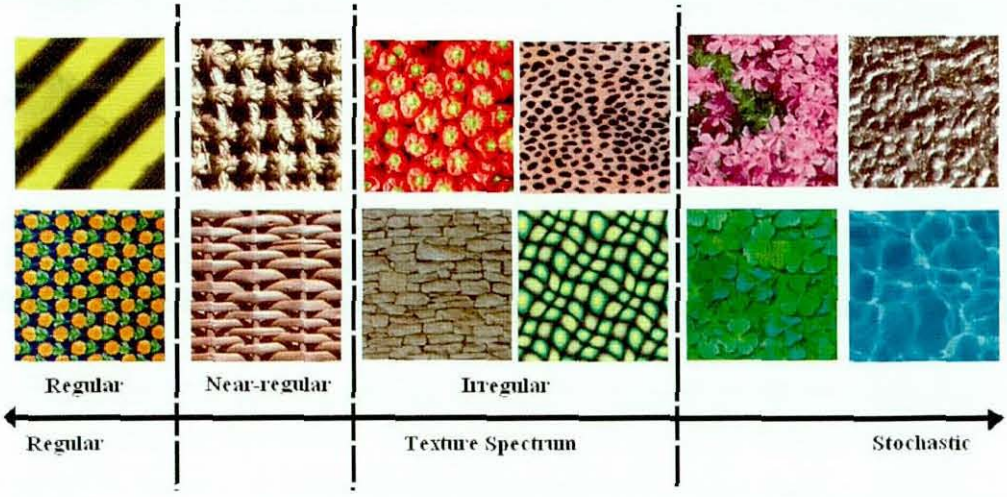


Figure 1.1 Textures spectrum

A texture synthesis algorithm starts from a sample texture image and attempts to produce a texture with a visual appearance similar to that sample, by repeated placement of micro patterns of texture elements onto a surface, such that when perceived by a human observer, the resulting texture appears to be generated by the same underlying stochastic process (see figure 1.2).

A large number of solutions to the problem of texture synthesis have been presented in the literature (see chapters 2 and 3), attempting to perform equally well for different categories of texture samples. Unfortunately, creating a real-time, robust algorithm that generally performs efficiently for all categories of texture has been proven to be

difficult. Hence finding an efficient solution to the texture synthesis problem still remains an open research problem.

Figure 1.2 and 1.3 illustrates examples of texture synthesis on 2D and 3D surfaces, respectively.

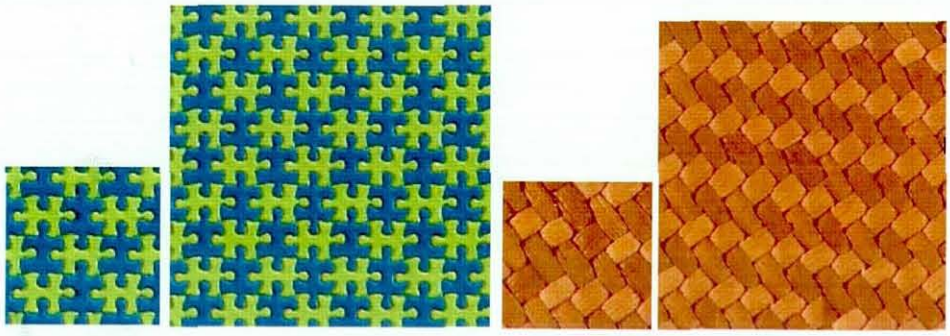


Figure 1.2 Texture synthesis on a 2D, planar surface

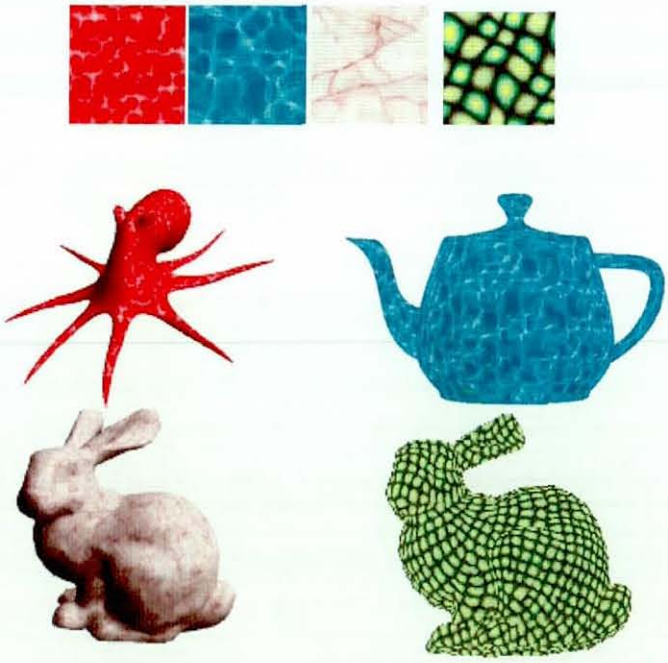


Figure 1.3 Texture synthesis on a 3D surface

1.3 APPLICATIONS

While creating a computer game, digital movie, or virtual reality production, texture synthesis is an essential part of the associated computer graphics application, as it gives realism to the scene. Samples for these applications can be obtained from a variety of sources such as hand-drawn artwork, photographs of natural scenes or artificially generated textures. Certain applications require the creation of a large area of texture at a remote end and thus may benefit from data compression, i.e. instead of first generating and subsequently sending/transmitting the information related to the texture of the entire synthesised surface, simply send the information related to a selected sample of texture and subsequently use it to synthesis the entire surface at the receiver end. This way, large 2D surfaces and arbitrarily shaped 3D object surfaces can be synthesised, maintaining the underlying pattern of the texture sample. This is where the necessity of a texture synthesis algorithm, which enables the creation of large areas of textured surfaces, originates.

Theoretically, texture synthesis enables rendering texture on any surface topology, without a visual appearance which indicates that the rendered texture is a repetition of the texture pattern specific to the sample used. However a number of practical challenges have to be met in fulfilling the above aspiration. Apart from its primary use in surface rendering of artificially created scenes, texture synthesis provides a useful solution to a number of other practical problem domains. These include, image denoising, occlusion fill-in, bump mapping, *animation* applications such as ocean waves, rising smoke, or a burning fire, and *data compression* of areas covering grassland, forests, sky or a beach.

1.4 MOTIVATION OF RESEARCH

During the past decade, a number of texture synthesis algorithms on 3D surfaces, have been introduced to the research community. However, the main focus of these algorithms has been obtaining the highest quality of texture synthesis for a wide variety of texture sample categories. Although vital, this research focus has often led to the introduction of algorithms that are too slow for many practical applications. With the rapid growth of the internet, heterogeneous networks providing for

multimodal devices and the digital movie industry, the requirement of a faster texture synthesis method is becoming inevitable. Despite ever increasing speed of computer hardware, decreasing cost of RAM and storage space, researchers have not yet been able to achieve real time texture synthesis on 3D surfaces. This highlights the fact that the texture synthesis algorithms themselves requires to be made faster and more efficient in the future, if one is to achieve real time texture synthesis on complex, 3D surfaces.

Recently some attempts have been made in the literature to consider only the visually significant information of a texture sample, in synthesis [5-12] on 2D surfaces. In these attempts the visually significant information has been obtained by first representing the spatial domain pixel information (i.e. pixel values themselves) in transform domains such as Discrete Cosine Transform domain (DCT) and Discrete Wavelet Transform (DWT) domain, and subsequently selecting only the visually significant information (i.e. transform coefficients). Thus less information is used in the searching and matching processes, increasing the speed of the texture synthesis algorithm by a significant amount, maintaining the same or sometimes better quality of synthesis. This research has already proven that it is possible to produce good quality texture synthesis with only 10-20 percent of the total texture information.

In this research we extend the work of [5-12] above to texture synthesis on arbitrarily shaped 3D surfaces. Though this research does not claim novelty in performing texture synthesis in the transform domain that allows the design of faster synthesis algorithms, a number of original contributions have been made in the application of transform domain texture synthesis ideas to arbitrarily shaped 3D surfaces, as detailed in the following section. Further, note that in the absence of a quantitative metric to measure the quality of a synthesised texture, our claim to improved quality is based purely on subjective judgment.

1.5 RESEARCH AIM, OBJECTIVES AND CONTRIBUTIONS OF THE THESIS

1.5.1 Aim & Objectives

The aim of this research is to "Design and implement novel algorithms for high quality, high speed texture synthesis on 3D surfaces that will have functionality which will extend their use into new application scenarios". The specific research objectives can be listed as follows:

- To study possible shortcomings of existing pixel domain 3D texture synthesis algorithms and suggest novel ideas towards their improvement.
- To investigate possible application of wavelet domain 2D texture synthesis algorithms to patch based 3D texture synthesis algorithms.
- To investigate the Bézier form of surface representation of arbitrarily shaped 3D objects and its use in the texture synthesis of such surfaces.
- To suggest how transform domain texture synthesis can be applied to the Bézier form of surface representation, enabling an effective and efficient method for 3D texture synthesis.
- To extend the above to progressive texture synthesis on 3D surfaces, extending the algorithm's practical use to application scenarios that are not supported by existing texture synthesis algorithms (see pages 86 and 100).
- To identify the limitations of the proposed algorithms and propose future directions of research.

1.5.2 Contributions of Research

The following original contributions have been made by the research presented in this thesis. The conference and journal papers have been included in the Appendix C, and are referenced as C1 to C5.

A. Introducing progressive texture synthesis capability to the popular "Texture synthesis on a surface", work of G. Turk. [62]

The main focus of this contribution was to introduce progressive texture synthesis capability to Turk's original work on texture synthesis of arbitrarily shaped 3D

objects. It was shown that this extended functionality will enable its use in many new application domains/scenarios.

B. Improvement of D.S.Wickramanayake et al.'s [5, 39] work on Transform Domain Edge Blending.

This work investigates the performance of the transform domain edge blending algorithm proposed by D.S.Wickramanayake et al. [5, 39] in blending (i.e. defining the boundary between two patches) texture samples of different characteristics and proposed improvements based on its limitations. In particular the algorithm proposes the use of a transform domain extension of the popular pixel domain, graph cut algorithm proposed by Kwatra et al.[36], to enhance the performance of the original edge blending idea of [12, 39]. The improved algorithm was demonstrated to be able to blend textures of a wide variety of kinds at comparatively better quality.

C. Transform domain progressive texture synthesis on a Bézier surface.

Bézier surfaces provide the means for flexible representation of geometric surfaces. By adjusting the surface parameters, Bézier surfaces can be easily redefined to fit many geometric surface topologies. In this work the original DWT domain texture synthesis ideas [12, 39] was used along with the improved form of edge blending (see contribution 2) to synthesise texture on a single Bézier surface. Whilst inheriting the advantages brought about by the use of transform domain texture synthesis such as enhanced texture quality, progressive nature and speed, this contribution enables the flexible adaptation of original ideas in DWT based texture synthesis to texture synthesis on geometric surfaces.

D. A control polygon based texture synthesis approach that aims to achieve significantly high speeds in 3D texture synthesis

In this work we propose the use of Bézier control polygons in texture synthesis. Firstly the texture is synthesised on a control polygon (i.e. a close representation of the actual surface) instead of on the actual 3D surface using a significantly improved version of the texture synthesis algorithm outlined in contribution 3, above. This texture is subsequently projected onto the actual surface that may consist of a number of patches forming a geometric surface. We show that this approach provides wider

applicability to the underlying texture synthesis process and flexibility of extension towards creating animations of 3D objects.

E. Transform domain texture synthesis on NURBS

NURBS (Non-Uniform Rational Basis Splines) can be used to represent an arbitrarily shaped 3D object. This work combines all previous contributions to enable fast, efficient texture synthesis on arbitrarily shaped 3D objects. This approach takes into account the possibility of using user defined tangential vectors to indicate the preferred direction of texture pattern on a given surface. We show that the proposed algorithm has wide applicability in areas such as CAD/CAM, digital movie production and computer games etc.

1.5.3 Scholarly Contributions

Refereed Conference Papers

1. R. N. Shet, H. E. Bez, E. A. Edirisinghe, "A Control Polygon Based Texture Synthesis on Bi-Quadratics Rational Bézier Patches", in 3rd International Conference on Computer Graphic Theory and Applications VISIGRAPP2008, Jan 2008, ISBN: 978-989-8111-20-3, Page No: 45-52.
2. R. N. Shet, H. E. Bez, E. A. Edirisinghe, "Progressive Texture Synthesis on Geometric Surfaces Parameterised by Bi-Quadratics Rational Bézier Patches" in the Proc. of CVMP07: 4th European Conference on Visual Media Production, November 2007, ISBN CD 978-0-86341-843-3.
3. R. N. Shet, E. A. Edirisinghe, H. E. Bez, "A Wavelets Based Max-Flow/Min-Cut Approach For Texture Synthesis" in The IET International conference in Visual Information Engineering: Convergence in Graphic and Vision (VIE2007), ISBN CD 978-0-86341-830-3.
4. R. N. Shet, E. A. Edirisinghe, H. E. Bez, "Progressive Texture Synthesis on 3D surfaces" in Proceedings of Sixth IASTED international conference on Visualization, Imaging, and Image Processing (VIIP 2006), ISBN Hardcopy:0-88986-598-1/CD:0-88986-600-7, pp.136-141.

Submitted Journal Paper

1. R. N. Shet, H. E. Bez, E. A. Edirisinghe, "Texture Synthesis on 3D surfaces using Bézier control polygons ", submitted to International Journal on Signal Processing: Image Communication, Elsevier, June 2008.

1.6 ORGANIZATION OF THE THESIS

The thesis is organised into eight chapters as described below.

Chapter 1 provides an overview of the research domain, identifies open research problems and provides a summary of the proposed solutions. The chapter provides the research aim, identifies the specific objectives and clearly states the original contributions made by the research presented in the thesis.

Chapter 2 presents details of the existing literature on 2D and 3D texture synthesis algorithms and analyses their performance. The chapter further discusses in detail, different 3D surface representation methods and identifies the advantages and disadvantages of each method.

Chapter 3 concentrates on providing the background knowledge related to the research context in which the novel texture synthesis algorithms have been proposed. In particular the chapter details Discrete Wavelet Transforms (DWT), the Embedded Zerotree Wavelet (EZW) based wavelet transform coefficient prioritisations scheme that is used in this thesis to identify visually significant texture content and details of four benchmark algorithms that are used for comparison purposes.

Chapters 4, 5, 6 and 7 provide information on original contributions made by the work presented in this thesis to the field of texture synthesis on 2D and 3D surfaces. Each chapter concludes with a discussion on the original contributions made by the proposed algorithms and possible improvements.

Finally Chapter 8 concludes with an insight into the future directions of research.

Chapter 2

Literature Review:

Texture Synthesis and 3D Surface Representation Methods

2.0 INTRODUCTION

This chapter provides a brief overview of the existing research on texture synthesis and the methods widely used for the generation of 3D models.

Due to the large amount of literature published in the texture synthesis research domain, providing a complete survey of algorithms is beyond the scope of this thesis (see [12, 13] for a complete survey). Therefore only the key algorithms/techniques have been considered in the literature survey. For clarity and ease of presentation, we classify the existing texture synthesis algorithms depending on the type of the surface, i.e. whether planar (2D) or 3D, on which the texture is to be synthesised. All texture synthesis algorithms could also be classified as parametric model based approaches (see section 2.1.1) and non-parametric model based approaches (see section 2.1.2) depending on the underlying mathematical model to be utilized. As the research focus of this thesis is the non-parametric model based category (see figure 2.1), a comprehensive analysis of the advantages and disadvantages of key papers in the area of planar and 3D surface texture synthesis (see section 2.2) using this approach, have been included and critically reviewed in this chapter.

The latter part of the chapter (see section 2.3) discusses the methods which are widely used for 3D model generation. These methods are important as they provide the basis of surfaces on which the texture is to be later synthesised.

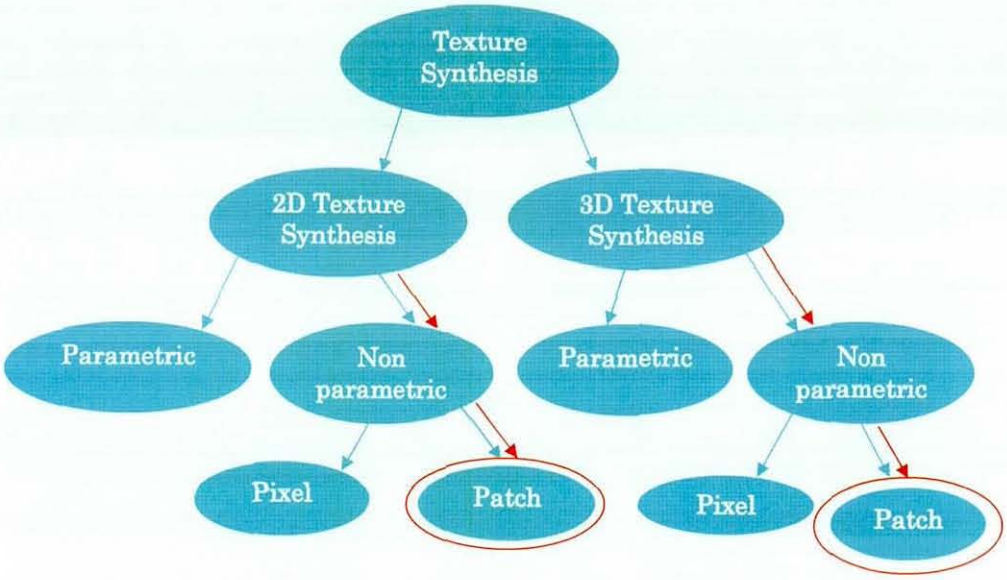


Figure 2.1: Classification of texture synthesis algorithms [Note: the paths in red illustrate the classification routes that have been the main focus of this thesis.]

2.1 TEXTURE SYNTHESIS ON PLANAR SURFACES

Texture synthesis algorithms on planar surfaces can be broadly classified into two groups, namely, A) *Parametric model based approaches*: use a number of parameters to describe a variety of textures. [Note that this topic does not form a part of the thesis, but is included in the literature survey for general discussion]. B) *Non-parametric model based approaches* (see section 2.1.2), or 'example based methods', which generate textures by directly copying pixels or patches from input/sample textures.

2.1.1 PARAMETRIC MODEL BASED APPROACHES

The parametric model based approaches are a comparatively older type of approach, invented during an era in which digital photographs could not be used to provide a sample texture. With this approach it is possible to generate textures such as water, a tree or a cloud, for e.g. using the summation and multiplication of *Sine* waves. However, only a limited set of textures, which can be modelled accurately with mathematical equations, can be synthesised. It also requires a substantial amount of

trial and error runs, which have been widely considered as a major drawback that needs further investigation. However the approach has advantages in that if a mathematical equation can be used to accurately define a required texture, there will not be any texture distortions. Further the output texture can be easily scaled in size, efficiently stored and transmitted.

In the following sections different types of parametric model based approaches are described briefly.

A. Reaction-Diffusion

Reaction-diffusion is a process first introduced by A. Turing [14] in which two or more chemicals diffuse at an unequal rate over a surface, and react with one another to form stable patterns such as spots and stripes. Various research projects have been successfully completed in this area to demonstrate how a simple texture pattern can be created by reaction-diffusion. In 1981 this research area was further explored by J. Bard [15] and J. Murray [16], independently. Their work mainly focused on texturing patterns on the coats of mammals; Bard managed to demonstrate the creation of deer and giraffe spots, where as Murray demonstrated the creation of spots according to size of an animal and took further efforts to demonstrate texturing patterns on butterfly wings.

B. Cellular Texturing

Materials such as masonry, tiling, shingles are called cellular patterns. These patterns are influenced by geometric features. In this approach the texture space is first divided into cells and each cell position is subsequently textured individually to create branching patterns, called cellular texture [17-18].

C. Weathering

Weathering is the process which causes decay, deterioration and change in appearance of the material due to the effect of surrounding environment. A few examples are corrosion of metals, efflorescence on stone and brick, fungal attacks on organic materials. With advancement of research in texture synthesis driven by the need for 'realism', different approaches to modelling or rendering surfaces with weathering effects have been introduced [19-21].

All the above mentioned techniques can generate only a limited set of texture patterns, such as the patterns on butterfly wings, spots, stripes, ice, rock, marble, stone. Hence these approaches will not be able to support the synthesis of some texture patterns. Thus, in spite of advantages such as lower memory usage and low-bandwidth requirements, this is a major drawback of the parametric approaches to texture synthesis and therefore calls for further research and investigation. Hence it is necessary to consider alternative approaches. In the next section we show that the non-parametric model based approaches provide a viable alternative.

2.1.2 NON- PARAMETRIC MODEL BASED APPROACHES

In this approach texture is created by using real world images as texture resources. In the synthesising process, pixels or blocks of pixels of the sample image are used to match a candidate pixel/block already synthesised and the process is continued until the entire output texture area has been synthesised. These algorithms are broadly categorised into two groups: pixel based approaches and patch based approaches. These approaches can be used to create a complete texture map covering the entire scene being textured, using multiple images as texture sources. Below are a complete collection of these approaches, presented and reviewed in the order they have evolved.

A. Pixel Based Approaches

In this approach the surface is synthesised gradually using pixels of the sample image, rather than a collection of pixels, such as blocks. Given a pixel that has been synthesised, a matching pixel (i.e. a pixel that will help maintain the smooth continuity of the texture pattern) is sought from the sample texture. This process is repeated until the entire output texture area is filled. In the literature many different approaches have been used to match the adjacent pixels. They include the use of image histograms, colour of neighbouring pixels, probability density function, etc., and are the key difference between the published literatures under this category of texture synthesis algorithms. However, all these approaches are highly time consuming and are often unable to capture global structure of the texture. Hence they are not applicable to regular and stochastic types of texture. Nevertheless for the

completeness of the literature survey, a brief survey of pixel based approaches is presented below.

D. J. Heeger and J. R. Bergen [22]

Heeger and Bergen focused on the synthesis of stochastic texture patterns and made the first attempt to demonstrate texture synthesis in colour. This algorithm converts a random noise image into a synthesised texture by matching the filter response histograms at different spatial scales. Their approach is mainly based on the principle (though not correct all the time) that all the spatial information characterizing a texture image can be captured in the first order statistics of an appropriately chosen set of linear filter outputs. The method consists of mainly two operations 1) Analysis phase; where the texture image is analysed with a number of texture parameter values. 2) Synthesis phase; where synthesis is done on 2D or 3D surfaces using parameter values. However this model captures an interesting set of texture properties though it is not complete. This method is more efficient than many of the previously published texture synthesis methods as the analysis phase is used to generate the parametric values which are later used in the synthesising process. This is considered as the main contribution of the approach.

J. S. De Bonet [23]

This paper outlines a technique for treating input texture images as probability density estimators from which new textures, with similar appearance and structural properties can be generated. De Bonet used a multiresolution filter based approach, in which a texture patch at a finer scale is conditioned on its 'parents' at the coarser scales. The algorithm works by taking the input texture sample and randomizing it in such a way, in order to preserve these inter-scale dependencies. Although this method is applicable to a wide range of textures, it fails to exhibit perceptually correct behaviour on smaller stochastic textures. The method is very sensitive to the choice of threshold parameters, which if chosen incorrectly detrimentally affects the output texture.

R. Paget and I.D. Longstaff [24]

Paget and Longstaff introduced a top down approach, where the frequency components of a texture are gradually introduced into a synthetic texture from low to high frequency. This algorithm uses a different approach to classification of textures. It captures the characteristics of the texture by setting up a unique statistical model. It stands on the assumption that a model captures all the visual characteristics of that texture if it is capable of synthesizing the texture visually indistinguishable from its training texture. Based on this MRF model, they propose an algorithm with the multi-scale synthesis, incorporating local annealing. Results show that this model is able to produce realistic texture. The main disadvantage of this algorithm is heavy computational cost. Although a multi-scale approach with local annealing improves the implementation of the MRF model, the speed is yet very low. As the nonparametric MRF model implemented in this algorithm captures sufficient high order statistical characteristics of texture, it can be used for synthesizing natural textures.

A. Efros and T. K. Leung [25]

Efros and Leung mainly focused on preserving local structure as much as possible in texture synthesis. They assumed a Markov random field model for the texture synthesis by repeatedly matching the neighbourhood around the target pixel in the synthesised image with the neighbourhood around all pixels in the input texture, the process starts from a seed pixel and grows outward. Neighbourhood matching is done by computing the Gaussian weighted normalised sum of square difference (SSD), between already synthesised pixels and the pixel neighbourhood of each candidate match from the sample texture. The degree of randomness of synthesis is controlled by a single parameter. However the algorithm inherits the so-called stability problem due to use of the sequential non-parametric Markov chain type approach. In other words the sequential nature of the algorithm may lead to a pixel neighbourhood that is highly dissimilar to any neighbourhood in the original image. Thus, the algorithm may become unstable resulting in synthesis artefacts. Instability depends on the choice of the local neighbourhood size. It is expected that the use of large neighbourhood or the employment of multiresolution analysis will a provide solutions

to this problem. Nevertheless, similarity measures from large neighbourhoods may not be effective, especially for highly irregular textures.

L.-Y. Wei and M. Levoy [26]

Wei and Levoy proposed an algorithm with significant changes to previous approaches, with the aim of enhancing the synthesised texture quality and speed. They used an approach similar to that of the Markov random model based approach, but used scanline neighbourhood searching in the synthesis process. Additionally they introduced the use of Tree-Structured Vector Quantization (TSVQ) to quickly search for the nearest neighbour. This pre-processing stage results in an increase of algorithmic complexity in the search for the best matching pixel. However it has been shown that the overall process is speeded up maintaining the same quality of texture synthesis as compared to the method of Efros and Leung's. Further the algorithm is extended to a multiresolution synthesis process using coarse-to-fine progression. It has been demonstrated that [26] the results obtained when using a smaller search neighbourhood within the multi-resolution approach is comparable to the results obtained when a relatively larger neighbourhood is used when using a single resolution texture synthesis process.

M. Ashikhmin [27]

The method proposed by Ashikhmin improves the quality of the synthesised texture and speed of processing only for specific classes of texture. The algorithm provides a solution to the time consuming procedure of exhaustive nearest neighbourhood search. To measure the best perceptual similarity a novel approach was proposed. While synthesizing neighbouring pixels, instead of an exhaustive search, only valid candidates of neighbourhood pixels from the input image were selected to synthesise the next pixel. From this it can be assured that each of its defined neighbours corresponds to a pixel within the input image. While searching for a limited neighbourhood of pixel locations, it results in a speed gain.

S. Zelinka and M. Garland [28]

Zelinka and Garland approached texture synthesis differently; instead of using nearest neighbourhood comparison during the synthesis process, they created a k nearest neighbour lookup table as a part of the input texture analysis stage. This table was subsequently used to make random jumps during their sequential texture synthesis stage. No neighbourhood comparison was done during the synthesis process, which is fundamental in the intended algorithmic speedup.

M. Tran and A. Datta [29]

Tran and Datta's method focused on avoiding the blurring of texture features due to the use of a fixed sized neighbourhood and the acceleration of neighbourhood search. It is noted that Wei & Levoy's and Ashikhmin's methods are based on a fixed neighbourhood search, which results in dominant artefacts in the output image. In order to attempt to overcome this problem, Tran & Datta used a variable size neighbourhood search. In this approach the pixel difference between the matching neighbourhood from the sample texture and synthesis pixel is first calculated and compared with an initial threshold. If the matching error is above the threshold the neighbourhood size is doubled, with the ultimate hope of finding a better match. To accelerate neighbourhood matching the authors have used the approximate nearest neighbours (ANN) [121] approach.

M. Sabha, P. Peer and P. Dutré [30]

Sabha et al.'s method focuses on reducing high frequency discontinuities present in pixel based synthesis algorithms. The algorithm is forced to select at least one of the pixels from the direct neighbourhood pixels in each causal neighbourhood, in order to match within a predetermined threshold. This enhances the speed of the overall texture synthesis algorithm due to the high probability of a direct neighbourhood match. They have also used the Kd-tree approach [122] to enhance the searching process. This algorithm provides a visually pleasing result.

B. Patch Based Approaches

This is the latest approach to texture synthesis, which was first introduced in the year 2000. This approach overcame some of the drawbacks of the pixel based approach, such as the low processing speed and the inability to capture global structure of the texture. In patch based approaches, instead of using one pixel, a collection of pixels which is known as a 'patch' is used as the basic processing element. The key papers proposing patch based approaches to texture synthesis are reviewed below:

Y. Q. Xu, B. N. Guo, and H-Y. Shum [31]

Xu et al.'s paper is considered the birth of "patch-based" texture synthesis algorithms. Instead of copying one pixel at a time from an input image, they copied entire patches, i.e. a specific collection of adjacent pixels. The algorithm randomly transferred patches from an input texture onto an output texture lattice, smoothing the resulting edges between overlapping patches with a simple cross-edge filtering technique. The random selection of adjacent patches often resulted in visible patch boundaries and thus the algorithm worked well only for a very limited set of stochastic textures.

L Ling, C E Liu, Y. Xing, B. Guo, and H-Y. Shum [32]

In this paper Ling et al. proposed a method that is an extension to their previous work. The texture patches were sampled in relation to the local conditional MRF density, which resulted in a reduction of artefacts in patch boundaries. The search for matching patches was optimised by using a Principle Component Analysis (PCA) of the input texture and a quad tree pyramid based ANN search. In addition a '*feathering*' (alpha-blending) [34] approach was used to address the problem of constrained texture synthesis, and achieve better visual quality at the patch boundaries. This algorithm produced good quality texture synthesis results for both constrained and unconstrained texture patterns, with considerable gain of speed, as compared to previous algorithms.

A.Efros and W. T. Freeman [33]

Efros and Freeman used an approach similar to that of Ling et al. However instead of using *'feathering'* (alpha-blending) [34] to blend patches they used an optimised Minimum-Error Boundary Cutting (MEBC) algorithm at the edges. Although the MEBC approach makes the algorithm computationally expensive, it results in an optimised patch boundary, given two matching blocks.

A. Nealen and M. Alexa [35]

Nealen and Alexa proposed an adaptive and hybrid algorithm, in which the patches are split and re-matched if the overlapping error between two patches increases beyond a user defined mismatch error tolerance. For a quick calculation of overlapping errors a Fast Fourier Transform (FFT) based approach was used. Using larger patches for texture synthesis wherever possible assures the global structure of the texture to be maintained. Any remaining errors in the overlap region of patches were eliminated using a pixel-based re-synthesis approach. It was shown that the proposed approach works well with most types of textures. However the algorithm suffers from excess computational cost related problems due to multiple uses of Fourier transforms and matching iterations.

V. Kwatra, A. Schödl and I. Essa [36]

Kwatra et al. proposed an algorithm for image and video texture synthesis, in which their approach is similar to previous approaches to copying a patch from a sample image to an output image, and stitch them together along optimal seams. In their approach, patch size is not chosen a priori, but instead a graph cut technique [37] is used to determine the optimal patch region. This technique can be applicable to structured, random and video texture, which produce good quality texture. This algorithm [36] is further discussed in detail in this thesis, as it is one of the benchmarks, and is extended in transform domain applications.

M. Cohen, J. Shade, S.Hiller and O. Deussan [38]

Cohen et al. use principles of provably non-periodic tilings of the plane to generate arbitrary amounts of non-repetitive texture. They used 'Wang Tiles' patches that are squares whose edges are each assigned a colour value. Their tile generation procedure randomly selects a set of base tiles from the input texture, one tile for each Wang Tile edge colour, and constructs Wang Tiles by assembling the necessary 4-permutations of the base tile set. For each 4-permutation, the overlap region is repaired by performing a minimum-error boundary-cut (MEBC [25]). If the resulting Wang Tile set is below a visual quality threshold (i.e. artefacts along the MEBC), a new base tile set is selected and the assembly procedure is repeated (optimization). However as a result of the random selection process, some diamond shaped artefacts may occur.

D. S. Wickramanayake, E .A. Edirisinghe and H. E. Bez [5, 12, 39, 40,]

Wickramanayake et al. proposed various techniques to synthesise texture using Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) domains, in which coefficient blocks are stitched together to form transform domain texture synthesis. Their approaches were shown to produce good quality texture synthesis for a wide range of textures, with higher speed and with added functionality that enables its use in bandwidth adaptive, compressed domain, progressive texture synthesis applications. In their work Wickramanayake et al. have discussed the issue of computational speedup, in which one should attempt to use less amount of texture related information within the synthesis process. However this may degrade the quality of texture synthesis. To avoid this, the authors suggested considering the use of only the visually significant information in the synthesis process.

The first algorithm proposed by Wickramanayake et al. introduced DWT based texture synthesis [5, 12, 39], where the main goal is to synthesise texture at a faster rate by using minimum computational power and considering a multiresolution approach. In this approach, coefficient blocks of the spatio-frequency components of the input texture are efficiently stitched together in the synthesis process. They proposed two methods; the first used a constant block size whilst the second used a variable block based on an adaptive block splitting algorithm. The later method makes

sure that the largest possible block that can capture global structure of the texture is ultimately used in the synthesis process, while maintaining the mismatched error of the overlapped boundaries below an error tolerance constant.

Wickramanayake et al. subsequently proposed a DCT [40] based, fast, texture synthesis algorithm that minimises computational power by using the minimum number of visually significant coefficients in the synthesis process. The authors demonstrated that the use of only the visually significant coefficients in the texture synthesis process leads to an increase in the quality of the synthesised texture, and also a significant reduction of the cost of the computations required. To overcome the artefacts at the boundary of patches the authors further proposed the use of a simple, transform domain weighted edge blending algorithm that was demonstrated to work well. Experimental results were provided to prove that their approach is applicable to a wide range of textures of regular to stochastic nature.

Wickramanayake et al. finally extended their original ideas in DWT domain texture synthesis by using an automatically generated threshold, which enabled the detection of significant coefficients of each sub-band which are used to define patch matching templates in the texture quilting process [5]. This extended approach was proven to significantly improve texture quality and reduce computational cost as compared to the original DWT domain texture synthesis idea. This algorithm [31, 33] is further discussed in detail within this thesis, as it is selected as one of the benchmarks to evaluate the performance of the novel algorithms proposed in this thesis. In our work this algorithm is further improved and later extended to enable its use in texture synthesis on 3D surfaces.

2.2 TEXTURE SYNTHESIS ON ARBITRARY SURFACES

Many algorithms for texture synthesis of arbitrarily shaped 3D objects have been proposed in the literature. These algorithms are generally more functionally demanding (e.g. the practical need of specifying the direction of texture) and computationally expensive due to the need of having to synthesise texture on non-planar surfaces using non-regularly shaped patches. However despite many efforts, producing a texture synthesis algorithm for arbitrarily shaped 3D objects which is fast,

enables the synthesis of a wide range of texture types and has added functionality that enables its efficient application in a wide practical area, still remains an open research problem.

Texture synthesising approaches on arbitrarily shaped 3D surfaces can be generally classified into two categories; 2.2.1 parametric model based approaches [note: this category does not form a part of the thesis, but a review has been included merely for completeness]. 2.2.2 Non-parametric model based approaches. The key algorithms that represent these categories are reviewed below.

2.2.1 PARAMETRIC MODEL BASED APPROACHES ON SURFACES

Parametric model based texture synthesis approaches on surfaces can be further classified into two sub-categories; Solid and Geometric texturing. These are described below.

A. Solid Texturing

Solid texturing is defined as 'colour variation throughout an entire 3D space'. It means that the object can be "carved" in a solid block. A significant amount of research has been carried out in this direction, which is presented under three further categorisations below:

Procedural Solid Texturing: The first approach to solid texturing was proposed by Gardner [42, 43] to define terrain, tree, water-stream, and cloud textures using summation and multiplication of sine waves. Later Peachey [44] proposed solid texturing approaches that extended some well-known 2D texture synthesis algorithms to illustrate the synthesis of solid wood and granite texture onto surfaces. Perlin [45] observed that most natural textures have a more or less random structure. Based on this assumption Perlin introduced a solid noise and a turbulence function, which was used to demonstrate the synthesis of marble, cloud and ocean waves textures. However, the main drawback of all these approaches is the difficulty of determining the function parameters, in order to obtain a desired kind of texture. Further the experimental background often results in high computational cost.

Analytical Solid Texturing: To overcome the problems of procedural texturing such as the need for expert knowledge in determining the parameters and the computational cost of experimentation, this approach is used to synthesise "automatically". This approach uses one or more 2D views (digitised photograph) of the desired texture. The first method uses spectral analysis of texture, introduced by Ghazanfarpour et al. [46] in order to obtain both a basis function and a perturbation function, which showed many interesting results. However it failed because of high computational time for noise filtering. This work which was further extended by Ghazanfarpour et al. [47] had the observation that a single view is not sufficient to capture all the aspects of solid texture. Dischler et al. [48] proposed a hybrid, spectral and spatial based approach for processing 3D blocks. The main drawback of this approach is that it is not able to deal with the structured model. Heeger and Bergen [49] proposed a pyramid method for analyzing and synthesizing noisy 2D texture, as well as stochastic solid texture. Their approach is based on psychophysical studies, which assume that the human texture perception mechanism applies a bank of space localised filters to pre-attentively discriminate texture fields. Therefore, the method allows one to generate some types of synthetic textures having a close visual resemblance with respect to the model. Beyond 2D texture they extend their work to a 3D white noise block to synthesise texture. This approach can only be applied to a limited type of texture, and also is computationally expensive due to the cost of analyzing the sample texture.

Physical Simulation Approaches to Texturing: A further alternative attempt made on solid texturing is to use a physical simulation, which means to simulate a process based on a known science such as physical, chemical or biological. Unfortunately, the research community has not carried out sufficient detailed investigations of this approach. Buchanan [50] proposed a model to synthesise wood using a voxel based approach, where a full model of a tree was shown, whereby wood texture was obtained by carving in the tree trunk. Hirota et al. [51] generated patterns of 'cracks' using the voxel based approach. The object voxels are considered to be connected to their neighbours by springs. A force is applied on these springs to deform the 3D lattice. If the force exceeds yielding point the spring will tear and generate a crack pattern. This method is applicable for the simulation of a wet cube of mud that becomes dry. Turk [52] proposed texture synthesis based on reaction-diffusion. In his

paper he showed how texture can be directly generated on the geometry of a surface. He illustrated the synthesis of the cluster spots on leopards and web-like patterns on giraffes. The major advantage of the physical approach is its closeness to reality and production of very good quality results. However, the limitation is that a suitable scientific model does not exist at all times. Further an expert's knowledge is required to generate a mathematical model for the texture. Thus simulations may be expensive and time consuming.

B. Geometric Texturing

Geometric texturing consists of adding "real" third dimension information to a surface in the form of "real" apparent geometry. These approaches can be used to render several complex 3D phenomena, such as fur, cotton, lawns which cannot be rendered correctly by simple colour variation, because roughness is related to geometry. In order to realistically simulate wrinkled surfaces, Blinn [53] introduced a normal perturbation technique, known as bump mapping. This technique modifies the surface of normal vectors and allows the geometric appearance to be retrieved. However, in some cases, bump mapping remains insufficient because the surface is not "really" deformed. Other methods presented by Cook [54] and Max [55] were proposed to resolve the limitations of bump mapping. Cook [54] proposed a displacement mapping method that consists of "really" displacing surface points along with their normal points. Hence, all the problems of bump mapping can be resolved correctly with higher computational costs. Max [55] proposed a less expensive method but limited to self-shadow. However, even these extensions of bump mapping cannot resolve the problem of a highly complex structured surface like cotton or fur. Therefore, other kinds of approaches, closer to geometric modelling, appeared later in the form of hypertextures by Perlin [56], and texel maps by Kajiya [57]. Texel mapping and hypertexturing turn out to be particularly useful for "soft" (translucent) features. However, these approaches are complex and extremely time consuming.

Another form of texture-mapping close to geometric modelling consists of directly distributing small geometric elements on surfaces, such as for cellular textures by Fleischer et al. [58] and macro structured textures by Dischler et al. [59].

All of these 3D texturing approaches (displacement maps, hypertextures, texels, cellular textures.) can be considered as "geometric" techniques, since they add some "real" geometric components to the surfaces, these approaches are seldom used due to high complexity, the time required for texture synthesis and the need for expert knowledge which is often difficult to quantify.

2.2.2 NON PARAMETRIC MODEL BASED APPROACHES ON SURFACES

Non-parametric model based approaches address some of the key limitations of parametric model based approaches and thus find many practical applications. These techniques can be divided into two categories, namely; pixel based approaches and patch based approaches.

A. Pixel Based Approaches

L.-Y. Wei and M. Levoy [60]

Wei and Levoy proposed a texture synthesis algorithm for arbitrarily shaped surfaces, based on their own 2D texture synthesis algorithm [26]. In this method each mesh vertex initially establishes a local parameterization of a surrounding vertex. Subsequently using this vertex as a centre, a small rectangular neighbourhood is created. Finally a modified version of a planar texture synthesis algorithm [26] is used for texturing the 3D surface. The modifications introduced include the replacement of the scan-line neighbourhood search by a random neighbourhood search and the use of rectangular parameterization with a tangent direction at each mesh vertex, coupled with a scale factor derived from the mesh vertex. Taking a multi-resolution approach to texture synthesis on 3D surfaces, Wei-Levoy extended Turk's [61] *'re-tiling polygonal surface'* method to create a multi-resolution representation of vertices. The proposed algorithm showed low texture discontinuity and low distortion for the particular type of texture experimented. Wei and Levoy also mentioned that the methods that were proposed for planar texture synthesis cannot be applied to general surfaces and need to be changed for surface synthesis. A similar non-parametric approach to texture synthesis on a 3D surface was previously proposed by Turk [52]

in the paper titled, *procedural texture synthesis.* However this algorithm worked well only for a limited class of textures and was thus practically limited.

G. Turk [62]

Turk demonstrated texture synthesis on a 3D surface using an extended version of Wei-Levoy's 2D texture synthesis approach, popularly known as known as Turk's [61] *'re-tiling polygonal surface'* method. The approach is based on a method that uses image pyramids. Turk proposed to use a mesh hierarchy to serve in place of such pyramids. This approach is similar to the Wei-Levoy surface approach, yet has three significant differences: 1) Turk used a user defined smooth vector field where as Wei-Levoy used a random and symmetric vector field based on a relaxation algorithm. 2) Turk used a sweeping order derived from the smooth vector field for vertex transversal, whereas Wei-Levoy's approach visited the mesh vertex in a random order. 3) Turk used surface marching to construct the mesh neighbourhood, whilst Wei-Levoy performed flattening and re-sampling of the mesh. Results of both methods are comparable.

In this thesis we consider Turk's algorithm above as a benchmark algorithm (see chapter 4) that is used for performance evaluation of the proposed novel approaches to surface texture synthesis.

L. Ying, A. Hertzmann, H. Biermann, and D. Zorin [63]

Ying et al.'s intentions were synthesis on a 3D surface. The authors proposed two algorithms to handle a variety of different texture samples; one is a multi-scale synthesis algorithm which is similar to that of Turk [62] and Wei-Levoy [60], i.e. to synthesise texture directly on to the surface, as synthesising texture on a rectangle, and then mapping on to a surface may lead to artefacts. Identifying the applicability of most texture synthesis algorithms to a limited set of texture types, the authors proposed an algorithm which generalised the texture Wei-Levoy [26], and the other is a coherent synthesis algorithm based on Ashikhmin [27]. To use Wei-Levoy's multi-resolution approach the authors have proposed to use a novel approach to 'chart sampling'. These charts are created from the surfaces by using the multi-resolution

subdivision approach. The texture is first synthesised at a coarse-level using a 'surface marching' algorithm [Note: In surface marching algorithm a grid of sample locations is collected which corresponds to a grid of locations in the plane, using a tessellated mesh representation of the surface. These grids are further used in a texture synthesis process], and coarse-to-fine refinement is carried out subsequently. Another method is a coherent synthesis run in a single pass over all samples, on the surface. This approach also uses a 'surface marching' algorithm which collects a grid from the sample location that corresponds to the grid location in the plane using a tessellated mesh representation in order to synthesise texture as per Ashikhmin algorithm.

This method has certain advantages over previous methods such as less memory usage, faster speed and being able to support a variety of texture samples, which are handled rather poorly by some multi-scale algorithms. Although this provides good results, it inherits same drawbacks of Wei-Levoy's [26] and Ashikhmin's [27] algorithms as it is time consuming and applicable to a limited type of texture, such as irregular and stochastic.

T. Xin , J. Zhang, L. Liu, X. Wang, B. Guo, and H-Y. Shum [64]

Tong et al. proposed a method for synthesising a bidirectional texture function on an arbitrary surface. The bidirectional texture function (BTF) is described as texture arising from both spatially-variant surfaces and surface meso-structures. Initially a so-called texton analysis [64] is carried out on a sample BTF. Texton analysis consists of three steps: 1) build 3D texton vocabulary, 2) assign texton labels to the pixels to get the 2D texton map and 3) construct a surface texton space by calculating the dot-product matrix, and discarding the appearance vectors. Afterwards, the surface is synthesised based on a surface texton. To gain speed of operation the algorithm uses Turk's [61] '*re-tiling*' method to achieve multi-resolution and K-coherence search, for each candidate pixel. Later, the surface is rendered to BTF using the texton map, computing the viewing and lighting conditions for each vertex in its local texture coordinate. This algorithm provides good results, though having extremely slow synthesis, rendering process and applicable to only those materials which can be described by 3D textons.

J. Zhang, K. Zhou, L. Velho, B.Guo, and H.-Y. Shum [65]

Zhang et al. proposed a method to show progressively variant textures (heterogeneous texture synthesis) which can model local texture variations, the scale, orientation, colour, and shape variations of a texture element. This approach is widely useful for texturing animal skin onto models of animals as the texture surface and patterns change according to the surface shape. They modelled progressively variant textures in 2D and 3D, using a feature based wrapping and blending technique. Feature based wrapping provides user control for shape variations of a texture element, whereas feature based blending is used to blend texture to provide smooth transitions between homogenous textures. To synthesise texture over a surface, they use a texton mask to avoid the breaking of texture. The texton mask captures the most prominent texture elements from a sample texture. This method is mostly based on user interaction to create a texton mask, vector field and is also very lengthy, time consuming as well as being an expensive process. Yet it exhibits good quality progressive variants of an animal 3D model.

S. Zelinka and M. Garland [66]

Zelinka and Garland extended their own 2D texture synthesis algorithm to 3D surfaces [28]. These algorithms gained speed compared to pervious algorithms by one or two orders of magnitude. The authors assigned an offset in texture space to every edge of the mesh, by walking over each mesh vertex on the basis of a pre-computed vertex ordering. Finally ensuring that each corner of a triangle received a texture coordinate, one may optionally employ texture blending to improve quality at seams. However, the results and a careful analysis indicate that the algorithm is best suited for stochastic textures.

S. Lefebvr and H. Hoppe [67]

Lefebvre and Hoppe used appearance vectors for comparison within the synthesis process. Appearance vectors that are built for this purpose contain non-local information such as features and radiance-transfer data. Vector dimensionality is reduced in order to create a new appearance-space exemplar to work in real time. This

approach completely operates on the image space of the atlas domain. The synthesising process is done from scratch for every frame, which allows the user to interactively adjust all synthesis parameters, including randomness, direction of texture, and feature scaling.

J. Han, K. Zhou, Li-Yi Wei, M. Gong, H. Bao, X. Zhang, and B. Guo [68]

Han et al. proposed to animate texture patterns over arbitrary 3D mesh surfaces. The animation of the texture pattern is controlled by flow fields over the target mesh. The texture synthesis approach is based on satisfying a MRF. The authors have used a discrete solver which is inspired by the k-coherence search method [64]. This allows interactive flow texture animation on a surface which avoids the blurry blending problems. Fast flow animation is obtained using a GPU implementation.

J. Kopf [69]

Kopf proposed a method for solid texturing from 2D exemplar based texture synthesis algorithms. Initially, they synthesise a solid 3D texture block using an exemplar based 2D texture optimization technique [69]. They have integrated a non-parametric texture optimization approach with histogram matching, which is forced to retain the pattern of 2D exemplar texture in to the 3D solids. This approach effectively models the interior material of the 3D models, which means texture is not only on the surface, but also throughout the entire volume occupied by a solid object. This approach has applicability in applications such as scattering simulation and object breaking.

B. Patch Based Approaches

F. Neyret and M.-P Cani [70]

Neyret and Cani's algorithms pre-compute the set of triangle texture samples that match together along the border. The surface is tiled at the desired scale into curved triangular areas in which these texture samples are mapped. Thus the surface is synthesised with low distortion, no cracks nor singularities. Moreover this method uses relatively small amounts of memory, since no global map needs to be stored.

However the method is restricted to synthesising stochastic texture patterns. In addition to this, sample texture is created by procedural texture synthesis or delicate manual editing. Lastly, the resulting textured mesh is more complex than the initial one, since the triangle of a mesh which crosses the limit of a texture tile is split during the process.

E. Praun, A. Finkelstein, and H. Hoppe [71]

Praun et al.'s approach is based on a patch-pasting algorithm [31] and texture mapping on 3D surfaces with a pre-computed vector field to direct irregular texture. In their system, the user specifies a tangential vector over the surface, controlling texture scale and orientation. A (possibly irregular) input texture sample is then repeatedly pasted onto the surface by growing a surface patch. The surface patch is parameterised in a texture space. The perceptibility of seams is reduced by applying alpha-blending at edges of pasted patches. The parameterization is optimised (by solving a sparse linear system) such that the vector field aligns with the frame of the texture patch. The resulting model was rendered with a texture generated by pasting in runtime-pasting (i.e. pasting texture during run-time). Later texture memory is significantly lowered reducing the texture memory at the cost of rendering certain faces, multiple times. This was extended trivially to bump maps, displacement maps, and other surface appearance fields. Praun et al.'s placement of oriented texture patches over an independent parameterization of a surface produces high-quality texture synthesis when using many types of texture samples. Our observations of the experimental results provided revealed that the algorithm fails when a texture sample with a large pattern as compared to the surface shape, is being synthesised. The algorithm is unable to capture low-frequency texture, without sacrificing high frequency randomness. This algorithm also suffers from mismatching at patch boundaries and is applicable to only a particular type of texture.

C. Soler, M.-P. Cani, and A. Angelidis [72]

Soler et al., showed how a mesh can be seamlessly textured with only the input texture and a set of texture coordinates for each vertex. A hierarchy of face clusters was set up for the input mesh and subsequently each cluster is flattened (if distortion

is too high, the cluster is subdivided for later processing). For each flattened cluster, a texture patch in the sample texture which is a best match to an already textured neighbour is found. Note that if the error due to texture discontinuities with the existing neighbour is too high, the cluster is subdivided for later processing. In three passes texture coordinates are mapped onto all polygons in the cluster. Unlike previous methods, this method does not use a vector field, instead it lets texture propagate itself, irregularly. Further, similar to the Lapped Texture approach's runtime-pasting [71], the texture memory overhead for rendering is minimal. Additionally, all faces are rendered only once. Soler et al.'s method is applicable to a wide variety of textures and produces a quality synthesis process. However drawbacks are that depending on the number of polygons in the surface it takes a few minutes, and even up to tens of minutes for the synthesis to complete. Even then small artefacts across patch boundaries still remain.

L. Wang, X. Gu, K. Mueller, and S.-T Yau [73]

Wang et al.'s algorithm is mainly based on global conformal parameterization of a surface, which was originally introduced by Gu and Yahu [74]. It guarantees that the shapes embodied in textures are preserved on the surface, without seams or cracks. Moreover, the parameterization can segment the surface into patches, where each patch is mapped to a planar rectangle. Wang et al. applied texture synthesis on a conformal parameterised rectangle, where they proposed to use a multi-scale texture synthesis method. This method is used to non-uniformly synthesise texture on a 2D rectangle, considering the area stretching factor which is obtained from the conformal factor. This means that the algorithm has control over the local scale of the texture. This texture synthesis algorithm is similar to a previous algorithm that synthesises texture by stitching texture patches together [31]. However Wang used a multi-scale texture sample as input instead of a single sample. In order to obtain a multi-scale sample the algorithm either enlarged or minimised the sample texture by using cubic interpolation. The texture synthesis part of the proposed algorithm is simple. However a significant amount of complexity is introduced by the parameterization adopted and the specific multi-scale decomposition strategy used.

S. Magda and D. Kriegman [117]

Magda and Kriegman separated the texture pre-processing stage from the synthesis stage and defined the overall texture synthesis process as an independent two phase process. Pre-processed texture was stored on a disk and later used when needed. Although this is very slow, in many application scenarios it needs to be performed only once. In the pre-processing of sample texture, sets of pixels whose neighbourhoods have similar appearance are assigned the same texton label. Likewise several neighbourhoods with different texton labels are created. This results in the creation of a large feature vector that will eventually require a longer computation time for processing. The information searching process is reduced by creating a lookup table using the distance between every pair of textons. This texture synthesis process is an extension to the well known image quilting algorithm, where stitching is done between triangular rather than square patches. It has been shown that this texture synthesis algorithm performs well for most of the texture types and visible seams can be reduced by edge blending. However the pre-processing time can run from minutes to a few tens of minutes.

C-W. Fu and L. M. Kang [118]

In this work the authors proposed a texture tiling mechanism, where a low distortion conformal quad based map is created for the input surface. It produces proper tile orientations on all quad faces, so that texture can be laid out on quads and mapped back to the input surface accordingly. This algorithm provides an additional facility to change texture patterns by referencing different tile sets. The synthesis algorithm is based on the Image quilting approach [33], and therefore inherits all the drawbacks of image quilting algorithms such as random selection of patches, expensive cost of minimum boundary cut, somewhat limited applicability to isotropic textures and computation cost, due to the need of generating pre-synthesis textures.

2.3 SURFACE REPRESENTATION

In parallel to the advancements of 3D texture synthesis algorithms, the ever evolving requirements from industries, such as Games, Movie, Architecture, and

Manufacturing has lead to the development of effective surface approximation techniques to be used within computer aided representation/visualization of 3D objects. In this regard, 3D object modelling can be defined as the process of developing a mathematical, wireframe representation of any three-dimensional object.

3D models are created (by hand, algorithmically or scanned) by collecting 3D data and form an essential part in 3D graphics. Today, 3D models are extensively used in a variety of fields. For example, the medical industry uses detailed models of organs for study and analysis purposes. The movie and video games industry uses 3D models as characters and objects for animated and real-life motion pictures. The science sector uses 3D data as highly detailed models such as chemical compounds, reactions etc. The architecture industry uses these models to illustrate and visualise proposed buildings and landscapes. The engineering community uses 3D models for designing new devices, vehicles and structures as well as a host of other uses. In recent decades the earth science community has started to construct 3D geological models as a standard practice [123].

The 3D surface representation methods can be classified in a number of different ways. In this thesis we classify them as 1) 3D scanner based and 2) Geometric surface representation schemes.

2.3.1 3D Scanner Based Surface Representation Schemes

In the recent decade, 3D scanner based technology has been used to collect data from real-world objects or the environment, in particular related to their shape and appearance [124]. This data can be further used to construct digital, three dimensional models which are useful for a wide variety of applications. These devices are used extensively by the entertainment industry in the production of movies and video games. Other common applications of this technology include industrial design, reverse engineering and prototyping, and computer vision and documentation of cultural artefacts. Many different technologies have been used to build these 3D scanning devices. Each technology comes with its own limitations, advantages and costs. It should be remembered that many limitations in terms of the kind of objects that can be digitalised are still present. The basic purpose of a 3D scanner is to create

a point cloud of geometric samples on to the surface of the object. These points can then be used to extrapolate the shape of the object (a process called reconstruction). If colour information is collected at each point, then the colours on the surface of the subject can also be determined.

3D scanners are very similar to cameras. Like cameras, they have a cone-like field of view and are only able to collect information about non-hidden surfaces [125]. Whereas a camera collects colour information, 3D scanners collect distance information about the surfaces within the field of view. The "picture" that results from a 3D scanner device describes the distance at each point in the picture of the surface. If a spherical coordinate system is defined in which the scanner is at the origin and the vector that projects out from the front of the scanner is $\varphi=0$ and $\theta=0$, then each point in the picture is associated with a φ and θ. Together with the distance, which corresponds to the r component, these spherical coordinates fully describe the three dimensional position of each point in the picture, in a local coordinate system relative to the scanner.

Due to the cone-like field of view a single scan will not produce a complete model and hence multiple scans are required from many different directions, in order to obtain complete 3D model data. These types of scans are to be done in a common reference system, a process called *alignment* or *registration*, and then merged to create a complete model. This whole process, ranging from the single range map to the whole model, is usually known as the 3D scanning pipeline [76].

The 3D scanners can be sub divided into contact and non-contact types. Non-contact 3D scanners can be further divided into active scanners and passive scanners. There are a variety of technologies that fall under each of these categories [77].

A. Contact

In this scanner the 3D object is created using a physical touch sensor [Note: This approach is used extensively in machining and manufacturing]. The disadvantage of the approach is that it requires physical contact with the object being scanned. Thus scanning activity might modify or damage the object. This fact is very significant when scanning delicate or valuable objects such as historical artefacts. The other

disadvantage is that they are relatively slow. Other examples are the hand driven touch probes used to digitalise clay models in the computer animation industry.

B. Non-Contact

Non-Contact approach is where physical contact is not required with the object. This approach is further classified into active and passive Non-Contact body scanning.

Non-Contact Active: An active scanner uses radiation or light to detect its reflection in order to capture an object or environment. Possible types of emissions used include light, ultrasound or x-ray, MMR, CT, MRI etc.

Non Contact Passive: These types of scanners do not emit radiation themselves, but instead rely on detecting reflected ambient radiation. Most scanners of this type detect visible light because it is a readily available in ambient radiation. Other types of radiation, such as infrared could also be used. Passive methods can be very cheap, because in most cases they do not need particular hardware.

C. Reconstruction

The point clouds resulting from 3D scanners are not used directly. Most of the applications do not work with point clouds information, but instead they use polygonal 3D models. The process of converting from point cloud into a polygonal 3D model is known as reconstruction. Reconstruction involves finding and connecting adjacent points in the correct order to create a continuous surface.

2.3.2 Geometric Surface

In technical applications of 3D computer graphics, such as computer-aided design and computer-aided manufacturing, surfaces are one way of representing objects. The other ways are wireframes (lines and curves) and solids. Point clouds are also sometimes used as temporary ways to represent an object, with the goal of using the points to create one or more of the three permanent representations.

In mid 1960s Europe, French researchers pioneered work into complex 3D curves and surface geometry computation. Citroen's de Casteljau made fundamental strides in

computing complex 3D curve geometry[126], and Bézier published his breakthrough research, incorporating some of de Casteljau's algorithms, in the late 1960s [126]. The work of both de Casteljau and Bézier continues to be one of the foundations of 3D geometric surface representation to the present time. The breakthrough insight was to use *control polygons* (*control point*), a technique that was never used before. Instead of defining a curve (or surface) through points on it, a control polygon utilises points near it. Instead of changing the curve (surface) directly, one changes the control polygon, and the curve (surface) follows in a very intuitive way.

Bézier's efforts were influenced by the knowledge of similar developments by Citroen, but he proceeded in an independent manner. Bézier moved to polynomial formulations of the initial concept and also extended it to higher degrees. The result turned out to be identical to de Casteljau's curves, only the mathematics involved was different. The most fundamental parametric curve forms are the Bézier curves. Some later results include conditions for C^* joins between Bézier curves. After the early work of Gauss and Euler, parametric surfaces were well understood. The most popular of all surface methods was to become the tensor product surface. It was first introduced by C. de Boor for the case of bicubic spline interpolation [78].

Ferguson [79] used an array of bicubic patches which interpolated to a grid of data points. A generalization, capable of interpolating a rectangular network of curves, was devised by Gordon [80, 81] at General Motors; Rectangular surfaces are a map of a rectangular domain into 3D.

De Boor's recursive B-spline evaluation was the natural generalization of the de Casteljau algorithm. B-spline curves include Bézier curves as a proper subset and soon became a core technique. A first B-spline-to-Bézier conversion was found by Boehm [82]. Several algorithms were soon developed that simplified the mathematical treatment of B-spline curves. The generalization of B-spline curves to NURBS (Non-uniform rational B-splines has become the standard curve and surface form). They offer a unified representation of spline and conic geometries: every conic as well as every spline allows a piecewise rational polynomial representation.

A special case of NURBS is given by rational Bézier curves. Even more specialised are conic sections, or rational quadratic Bézier curves. Chaikin's [83] algorithm was the starting point for the initial work on subdivision surfaces, Chaikin's algorithm can be generalised to tensor product surfaces in a straightforward way. Such surfaces have a rectilinear control mesh, and after analyzing the tensor product algorithm, Doo and Sabin [84] reformulated it such that it could also be applied to control meshes of arbitrary topology.

Catmull and Clark [85] first generalised Chaikin's algorithm to uniform cubic B-spline curves and its tensor product counterpart. Then, they also reformulated it for the case of control meshes of arbitrary topology. Both surface schemes yield smooth (G1) surfaces. The Doo and Sabin surfaces have a piecewise biquadratic flavour. The Catmull and Clark ones have a piecewise bicubic flavour. Subsequently, subdivision surfaces gained more popularity, most notably in computer animation.

Hardware development in the 1980s made it possible to use the above technique. In the 1990s it became possible to produce computer-generated special effects for movies such as Jurassic Park, Toy story, Finding Nemo and Shrek [41] .

NURBS are nearly ubiquitous for computer-aided design (CAD), manufacturing (CAM), and engineering (CAE) and are part of numerous standards used industry wide, such as IGES, STEP, ACIS, and PHIGS [126]. They allow representation of geometrical shapes in a compact form. They can be efficiently handled by computer programs and yet allow for easy human interaction. NURBS surfaces are functions of two parameters that map to a surface in a three-dimensional space. The shape of the surface is determined by control points.

2.4 SUMMARY AND CONCLUSION

In section 2.1, we have reviewed some of the important algorithms out of many approaches that were proposed to the research community to perform efficient texture synthesis. We have seen that parametric model approaches, which marked the birth of texture synthesis, are still being widely used in certain simulations of texture patterns due to the advantages of being able to scale without any loss of quality, requirement

of little memory space and ease of transmission. However, it has a disadvantage as it is only applicable to a limited class of textures and requires expert knowledge for texture modeling. It was discussed that further research in this area subsequently introduced a new approach to the graphics community, i.e., non-parametric based model approaches, which are fully dependent on real-world images as a texture resource. It was shown that these approaches are applicable to a wide variety of textures and is further divided into two categories, pixel and patch based. It was shown that the pixel based category suffers from the requirement for lengthy computations. As a solution, at present GPU's are being used to accelerate the pixel based synthesizing process. However, the use of the acceleration methods that attempt to gain speed, does actually affect the quality. It was discussed that patch based approaches are better due to large basic unit construction, which enables the ability to capture the global structure of texture. It was further shown that amongst the patch based approaches introduced so far, Kwatra et al.'s [36] and Wickramanayake et al.'s [39] are the most successful due to their applicability to a large variety of texture types and final texture synthesis quality. Kwatra's algorithm [36] has speed constraints because of the use of an iterative texture synthesis procedure. As a result the algorithm consumes more computational power compared to Wickramanayake et al.'s [39]. Therefore Wickramanayake et al.'s [39] algorithm, was selected as one of the benchmarks and has been identified as having potential for further improvement and extension due to the following reasons; (a) the algorithm synthesises texture of better quality (b) the speed of the synthesising process is high due to only considering visually significant coefficients in the searching process (c) works well with most type of textures and (d) applicable in compressed domain, due to the wavelet based multi-resolution approach. This algorithm is discussed in detail in chapter 3.

Extensive experiments reveal that Wickramanayake et al.'s [39] algorithm is somewhat limited and performs sub-optimally for certain types of texture (such as regular textures). Further, the use of the simple weighted edge blending algorithm causes a blurring effect, whereas the blending approach presented within Kwatra's [36] algorithm has been proved to perform optimally in patch blending. Due to the positive features of some aspects of Kwatra's algorithm it has also been selected as a benchmark (see chapter 3) and has been discussed in chapter 3. The proposed improvement to [39] using the positive features of [36] is presented in chapter 5.

In addition to the above, in section 2.2., we have looked at various algorithms that enable texture synthesis on 3D surfaces. We have seen that 3D texture synthesis algorithms are generally classified into two categories; 3D Parametric model based approaches and the 3D Non-parametric model based approaches. It was noted that the 3D Parametric model based approaches purely depend on either mathematical functions and biological or chemical processes. These 3D texture synthesis approaches have been developed on the basis of similar 2D synthesis algorithms and thus inherit the same drawbacks, i.e., the approaches often result in high computational complexity, can be time consuming and are based on parametric models which require expert knowledge.

Resolving the limitations of 3D parametric model based approaches, 3D Non-parametric model based approaches has recently being proposed by researchers to synthesise texture on a 3D surface. These techniques can be further divided into two categories; pixel and patched based 3D approaches. It has been discussed that the pixel based 3D approaches are in general applicable to only particular classes of texture. In this thesis we have chosen Turk's [62] algorithm as a third benchmark to analyse 3D texture synthesis in the pixel domain, due to a number of reasons namely; (1) simplicity (2) availability of code (3) applicability to a wide range of textures (4) good quality of texture synthesis on a surface (see chapter 3). Most of the algorithms that accelerate texture synthesis degrade the quality of the synthesised texture. Therefore further research is required for better quality texture synthesis in real-time. As developments in patch based 2D technique progressed, researchers have been attracted to consider this approach to texture synthesis on 3D surfaces. The first algorithm to this effect was proposed by Praun et al. [71] to show how easily a surface can be rendered. However, this leads to other problems such as discontinuity, distortion and boundary artefacts, which were attempted to be resolved later by Soler et al. [72].

Although all the above methods produce good quality of texture, their application in real-time remote visualisation or display is somewhat limited, due to the enormous amount of information required, complexity of the computational processes and/or lack of applicability to the synthesis of all types of texture. To overcome many of the

above mentioned problems in this thesis we introduce a number of new algorithms (see chapters 4, 6 and 7).

Section 2.3, introduced the evolution of 3D surface representation systems in computer graphics driven by the demand in application domains such as medical, architecture, movie and video animation. It was mentioned that the surface generation is generally classified as scanner based surface generation or geometric surface generation. It was said that the scanner based surface generation approaches are extensively used in texture synthesis whereas geometric surface generation approaches although being widely used by CAD/CAM systems, games and animation applications has not been used in texture synthesis applications supported by limited bandwidth. This observation has provided a key motivation for our work in this direction. Geometric surface generation approaches nevertheless have several functional advantages such as animation and compression abilities. The geometric surface generation approaches and the mathematical concepts/theory behind such approaches are described in detail in chapter 3. The application of texture synthesis techniques on geometric surfaces will enhance realism of such surfaces extending the benefits to the wider graphics community (see chapters 6 and 7).

Chapter 3

Research Background

3.0 INTRODUCTION

The texture synthesis algorithms proposed in this thesis aim to overcome the limitations of existing texture synthesis techniques, in order to perform effectively within a limited bandwidth/channel at an optimised rate. This chapter introduces the fundamental concepts/theories on which the proposed texture synthesis algorithms have been developed. It further presents detailed information on existing texture synthesis algorithms that have provided the inspiration for the novel algorithms that are presented in this thesis. These are used as benchmark algorithms to compare the performance of the proposed novel algorithms.

This chapter is divided into 5 sections. Section 3.1 provides the fundamental theory of wavelets, wavelet families and their applications. In particular it provides details of the multi-resolution Discrete Wavelet Transform (DWT) decomposition and analysis of a texture image. Section 3.2 presents the Embedded Zerotree Wavelet (EZW) concept that is popularly used in image compression, which is altered and used to enable progressive functionalities of the proposed algorithms. Section 3.3 critically reviews the benchmark algorithms to provide valuable insights into possible directions for their further improvement. In addition this section builds up the design specifications of the novel algorithms to be proposed in this thesis, via careful analysis of the limitations of existing algorithms. Section 3.4 presents the theory of Bézier surfaces, their formation and possible use in texture synthesis. Finally section 3.5 summarises and concludes the chapter.

3.1 WAVELETS

Multi-resolution techniques have a long history in the field of computer graphics. Recently these approaches have gained importance and interest through the introduction of the mathematical framework of *wavelets*. Wavelets are efficient, easy to implement and have a major impact on several areas of computer graphics including image compression, and processing [88, 89] (using wavelet transform in order to achieve high compression), Wavelet radiosity [90] (which uses wavelets for global illumination), hierarchical modeling [91] (which uses wavelets to perform common editing tasks), in volume rendering and processing [92,93] (where wavelets are used for huge data set compression, and feature detection and enhancement), multiresolution painting (which uses wavelets to create 'infinite' resolution paint systems), image query approaches (which use a small number of the largest wavelet coefficients of an image's results in a perceptually useful signature for fast search and retrieval) [94]. A complete overview of the wavelet based algorithms in the field of computer graphics can be found in [95, 96], whereas details of their mathematical definitions can be found in [97, 98].

The wavelet transform [99-101] was developed as an alternative approach to the Short Time Fourier Transforms (STFT) in order to overcome the constant resolution problem, and to provide a time-frequency representation of a non-stationary signal. The wavelet transform uses multi-resolution techniques by which different frequencies are analysed with different resolutions. The wavelet and STFT analysis is done in a similar way i.e. its signal is multiplied with a wavelet function similar to the window function in STFT, and then the transform is computed separately for the different segments that are generated. However, there are two main differences between STFT and Wavelets, 1) in wavelets the negative frequencies are computed. 2) In wavelets the width of the window is changed as the transform is computed for every single spectral component.

3.1.1 The Continuous Wavelet Transform (CWT)

The continuous wavelet transform is mathematically represented as follows:

$$CWT_x^{\Psi}(\tau,s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} x(t)\ \Psi\ (\frac{t-\tau}{s})dt \qquad (3.1)$$

Where $x\ (t)$ is a signal to be analysed and Ψ(t) is the mother wavelet or basis function. As seen in the equation 3.1, the wavelet function (derived from mother wavelets) is used in the transformation, which is a function of two variables τ and S, the translation (shifting) and scale parameters (dilation and compression), respectively.

The mother wavelet is a prototype for generating the other window functions, which acquire the desired characteristics associated with the mother wavelet. The translation parameter τ is related to the location of the window, as the window is shifted through the signal, which corresponds to time information in the transform domain. The scale parameter S is defined as | 1/frequency | and corresponds to the frequency information. Scaling either dilates (expands) or compresses the signal. High scale corresponds to the non detailed global view of the signal, whereas low scale corresponds to the detailed view. Similarly, in term of frequency, low frequencies (high scales) correspond to the global information of a signal and high frequencies (low scales) correspond to detailed information of a pattern in the signal.

3.1.2 The Discrete Wavelet Transform (DWT)

The DWT provides sufficient information for both analysis and synthesis of a signal (i.e. the continuous signal), but with less computation time. In 1976 the technique was invented to decompose discrete time signals [99]. In DWT, the filtering technique is used to obtain a time-scale representation of a digital signal, whereas in CWT it is computed by changing the scale of the analysis window, shifting the window in time, multiplying by the signal, and by integrating over all times.

In DWT, the filters of different cut-off frequencies are used to analyse the signal at a different scale. The signals are analysed using a series of high pass and low pass filters to generate high and low frequencies, respectively. Wavelets can be realized by iteration of filters with rescaling. The filtering operations are applied on resolutions of

the signal which measures the amount of detailed information in the signal. The scale of the signal is determined by up-sampling and down-sampling (sub-sampling) operations [99]. For example, subsampling by a factor of n reduces the number of samples in the signal n times, whereas upsampling a signal by a factor of n increases the number of samples in the signal by a factor of n.

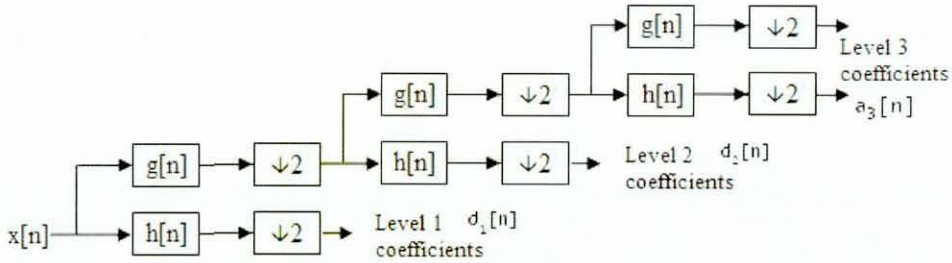


Figure 3.1 Wavelet decomposition tree

Figure 3.1 shows that DWT is computed by successive low-pass and high-pass filtering of the discrete time-domain signal. This is known as the Mallat-tree decomposition [102]. Its significance is in the manner it connects the continuous-time multi-resolution to discrete-time filters. In the figure 3.1, the sequence $x[n]$ is denoted by the signal, where n is an integer. The low pass and high pass filters are denoted by $g[n]$ and $h[n]$ respectively. At each level, the high pass filter produces detailed information, $d[n]$, while the low pass filter associated with scaling the function produces coarse approximations, $a[n]$.

At each decomposition level, the half band filters produce signals spanning only half the frequency band. This doubles the frequency resolution as the uncertainty in frequency is eliminated by half. In accordance with Nyquist's rule if the original signal has a higher frequency of ω, which requires a sampling frequency of 2ω radians, then it now has a higher frequency of $\omega/2$ radians. It can now be sampled at a frequency of ω radians thus discarding half of the samples with no loss of information. This halves the time resolution as the entire signal is now represented by only half the number of samples. Thus, while the half band low pass filtering removes half of the frequencies and thus halves the resolution, the removal of every other sample doubles the scale.

With this approach, high frequencies give a good time resolution, while low frequencies give a good frequency resolution. The filtering and sampling process is continued until the desired level is reached. The maximum number of levels depends on the length of the signal. The DWT of the original signal is then obtained by concatenating all the coefficients, $a[n]$ and $d[n]$, starting from the last level of decomposition.

3.1.3 Wavelet Families

In the Wavelet Transformation, there are a number of basis functions that can be used as the mother Wavelet. All wavelet functions generated from the mother wavelet are used in the transformation through translation and scaling. It determines the characteristics of the resulting Wavelet Transform. Therefore, the details of the particular application should be taken into account and the appropriate mother wavelet should be chosen in order to use the Wavelet Transform effectively.

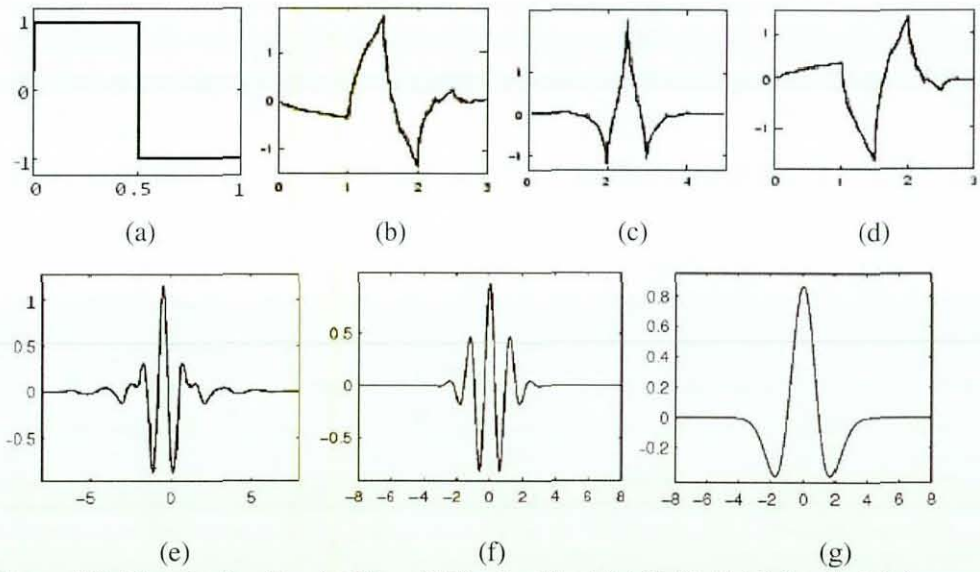


Figure 3.2: Wavelet families (a) Haar (b) Daubechies4 (c) Coiflet1 (d) Symlet2 (e) Meyer (f) Morlet (g) Mexican Hat

Figure 3.2 illustrates some of the commonly used wavelet functions. The Haar wavelet is one of the oldest and simplest wavelets. Therefore, any discussion of wavelets starts with the Haar wavelet. Daubechies wavelets are the most popular

wavelets. They represent the foundations of wavelet signal processing and are used in numerous applications.

The Haar, Daubechies, Symlets and Coiflets are compactly supported orthogonal wavelets. These wavelets along with Meyer wavelets are capable of perfect reconstruction. The Meyer, Morlet and Mexican Hat wavelets are symmetric in shape. The wavelets are chosen based on their shape and their ability to analyse the signal in a particular application.

3.1.4 Discrete Wavelet Transform on a Image

Wavelet Analysis: A large amount of perceptual data is contained in texture images. Therefore the number of bits required to represent/encode a texture image is high. However a wide range of frequency components spread throughout the image. Some of these frequency components have a highly significant effect, whilst others have a very low significance in human observation.

According to the Heisenberg uncertainty principle, obtaining an exact time frequency analysis of a signal is impossible. Simply, this principle states that one cannot know the exact time-frequency representation of a signal, i.e. one cannot know what spectral components exist at what instances of time. What one can know is the time intervals in which a certain band of frequencies exist, which is a resolution problem. Although the time and frequency resolution problems are results of a physical phenomenon, and exist regardless of the transform used, it is possible to analyse any signal by using a popular approach called multiresolution analysis (MRA). MRA, as implied by its name, analyses the signal at different frequencies with different resolutions. It is designed to give good time resolution and poor frequency resolution at high frequencies, and good frequency resolution and poor time resolution at low frequencies.

Images do not contain time varying information, instead they contain spatial information. All the above mentioned concepts are applicable for spatial-frequency relationships of images. MRA makes sense especially when the signal at hand has

high frequency components for short durations and low frequency components for long durations. Fortunately texture images are often of this type. The DWT provides a compact multi-resolution representation of an image. It gives a signal representation in correspondence to a narrow band, low frequency range, and some of the coefficients represent short data lags corresponding to a wide band, high frequency range. Using the concept of scale, data representing a continuous trade off between space and frequency can be made available for further processing (see figure 3.3 (b)).

Implementation: The application of two-dimensional DWT to an image involves the application of horizontal and vertical DWT filters over the image pixels. This subdivides texture image into four sub-bands. The resulting sub-bands labelled *LH1*, *HL1* and *HH1* represent the finest scale wavelet coefficient whereas the sub-band labelled *LL1* represents low-resolution coefficients. In order to obtain the next level of wavelet sub-bands, the sub-band labelled *LL1* is further decomposed and sampled using the vertical and horizontal DWT filters. This process is repeated until the required final decomposition is reached (see figure 3.3).

The texture synthesis process starts by applying 2D DWT (e.g. Haar Transform) on the sample texture image, which is denoted as I_{sample}. The process is defined mathematically as follow,

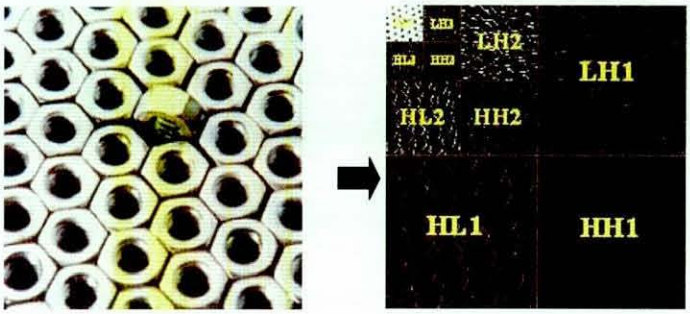
$$(I_{LL1}, I_{HL1}, I_{LH1}, I_{HH1}) = DWT(I_{sample}) \tag{3.2}$$

$$(I_{LL2}, I_{HL2}, I_{LH2}, I_{HH2}) = DWT(I_{LL1}) \tag{3.3}$$

$$(I_{LL3}, I_{HL3}, I_{LH3}, I_{HH3}) = DWT(I_{LL2}) \tag{3.4}$$

$$(I_{LLn+1}, I_{HLn+1}, I_{LHn+1}, I_{HHn+1}) = DWT(I_{LLn}) \tag{3.5}$$

We generalise the notation used in equation (3.2)-(3.5) as I_{pl} where $p \in \{LL, HL, LH, HH\}$ and $l \in \{0,1,2,..n\}$ where p represent the sub-bands within each decomposition level (*LL*-low resolution, *LH*-horizontal, *HL*- vertical, *HH* diagonal) and l represents the decomposition level. *DWT* represents the forward discrete wavelet transform.



(a) (b)

Figure 3.3: The three level discrete wavelet decomposition on a sample texture image.

3.1.5 Inverse Discrete Wavelet Transform on a DWT Decomposed Image [12]

By inspecting the DWT sub-bands (figure 3.4), it is observed that the visibilities of different frequency sub-bands are different. As a whole it increases along with increasing frequency. The highest levels of sub-bands contain the most visually significant data of an image. The inverse DWT is applied over an entire decomposed wavelet coefficient in order to reconstruct the texture image (see figure 3.4 (a)). Figure 3.4 shows various reconstructed texture images obtained by using different combinations of low frequency sub-bands. Figure 3.4 (f) was reconstructed using 6% of the wavelet coefficients.

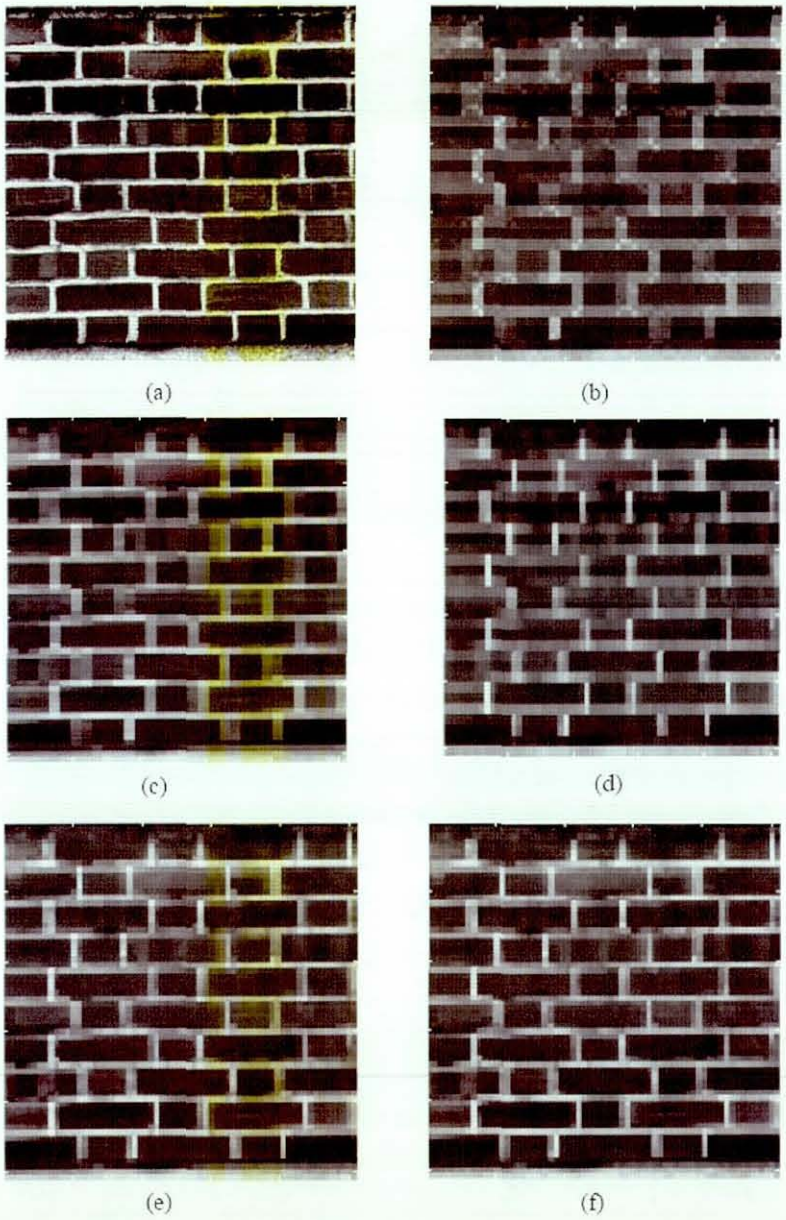


Figure 3.4: Reconstruction of "Brick" image using only 3rd level wavelet coefficient information (a) Original texture image, reconstructed; (b) only LL3 and HH3 (c) LL3 and LH3 (d) LL3 and HL3 (e) LL3, HL3 and LH3 (f) LL3, HL3, LH3 and HH3.

3.2 EMBEDDED ZEROTREE WAVELET [103]

The Zerotrees of the wavelet coefficient concept was originally introduced by Shapiro [103] for progressive encoding of images, which is also known as embedded coding. It was based on two observations on a DWT decomposed image, namely,

1. Natural images in general have a low pass spectrum. Therefore when an image is wavelet transformed the energy in the sub-bands decreases as the scale decreases, so the wavelet coefficients will, on average be smaller in the higher subbands than in the lower subbands. This shows that progressive encoding is a very natural choice for compressing wavelet transformed images, since the higher subbands only add detail.
2. Large wavelet coefficients are visually more perceptually important/significant than smaller wavelet coefficients.

These observations summarise to the fact that EZW provides a compact representation of perceptually significant coefficients and therefore enables the multi-resolution reconstruction capability of an image. The idea is to organise DWT coefficients of an image (see figure 3.3 (b)) in a prioritised order of visual significance, depending on their position and magnitude in the DWT decomposition and to subsequently encode the ordered list of coefficients following an embedded coding algorithm. Within this process if a wavelet coefficient is larger than a specific threshold (see eqn. 3.6) it is encoded (and removed from the image) and if it is smaller, it is left for the next pass. When all the coefficients have been visited the threshold is lowered and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the coefficients have been fully encoded.

In an embedded coding algorithm the encoder can terminate the encoding at any point thereby allowing a target bit rate or target distortion metric to be met exactly. On the other hand, given a bit stream, a decoder can cease decoding at any point in the bit stream. Thus it is capable of producing exactly the same image that would have been encoded at the bit rate corresponding to the truncated bit stream. EZW encoding does not really compress anything but only reorders wavelet coefficients in such a way that

they can be compressed very efficiently.

Within the research context of this thesis we are only interested in the re-ordering of wavelet coefficients in the above mentioned prioritised manner, as we intend to use it in making the texture information progressively available for synthesis.

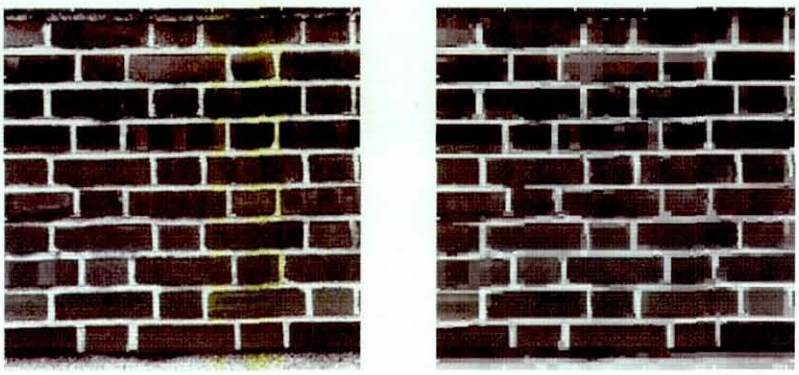
An extension of the EZW idea lies with the visibility threshold equation, which gives the minimum number of coefficients, distributed throughout all frequency bands, which will result in the best possible visual quality. The threshold can be calculated as

$$t = \frac{2^{\lfloor \log_2 (Max(|LL_n(x,y)|)) \rfloor}}{k} \tag{3.6}$$

where *MAX()* means the maximum coefficient value, k is a constant and $LL_n(x, y)$ denotes a general coefficient in *LLn* sub-band. The mask is created for the coefficient of sub-bands, which are larger than a given threshold and ignoring all others (i.e. setting to zero). By applying an inverse DWT on the mask, the texture quality at a given threshold can be generated.

Figure 3.5: shows the effect of selecting the threshold. The visual quality of figure 3.5 (b) uses only 3% of the coefficients which is better than figure 3.4 (f) which uses 6% of the coefficients. It is noticed that that threshold base coefficient selection method outperforms the low frequency only selection for reconstruction as in section 3.1.5. Further experiments performed reveal that the optimum threshold is dependant on the details contained in the texture.

Analysis: It has been shown that EZW provides flexibility of implementation, better reconstruction and progressive encoding capability [103]. In this thesis the EZW algorithm is used as an initial coefficient prioritisation procedure, to prioritise texture image coefficients. The shortcoming of the technique used visually prioritizes availability of coefficients to show the seamless progressive texture synthesis capability on a surface (see chapter 4, 5, 6, and 7).



(a) (b)

Figure 3.5: Effect of global threshold. (a) original images, (b) reconstructed images with only 3% of information picked using EZW from all the bands and all three levels.

3.3 THREE BENCHMARK ALGORITHMS FOR TEXTURE SYNTHESIS

This section introduces and critically reviews three texture synthesis algorithms that are used as benchmarks to compare the performance of the novel algorithms presented in this thesis. Amongst these, two are for texture synthesis on planar surfaces and one is for texture synthesis of arbitrary surfaces.

3.3.1 BENCHMARK ALGORITHM 1: FAST, WAVELET TRANSFORM DOMAIN TEXTURE SYNTHESIS [5, 12, 39]

This algorithm [5, 39] is an extension to the '*Image Quilting*' based texture synthesis algorithm [33] originally proposed by Efros and Freeman (see following section A). The basic idea of the algorithm is to create a large output texture from a given sample texture image as follows:

A. Image quilting [33]

The Efros and Freeman's algorithm can be summarized as follows:

- Go through the image to be synthesised in raster scan order in steps of one block (minus the overlap). (See figure 3.6(a))

- For every location, search the input texture for a set of blocks that satisfy the overlap constraints (above and left) within some error tolerance. Randomly pick one such block. (See figure 3.6 (b) & (c).)
- Consider the overlap region/surface between the chosen block and the previously synthesised block. Find the minimum cost cutting path along this surface and make that the boundary between the two blocks. Paste the block onto the texture. (See figure 3.6(d).)
- Repeat the procedure.

The quality of match between the overlapping areas of two blocks is calculated in terms of the Sum of Squared Error (SSE, i.e. the L2 norm), as follows:

$$\text{SSE} = \sum_{p \in O} [I_1(p) - I_2(p)]^2 \qquad (3.7)$$

Where I_x(p) is the intensity value of the pixel p, O is the set representing all pixels belonging to the overlap area of overlapping blocks, I_1 and I_2.

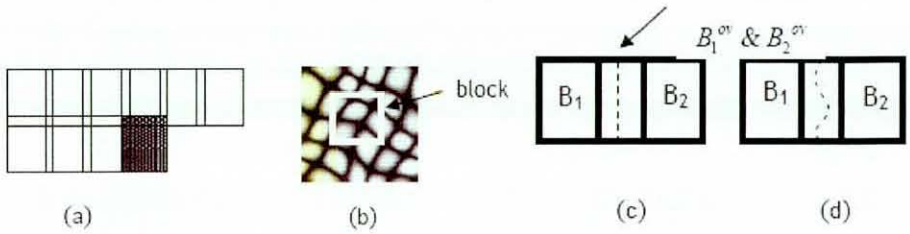


Figure 3.6: Pixel quilting algorithm- (a) pasting blocks in raster scan order (b) randomly picking blocks from the sample (c) overlapping randomly picked block, *B2* with already pasted block, *B1*. (d) Cutting through minimum error boundary.

The algorithm uses a 64 x 64 block size, and an overlap of 1/6th of the width of the block. The block having the best matching overlap is selected from a randomly picked up block from all the other blocks whose matching error is within 10%. This block will be used to patch the output image at the location to the right of the previously patched block. This process is continued until the whole output image is formed. In selecting non-boundary type blocks (and the last column of blocks), the overlap considered includes both the overlap with the block in front (as discussed above) and the block above. The output image formed following the above procedure is then subjected to a second stage in which each overlapping set of blocks is combined

together along a line of best fit, i.e. by performing a minimum error boundary cut (MEBC).

According to Wickramanayake et al. [12] the above algorithm cannot be used in a real-time texture synthesis process due to its complexity. An exhaustive search for a best match causes computational power to be wasted. A minimum boundary cutting technique which is used in order to smoothen sudden changes in overlapping texture is computationally expensive. In order to resolve these problems they introduced a multi-resolution framework for the image quilting, discussed in the following section.

B. Multiresolution Image Quilting Algorithm

Wickramanayake et al.'s multi-resolution approach [12] starts by applying a 2D DWT (e.g. Haar Transform) to the sample image. For example a three level decomposition of the sample and output texture consists of ten sub-bands each (see figure 3.3). The algorithm proposed is to synthesise each sub-band of the output texture by the corresponding sub-band of the sample texture. The block size used for texture synthesis is maintained constant at $2^{5-l} \times 2^{5-l}$, where l is the number of decomposition levels.

Initially the lowest resolution sub-band is synthesised. The synthesis starts with randomly picking a block from the lowest resolution sub-band of the decomposed input image, and placing it on the top left corner of the output coefficient image. Subsequently blocks on other sub-bands, which correspond in location to the above randomly picked block, are transferred to the top left hand corners of the associated sub-bands (figure 3.7 (a) & (b)).

Once the first block has been randomly selected and transferred to the output representation as discussed above, all possible blocks, of similar size from the input sample image's components are picked and matched, for a good overlap with the first block. The matching procedure can be described as follows:

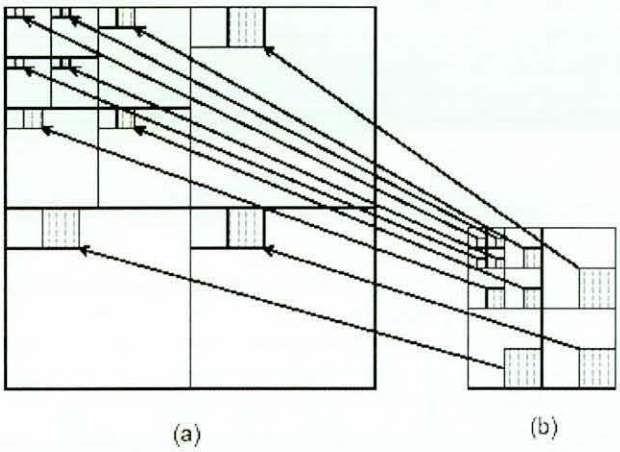


Figure 3.7: Construction of the output texture. (a) First random block ( ) and its best match ( ), (second block) is placed on the top left hand corner of the output texture together with the corresponding coefficient blocks of the detailed components in all three levels. (b) selection of the best match for the first block with its corresponding coefficient blocks from the input texture.

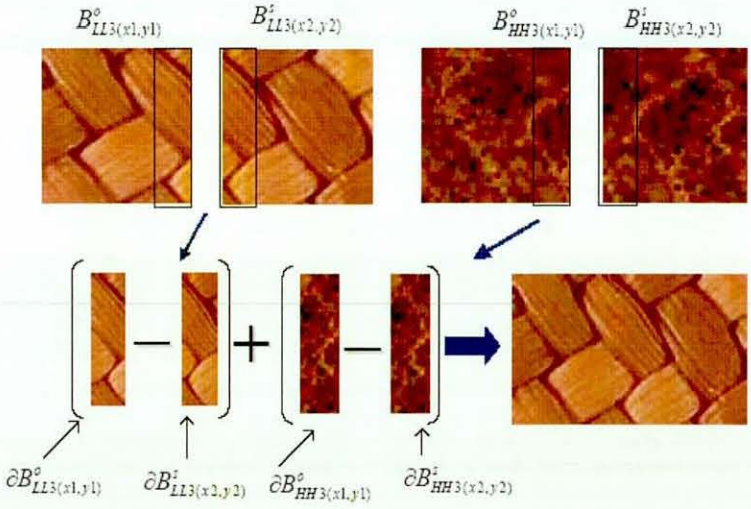


Figure 3.8: The matching criteria [Note the notation used: *LL3, HH3* -low resolution and diagonal sub-bands of the 3rd level of decomposition, respectively, ∂B is the *edge* zone of block B, B^o is a block already synthesized and B^s is the block being matched.]

In general, if $B^{o}_{(x1,\ y1)}$ and $B^{s}_{(x2,\ y2)}$ are two blocks to be matched, we say $B^{s}_{(x2,\ y2)}$ is the best match for $B^{o}_{(x1,\ y1)}$ if the mean squared matching error $d(B^{o}_{(x1,\ y1)},\ B^{s}_{(x2,\ y2)})$ is minimum for all possible blocks in the input sample where,

$$d(B^{o}{}_{(x1,y1)}, B^{s}{}_{(x2,y2)}) = \sum_{i \in \partial B} \{\partial B^{o}{}_{LL3(x1,y1)}(i) - \partial B^{s}{}_{LL3(x2,y2)}(i)\}^{2} \qquad (3.8)$$

Where ∂B^{O} is the edge zone of the block $B^{o}_{(x2,\ y2)}$ (figure 3.8) and i is an element (coefficient) within this edge zone.

In the proposed scheme the authors have computed the total matching error '*d*' by combining the matching errors of the blocks in its approximation sub-band, *LL3*, and one other sub-band out of the three detailed sub-bands, *LH3, HL3, HH3*. Figure 3.8 illustrates the case where *HH3* sub-band is used. In general, the decision to select a combination between the matching errors (as described above) in the horizontal detail, vertical detail, or diagonal detail sub-bands, is taken by comparing the energy levels of these components [12]. Wickramanayake et al. claims that for most textures the horizontal details contain more visually significant information than the vertical or diagonal detail. Using further higher frequency sub-bands is avoided, in checking the suitability of the candidate sample, as they may contain noise, which could be falsely interpreted as important texture detail.

In combining the two overlap errors, i.e. the overlap error between the blocks in the low-resolution component and the overlap error between the corresponding blocks in the selected detail components ('diagonal' in the illustration of figure 3.8); we use the sum of the squares errors (eq. 3.7) of the two components, as the matching criteria. For example, if the diagonal detail is prominent then the comparison equation (3.9) can be modified as below:

$$d(B^{o}{}_{(x1,y1)}, B^{s}{}_{(x2,y2)}) = \sum_{i \in \partial B} (\{\partial B^{o}{}_{LL3(x1,y1)}(i) - \partial B^{s}{}_{LL3(x2,y2)}(i)\}^{2} + \{\partial B^{o}{}_{HH3(x1,y1)}(i) - \partial B^{s}{}_{HH3(x2,y2)}(i)\}^{2}) \qquad (3.9)$$

Where ∂B_{pl} is an edge zone of block $B_{pl(x,y)}$ and i is an element (coefficient) within the edge zone.

If more than one sub-band is used the equation can be modified accordingly. Once the best matching block to the first block of the output image is found and located in the

output images, the corresponding coefficient blocks from all detailed images of the sample texture image are transferred to the appropriate components of the output texture image (figure 3.7 (a) & (b)). This process is continued until the whole output texture image is constructed in a three-level decomposed format. When constructing blocks other than those which belong to the first row and first column of blocks, two overlap areas, where the first corresponds to the overlap with the block in front and the second corresponds to the overlap with the block above, need to be considered. Finally an inverse DWT is performed on an output multi-resolution image to create the output texture image.

The above wavelet transform domain extension to Efros's algorithm proposed by Wickramanayake et al. was proved to work with most types of sample texture. However in a subsequent research effort [5] Wickramanayake et al. worked towards minimizing computational complexity and generating comparatively better quality of synthesised textures in their 'zerotree wavelet based image quilting' algorithm.

C. Zerotree Wavelet Based Image Quilting [5]

In this algorithm Wickramanayake et al. further extended and improved their previous work [12]. Instead of using all coefficients, an automatically generated threshold (see section 3.2) was used to determine the significant coefficients of each sub-band to define a matching template that is subsequently used in the texture quilting process. The work shows that the use of a limited set of visually significant coefficients, regardless of their level of resolution, not only reduces the computational cost, but also results in a more realistic texture synthesis.

Analysis of Wavelet based Texture Synthesis:

The transform domain extensions to the original image quilting approach to texture synthesis proposed by Wickramanayake et al. [5, 39] have various positive features such as ease of implementation, resulting generally in better synthesised quality as compared to Efros's [33] original algorithm and the ability to be used in real-time, due to the optimised transform domain search and the use of the multi-resolution

approach. This algorithm can cater for the needs of many additional application scenarios due to the added functionality that can handle computational and bandwidth constraints.

After analysing the synthesis results of the above algorithms [5, 12], we have realized that they may suffer from edge blending problems, when synthesizing certain texture patterns. Wickramanayake et al. [12] used a simple weighted edge blending approach to minimise computational complexity. However this simplistic blending approach compromises the quality of sharp edges in the patch overlap areas of the synthesised texture. In order to resolve this shortcoming, in chapter 5 we propose the replacement of this simple approach by an optimised, transform domain patch quilting algorithm.

After further analysing Wickramanayake et al.'s algorithms, we have realized that they can be extended to address texture synthesis on arbitrarily shaped 3D surfaces. However the design and implementation of this extension is not straightforward. It is noted that Wickramanayake et al.'s algorithm first builds the synthesised texture in a multi-resolution format in the DWT domain and the final synthesised texture in the pixel domain is obtained by an inverse DWT process. However, creating a multiresolution decomposition of an arbitrarily shaped 3D surface is challenging. A further challenge is the use of fixed size, square shaped patches in the texture synthesis process, such as either, 32 x 32, 64 x 64, or 128 x 128. However, in general 3D surfaces are represented by triangular or quadrilateral meshes which are of irregular shape and size. Hence, extending Wickramanayake et al.'s algorithm, which assumes the blocks to be of a fixed, shape and size, to 3D texture synthesis is infeasible. In chapters 6 and 7 we propose the modification and adaptation of Wickramanayake et al.'s original DWT domain 2D texture synthesis ideas to 3D texture synthesis.

3.3.2 BENCHMARK ALGORITHM 2: TEXTURE SYNTHESIS USING GRAPH CUTS [36]

As mentioned earlier in chapter 2, this is the second algorithm (based on *image quilting*) that is used in this thesis for benchmarking the novel texture synthesis

algorithms to be proposed. However, instead of dynamic programming, a new patch cutting approach is introduced by Kwatra et al. [36]. The initial patch stitching method used is similar to the patch selection method of the *Image Quilting* (section 3.3.1.A) approach. The texture is synthesised by copying irregularly shaped patches from the sample texture image into the output image. The output image is generated by copying patches which must make two decisions for each patch: (1) where to position the input texture relative to output texture (the offset of patch), and (2) to compute an optimal (irregularly shaped) portion of the rectangle, and then only those pixels are copied to the output image (figure 3.9). [Note that all the figures included in this section are from Kwatra et al.'s [36] paper]

The above process is repeated iteratively to improve the quality of the synthesised texture.

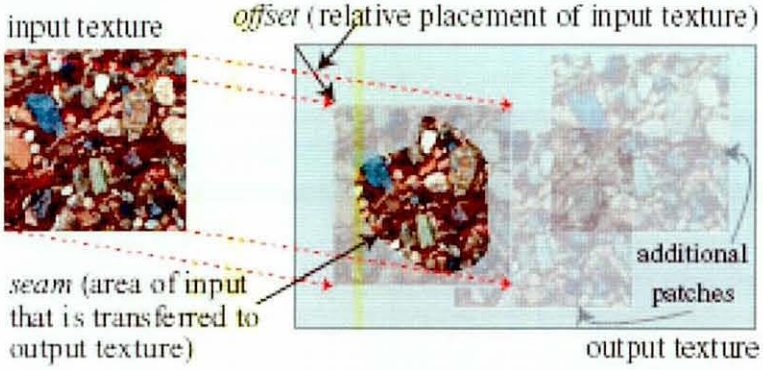


Figure 3.9: Image texture synthesis by placing small patches at various offsets followed by the computation of a seam that enforces visual smoothness between the existing pixels and the newly placed patch.

A. Patch Fitting using Graph Cuts

The texture synthesis process is based on copying an irregular shape of texture patches from an input image to an output image. This process is performed in two stages: (1) select the candidate patch by comparing the pixels that are already in the output image, (2) select the optimal portion of the rectangular patch to be copied into the output image. The process for optimal patch calculation is carried out by following a graph cut algorithm as explained below.

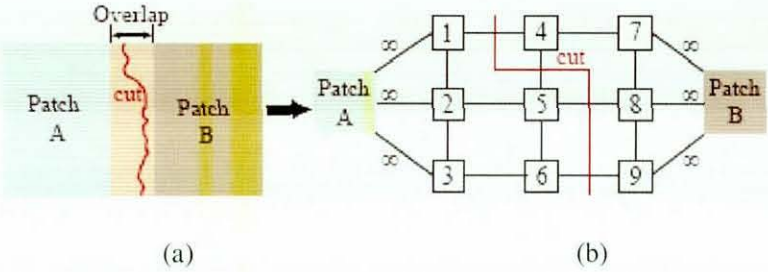
Referring to section 3.3.1.A, the first block (e.g., 32 x 32 pixels) is placed randomly and the subsequent block is placed partly overlapping with the previously placed block of pixels. The overlap between the old and the new block is typically 4 to 8 pixels in width. The path is formulated by dynamic programming, which will determine which patch pixels will contribute towards the overlapping region.

To cast this problem as a graph cut problem, a matching quality measure for a pixel between a new and an old patch is required. The simplest quality measure is a measure of the colour difference between pairs of pixels. Let s and t be two neighbouring pixel positions in the overlapping region. Also assume at the position s, $A(s)$ and $B(s)$ to be the pixel colours in the old and new patches respectively. The matching quality cost M between the adjacent pixels s and t is calculated as,

$$M(s, t, A, B) = \| A(s) - B(s) \| + \| A(t) - B(t) \| \qquad (3.10)$$

Where $\| \cdot \|$ denotes an appropriate norm. This matching cost is used in the graphcut algorithm to solve the patch finding problem.

Consider the graph shown in figure 3.10 (b) consisting of one node per pixel in the overlap region. The arcs connecting any two adjacent pixel nodes s and t are labelled with the matching quality cost $M(s, t, A, B)$ (eq. 3.10). The nodes A and B represent old and new patches. Finally an infinitely high cost arc is added between some pixels spanning between nodes A or B. These are constrained arcs which insist that the pixels to be selected come from a particular patch. In figure 3.10 (b) constrained pixels 1, 2, 3 are to come from the old patch, and 7, 8, 9 must come from the new patch. To find out from which patch each of the pixels 4, 5, and 6 should come from, a minimum cost cut of the graph is calculated by using a so-called max-flow/min-cut algorithm. In figure 3.10 the red line shows the minimum cut, and it means that pixel 4 will be copied from patch B, whereas pixels 5, and 6 will come from the old patch, A.



(a) (b)

Figure 3.10: (a) Schematic showing the overlapping region between two patches. (b) Graph formulation of the seam finding problem, with the red line showing the minimum cost cut [36].

B. Surrounded Regions

The above algorithm can be used further to place a new patch over a region where the entire area has already been covered by pixels from the earlier patch placement step. This is done to overwrite potentially visible seams with the new patch, and an example of this is shown in figure 3.11. In this example the red line shows that the resulting graph cut forms a closed loop, which defines the best irregular shape to be copied into the output image.

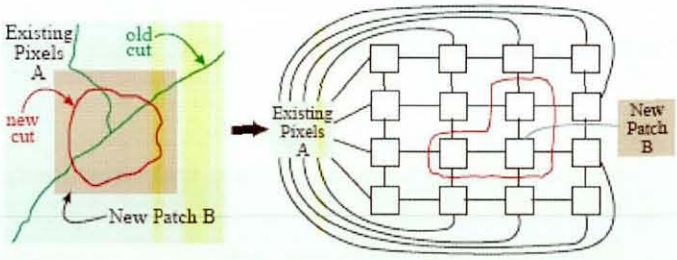


Figure 3.11: (Left) Placing a patch surrounded by already *filled* pixels. Old seams (green) are partially overwritten by the new patch (bordered in red). (Right) Graph formulation of the problem. Constraint arcs to **A** force the border pixels to come from the old image. Seam nodes and their arcs are not shown in this image for clarity [36].

C. Patch Placement & Matching

Three algorithms are used for selecting candidate patches based on the type of texture being synthesised. In all these algorithms, the patch selection is restricted to

previously unused offsets. The following are the three patch selection algorithms used;

Random placement:

In this approach, the new patch, (the entire input image), is translated to a random offset location. The graph cut algorithm selects a region of this patch to place into the output image, and then this process is repeated. This is a direct application of Guo & Shum's [31] algorithm with graphcut based edge cutting. This method provides better results for only a few stochastic textures. Therefore only the following two patch placement methods are used in the general algorithm.

Entire patch matching:

This step, called the initialization step, starts by placing the sample at the top left hand corner of the output representation. This involves searching for translations of the input image that match well with the currently synthesised output. To account for partial overlaps between the input and the output, they normalize the sum-of-squared differences (SSD) cost with the area of the overlapping region. After calculating the cost of translation t of the input $C(t)$ of all possible offsets, the following probability distribution function (eq. 3.11) is used for selecting the best patch amongst them:

$$P(t) \propto e^{-\frac{c(t)}{k\sigma^2}}. \qquad (3.11)$$

Where σ is the standard deviation of pixel values in the input sample and k controls the randomness parameter in the patch selection.

Sub-patch matching:

This is the step after the entire patch matching. A small sub-patch (which is smaller than the input texture) in the output texture is initially picked. The selection is based on the seam cost in the previous steps (figure 3.12 bottom left). Subsequently a search is made for a sub-patch in the input texture that matches this output-sub-patch well. Only those translations that allow a complete overlap of the input with the output-sub-patch are considered. The cost of a translation of the input texture is again calculated and a sub-patch is picked stochastically using a probability function similar to eqn. 3.11.

Analysis of Graphcut Algorithm:

With the initial straightforward implementation, the complexity of the graphcut textures of Kwatra et al.'s [36] algorithm was $O(n^2)$. But a FFT based acceleration technique [106, 107], which uses Summed Area Tables (SAT) and convolutions has been subsequently adopted for reducing the complexity to $O(n \log n)$. However the algorithm yet results in high synthesis time requirements. Due to the use of a whole sample patch as the unit of construction in the synthesis, there can be mismatching areas in the overlapped area leading to artefacts. Kwatra at el.'s algorithm uses several refinement passes in order to minimise these artefacts which is a time consuming process. In chapter 5 we introduce further improvements to Kwatra's original ideas on using graph-cuts for image quilting based texture synthesis.

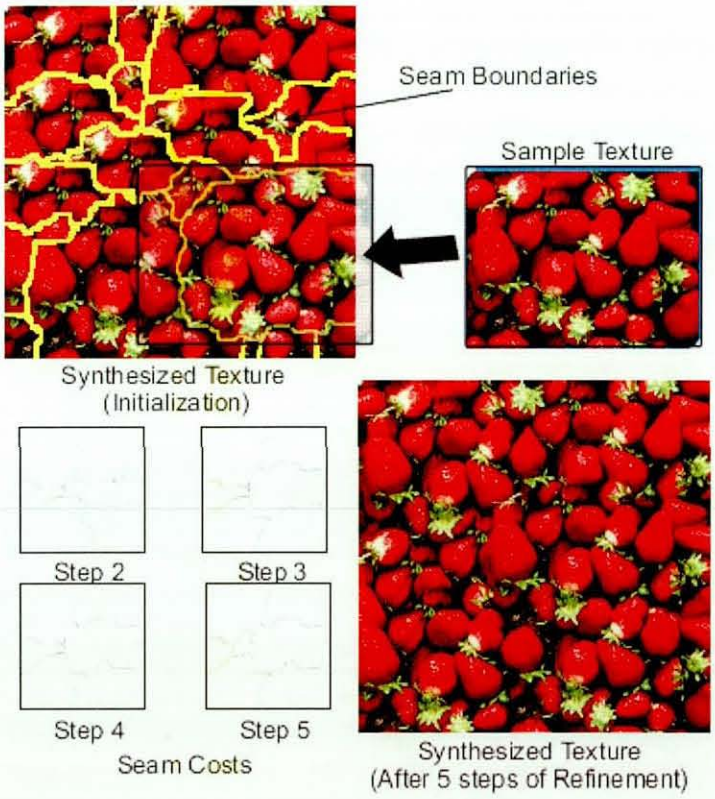


Figure 3.12: The process of synthesizing a larger texture from an example input texture. After initialisation, new patch locations are found depending on the seam costs within the refinement passes (bottom left).

3.3.3 BENCHMARK ALGORITHM 3: TEXTURE SYNTHESIS ON SURFACES [62]

This section discusses Turk's algorithm popularly titled as, *Texture Synthesis on Arbitrary Surfaces*. Figure 3.13 illustrates the basic block diagram of Turk's algorithm.

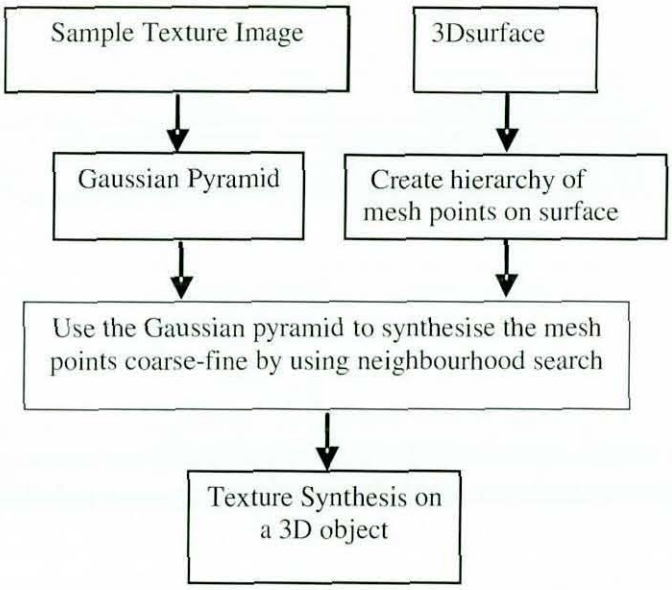


Figure 3.13: Turk's texture synthesis algorithm

This algorithm is based on the principles of texture synthesis on surfaces, independent of parameterization and the use of the neighbourhood search. The method draws upon texture synthesis methods that use image pyramids [49] for texture decomposition/representation, and a mesh hierarchy to serve in place of a pyramidal structure. Firstly a hierarchy of sample points from a low to high density are created in a random order, over the given surface (figure 3.14 (a)). Using a point repulsion method [52], the points are subsequently repulsed and separated from each other to uniformly distribute over the surface. By connecting these points, a mesh is formed on the surface. Likewise a hierarchy of meshes are created on the surface by adding additional mesh points in stages, resembling an image pyramid representation. Subsequently a user specified vector field that indicates the orientation of texture patterns is created over the surface. Mesh vertices are sorted in such a manner that the

vector field will be followed when visiting the points (see figure 3.14 (b)). Then for each mesh vertex, a local parameterization of the surrounding vertices is established. Using this parameterization a small rectangular neighbourhood with the vertex as the centre is created (see figure 3.15). Each point is then scanned over the surface to determine the colour. The colour of a particular point is established by examining the colour of neighbouring points, and finding the best match to a similar pixel neighbourhood in the given sample texture image. A coarse-to-fine refinement is done on the mesh hierarchy to retain the pattern and to achieve good quality of texture on the surface. A multi-level representation of the synthesised texture on the arbitrary surface is finally generated as shown in figures, 3.14 (c), (d), (e), and (f).

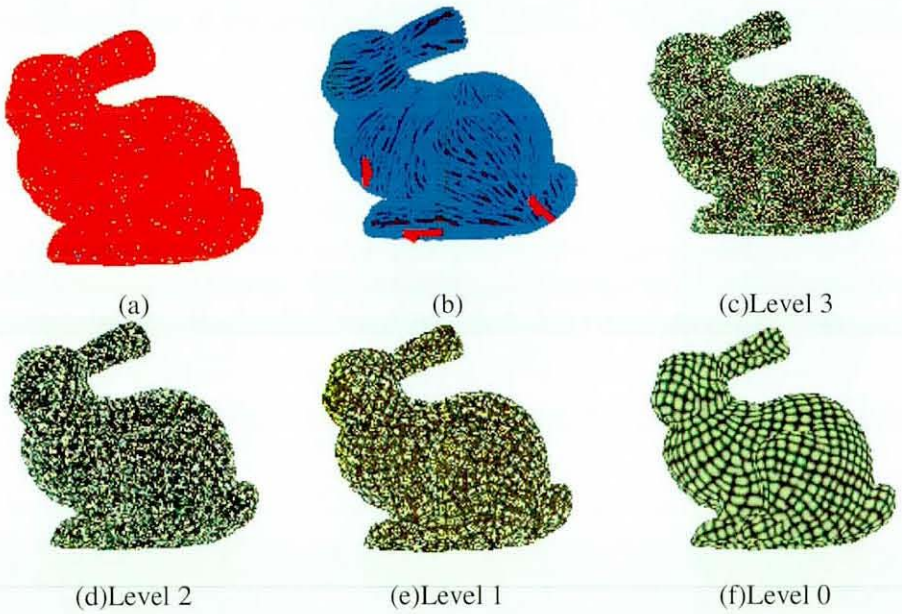


Figure 3.14: Turk's Method (a): dense points on surface (b): vector field creation on the surface (c) (d) (e) (f): multi-level texture synthesis

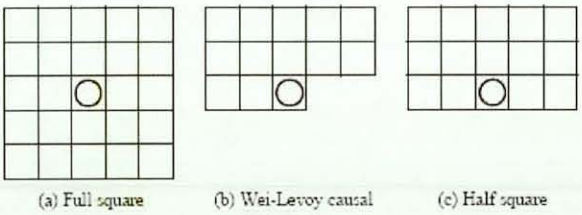


Figure 3.15: Three different pixel neighbourhood searches.

Pseudo code for the texture synthesis algorithm:

```
For each vertex v on mesh
    C(v) = colour of random pixel from I
For each vertex v on mesh (ordered by S (v))
    construct neighbourhood colours N(v)
    smallest match = BIG
    For each pixel (a, b) in I
        construct neighbourhood colours M (a ,b)
        new match = D (N(v), M(a ,b))
        If (new match < smallest match)
            smallest match = new match
            C_best = I (a, b)
    C(v) = C_best
```

Where $C(v)$ colour at vertex v, $N(v)$ neighbourhood colour at vertex v, $M(a,b)$ neighbourhood mask from I, I Input sample image, $S(v)$ visiting order of vertices over mesh.

Analysis of Turk's Algorithm:

This algorithm is a combination and extension of various techniques such as 1) neighbourhood matching technique, 2) TSVQ, 3) Interpolation for neighbourhood and 4) Gaussian based multiresolution representation of sample textures. Our detailed experiments on ten different samples of texture proved that the algorithm works well with irregular textures, confirming the conclusions made by Turk, in [62]. Further the algorithm suffers from practical applicability to work in a limited bandwidth channel due to the non-progressive nature of texture synthesis. In addition it cannot be used in real-time, in particular due to the iterative approach being time consuming. In chapter 4 we propose an improvement to this algorithm that removes some of the above shortcomings.

3.4 GENERATION AND REPRESENTATION OF GEOMETRIC SURFACES

As discussed in chapter 2, there are two ways to generate a 3D surface, 1) 3D Scanner based surface generation and 2) Geometric surface generation. This part of the chapter discusses geometric surface generation using polynomial parametric curves, which is a type of curve used in geometric modeling. These curves are useful for computational purposes. Thus their properties are discussed in detail. We further discuss the extension of such curves to Bézier surfaces and detail their properties.

3.4.1 Bézier Curves [108]

The first method of generating Bézier curves was developed by P. Bézier, where points on a curve were located via a weighted interpolation of control points, i.e., a limited set of specific points. Since Bézier's initial proposal several different ways of interpolating control points of a 2D Bézier curve have been introduced. Although these approaches differ algorithmically and in complexity, they generate the same set of points on a given parametric curve.

Figure 3.16 illustrates the calculation of the cubic Bézier curve for four control points.

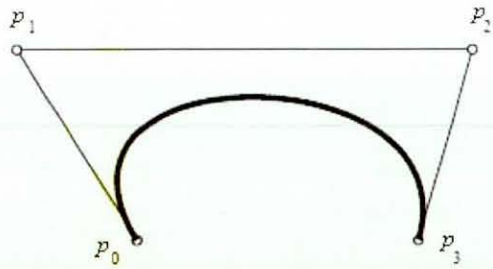


Figure 3.16: Bézier curve using four control points.

The contribution that each control point makes to a curve depends on a single parameter, s, with $0 \leq s \leq 1$. As s varies along the curve each control point's contribution varies as a function of s. In fact, each control point has a function which controls its influence; these are known as the Bernstein basis functions. Figure 3.17 illustrates how the function for the first control point has a value of 1 when $s = 0$, and decays to 0 as s approaches 1. This means that the first control point contributes 100%

to the curve when the curve is very close to it, and that when the curve is far away, it hardly contributes anything. The Bernstein basis function [127] cause Bézier curves to pass through the starting and ending control points.

The Bernstein basis functions for $n = 3$ are plotted in figure 3.17 and illustrate the control points' influence on Bézier curves:

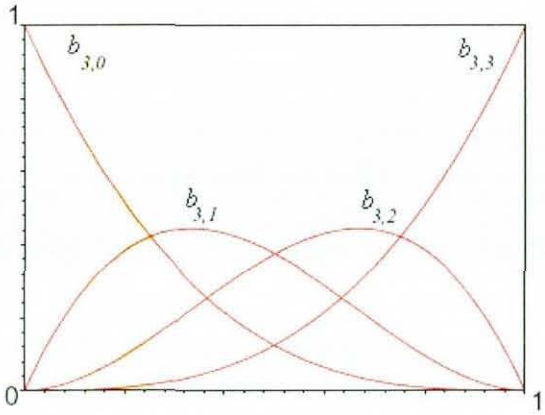


Figure 3.17: Bernstein basis functions for $n = 3$

Given n+1 control points, P_0, P_1, P_2, ... and P_n in space, the Bézier curve C (s) is defined by,

$$C(s) = \sum_{i=0}^{n} b_{n,i}(s) P_i \tag{3.12}$$

The $n + 1$ Bernstein basis polynomials of degree n are defined as

$$b_{n,i}(s) = \binom{n}{i} s^i \ (1-s)^{n-i} \tag{3.13}$$

Where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{3.14}$$

For a given value of s, a point C on a Bézier curve is defined as the sum of the four control points, weighted by the value of the basis functions for that value of s:

The following are important Bézier curve properties:

1. The degree of a Bézier curve defined by n+1 control points is n: In each basis function, the exponent of s is $i + (n - i) = n$. Therefore, the degree of the curve is n.
2. $C(s)$ passes through P_0 and P_n: This is shown in the figure 3.16 above. The curve passes through the first and the last control points.

3. Non-negativity: All basis functions are non-negative.
4. Partition of Unity: The sum of the basis functions at a fixed s is 1. It is not difficult to verify that the basis functions are the coefficients in the binomial expansion of the expression $1 = (s+ (1 - s))^n$. Hence, their sum is one. Moreover, since they are nonnegative, we conclude that the value of any basis function is in the range of 0 and 1.

3.4.2 Bézier Surfaces [109]

The Bézier patch (surface) is a surface extension of the Bézier curve. Whereas a curve is a function of one variable and takes a sequence of control points, the patch is a function of two variables with an array of control points. Most of the methods used for Bézier patch generation are direct extensions of those used for Bézier curves. The Bézier patch is the most commonly used surface representation in computer graphics. Patches are eventually rendered as polygons. However as the polygons are generated dynamically, faster machines end up with smoother models, the ultimate in scalability.

A Bézier patch is specified by a number of control points and a tessellation factor to determine smoothness (higher factor equals smoother surfaces). The patch is constructed from an $n+1 \times m+1$ array of control points $\{P_{k,j}; 0 \le k \le n,\ 0 \le j \le m\}$ (figure: 3.18).

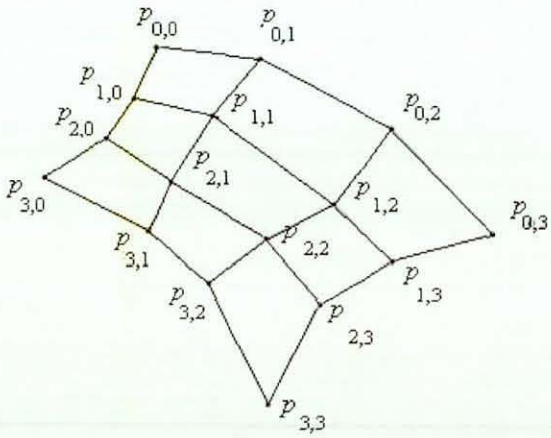


Figure 3.18 $n+1 \times m+1$ arrays of control points ($n=3$, $m = 3$)

The resulting surface (see figure 3.19), is subsequently parameterised by two variables using equation 3.15 as,

$$P(s,t) = \sum_{k,j=0}^{n,m} b_{n,k}(s) b_{m,j}(t) P_{k,j}, 0 \le s,t \le 1 \qquad (3.15)$$

Where $P_{k,j}$ are the Bézier vectors.

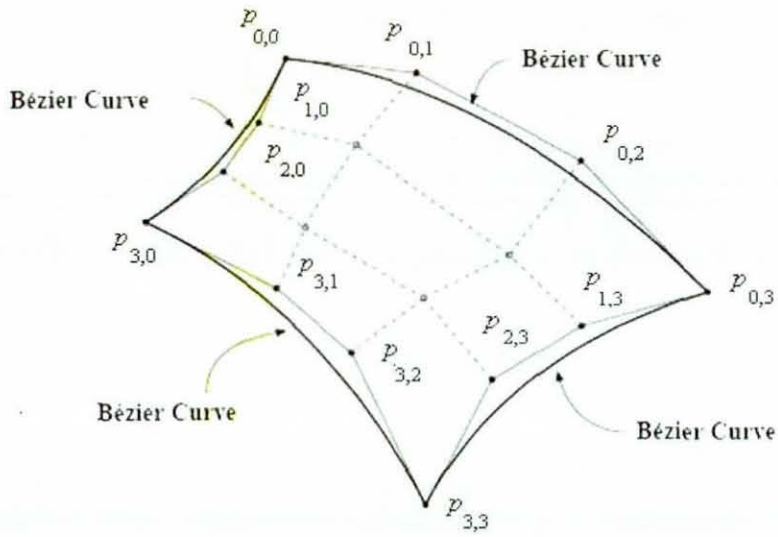


Figure 3.19: Resulting surface using $n+1 \times m+1$ array of control points

It is easily seen that this is in the same general form as the Bézier Curve – with the summations running over all the control points, and the bi-variate Bernstein polynomials serving as the functions that blend the control points together. Note that the uni-variate Bernstein polynomials are $b_{n,k}(t), b_{m,j}(s)$.

Properties of a Bézier Patch

A Bézier patch has properties similar to those of a Bézier curve. These can be verified directly from the patch defining equations.

1) If control points P_{00}, P_{0m}, P_{n0} and P_{nm} are on a patch, all other control points are on the patch only if the patch is planar.
2) The patch is continuous and partial derivatives of all orders exist and are continuous.
3) The patch lies within the convex hull of its control points.

3.4.3 Rational Bézier Surfaces

A homogeneous form of a parametric surface is referred to as a rational surface. The rational parameterisation is a de-facto standard representation in computer graphics allowing portability across applications and systems. In addition to possessing desirable geometric properties, rational parameterisation:

- requires the evaluation of only polynomial functions,
- gives rise to a compact data-structure,
- facilitates interactive control of shape,
- is complete in the sense that approximation of any shape to a specified tolerance δ can be achieved, and exact parametrisation (i.e. δ= 0) is often possible.

The bi-quadratic and bi-cubic type of patches depends on the number of control points that are in use (9 points and 16 points, respectively). While higher order Bézier curves are possible, they are not feasible for use in applications such as computer games due to their computational complexity.

Rational parametrisations of surfaces comprise local atlases, or patches, of the form:

$$\tau(s,t) = \frac{\sum_{k,j=0}^{n.m} b_{n,k}(s) b_{m,j}(t) v_{k,j}}{\sum_{k,j=0}^{n.m} b_{n,k}(s) b_{m,j}(t) \omega_{k,j}}, 0 \leq s,t \leq 1 \qquad (3.16)$$

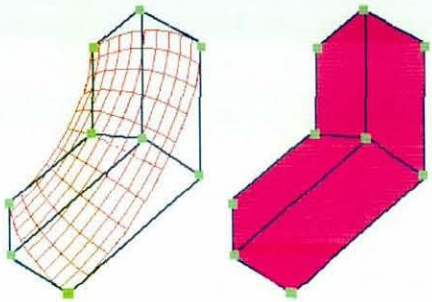
where $\omega_{k,j}$ are the weights defining the specific shape of the smooth surface, $v_{k,j}$ are the Bernstein vectors, $b_{m,j}(s)$, $b_{n,k}(t)$ are Bernstein polynomials. If all the weights are non-zero this may be expressed as:

$$\tau(s,t) = \frac{\sum_{k,j=0}^{n.m} b_{n,k}(s) b_{m,j}(t) \omega_{k,j} v^{*}_{k,j}}{\sum_{k,j=0}^{n.m} b_{n,k}(s) b_{m,j}(t) \omega_{k,j}}, 0 \leq s,t \leq 1 \qquad (3.17)$$

where $v^{*}_{k,j} = \frac{v_{k,j}}{\omega_{k,j}}$ are the Rational Bézier vertices.

We have seen that the values of n and m determine the degree of the parametrisation; if $n = m = 2$ the patch is said to be biquadratic and if $n = m = 3$ it is

bi-cubic. In figure 3.20 we illustrate the case in which, given nine control points we compute and draw the biquadratic surface patch defined by them.



(a) (b)

Figure3.20: (a) Green Colour: Biquadratic control polygon point, Red Colour: Smooth surface mesh generated using the control polygon (b) Control polygon mesh (note that 9 control points generate 4 faces).

Many of the desirable geometric properties of rational representation, e.g. the convex hull property and the existence of Bézier vertices, are lost if negative or zero weights occur, hence in computer graphics and geometric modelling applications, positive weight parametrisations are always preferred. For computational efficiency, low degree parametrisations are desirable.

3.4.4 An Example: The parametrisation of Ring Dupin Cyclides by trigonometric functions

Dupin cyclides may be defined implicitly as:

$$(x^2 + y^2 + z^2 - \mu^2 + b^2)^2 - 4(ax - c\mu)^2 - 4b^2 y^2 = 0 \qquad (3.18)$$

where the parameters a, b and c satisfy $c^2 = a^2 - b^2$. The surfaces for which $c < \mu \leq a$ have found applications in geometric modelling (see [114] and [115]), and are known as ringed cyclides. Ringed cyclides with $c = 0$ are tori, and the ringed cyclides may therefore be regarded as generalised tori. Figure 3.21 (a) shows a ringed cyclide for which $a = 6$, $b = 4\sqrt{2}$, ($c = 2$) and $\mu = 3$. Cyclides admit trigonometric parametrisations which may be written as:

$$\tau_\mu(\theta,\varphi) = \frac{1}{a - c\cos\theta\cos\varphi}\left[\mu(c - a\cos\theta\cos\varphi) + b^2\cos\theta, b(a - \mu\cos\varphi)\sin\theta, b(c\cos\theta - \mu)\sin\varphi\right] \qquad (3.19)$$

for $0 < \theta < 2\pi$, $0 < \phi < 2\pi$. Figure 3.21 (b) shows the origins and directions of the angular parameters, θ and ϕ, on the surface.

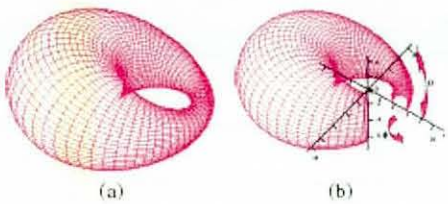


Figure 3.21: (a): The surface, (b): The trigonometric parametrisation τ_μ of the surface

A. The rational parametrisation induced from τ_μ [112]

This section will briefly introduce biquadratic parametrisation of a Dupin cyclide.

The restriction of the trigonometric parametrisation τ_μ to the boundary is defined by,

$$\theta_0 \le \theta \le \theta_1 \qquad \varphi_0 \le \varphi \le \varphi_1$$

is written as $\tau_{\mu.\theta}$ and may be used to 'induce' rational parametrisations of the same surface patch as that parametrised by $\tau_{\mu.\theta}$. In particular it may be used to induce bi-quadratic rational Bézier representations (see [111] and [110]). The induction process discussed in [110] produces a rational bi-quadratic patch, that parametrises the same region of the surface as $\tau_{\mu.\theta}$, with weights, w_{ij}, $0 \le i, j \le 2$, given by:

$$\begin{aligned}
w_{00} &= a - cx_{0\theta}x_{0\varphi} & w_{01} &= aw_{1\phi} - cx_{0\theta}x_{1\varphi} & w_{02} &= a - cx_{0\theta}x_{2\varphi} \\
w_{10} &= aw_{1\theta} - cx_{1\theta}x_{0\varphi} & w_{11} &= aw_{1\theta}w_{1\phi} - cx_{1\theta}x_{1\varphi} & w_{12} &= aw_{1\theta} - cx_{1\theta}x_{2\varphi} \\
w_{20} &= a - cx_{2\theta}x_{0\varphi} & w_{21} &= aw_{1\phi} - cx_{2\theta}x_{1\varphi} & w_{22} &= a - cx_{2\theta}x_{2\varphi}
\end{aligned} \tag{3.20}$$

Where

$$\begin{aligned}
w_{1\theta} &= \cos(\theta_1 - \theta_0)/2 & w_{1\phi} &= \cos(\phi_1 - \phi_0)/2 \\
x_{0\theta} &= \cos(\theta_0) & x_{0\phi} &= \cos(\phi_0) \\
x_{1\theta} &= \cos(\theta_1 + \theta_0)/2 & x_{1\varphi} &= \cos(\phi_1 + \phi_0)/2
\end{aligned}$$

$$x_{2\theta} = \cos(\theta_1) \qquad x_{2\phi} = \cos(\phi_1)$$

$$y_{0\theta} = \sin(\theta_0) \qquad z_{0\phi} = \sin(\phi_0)$$

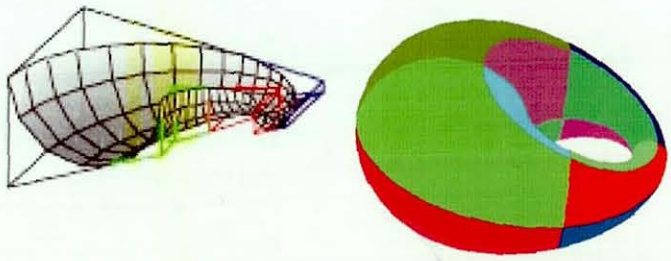
$$y_{1\theta} = \sin(\theta_1 + \theta_0)/2 \qquad z_{1\phi} = \sin(\phi_1 + \phi_0)/2$$

$$y_{2\theta} = \sin(\theta_1) \qquad z_{2\phi} = \sin(\phi_1)$$

Using the formulae for the weights (eq. 3.20), in [128] it has been proved that for any cyclide (i.e., for any a, b, (c > 0), μ), there exist 16, positive-weight, bi-quadratic Bézier patches that parametrise the entire surface (see figure 3.22 and [112]). Further, the 4-way symmetry of the cyclide can be exploited, enabling 12 of the 16 patches to be determined, by vertex transformations, from the 4 patches defined by the following angular displacements:

- $0 \le \phi \le \frac{\pi}{2}, \quad 0 \le \theta \le \frac{\pi}{2}$
- $\frac{\pi}{2} \le \phi \le \pi, \quad 0 \le \theta \le \frac{\pi}{2}$
- $0 \le \phi \le \frac{\pi}{2}, \quad \frac{\pi}{2} \le \theta \le \pi$
- $\frac{\pi}{2} \le \phi \le \pi, \quad \frac{\pi}{2} \le \theta \le \pi$

These 4 patches are shown, with their Bézier polygons, in Figure 3.22. These patches parametrise ¼ of the surface. Their weights and the associated computations are given in appendix A of the thesis. As 0 < c < a, it is clear that all the patches have positive weights. The Bézier vertices of the four patches are also given in the appendix A of the thesis.



(a) (b)

Figure 3.22: (a) ¼ cyclide comprising 4 positive weight quadratic rational Bézier patches (b) 16 patch NURBS representation obtained from ¼ of a patch.

A 16 Patch cyclide and a NURBS representation which is generated from the ¼ patch equation is used in our experiments. Note that different size patches can be created for different models. In general the proposed algorithm supports any number of patches. Similarly it can be designed for geometric shapes such as a sphere or a torus.

Analysis of Geometric Surface Generation Schemes:

The control points in space roughly indicate the shape of a desired surface. If the surface is not up to our expectation, we can move the control points around. As one or more control points are moved, the shape of the Bézier surface changes accordingly. Texturing such a geometrically flexible surface representation gives further realism to it as seen in chapter 1. Such geometric surfaces have wide applications in computer animations, digital movies, and architecture.

It is noted that the surface mesh representation depicting an object's shape, and texture information depicting an object's appearance, are both in uncompressed format. Therefore a full representation (i.e. a textured 3D object) will require excessive bandwidth for transmission and/or storage space. A solution to this problem is to use a geometric surface representation where possible and to carry out progressive texture synthesis in the transform domain. In chapter 6 and 7, we propose a novel algorithm to progressively synthesise texture onto an arbitrarily shaped geometric surface.

3.5 SUMMARY AND CONCLUSION

This chapter has introduced the basic concepts, which will be fundamental to the understanding of the novel algorithms proposed in the rest of this thesis. Fundamentals of DWT and the possible use of EZW concepts in wavelet coefficient prioritisation have been explained. The chapter also introduced the three state-of-the art texture synthesis algorithms that have been selected in this thesis for benchmarking the performance of the novel algorithms to be proposed. Among these, two are planar texture synthesis algorithms and one is a 3D surface texture synthesis

algorithm. During the course of the research presented in this thesis, each of the benchmark algorithms was implemented and critically analysed in order to identify possible shortcomings. The novel algorithms presented in this thesis are improvements and extensions to these benchmark algorithms and are introduced in chapters 4, 5, 6 and 7.

The final section of this chapter further discussed the generation and representation of geometric surface, where fundamental theory of Bézier curves and surfaces was presented. Rational parametrisation of Bézier surfaces was finally discussed and possible applications to the proposed novel texture synthesis algorithms have been hinted at.

The analysis of the state-of-the art of texture synthesis algorithms revealed that all algorithms have limitations with regards to their capability to synthesise a wide range of texture patterns, and their ability to synthesise texture in real-time using a flexible, progressive approach. Thus it was observed that regardless of the vast amount of research carried out in this area, designing a fast, efficient and flexible texture synthesis algorithm still remains an open research problem.

Chapter 4

Progressive Texture Synthesis on Arbitrary Surface

4.0 INTRODUCTION

This chapter discusses in detail a novel algorithm that can be used in the creation of progressively better quality texture on an arbitrary shaped surface of a 3D object. We discuss that the algorithm has applications in video, movie, gaming and remote visualization applications [see section 4.3]. It is an extension to the pixel domain texture synthesis work of Turk et al. [62], *'Texture Synthesis on Surfaces'*, the fundamentals and principles of which were discussed in detail in chapter 3, section 3.3.3. The proposed extension addresses some shortcomings of the original algorithm by Turk, namely, its limited ability to only generate a discrete number of resolution levels of texture on a surface, generation of synthesis artefacts, particularly at lower resolution levels and difficulty of being used in conjunction with bandwidth constrained transmission media. The proposed approach meets the above challenges by adopting a multi-resolution DWT domain, progressive texture synthesis procedure. We show that the progressive nature of the proposed texture synthesis algorithm enables the creation of infinite levels of intermediate texture qualities between any two levels of quality supportable by Turk's algorithm.

For clarity of presentation, this chapter is divided into several sections. Section 4.1 discusses the motivation behind this work; Section 4.2 discusses the proposed algorithm, the benchmark algorithm and the underlying texture synthesis techniques used; section 4.3 provides experimental results and a comprehensive analysis of the performance of the proposed algorithm, whilst section 4.4 summarises and concludes the chapter.

4.1 MOTIVATION OF WORK

The motivation to the proposed work comes from the current demand for applications requiring progressive texture synthesis on arbitrary shaped 3D surfaces. The progressive texture synthesis provides a practical solution to the extensive use of bandwidth in application domains that are supported by bandwidth limited transmission media, such as remote visualisation, distributed/collaborative gaming etc. The idea is to provide texture synthesis quality *on-demand*. The use of existing texture synthesis algorithms on arbitrarily shaped 3D objects, such as the popular Turk's method (see section 3.3.3 for details) though attempting to provide a partial solution to this problem is limited. This is due to the support of only a discrete number of resolution levels that are determined by the distinct levels of the Gaussian pyramidal decomposition of the sample texture. The intermediate quality generated in the multi-level resolution is not visually appealing [see figure 4.4], though it retains the pattern of texture. Therefore a texture synthesis approach that allows the creation of an arbitrarily large number of texture resolutions between textures defined by two given distinct levels of multi resolution decompositions should be sought.

4.2 PROPOSED APPROACH

To overcome the above mentioned limitation and to further improve Turk's method's practical applicability, in this chapter we propose an extension that replaces the pyramidal decomposition of a sample texture, by DWT decomposition.

Figure 4.1 illustrates the block diagram of the proposed algorithm. [Note: the complete, progressive texture coding, transmission and synthesis process has been illustrated.] The key additional steps used in the proposed approach as compared to those used by Turk's approach can be listed as follows (discussed in detail in chapter 3):

1) DWT to provide a compact multi-resolution representation of the sample texture.
2) EZW algorithm to prioritize the wavelet coefficients according to their perceptual significance.

3) Architecture to support the integration of the proposed novel aspects into Turk's texture synthesis algorithm.

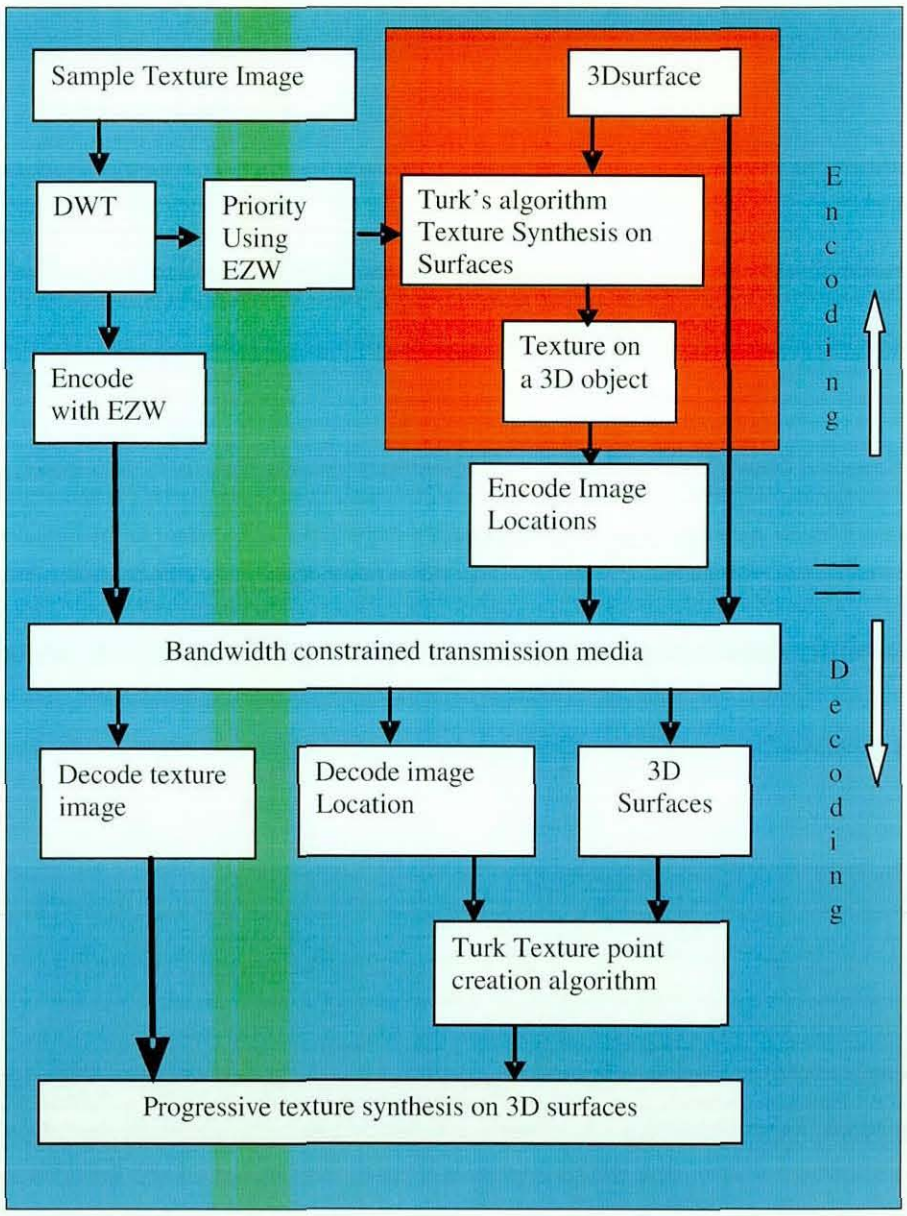


Figure 4.1: proposed progressive texture synthesis and transmission algorithm [Note: sky blue colour represents elements of the proposed algorithm and orange colour is Turk's algorithms].

A close inspection of figure 4.1 shows that it resembles the block diagram of Turk's algorithm [62], where the process of synthesizing texture is similar, though with two important differences. Firstly, in our design, the Gaussian pyramid based texture representation used by Turk is replaced by a DWT representation supported by an EZW based coefficient prioritisation scheme as illustrated by the modules named, 'DWT' and 'Priority Using EZW', in figure 4.1 and detailed in figure 4.2. EZW provides the facility to prioritise the visually significant wavelet coefficients. This ensures the seamless texture representation capability as against the discrete quality level texture representation of Turk's original algorithm. We have used the same multi-level synthesis approach while synthesizing texture on to a surface which produces the same quality results as Turk's algorithm. In addition to this texture encoding via EZW enables embedded texture decoding capability, allowing almost any intermediate texture quality to be readily reconstructed/ made available at the receiver depending on bandwidth constraints.

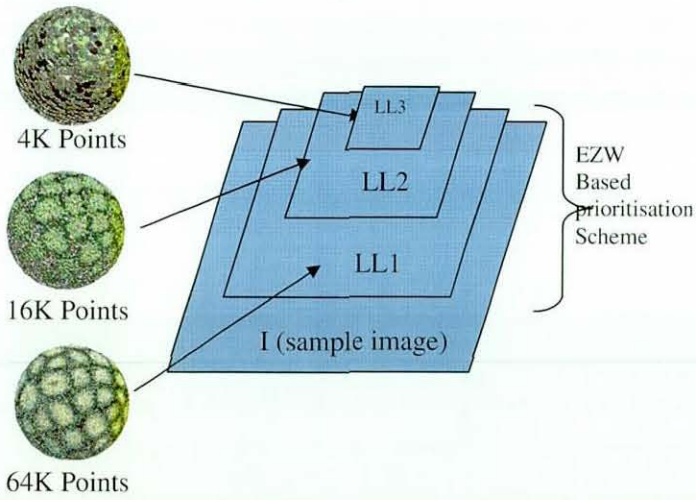


Figure 4.2: DWT and EZW based prioritisation scheme

Secondly in the proposed method, while synthesising texture using the modified (with EZW) Turk's method, we maintain a record of sample image locations from where each 3D surface point is textured. However this incurs additional memory cost. These are denoted as *'point image locations'* (see figure 4.3) see module *'Encode Image Location'*. These records are maintained for each representation of a 3D surface

shape. In order to avoid any searching at the decoder end and enable direct texture mapping from the received sample image, we propose the suitable encoding of the 'point image locations' and their transmission. We have adopted differential pulse code modulation which provides an effective means of coding this high redundancy information. However if one wishes texture synthesis to be carried out at the decoder as well, this extra information will be redundant and thus need not to be transmitted. We recommended that the decision on whether or not the 'point image locations' need to be transmitted be taken depending on whether the bottle-neck is in the transmission bandwidth or the receiver capability and texture quality required on the surface.

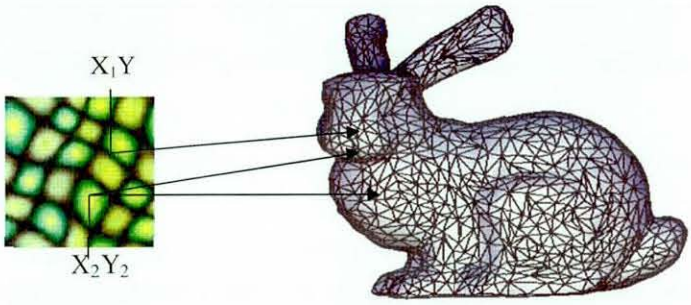


Figure 4.3: Single and multiple mappings of 'point image locations' between the sample texture and 3D surface

At the receiver, all texture sample image information transmitted using EZW encoding is decoded to generate the texture sample at seamless quality settings. As more bits are received by the decoder, better quality textures are produced. At any given instance of time and bit rate, using the re-generated texture sample image (i.e. via EZW decoding of texture), and separately received (and decoded) point image locations and the recovered 3D surface shape, the texture pixel value (colour) of each 3D point can be obtained. When the pixel values of all 3D points are obtained, the 3D object will be completely textured with the corresponding instantaneous texture quality at the given 3D surface representation level, i.e. number of vertex points accuracy to which the 3D surface is defined at the given moment. When more bits representing the texture information are received, the available texture quality at the decoder improves and hence the accuracy of texture representation on the 3D surface can be improved. Thus the process ensures a seamless quality texture synthesis

capability on 3D surfaces.

4.3 EXPERIMENTAL RESULTS & ANALYSIS

In order to analyse the performance of the proposed algorithm experiments were performed on a widely used set of texture samples and the 3D object bunny [62]. The proposed algorithm is implemented and tested using C, C++, openGL, Linux. Figure 4.4 shows the results obtained from Turk's algorithm, whereas figures 4.5, 4.6, and 4.7 illustrate the results obtained from our work which indicate the ability of the proposed technique to synthesise texture at seamlessly different levels. The results obtained are comparable. Our approach generates seamless texture on the surface (see figure 4.5(a)-4.5(h)) as compared to figure 4.4. Although for practical reasons we have limited the number of texture synthesis examples to eight, the proposed method is capable of progressive texture transmission aimed at seamless texture mapping. Note that our demonstration here is for a set of 256,000 (256K) points on the surface of the Bunny. In general, the proposed algorithm is capable of providing any intermediate texture level from texture-less to the quality of texture depicted in figure-4.4 for 4K, 16K, 64K, 256K point accuracy of the surface representations.

Texture Sample 4K 16K 64K 256K

Figure 4.4: Multi-resolution of Turk Algorithm

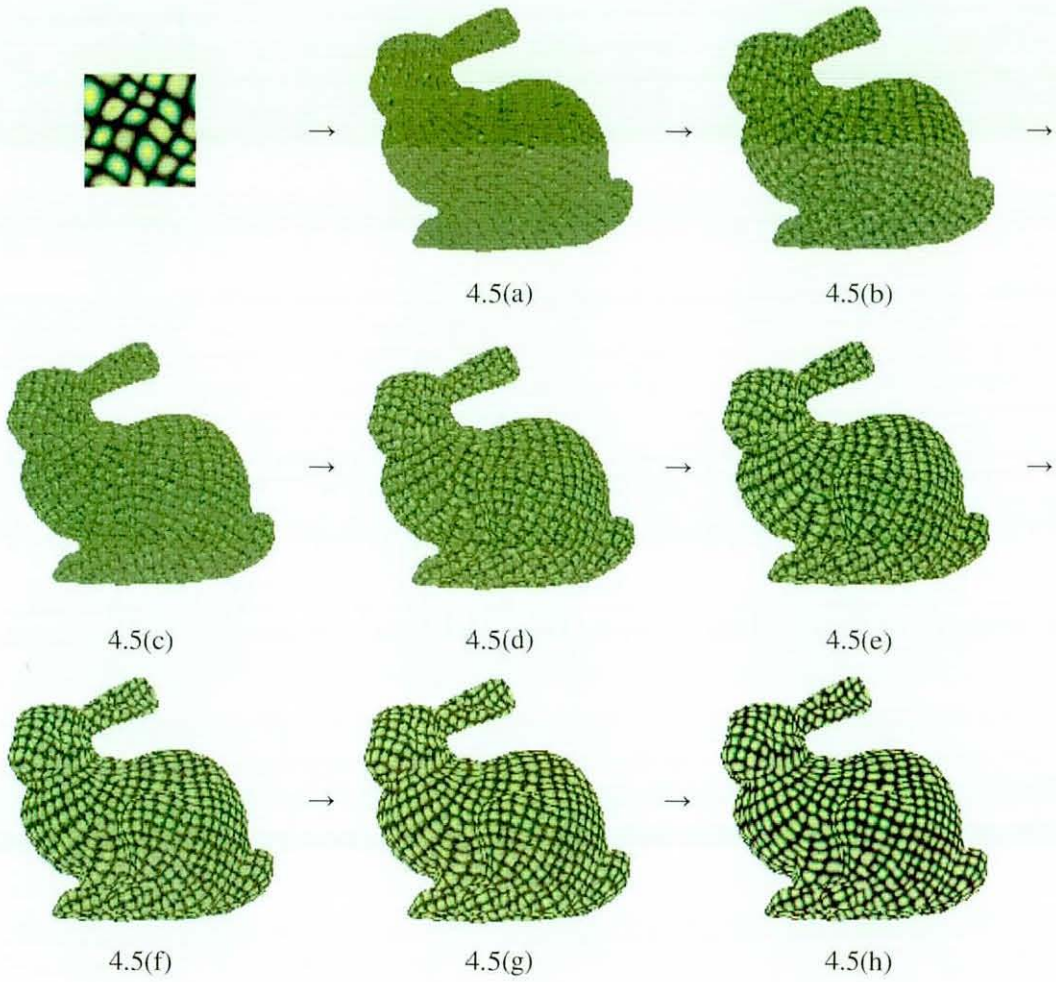


Figure 4.5: Progressive texture synthesis on the bunny by the proposed algorithm using texture sample-1 and 256 K points.

The nature of the design and implementation of the proposed design enables its use in application domains where existing texture synthesis algorithms perform ineffectively due to functional constraints. The following is a summary of applications that could benefit from the specific progressive/ multi-resolution design of the proposed algorithm.

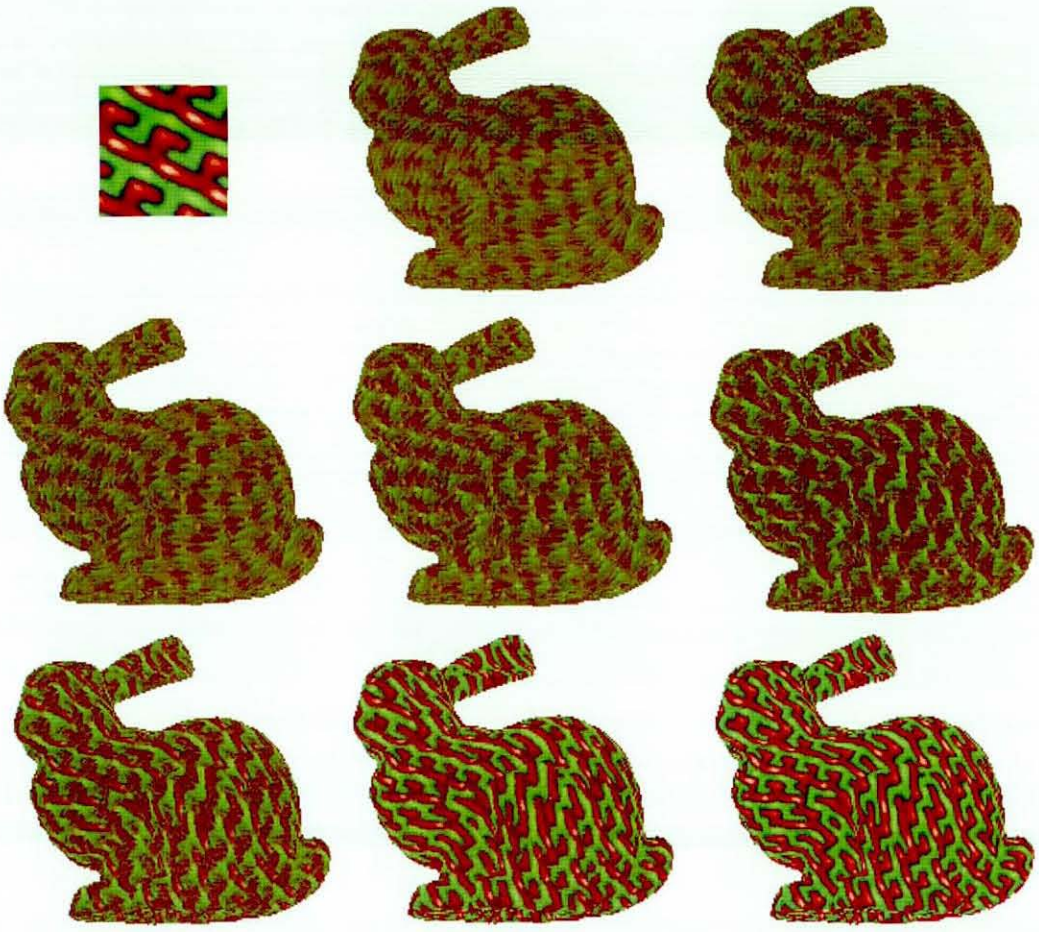


Figure 4.6: Progressive texture synthesis on the bunny by the proposed algorithm using texture sample-2 and 256 K points

Progressive 3D texture transmission: Within a progressive transmission scenario, data is transmitted according to visual and/or decoding significance. Special design of the proposed texture synthesis algorithm allows DWT coefficient significance based progressive creation, transmission and reconstruction of the synthesised texture. Current applications such as video/movie on demand, online gaming and remote visualisations are a few applications that would benefit from the above feature.

Compressed domain texture synthesis: Synthesizing a compressed output texture with the use of a compressed original texture sample. This approach is useful in fast, on demand, internet based applications such as gaming and medical visualisations.

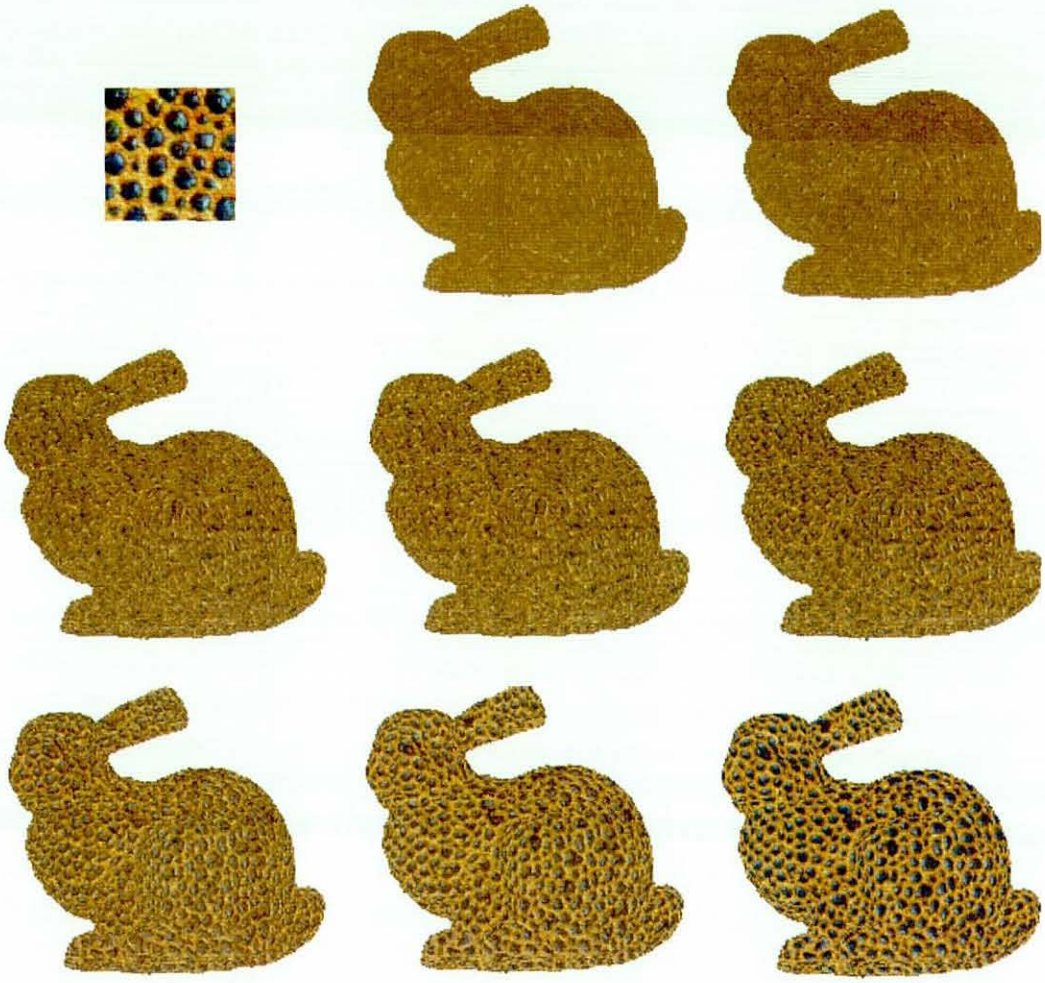


Figure 4.7: Progressive texture synthesis on the bunny by the proposed algorithm using texture sample-3 and 256 K points.

4.4 LIMITATIONS

The proposed algorithm suffers from the following limitations:

1) *It inherits some drawbacks of Turk's algorithm, e.g., applicable to limited* classes of texture.
2) The synthesised texture quality is dependent on the neighbourhood size and user defined vector fields.
3) The time required to synthesise texture is relatively high and is dependent on the number of hierarchical mesh points created on the surface, sample texture

size and level of pyramid in coarse-to-fine refinement.

4) Extra memory and computation is required for encoding and decoding the '*point image locations*'.

In the subsequent chapters of this thesis we propose a number of novel, transform domain, patch based texture synthesis approaches that can resolve the above shortcomings.

4.5 SUMMARY AND CONCLUSION

In this chapter we have proposed a novel approach to progressively synthesizing texture based on the representation and analysis of the texture sample in the DWT domain. We have provided experimental results and a detailed analysis to highlight the added functionality provided by the proposed scheme and its ability to perform accurate, good quality progressive texture synthesis on 3D surfaces. We have shown that the proposed method has the capability of synthesizing texture at seamlessly different quality settings, a functionality which is not achievable via existing state-of-the-art techniques. It has been proved that the use of visual prioritisation of information in the sample image during texture synthesis allows the tasks to be carried out at a higher speed at equivalent visual quality levels, as compared to the existing techniques.

Chapter 5

Max-Flow/Min-Cut Approach for Planar Texture Synthesis

5.0 INTRODUCTION

In this chapter we propose a DWT domain design and implementation for the popular max-flow/min-cut algorithm that is used to provide the optimal cutting path between two matching patches in patch based planar texture synthesis. A coarse cutting path is initially generated using the lowest resolution sub-band of the decomposed texture image and subsequently the path is refined using the coefficients of the higher resolution sub-bands. The Embedded Zero-tree Wavelet (EZW) concept has been used to prioritise and use the wavelets coefficients according to their impact on visual quality, thus optimising the visual quality at patch overlapping boundaries. Throughout the algorithm design, we have given special consideration to minimizing the computational cost. We have shown that the adaptation of the max-flow/min-cut algorithm in the wavelet transform domain results in an efficient texture synthesis algorithm that is capable of being used in conjunction with modern, band-width adaptive applications. A set of regular to stochastic textures have been used to prove the effectiveness of the proposed algorithm and to compare them with existing state-of-the-art techniques. It should be noted that in the subsequent chapters we use this algorithm in optimising the texture synthesis quality in patch overlapping areas.

For clarity of presentation, this chapter is divided into several sections. Section 5.1 gives motivation behind this work; Section 5.2 discuss the proposed algorithm, benchmark algorithm and underlying techniques that are used; section 5.3 provides the experimental results and analysis of the proposed work and section 5.4 will summarise and conclude the chapter.

5.1 MOTIVATION OF WORK

It was discussed that a number of key recent contributions to transform domain texture synthesis have been made by Wickramanayake et al. [12, 39]. The authors designed and implemented block based texture synthesis algorithms in transform domain as a solution to the speed related problems in traditional patch based texture synthesis approaches. In particular Shapiro's EZW concept was used for limiting the number of coefficients used in the texture synthesis process based on their visual significance, thereby reducing the computational cost (see chapter 3, section 3.3.1).

It has been shown that the original transform domain texture synthesis algorithms proposed by Wickramanayake et al. perform well in synthesizing a wide range of texture patterns at high operational speeds. However our detailed analysis indicated that in synthesizing some regular and near-regular textures consisting of patterns with sharp edges, the algorithms perform sub optimally. The main reason for this is the use of the simple weighted edge blending technique that results in the blurring of edges. Therefore to meet optimal results, one needs to explore further methods for resolving patch boundary mismatches.

In [36], Kwatra et al. proposed the use of their popular graphcut technique of texture synthesis that uses the max-flow/min-cut algorithm [37] to optimise the quality at patch edges (see chapter 3, section 3.3.2). They demonstrated that the graph cut algorithm is capable of better quality texture synthesis as compared to most other patch/block based texture synthesis techniques that use simple edge blending approaches. Note that the graph cut algorithm provides an optimum cut along which the sum of pixel value differences is minimized. Hence extending the max-flow/min-cut algorithm to the transform domain and its subsequent use in minimising edge artefacts in transform domain patch based texture synthesis [12, 39] should provide a plausible solution. It is noted that the quality of edges are measured subjectively rather than using an objective metric.

5.2 PROPOSED ALGORITHM

This section discusses design and implementation details of the proposed transform domain max-flow/min-cut technique which is capable of enhancing the overall synthesised texture quality in transform domain texture synthesis [5, 39].

The novel algorithm combines the merits of two algorithms that were discussed in detail in chapter 3. They are:

1) Benchmark Algorithm 1 - *Multiresolution texture synthesis in wavelet transform domain*: limits the number of coefficients in the searching process for best matching blocks, hence reduces the computational cost.
2) Benchmark Algorithm 2 - *Graphcut Textures: Image and Video Synthesis Using Graph Cuts*: provide optimal seams at patch boundaries.

In [5, 39] a n level DWT is initially applied on the sample texture (see figure-5.1(a)) which results in a decomposed image. (See figure-5.1(b) where n=3 decomposition is applied on sample images.)

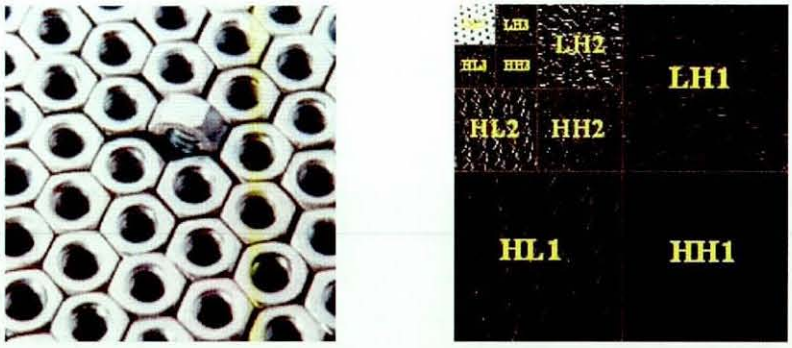
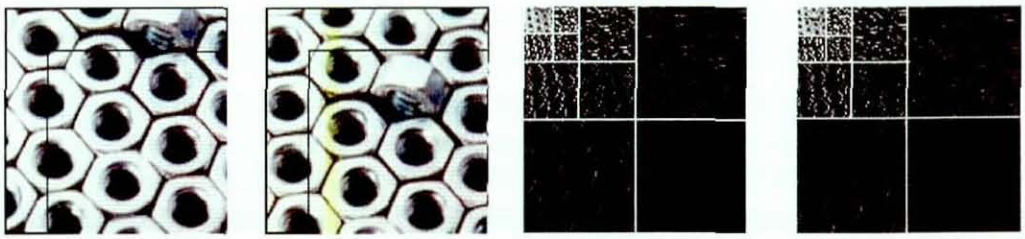


Figure 5.1: (a) Sample Texture (b): n level decomposed sample image

The texture synthesis algorithm is required for patching blocks with seamless boundaries. Once a block has been synthesised, searching for its best matching block (to be patched as the subsequent block to be synthesised) is carried out by effectively using the coefficients of the LLn and HHn sub-bands [5, 39]. The coefficients in the sub-bands can be prioritised according to their visual impact, providing the ability to synthesise textures faster than traditional approaches.

Once a matching pair of adjacent blocks is selected, their overlapping boundary should be further refined to remove edge artefacts. In general, given the fact that a certain amount of texture would have been already synthesised, synthesizing a new block will require the consideration of an L-shaped boundary region. For clarity of presentation we used two complete patch blocks and illustrated a typical overlapping region in figure-5.2(a) and its transform domain representation in figure-5.2(b).

The idea of using the max-flow/min-cut algorithm is to make a cut in order to minimise any potential seam artefact in the overlapping region. In the proposed transform domain extension to the max-flow/min-cut algorithm, the process begins with a sub-ordinate pass (see section 5.2.1) which gives an approximate cutting path. This path is further refined using a subsequent refinement pass (see section 5.2.2).



Block A Block B Block A Block B

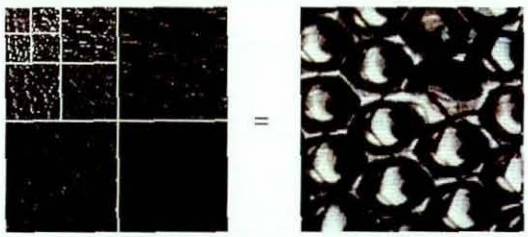
(a) (b)

Figure 5.2: (a) Pixel image of overlapping region for block A and B (b) Transform domain of overlapping region for block A and B.

5.2.1 Sub-Ordinate Pass of Patch Fitting

The first step of the sub-ordinate pass is to cast the above problem into graphcut problems. We measure the matching quality for wavelet coefficients of block A and B. The simplest quality measure is the wavelets difference between the pair of wavelet coefficients of the 'difference block' sub-bands. The 'difference block' is the coefficient difference between the transform domain overlapping blocks A and B as illustrated in figure-5.3(a). For clarity of presentation and explanation, figure-5.3(b) represents the 'difference block' in the pixel domain.

In our experiments, after a third level decomposition and difference, all ten sub-bands are initially extracted. The max-flow/min-cut algorithm is then applied on the overlapping region using the lowest resolution sub-bands, i.e. LL_3 bands to obtain the sub-ordinate minimum cutting path. (see figure-5.4). The cutting path obtained will run between the pairs of wavelet coefficients and is called the 'primary cut'.



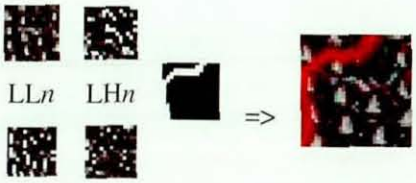
(a) (b)

Figure 5.3: Difference Image

LL*n* Cutting Path

Figure 5.4: Cut on Low Level Band (Note: LL*n* has been zoomed for clarity).

The magnitude of the wavelet coefficient in higher decomposed levels of the sub-bands is high which will provide the more appropriate cut. Considering the 'primary cut' as the minimum cutting path, it is then used to cut the sub-bands, HH_3, LH_3, HL_3. Subsequently using inverse-DWT on sub-bands LL_3, HH_3, LH_3, HL_3, the sub-band LL_2 is obtained (See figure 5.5.) For ease of reference, the cut obtained in the LL_2 sub-band is named the 'incorporated path' (shown in red).



LL*n* LH*n*

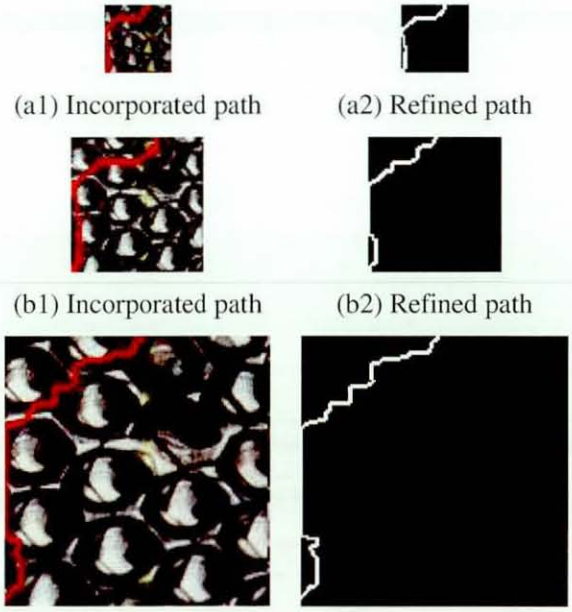
HL*n* HH*n* LL*n-1*

Figure 5.5: LL_2 with incorporated cut (note: LL*n*, LL*n*-1 has been zoomed for clarity).

5.2.2 Refinement Pass of Patch Fitting

The primary cut is further refined by considering only the neighbouring wavelet coefficients around it in the subsequent levels of inverse discrete wavelet transform i.e. IDWT, starting from the incorporated path of LL_2. This makes sure the primary cut gets refined accurately and is influenced by a number of wavelet coefficients. In practice this refinement process is continued through all the IDWT levels. It is noted here that the process of refinement passes through all n levels of decomposition. Therefore the number of refinement steps is n, i.e. n =3 in our experiments.

Figure-5.6 (a1) illustrates that the incorporated cut in LL_{n-1} is first refined by max-flow/min-cut using neighbourhood coefficients (see figure-5.6 (a2)). This refined cut is then incorporated into the three detailed sub-bands of the same resolution level. The incorporated cut at level LL_{n-2}, i.e. LL_1 is obtained via inverse DWT of the four sub-bands (see figure 5.5). The incorporated cut at level LL_1 is illustrated in figure-5.6 (b1). This process is repeated through all the levels until the refined path at level LL_0 (full resolution) is obtained (see figure 5.6 (c2)).



(a1) Incorporated path (a2) Refined path

(b1) Incorporated path (b2) Refined path

(c1)Incorporated path (c2)Refined path

Figure 5.6: Refinement of primary cut using corresponding bands and neighbouring coefficients.

The cutting path produced by following the above process results in a minimum cutting path for the final pixel domain overlapping block. The final blended quality similar to that illustrated by figure 5.7 is obtained.

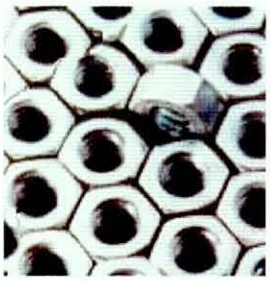


Figure 5.7: Final blended block

5.2.3 Feathering

Although the above approach produces an optimum seam, the resulting quality is restricted by the quality of the original edge blocks. Therefore, it is likely that some artefacts still remain after the application of the proposed approach. To address this problem we have used the popular feathering [87] approach that uses suitable weighting factors to combine those pixels at the seams. Feathering results in a further improvement of the seam quality.

5.3 EXPERIMENTAL RESULTS AND ANALYSIS

Experiments were carried out on a large database of commonly used texture samples representing all textures on a texture spectrum [4] i.e. regular to stochastic. This section discusses experimental results in detail. The proposed approach is implemented in MATLAB.

Figure 5.8 illustrates a comparison between the edge blending approach used in Wickramanayake et al. [5, 39] and our approach. To clearly illustrate the possible improvements, a block selected for demonstration purposes above was one in which the edge-blending algorithm would perform sub-optimally (note the artefacts at the top and front rows of figure 5.8(a)).

(a) Edge blending (b) Our algorithm

Figure 5.8: Comparison between edge blending and the proposed transform domain max-flow/min-cut algorithm.

For a more comprehensive analysis we compare the results of our algorithm against Kwatra et al. [36] pixel domain max-flow/min-cut algorithm and Wickramanayake et al.[5, 39] transform domain edge-blending algorithm. The results clearly illustrate that the proposed algorithm provides better visual quality at block edges as compared to using the non-optimised edge-blending algorithm adopted by Wickramanayake et al. Though the Wickramanayake et al. algorithms provide faster texture synthesis as compared to the Kwatra et al. algorithm, it was stated previously that the simple edge-blending approach adopted was non-optimum. However the Kwatra et al. algorithm is based on the max-flow/min-cut algorithm, which is capable of providing the optimum seam between two matching blocks in the pixel domain. Our extension of this algorithm to the transform domain has enabled its application in Wickramanayake et al.'s fast texture synthesis algorithm. The detailed experimental results show that the proposed approach provides equivalent visual quality at block boundaries, at a much lower computational cost, when compared to the pixel domain max-flow/ min-cut algorithm adopted by Kwatra et al. [Note : The random selection of the initial block removes the ability to reproduce the results of the above algorithm. This can be resolved by selecting the first block to be specified.]

The texture synthesis results obtained for the 'nut' texture sample which can be categorised as of type near–regular, when using our method, Wickramanayake et al.'s and Kwatra et al.'s. algorithms, are illustrated in Figure-5.9. It is clear from the results that the proposed approach provides an improved quality with less blurring at patch

boundaries, as compared to that of Wickramanayake et al.'s algorithm. The results are visually similar and comparable with the results of Kwatra's algorithm.

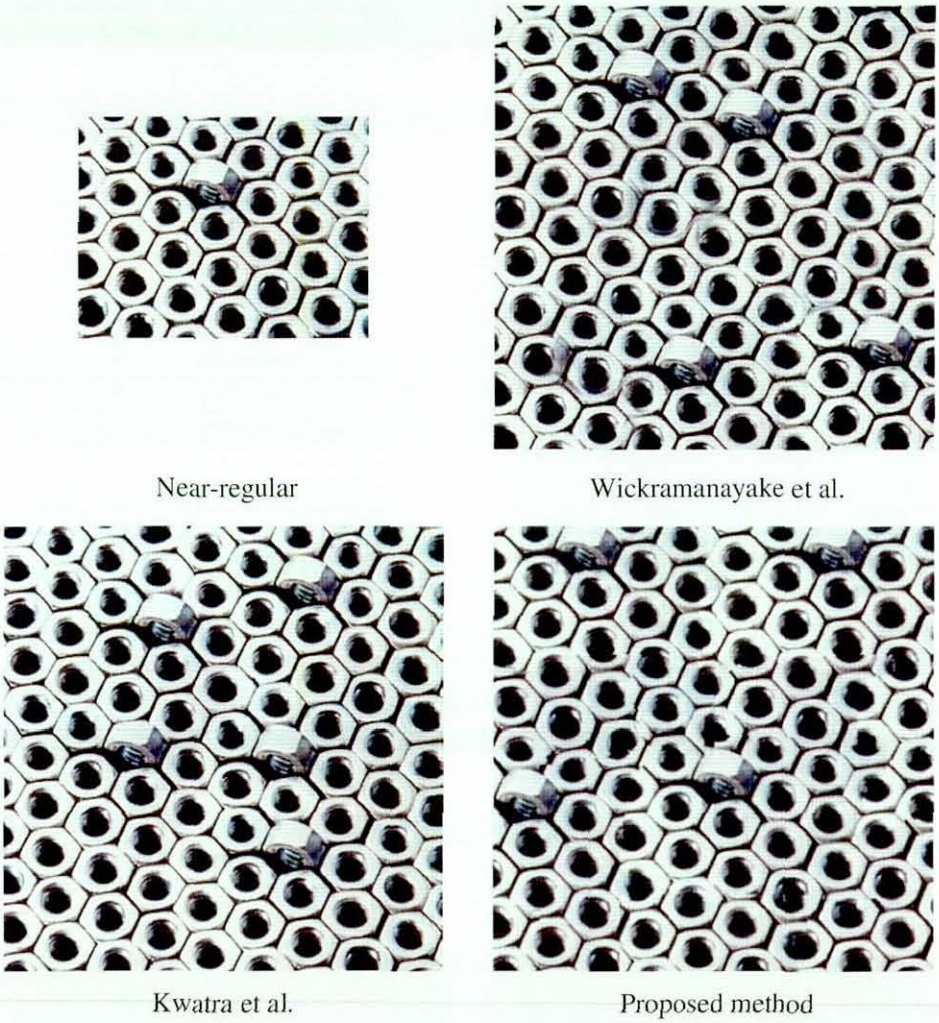


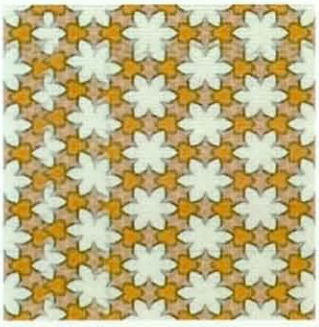
Figure 5.9: Comparison of synthesised texture qualities for the texture, 'nut'

Figure-5.10 illustrates the texture synthesis results obtained for further eight different texture samples, when using the proposed, Wickramanayake et al.[5, 39] and Kwatra et al.'s[36] algorithms. Figure-5.10 (a1, b1) are examples of sample textures or the 'regular' type. Carefully observing the results of texture synthesis obtained when using the proposed technique illustrates that (see figure-5.10(a4, b4)) the proposed technique performs subjectively better than Wickramanayake et al.'s (see figure-5.10(a2, b2)) algorithm and performs similar to Kwatra et al.'s (see figure-5.10(a3 b3)) algorithm. Figure-5.10(c1, d1, e1, f1) are sample textures of the category, 'near-

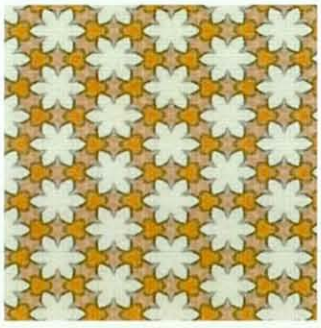
regular', where the proposed algorithm (see figure-5.10(c4, d4, e4, f4)) performs better than Wickramanayake et al.'s algorithm (see figure-5.10 (c2, d2, e2, f2)) and similar to Kwatra et al.'s algorithm (see figure-5.10(c3, f3), and better than Kwatra et al.'s algorithm (see figure-5.10(d3, e3)) in certain other cases. Note that the low probability of an improved quality texture compared to Kwatra's approach, when using the proposed multi-resolution transform domain approach, exists due to the non-consideration of noise and unnecessary texture detail in matching, when the proposed approach is used. Figure-5.10(g1) irregular sample textures where the proposed algorithm (see figure-5.10(g4) performs better than Wickramanayake et al.'s (see Figure-5.10(g2)) and similar to Kwatra et al.'s (see figure-5.10(g3)) algorithms. Figure-5.9(h1) are stochastic sample textures where the proposed algorithm (see Figure-5.10(h4)) performs similar to Wickramanayake et al.(see figure-5.10(h2)) and to Kwatra et al.'s (see figure-5.10(h3)) algorithms.

(a1) Regular texture

(b1) Regular texture

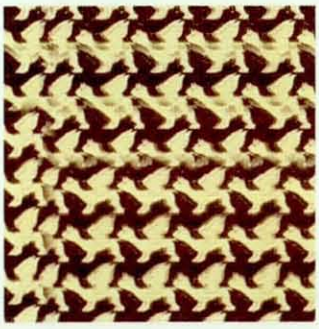


(a2) Wickramanayake et al.

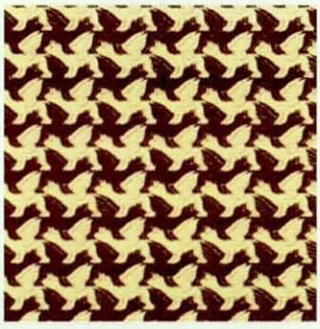


(a3) Kwatra et al.

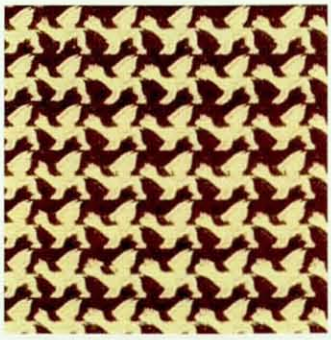
(a4) our method



(b2) Wickramanayake et al.



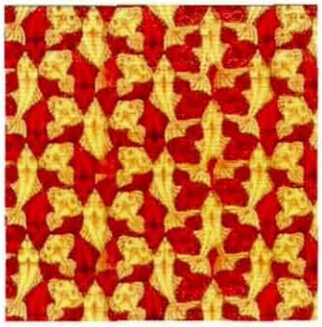
(b3)Kwatra et al.



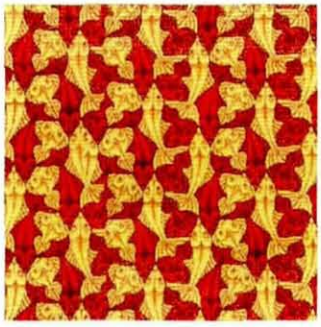
(b4) our method

(c1) Near-regular texture

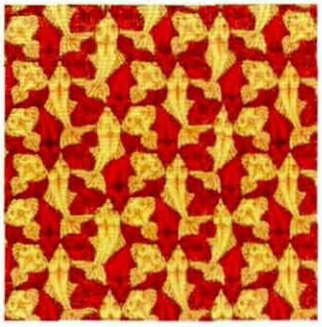
(d1) Near-regular texture



(c2) Wickramanayake et al.

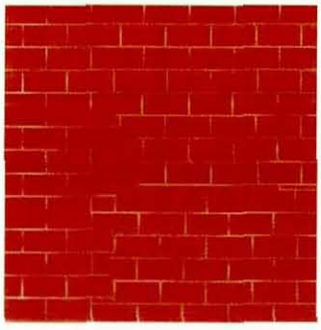


(c3) Kwatra et al.



(c4) our method

(d2) Wickramanayake et al.

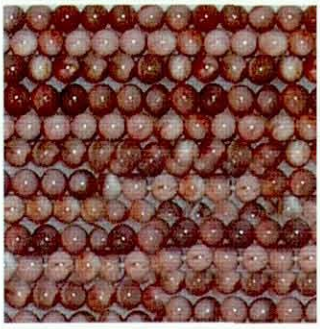


(d3) Kwatra et al.

(d4) our method

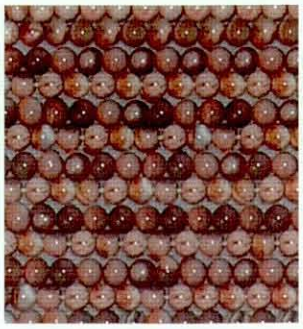
(e1) Near-regular texture

(f1) Near-regular texture



(e2) Wickramanayake et al.

(e3) Kwatra et al.



(e4) our method

(f2) Wickramanayake et al.

(f3) Kwatra et al.

(f4) our method

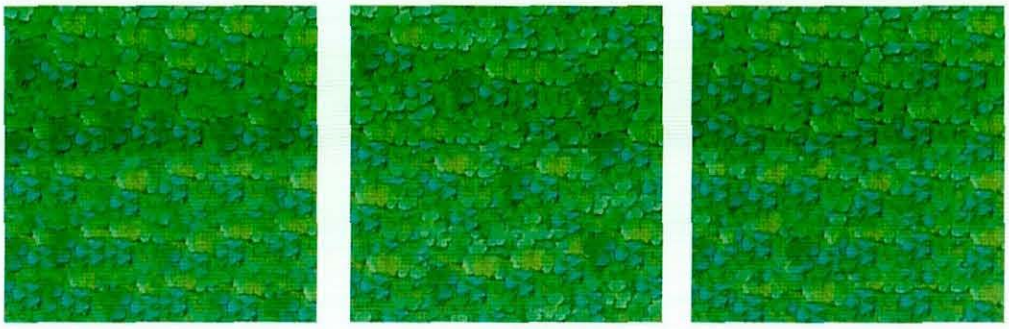
(g1) Stochastic texture

(h1) irregular texture

(g2) Wickramanayake et al.

(g3) Kwatra et al.

(g4) our method



(h2) Wickramanayake et al. (h3) Kwatra et al. (h4) our method

Figure 5.10: Comparing the results of the proposed, Kwatra et al.'s and Wickramanayake et al.'s, algorithms.

The replacement of the edge-blending algorithm adopted by Wickramanayake et al. in [5, 39] by the proposed transform domain max-flow/min-cut algorithm, provides a fully transform domain, fast texture synthesis algorithm (due to transform domain analysis/synthesis, see [5,39] for quantitative timing comparisons) with optimum quality at edges (due to max-flow/min-cut algorithm). Such a texture synthesis algorithm has applications in many practical domains where existing algorithms perform ineffectively due to functional constraints. Some examples of such application areas are:

Compressed domain texture synthesis: Refers to the synthesis of a compressed output texture with the use of a compressed original texture sample. This is a useful feature in fast, on demand applications, with resource/device constraints such as mobile and hand held devices.

Computer Games: Transform domain texture synthesis can be used in generating rendered textures of surroundings, such as walls, background or foreground objects, in which the quality of the rendered texture is kept to a minimum depending on its required visual significance at any given time.

MPEG4: MPEG4 AFX uses Bézier surfaces to model and animate (describe smooth motion) three dimensional objects. Transform domain texture synthesis approaches can be easily adopted to provide progressive, surface rendering to these objects.

It is noted that the current implementations of the proposed transform domain texture synthesis algorithms are limited to a proof of concept that they can be fully utilised to support the above applications if required.

5.4 SUMMARY AND CONCLUSION

In this chapter we have discussed transform domain extensions to the max-flow/min-cut algorithm, popularly used in the edge enhancement of patch based texture synthesis algorithms. We have provided experimental results and detailed analysis to prove that the proposed technique is capable of being used to replace the simple non-optimised edge-blending strategy adopted by Wickramanayake et al. in their pioneering work of transform domain texture synthesis, thus further improving its efficiency and extending its application.

In designing the proposed multi-resolution, transform domain extension to the max-flow/min-cut framework, we have given special attention towards reducing algorithmic complexity by introducing refinement passes and progressively improving the visual quality at block edges.

Chapter 6

Texture Synthesis on Geometric Surfaces

6.0 INTRODUCTION

It has been shown that the existing 3D texture synthesis algorithms fail to deliver effectively in application areas where a progressive texture rendering capability is required. The novel algorithm proposed in this chapter provides a practical solution to this challenge. The algorithm is an extension of the novel DWT based approach to texture synthesis on planar surfaces, presented in chapter 5. In this chapter we demonstrate the use of the proposed algorithm in progressive texturing of geometric surfaces such as Ring Dupin Cyclides, Tori, Spheres, which are parametrised by biquadratic rational Bézier patches. We provide experimental results to prove the efficiency and effectiveness of the proposed approach, and demonstrate its use in resource constrained domains.

For clarity of presentation this chapter is divided into several sections. Section 6.1 discusses the motivation behind the work. Section 6.2 presents the novel texture synthesis algorithm on biquadratic rational Bézier patches. Section 6.3 extends the fundamental ideas presented section 6.2 to support texture synthesis on a number of selected geometric surfaces. Section 6.4 specifically discusses the proposed method of embedding texture onto individual patches. Section 6.5 presents an EZW based algorithm that enables progressive texturing of geometric surfaces. Section 6.6 provides experimental results and a comprehensive analysis that includes a comparison of the proposed method's performance with that of the existing state-of-the-art techniques. Finally section 6.7 summarises and concludes the chapter.

6.1 MOTIVATION

In patch based texture synthesis techniques, the use of a triangular or quadrilateral mesh, results in seams at the edges. The size of triangles in a triangular mesh varies, which makes prominent visual artefacts on the surface. Further to this, the graphical object will use extensive bandwidth in transmission since the surface mesh representation depicting the objects shape, and texture information depicting its appearance, are in an uncompressed format. Therefore, their applicability in a constrained bandwidth channel is limited. Further to this, animation and deformation of triangular meshes are relatively difficult and costly. To overcome many of the above problems, and to be used effectively in bandwidth-limited transmission media, we propose a novel algorithm that enables fast progressive-texture synthesis on Bézier patches. We show that it can be further extended and generalised for texture synthesis on arbitrary shaped 3D surfaces, achieving faster synthesis rates and enhanced quality as compared to that of the current state-of-the-art techniques.

6.2 TEXTURE SYNTHESIS ON BIQUADRATIC RATIONAL BÉZIER PATCHES

The underlying techniques used in this proposed work are as follows (see chapter 3 for details):

1) DWT provides a compact multi-resolution representation of the sample texture (see section 3.1.4).
2) EZW prioritizes the wavelet coefficients according to their perceptual significance (see section 3.2).
3) EZW can be used in adaptive systems which require dynamic quality/speed adjustment (see section 3.2).

An overview of the operation of the fundamental unit of texture synthesis illustrated in figure 6.1, can be given as follows: The sample texture is first decomposed using an n-level DWT. A random decomposed block of this sample is initially picked, converted into the pixel domain using an n-level IDWT procedure and subsequently embedded as a starting, corner block of a patch of the surface. An overlap area of this

block with the subsequent texture block to be synthesised is selected, and converted into the DWT domain using an n-level forward DWT. The LL and HH bands of this overlap area are then used to find a suitably matching block from the decomposed sample texture image. The best matching block is subsequently converted back into the pixel domain using an n-level IDWT and is placed adjacent to the previously synthesised texture block. An edge-blending algorithm [116] is used to improve the quality of the seam between the two adjacent blocks. The above process is continued until the entire surface of the patch (i.e. small rectangular mesh) is textured.

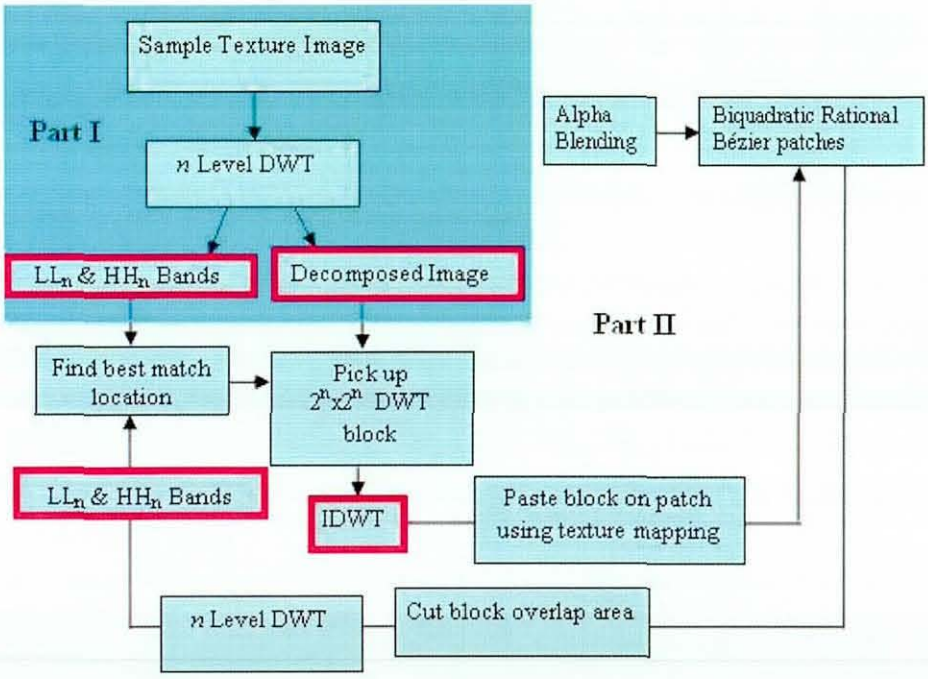


Figure 6.1: Texture synthesis on a single bi-quadratic rational Bézier patch.

The block diagram (figure 6.1) is divided into two parts. Part I shows the decomposition of a texture image and extracting LLn and HHn sub-bands (explained in detail in section 3.1.4). Referring to figure 6.1 Part II, Let $B_{pl(x,y)}$ represent a general square block of the decomposed sample image located at position (x, y) relative to the sub-bands (p, l)'s origin. [Note that in our experiments the size of the above block was set to be 2^{5-l} x 2^{5-l}, where l (= 0, 1, 2).]. Initially we randomly pick a block, $B_{LLn-1(x,y)}$ from the lowest resolution, (LL, n-1) sub-band of the decomposed sample image. Subsequently by combining this block with the corresponding blocks of the other sub-

bands and performing an n-IDWT the randomly selected block is obtained in the original pixel domain, (i.e. B_{LL}).

$$B_{LLn-2} = IDWT(B_{LLn-1}, B_{HLn-1}, B_{LHn-1}, B_{HHn-1}) \quad (6.1)$$

$$B_{LL} = IDWT(B_{LL0}, B_{HL0}, B_{LH0}, B_{HH0}) \quad (6.2)$$

After randomly picking up the block as described above and converting it into its pixel domain, the next stage is to map the texture block onto a Bézier patch (for Bézier patch refer to section 3.4.2). The Bézier patches of the surface will be divided into a quadrilateral mesh, where each mesh element is referred to as a surface division. The above mentioned process randomly picked texture block is embedded to surface division (see figure 6.6) by following the texture embedding method detailed in section 6.4. This procedure is popularly known in the literature as *texture mapping*. Once the first, randomly selected block is mapped as above, an overlapping area of this block (with the adjacent, potential block to be mapped) is selected to be used in locating the next block from the sample texture to be mapped onto the patch. Note that in our experiments this area was 8 pixels wide, and was selected from the left end, bottom end or both above ends of the block (depending on the location of the original texture block on the Bézier patch).

The overlap area is subsequently *n-level* decomposed (n=3 in our experiments) and used to find a suitable match for the adjacent block from the decomposed sample texture using minimum sum of squared-differences (see section 3.3.1 figure 3.8). To speed up the block matching we only use all coefficients of the LLn and HHn sub-bands for matching. In our experiment where n=3 and a sample texture size of 128 × 128 or 256 × 256 is used, out of the 16384 or 65536 possible coefficients, we only use 1/64 of total number coefficients in the sample texture. Therefore, the matching speed is significantly enhanced. Although adjacent blocks on a Bézier patch are matched using an overlapping area, block seam mismatches are likely to exist. However due to the efficiency of matching in the DWT domain [39] these seam mismatches are minimal as compared to using an alternative patch based texture synthesis technique and are therefore easily eliminated using an edge-blending approach or max-flow/min-cut approach [116, chapter 5]. The above process of block matching,

selection and placement is continued until the texture of the entire patch is synthesised.

6.3 TEXTURE SYNTHESIS ON A GEOMETRIC SURFACE

A high level block diagram of the proposed texture synthesis algorithm on a geometric surface is illustrated as in figure 6.2.

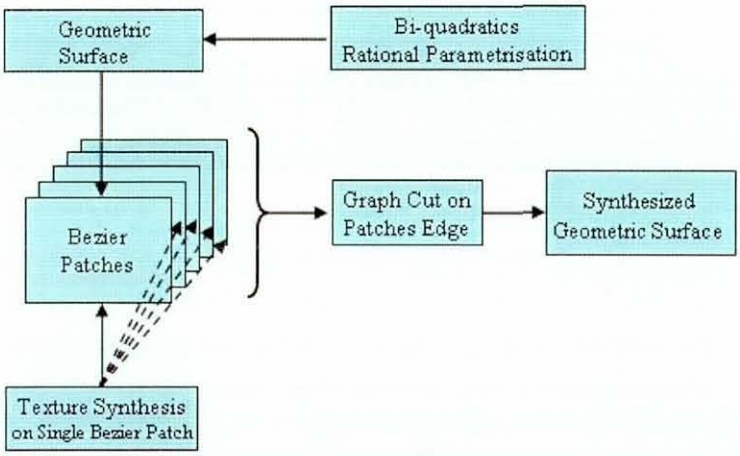


Figure 6.2: Proposed block diagram for texture synthesis on geometric surface.

The geometric surface (a Ring Dupin Cyclide in our experiments) to be textured, is first parametrised into a collection of Bézier patches (i.e. set of small rectangular mesh) (see figure 3.2.2). Subsequently, texture is independently synthesised into individual Bézier patches (see section 6.1). The texture synthesis techniques adopted result in a seamless texturing within patches. Depending on whether or not seamlessly progressive texture synthesis is required, the use of the EZW algorithm (see section 6.5) is considered for coefficient prioritisation/selection in forming the texture blocks (see figure 6.1) of a Bézier patch. The Bézier patches are subsequently embedded into the surface mesh representation obtained using the rational parametrisation approach which was presented in section 3.4.4. The visibility of seams between Bézier patches is then reduced using a minimum boundary cut technique, i.e. graph-cut (see section 6.3.1). Figure 6.1 illustrates the block diagram of the fundamental unit of the proposed overall texture synthesis algorithm of figure 6.2, i.e. the block diagram depicting the

detailed procedure adopted in texture synthesis on a single bi-quadratic rational Bézier patch.

As individual Bézier patches are textured independently, it is likely that visible seam boundaries exist between patches (see figure 6.3). In the following section we describe the use of a minimum boundary cut technique to minimise these artefacts.

(a) (b)

Figure 6.3: Artefacts at Bézier patch edges.

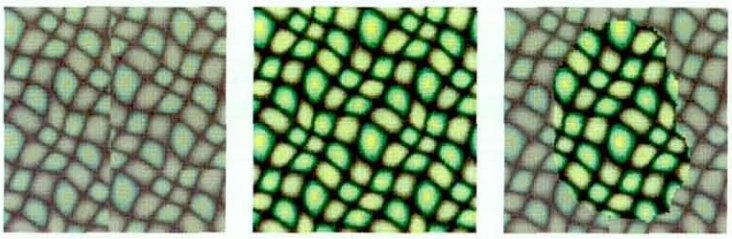
6.3.1 Using a Modified Graphcut Technique to Improve Bézier Patch Seams

To remove the edge artefacts between Bézier patches (see figure 6.3) we use a modified version of the popular Graphcut technique proposed in [36]. Its operations can be explained as follows:

Patch selection and matching: The idea of this stage is to replace Bézier patch seam regions, with visible straight-line artefacts, with matching regions from the sample texture, which can be better blended with the non-seam region texture. To accomplish this task a block of size $2^{s-l} \times 2^{s-l}$ (l=3 in our experiments), which will be called an edge block is first selected from the patch boundary (see figure 6.4(a)). It is noted that seams are included within this edge block. Subsequently we apply an n-level (n=3) DWT on the selected edge block to create a DWT decomposed edge block. The LLn and the HHn sub-bands of the decomposed block is then used to search the decomposed sample texture for the best matching block, which is found using block matching criteria (see figure 6.4(b)). Once the best matching block is found (see

Figure 6.4(b)) it can then be combined with the edge block using the graphcut approach in the transform domain (see section 4.4.2) in order to minimise seam artefacts. After this an n-level IDWT is performed to obtain a pixel block.

Graphcut Approach: Once the best match to the edge block is found as above, we apply the graphcut technique of [36] which is capable of optimally replacing the seam region of the edge block with regions from the best matching block (see section 4.4.1) in order to provide an optimal minimisation of boundary artefacts (see Figure 6.4(c)). We apply the Graphcut technique (see figure 6.5) in the transform domain to reduce the algorithm complexity and to get the best possible seams. The above modified edge block is finally embedded back in its original position in the Bézier patch seam area. This process is continued for all edge blocks along the Bézier patch seams. The results illustrated in figure 6.6 when compared to the results illustrated in figure 6.3 prove the improved quality at Bézier patch seams.



(a)Artefact block (b) Best Block (c) Embedded block

Figure 6.4: Seam selection

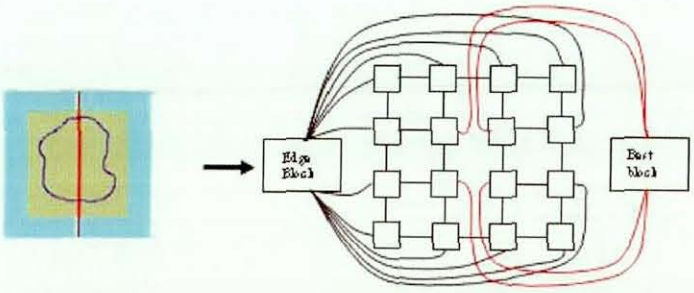
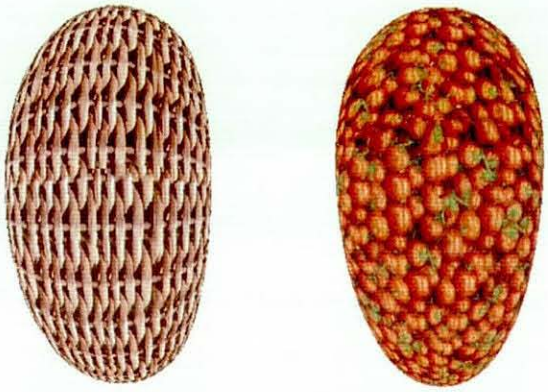


Figure 6.5: The Graphcut approach



(a) (b)

Figure 6.6: Refined Bézier patch edges

Feathering: Although the graphcut approach produces an optimum seam, the optimality is restricted by the quality of the original edge blocks. Therefore, it is likely that some artefacts still remain after the application of the graphcut approach. To address this problem we use the popular feathering [87] approach that uses suitable weighting factors to combine those pixels at the seams. Feathering results in further improvements of the seam quality.

6.4 EMBEDDING TEXTURE ON RATIONAL BÉZIER PATCHES

To embed texture on Bézier patches, we use a modified version of the approach proposed by Soucy et al. [119]. The modification required is due to the use of a quadrilateral mesh in the proposed approach as against a triangular mesh in the original approach of [119]. The texture of each quadrilateral in the original mesh is obtained via direct mapping [119] from the corresponding quadrilateral in a texture space. The texture space referred to above is created as a result of the block placement and matching procedure on Bézier surfaces, described in detail in section 6.1. The quadrilaterals in texture space that we use are rectangular in nature and are uniform in size. It is further noted that the above texture embedding can be performed at interactive (real-time) rates. In our experiments all surfaces are rendered at interactive rates using 1024 × 1024 sample textures. The surface models are composed of between 1,000 and 50,000 quadrilaterals.

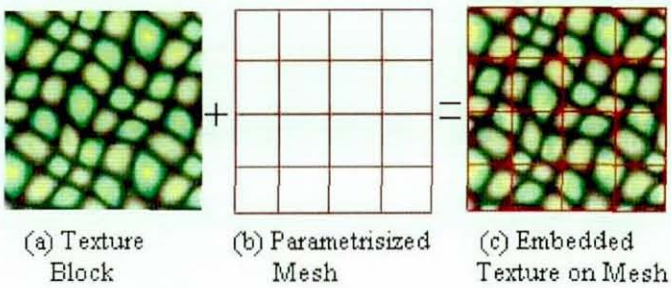


Figure 6.7: Texture Embedding

6.5 PROGRESSIVE TEXTURE SYNTHESIS ON A 3D SURFACE

In order to extend our idea further and improve its applicability, the concept of zerotrees of wavelet coefficients is used for progressive encoding of images (see section 3.2). This uses the EZW algorithm, where the initial coefficient prioritization procedure is used to prioritise their use within the proposed texture synthesis algorithm, where we have replaced the IDWT with the EZWIDWT module (see figure 6.1). The threshold can be calculated using the magnitude of wavelet coefficients of the decomposed sample image (see eq 3.6). The visually prioritized availability of coefficients and the subsequent embedded coding of the coefficients enable seamless progressive texture synthesis capability on bi-quadratic rational patches.

6.6 EXPERIMENTAL RESULTS AND ANALYSIS

In order to analyse the performance of the proposed progressive texture synthesis approach we use three different geometric surfaces namely, a Dupin Cyclide, a Sphere and a Torus. We carried out efficient implementations of the proposed algorithms in OpenGL and C++, following a number of iterative stages of program design and implementation. A range of sample textures with different characteristics were used in the experiments [4]. Figure 6.8 and figure 6.9 illustrate the results. Figure 6.8 compares our new patch based approach to the pixel based approach proposed by Turk's [62]. It is observed that the proposed approach results in a marginally improved texture quality with no visible seams. However the new algorithm is significantly faster than the previous pixel based approach. Our experiments revealed that the previous approach takes approximately an average of 15 to 20 minutes for

progressive texture synthesis, whereas our present approach takes only an average of the order of five seconds for synthesizing identical 3D objects.

Figure 6.10 shows the effectiveness of the proposed algorithm in the progressive texture synthesis domain. It is noted that the average time required to synthesise the textures on the geometric surfaces was in the order of five seconds.

A closer analysis of the results illustrated in figure 6.10 reveals the following important facts:

- By varying the threshold selected for DWT coefficient selection, within the EZW scheme, discrete seamless levels of output quality can be obtained. See figure 6.10 (a1 to a4, b1 to b4, c1 to c4).
- It is observed that for most types of textures, 15% of the DWT coefficients from the sample texture are sufficient for creating output textures of sufficient visual quality. See figure 6.10 (a4, b4, and c4). For regular textures this percentage can be significantly lower (5% in our experiments) figure 6.10 (a2).
- It is observed that the seams are rarely visible.

Given the above observations, the proposed technique can be proved to be beneficial in applications that require fast and accurate progressive texturing capabilities such as distributive and collaborative gaming or low bandwidth transmission applications. Although due to space limitations we have restricted the number of progressive texture synthesis results illustrated in figure 6.10 to four distinct quality levels, the proposed method is capable of progressive texture synthesis at seamlessly different numbers of quality levels, and a large number of sample textures covering broad statistical texture properties. Due to the flexibility of the extension of the Bézier patch based rational parametrisation scheme, the proposed approach can be further extended to efficient progressive texture synthesis on any arbitrarily shaped 3D object (see chapter 7).

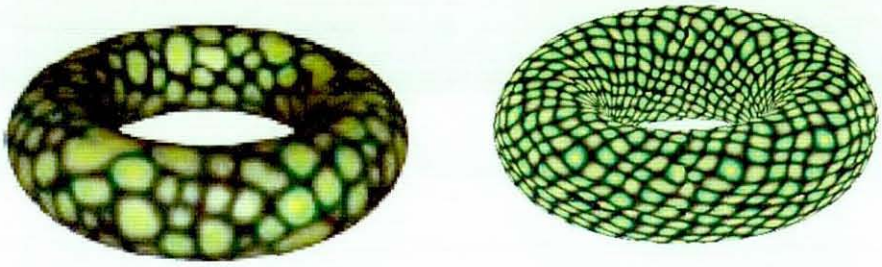
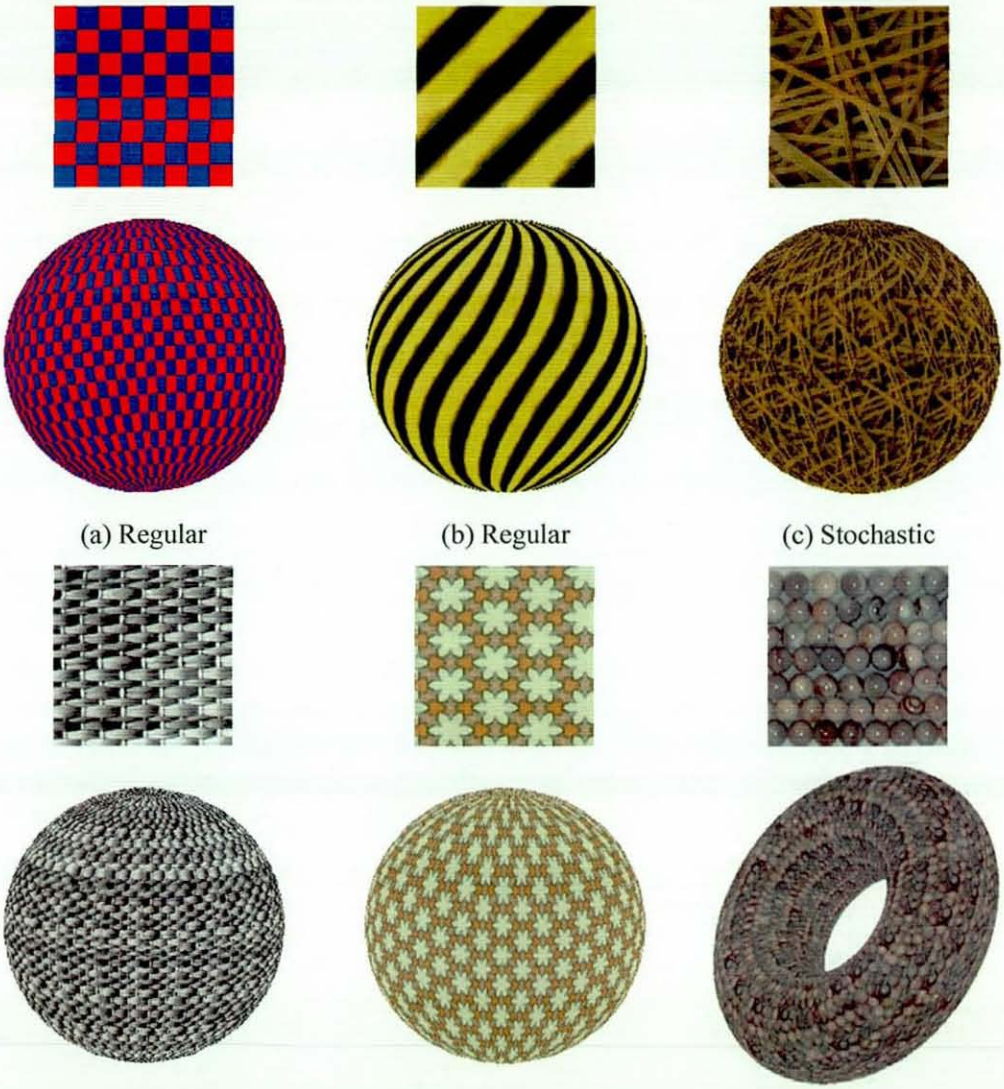


Figure 6.8: (a): Benchmark algorithm (Turk's, see chapter 4), (b) Our present patch based algorithm.

6.7 SUMMARY AND CONCLUSION

In the proposed algorithm a rational parametrisation stage is initially used to represent the surface as a collection of bi-quadratic Bézier patches, onto which the texture is synthesised using a multiresolution DWT approach. We have used the popular EZW idea of prioritizing the DWT coefficients and subsequently embedding the coding in order to introduce progressive texture synthesis. To minimise seam artefacts at the block boundaries within patches, we have successfully used an alpha-blending algorithm, whereas to minimise artefacts at seams between patches we have successfully proposed and used a transform domain extension to the popular random patch cutting algorithm, the 'graph-cut approach', followed by a further feathering algorithm at the exact seam locations. We have provided results to show the effectiveness of the overall surface parametrisation and texture synthesis algorithms. Further experimental results have been provided to show the effectiveness of the measures taken to minimise seam artefacts, and the flexibility and efficiency of the progressive texturing capability of the proposed algorithm.

We have used a Dupin Cyclide [6, 15], a Sphere and a Torus as examples to demonstrate texture synthesis as these surfaces are widely used in computer graphics and in the development of a number of CAD tools. It is further noted that the presented algorithm is the first attempt of progressive and non-progressive texture synthesis on geometric surfaces generated using bi-quadratic rational Bézier equations.



(a) Regular (b) Regular (c) Stochastic

(d) Near regular (e) Regular (f) Irregular

(g) Near regular (h) Irregular (i) Irregular

(j) Stochastic (k) Irregular (l) Irregular

Figure 6.9: Texture synthesis on a Sphere, Torus and Cyclide

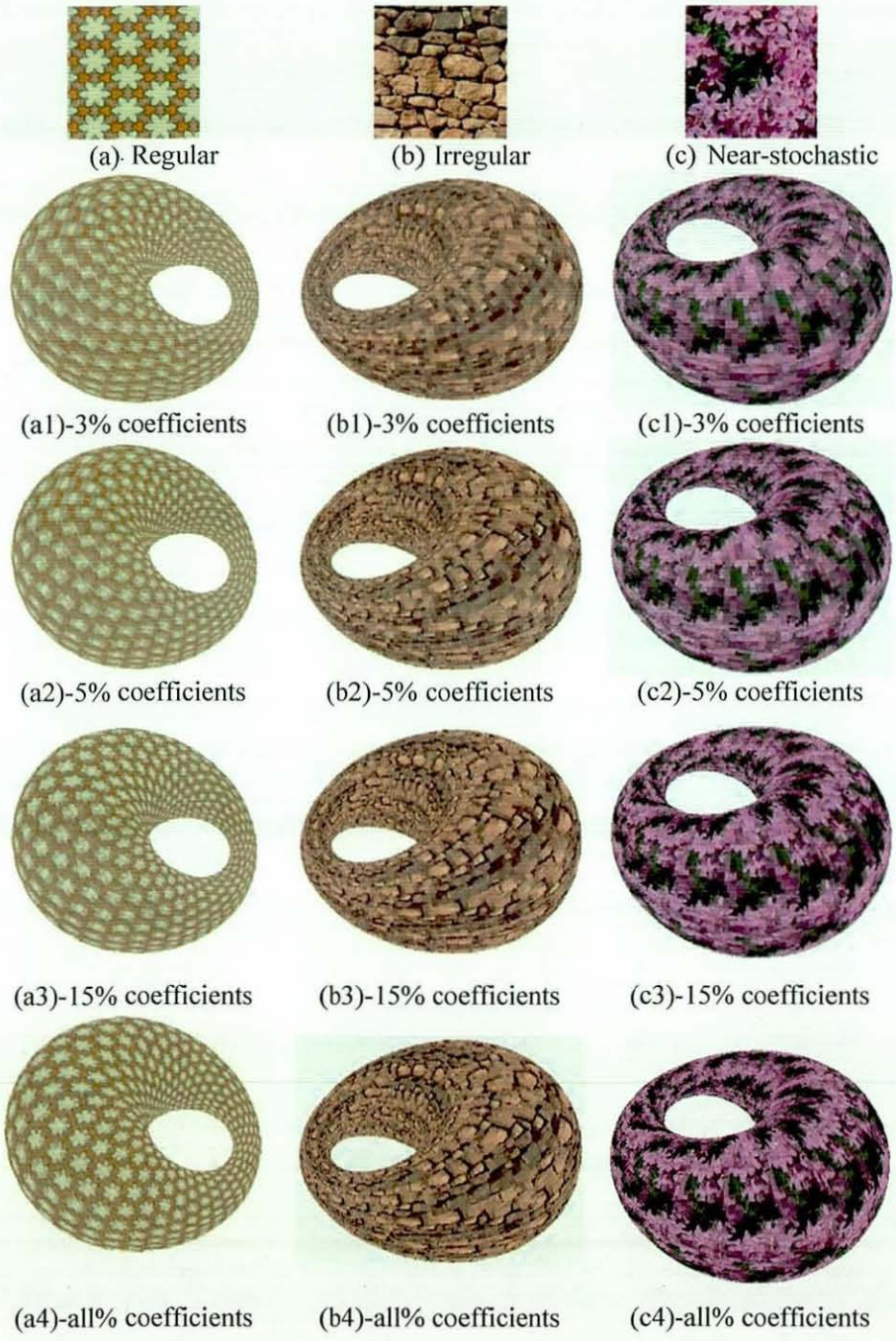


Figure 6.10: Progressive texture synthesis on a Cyclide

Chapter 7

Control Polygon Based Texture Synthesis on 3D Surfaces

7.0 INTRODUCTION

In this chapter, we extend and generalise the algorithm proposed in chapter 6 to enable texture synthesis on arbitrary shaped 3D geometric surfaces. In addition the generalised approach presented in this chapter is supported by an interactive user interface, which increases the speed and quality of texture synthesis due to the introduction of number of additional features such as the ability to define user defined vector fields indicating the direction of texture synthesis in a given locality, use of control polygons and effective projection techniques. Further experimental results are provided to illustrate the practical use of the proposed texture synthesis algorithm in resource constrained application domains.

It is observed that the algorithm proposed in chapter 6 suffers from two limitations. Firstly, no control is provided on the size of the texture pattern in relation to the geometry of a surface; secondly, no control is provided over the direction of the texture pattern to be synthesised on a surface as it is determined by the algorithm and therefore out of the control of the user. To overcome these shortcomings, in this chapter we propose a control polygon based texture synthesis algorithm on 3D surfaces. In overcoming the first limitation above the algorithm has been developed based on texture synthesis on control polygon (see section 7.1). Initially, we demonstrate the use of the proposed algorithm for texturing a single Bézier patch and two smoothly joined Bézier patches. This algorithm is later extended to overcome the second limitation by enabling the definition of the texture direction on a 3D surface (see section 7.2). We show that the proposed texture synthesis approach that uses

Bézier patches as the basic unit of texture synthesis allows the algorithm's general use in texture synthesis on many arbitrary shaped surfaces. Section 7.3 briefly discusses the texture mapping approach used. Section 7.4 provides the experimental results and analysis of proposed work and finally section 7.5 summarises and concludes the chapter.

7.1 A CONTROL POLYGON BASED TEXTURE SYNTHESIS ON BÉZIER PATCHES

This algorithm is designed to address the first problem i.e. to gain control over the size of a texture pattern in relation to the shape of the geometry of a surface. The algorithm is similar to the previous algorithm (chapter 6), but with two major changes; First, texture is synthesised on control polygon faces, Second, a random block size is used, which is of better fit to the size of the control polygon face (see figure 7.2). The detail of this algorithm is discussed below.

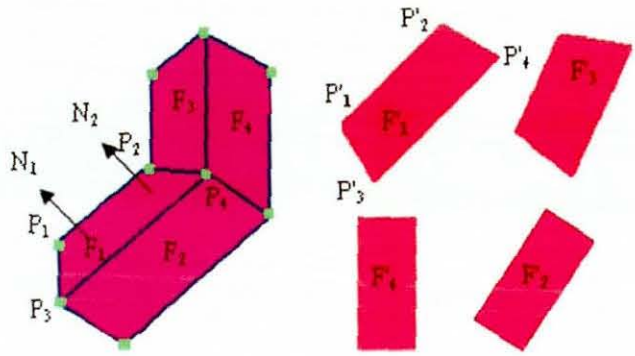
7.1.1 Projecting a 3D Face on to a 2D Plane

To synthesise a texture on a face of the Bézier control polygon and to obtain a better fit of texture, one needs to appropriately align and project the face onto the texture space. The mapping between control polygon faces and planer polygons on the texture space should be isometric, i.e. preserving both angle and distance.

The alignment and projection procedure is illustrated in figure 7.1. A given arbitrary surface F_1 (P_1, P_2, P_3, P_4) can be assumed to be comprised of two triangles i.e. $t_1 = \Delta$ (P_1, P_3, P_4) and $t_2 = \Delta$ (P_1, P_2, P_4) (see figure 7.1 (a)), with the surface normals denoted by N_1, N_2. The angle between the normal of each triangle and the normal of the 2D plane to which it is to be projected is measured initially. Subsequently the vertices of each triangle are rotated such that their normals are aligned with the normal of the 2D texture space. Finally the adjusted vertices are projected on to the 2D texture space (see figure 7.1 (b)) and the two triangles are then combined together carefully such that no overlap occurs (note: additional displacement of vertices of the shared edges of the two triangles may be required). In figure 7.1, the arbitrary Bézier surface, F_1 is

rotated/aligned and projected as F'_1 (P'_1, P'_2, P'_3, P'_4) (see figure 7.1 (b)) on the 2D texture space.

These faces are then moved to the texture space and arranged next to each other as illustrated in figure 7.1 (b). Subsequently we assign the texture space coordinates to the corresponding control polygon faces. We begin with an actual texture synthesis process which is described in the following section.



(a) (b)

Figure 7.1: 3D faces to 2D faces

7.1.2 Texture Synthesis on a Control Polygon Mesh

Figure 7.2 illustrates the texture synthesis algorithm which is an extension of our previously proposed work. The texture synthesis process starts similarly by applying 2D DWT (e.g. Haar Transform) on a sample texture image, denoted as I_{sample}.

The basic idea of the proposed algorithm is to synthesise texture on control polygons of a given Bézier patch. Let $B_{LLn(x,y)}$ represent a general polygonal block of a decomposed sample image located at position (x, y) relative to the sub-bands (p, l)'s origin. Initially we randomly pick block $B_{LLn(x,y)}$ from the sample texture image with identical size and shape to that of the control polygon face. This randomly created texture block is mapped to the control polygon surface shown in figure 7.2. [Note: the details of the mapping process are described in the following section.]

In locating the next block to be synthesised, we cut an 8 pixel wide template of pixels along the edge of already synthesised, neighbouring blocks, we then apply a 3 level DWT decomposition on the template block and extract low-resolution and diagonal detail bands, which are used for searching in sample LL3 and HH3 bands. The best overlap area is subsequently *n-level* decomposed (n=3 in our experiments), and used to find a suitable match for the adjacent block from the decomposed sample texture, by minimizing the L2 norm (see section 3.3.1.2 figure 3.8). The EZW algorithm can be used if coefficient prioritisation is used to further reduce complexity and enhance quality (see section 3.2).

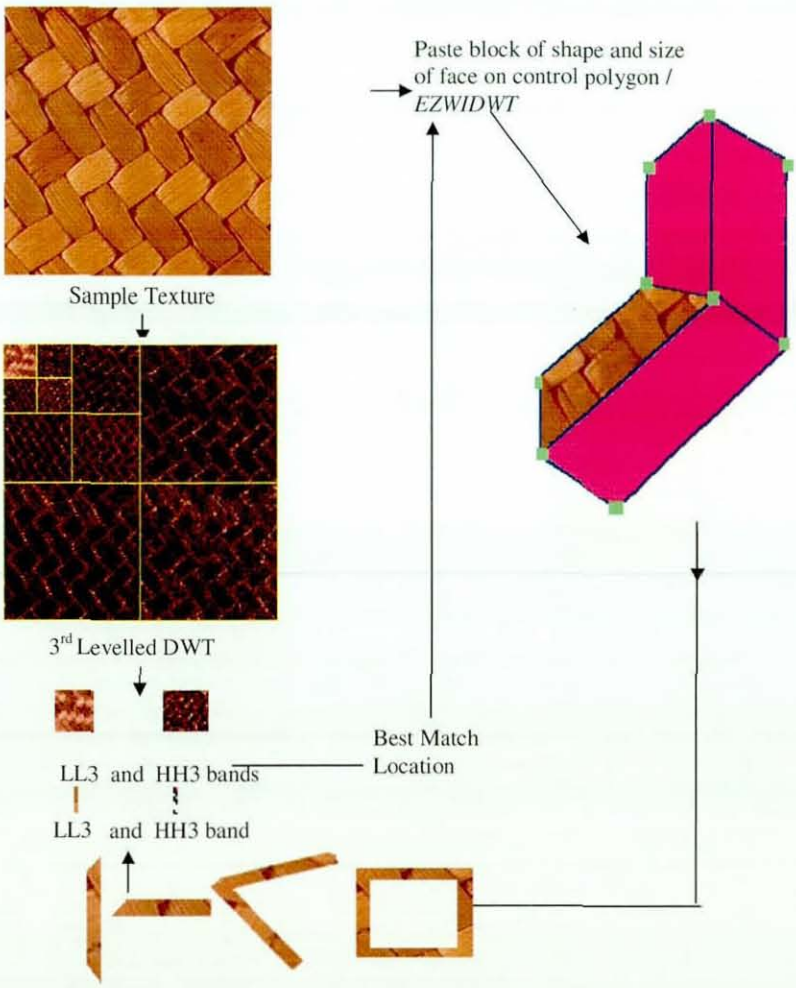


Figure 7.2: Proposed block diagram for texture synthesis on a Bézier control polygon.

Finally the overlap area of the best matching edge is blended with the overlap area on the original block using weighted edge blending. The non-overlapping area of the block is picked from the sample texture and subsequently appended to the synthesised texture. This process will continue till all the faces of the control polygon are mapped. In some cases we have considered two or more overlapping areas for finding the best match. Once texture is synthesised we project it on to a smooth surface, as discussed in the following section.

7.1.3 Texture Projection on to a Smooth Surface

In order to project texture from the control polygon to the smooth surface, initially we calculate a length/distance between each of the control points of the control polygon, using the standard distance formula between two control points in 3D space. Subsequently depending on these distances and using the correspondence between points, we calculate the relative location of projections of these points on the smooth surface, parameterised by $0 \leq s,t \leq 1$. We then decide on the area of projection from the control polygonal mesh to the smooth surface.

For clarity of presentation the projection of texture on a single dimension (i.e. a Bézier curve) can be presented as follows. Assume that curve c' (figure 7.3) is generated using four control points P_1, P_2, P_3, P_4 via the use of the Bézier curve equation 7.3. Further the lines, P_1 to P_2, P_2 to P_3 and P_3 to P_4 are assumed already textured (see figure 7.3). We calculate the length between these control points, as say l_1, l_2 and l_3. The total length $l_{T,k}$ between the control points is calculated as the sum,

$$l_{T,k} = \sum_{i=1}^{k} l_i \qquad (7.1)$$

Where l_i is the distance between two adjacent control points, and k is the total number of adjacent pairs of control points. We normalize all the control point lengths to a unit length l_i^* which is calculated as.

$$l_{T,k}^* = \frac{1}{l_T} \sum_{i=1}^{k} l_i \qquad (7.2)$$

Once the parameterization of the curve has been completed as described above, projecting the texture onto the curve becomes fairly straightforward. It was mentioned that the texture on line segment P_1 to P_2 gets projected on to the smooth curve P_1 to c'_2, where c'_2 can be calculated using the Bézier curve equation 7.3 as;

$$c_i'(l_{T,k}^*) = \frac{\sum_{j=0}^{m} b_{m,j}(l_{T,k}^*)\omega_j v_j^*}{\sum_{j=0}^{m} b_{m,j}(l_{T,k}^*)\omega_j}, \qquad 0 \leq l_{T,k}^* \leq 1, \tag{7.3}$$

where ω_j are the weights defining the specific shape of the smooth surface, v_j^* are the Bernstein vectors and $b_{m,j}(s)$ is a Bernstein polynomial. The texture projection in general is represented as follows:

$$\text{Texture } (c'(s)) \equiv \text{Texture } (p(s)) \quad 0 \leq s \leq 1 \tag{7.4}$$

Therefore in general a patch can be textured using the following equation,

$$\text{Texture } (\textit{smooth patch } (t, s)) \equiv \text{Texture } (\textit{control polygon } (t,s)) \quad 0 \leq s,t \leq 1 \tag{7.5}$$

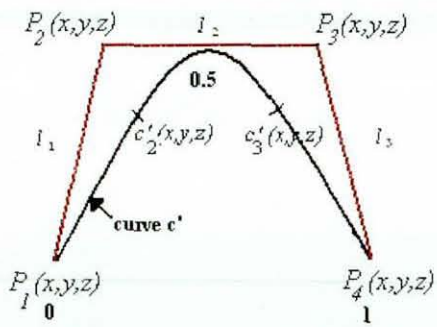


Figure 7.3: Basic projection approach

We used the above projection approach to project texture from the control polygon to a given smooth surface. (see figure 7.4(a) and 7.4(b) illustrates the mapping of texture from the control polygons onto the smooth rational surface).

Note that the closer the control polygon is to the smooth surface representation, the lesser will be the distortion in projection and vice –versa.

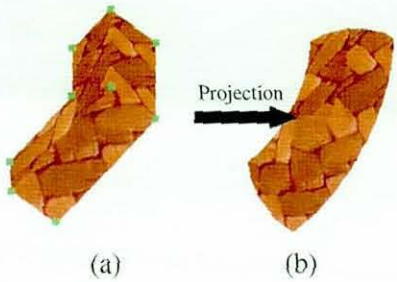


Figure 7.4: (a) Synthesised texture on the control polygon (b) Projection from control Polygon to smooth surface.

7.1.4 Experimental Results and Analysis

In order to analyse the performance of the proposed algorithm, and to show that arbitrarily shaped 3D surfaces can be rendered effectively, we have implemented the proposed algorithm in OpenGL, C++. The operating system used was Windows XP (Professional). A Pentium 4 CPU, 2.80 GHz was used with 2.00 GB RAM.

Experiments were performed on a diverse range of texture samples that include regular, near-regular, irregular and stochastic (Lin et al., 2004) textures. Results illustrated in figure 7.5 indicate the ability of the proposed technique to efficiently map and synthesise texture on surfaces, with minimal artefacts. Furthermore, as matching and searching is performed in the wavelet domain, the texture synthesis is fast. Further experimentation revealed that the time required to synthesise these textures is in the range of a few milliseconds.

To further extend the functionality of the proposed method, we have extended our work to progressive texture synthesis on surfaces. We have preformed a wide range of experiments (see figure 7.6 & 7.7) to show that texture can be synthesised at seamlessly different levels of quality on surfaces, without consuming noticeable processing time. Figure 7.6 (c) illustrates the synthesis of a stochastic texture of a flower. It is evident from the results that only 10% of the information from the sample texture is sufficient to create a texture with sufficiently rough quality. By increasing the percentage of coefficients further, the quality of the synthesised texture can be seamlessly improved. Further experiments revealed that for this texture, 20% of the

coefficients were sufficient to synthesise a texture visually equal to the texture that can be synthesised when all coefficients are utilized. Progressive texture synthesis gives the added advantage of being able to truncate a bit stream representing the sample texture at any intermediate stage, still being able to synthesise texture at some intermediate quality level.

To further illustrate the application of the proposed idea, we have extended our approach to synthesizing texture on two smoothly joined rational Bézier surfaces, shown in figure 7.7. Figures 7.7(a1, b1, c1) to 7.7(a4, b4, c4) illustrate progressive texture synthesis on this surface using three different types of texture. This proves that the proposed technique can be extended to cover texture synthesis on many different geometric topologies. The results in figure 7.7 further illustrate that when using regular and near-regular texture samples, texture variations across patch boundaries are smooth.

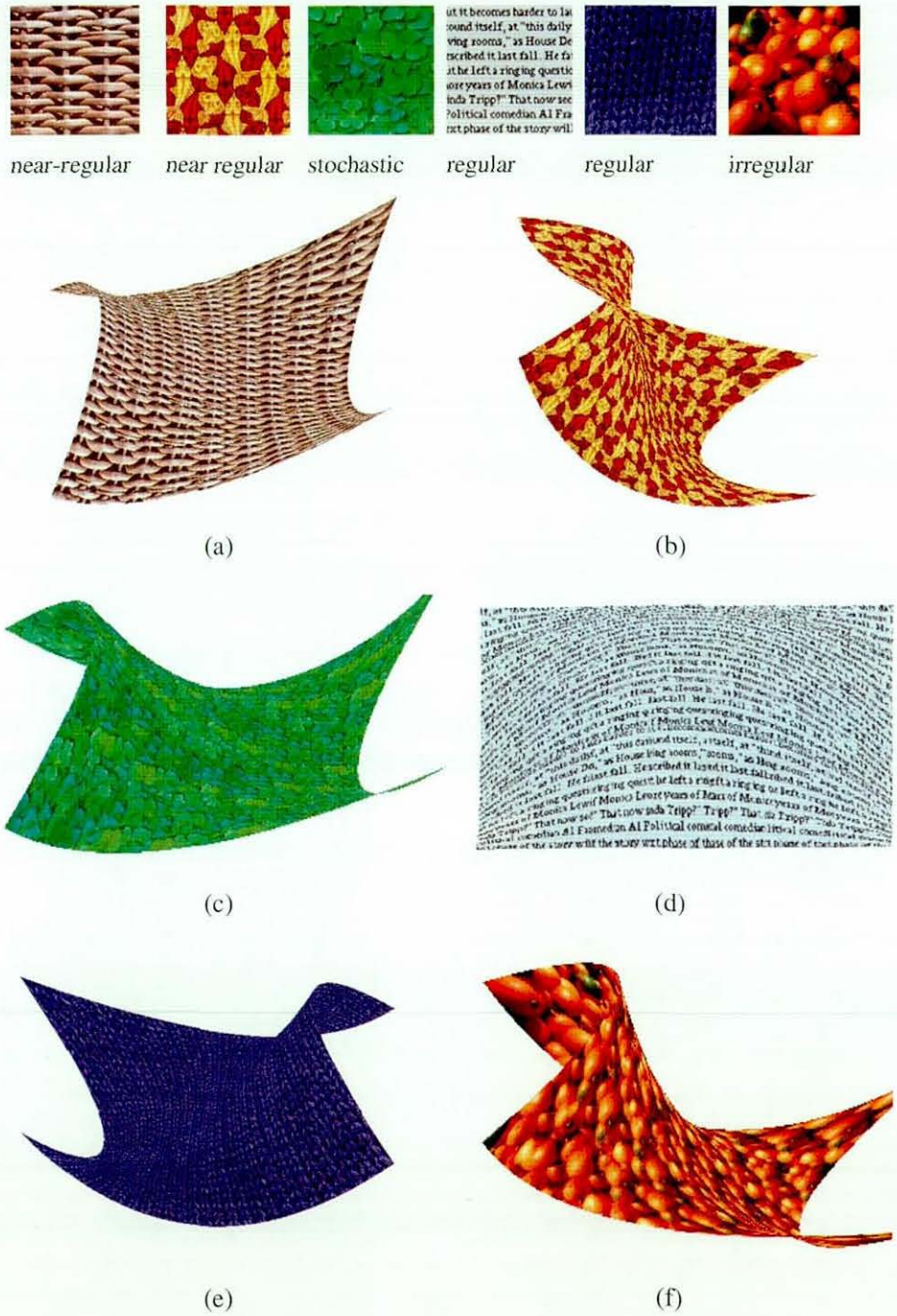


Figure 7.5:Texture synthesis on a biquadratic surface

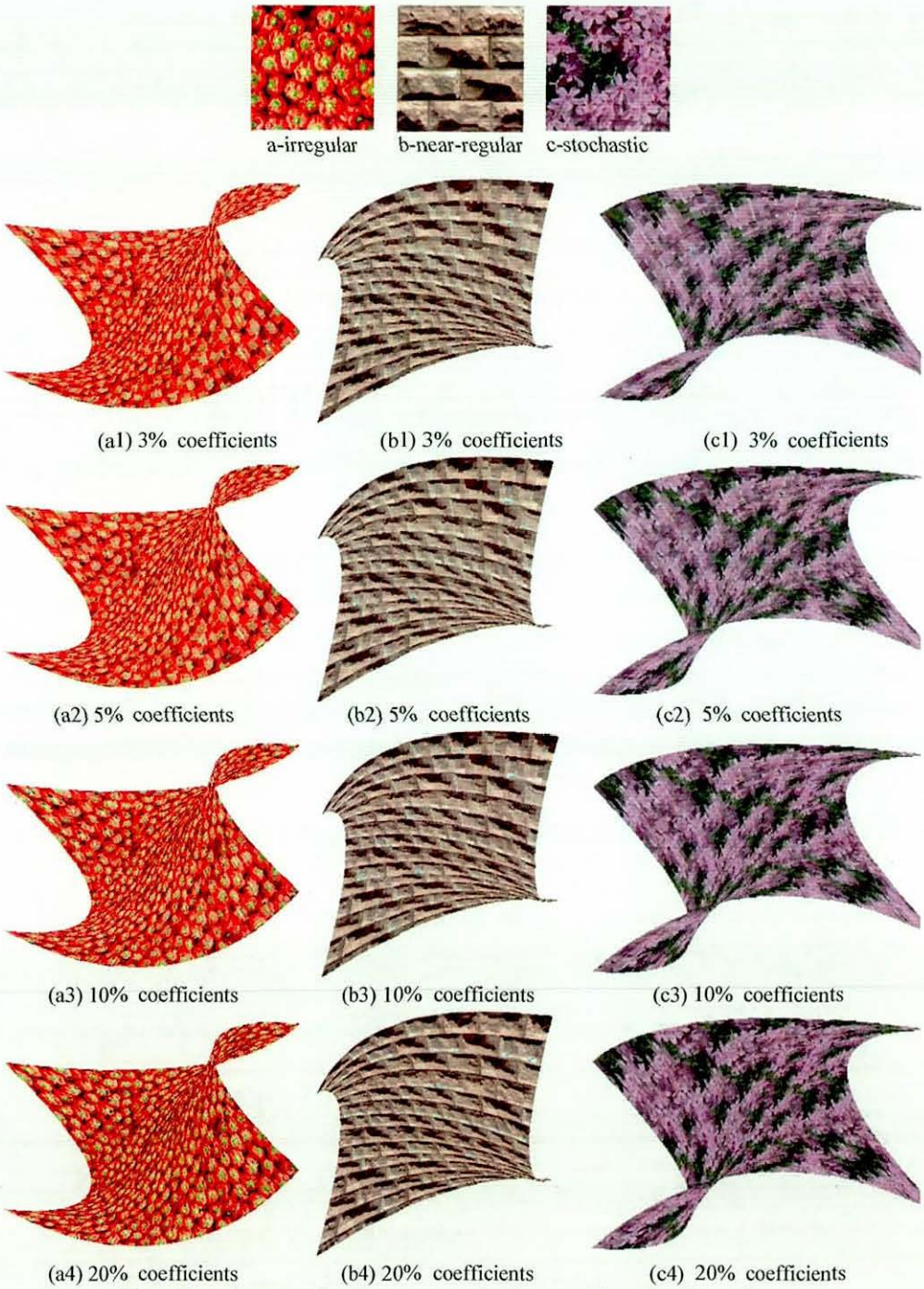


Figure7.6: Progressive texture synthesis on a biquadratic surface.

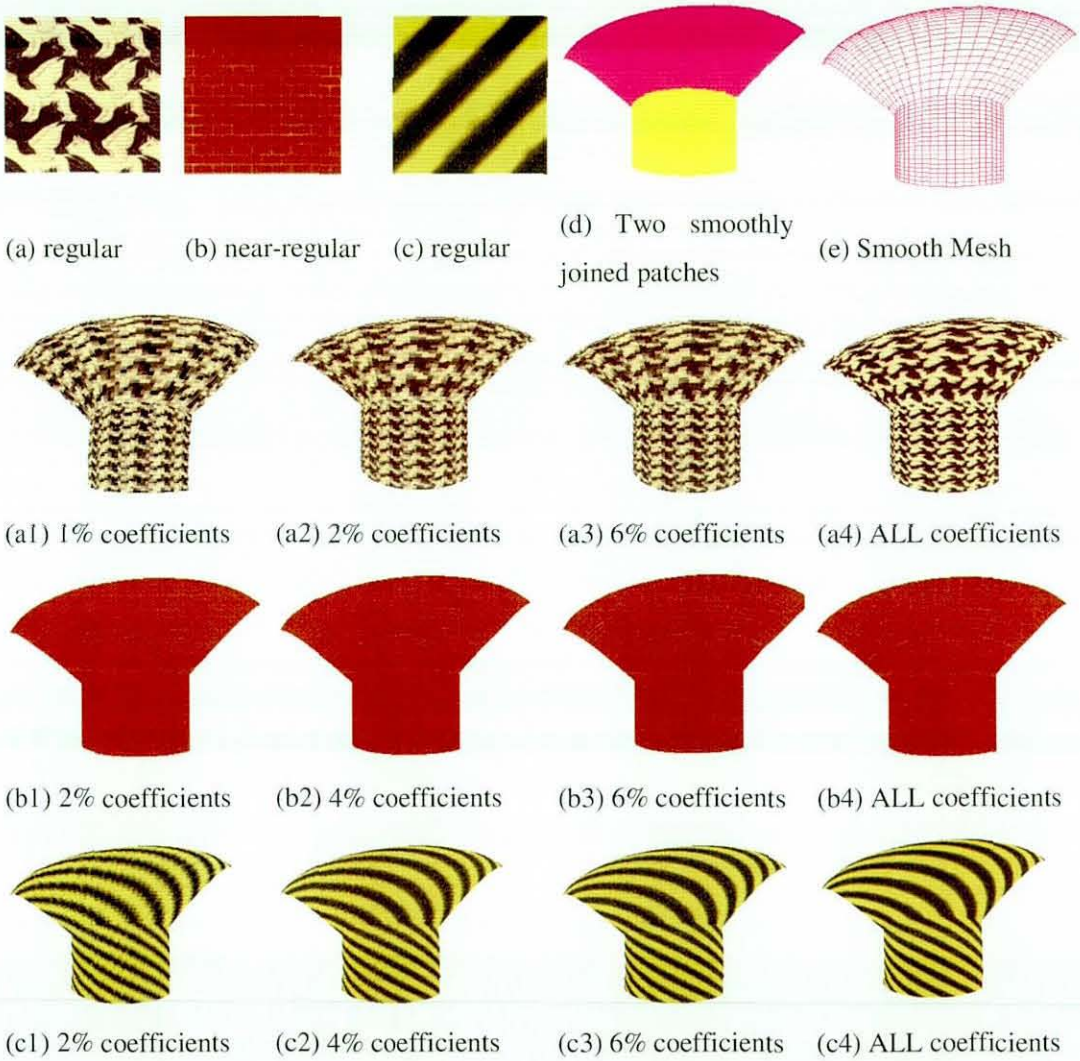


Figure 7.7: Progressive texture on two, connected biquadratic surfaces

7.2 A CONTROL POLYGON BASED TEXTURE SYNTHESIS ON BÉZIER 3D MODELS

This algorithm is designed to address the second limitation i.e. to gain control over the direction of a texture pattern over a surface and to provide the complete GUI for a user in order to mark the vector direction on a given surface. The algorithm is an extension of the previous algorithm (chapter 7.1) which includes all previous features. The algorithm is further generalised to cover many surface topologies. The details of the algorithm can be discussed as follows:

7.2.1 An overview

The basic logical flow of the proposed algorithm is illustrated in figure 7.8. The process begins with the reading of the Bézier control polygonal information from the Bézier Patch Format file (see section 7.2.2) of the 3D surface. Subsequently the user interactively marks the required texture directions on the control polygons in the form of vectors (see section 7.2.3). All faces of the control polygons are then projected on to a 2D plane (see section 7.1.1). The user defined vector fields representing the expected texture directions along with the polygonal faces gets projected on to the 2D plane (7.2.3). The first marked user defined vector is considered the anchor vector and is used to initiate vector propagation (see section 7.2.4). It is noted that the time taken for vector propagation depends on the time spent by the user in manually marking the vector fields.

Once the above process has been completed, the texture synthesis commences, where each of the Bézier control polygonal faces are synthesised (according to sections 7.1). Finally, the synthesised textures on control polygons are projected back onto the actual Bézier patches representing the 3D surface (section 7.1.3), thus completing the texture synthesis process.

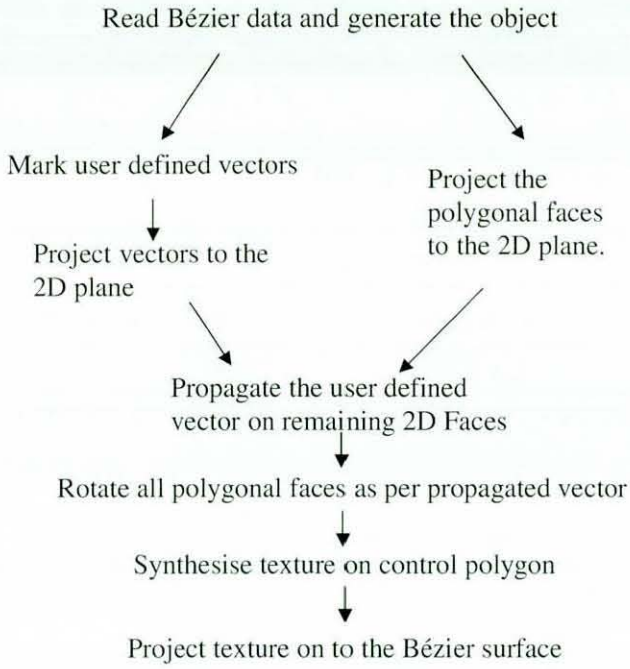


Figure 7.8: Logical flow of the proposed algorithm

The following sections provide further design and implementation details of each sub process of the proposed texture synthesis approach.

7.2.2 Bézier Patch File Format[120]

The BPT file format is a very simple text file format. The first line contains the information of the number of patches in the file. Each patch record is followed immediately by the next, and consists of the following items: Two integers, u and v representing the dimensions of the patch (e.g. 2, 2 for biquadratic and 3, 3 for a bicubic patch); one control point per line, for u * v lines (e.g. 16 control points for the bicubic and 9 for the biquadratic). The control points are four double values, separated by spaces. The first three double values are x, y, z and fourth double value is the weight. Control points move along the first dimension are specified first, then the second. Therefore, for a "1, 2" patch, the first two would be the first edge, the next two would be the centre, and the final two would be the far edge of the particular

Bézier patch. For ease of understanding, the format of the file can be summarised as follows:

Example of file format:

<n - number of patches>

$<u_1> <v_1>$

$<x_{0\,0}> <y_{0\,0}> <z_{0\,0}> < w_{0\,0}>$

$<x_{1\,0}> <y_{1\,0}> <z_{1\,0}>< w_{1\,0}>$

.

$<x_{u-1\,0}> <y_{u-1\,0}> <z_{u-1\,0}>< w_{u-1\,0}>$

$<x_{0\,1}> <y_{0\,1}> <z_{0\,1}>< w_{0\,1}>$

.

$<x_{u-1\,v-1}> <y_{u-1\,v-1}> <z_{u-1\,v-1}>< w_{u-1\,v-1}>$

$<u_2> <v_2>$

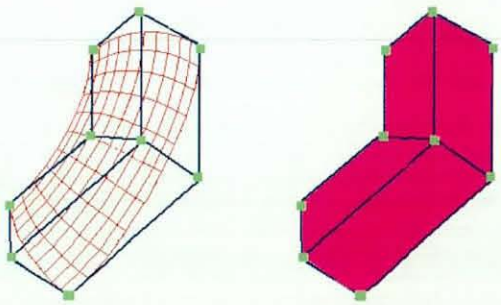
.

$<u_n> <v_n>$

.

$<x_{u-1\,v-1}> <y_{u-1\,v-1}> <z_{u-1\,v-1}>< w_{u-1\,v-1}>$

The following (see figure 7.9) is an illustration of a rendered patch. For the given nine control points, the biquadratic surface patch defined by them has been computed and drawn.



(a) (b)

Figure7.9: (a) Green Colour: Biquadratic control polygonal points, Red Colour: The smooth surface mesh generated using control polygons (b) Control polygonal mesh (note: for 9 control points, 4 faces are generated.).

7.2.3 User Defined Tangential Vectors and their Propagation

This is the most frequently used approach in maintaining realism in 3D texture synthesis. It allows the user to interactively specify texture directions (user defined tangent vectors) on selected patches, which are (i.e. the directions) subsequently propagated to provide the texture directions of all other patches defining the surface.

In our approach the user initially marks texture directions on a selected number of faces (note: minimum of one face is required). The algorithm then considers one of the marked faces (usually the one selected first) to be a 'seed face' and will select a neighbouring non-marked face to commence vector propagation.

The vector $T_{f,c}$ of a non marked face, c, is calculated as follows:

$$T_{f,c} = (\frac{1}{k})\sum_{i=1}^{k} T_{f,i} \qquad (7.6)$$

where $T_{f,c}$ is the vector of the non-marked face (i.e. the field being calculated) and $T_{f,1}, T_{f,i+1}, \ldots T_{f,n}$ represent the vector fields of the marked faces, from the k-connected neighbourhood. (note: in our experiments, k = 8). Starting from a neighbouring block of the seed face we use the above equation to propagate the texture field to cover all non-marked faces of the 3D object.

Once $T_{f,c}$ is obtained using equation 7.6, it may not be parallel to the face. Therefore we project the vector field into the plane of the face, $T_{f,c}^{*}$, as follows:

$$T_{f,c}^{*} = T_{f,c} * \alpha_1 + T_{f,c} * \alpha_2 \qquad (7.7)$$

where,

$$\alpha_1 = b_1 \bullet T_{f,c} \qquad \alpha_2 = b_2 \bullet T_{f,c} \qquad (7.8)$$

and b_1, b_2 are the face basis vectors i.e., they lie in the plane of the face. The vector fields thus obtained are used for the appropriate rotation of the face after being

projected onto the 2D plane, which is discussed in the next section. This results in minimizing distortion at the patch boundaries.

7.2.4 Project 3D Face on to 2D Plane

To synthesise a texture on a Bézier surface, one needs to appropriately align and project all faces onto the texture space, which is discussed in section 7.1.1.

The alignment and projection procedure is illustrated in figure 7.10. In figure 7.10, the arbitrary Bézier surface, F_1 is rotated/aligned and projected as F'_1 (P'_1, P'_2, P'_3, P'_4) (see figure 7.10 (b)) on the 2D texture space. In carrying out the above explained projections, if a given Bézier polygon has been assigned a user defined vector field, it is appropriately transferred to its projection in the texture space figure 7.10 (b)). Subsequently following the vector field propagation procedure explained in section 7.2.3, the user defined, projected vector fields are propagated to obtain the vector fields of all projected polygons. All projected faces are finally rotated so that their vector fields (i.e. vectors that define their intended texture direction) are aligned with the y-direction of the texture space (see figure 7.10 (c) and (d), i.e., F'_1 is rotated to F''_1 (P''_1, P''_2, P''_3, P''_4) in order to obtain a desired, realistic texture pattern on to the surface.

These faces are then moved to the texture space and arranged next to each other as illustrated in figure 7.10 (d). Subsequently we assign the texture space to the corresponding control polygon faces. We begin with the actual texture synthesis process which is described in the section 7.1.2.

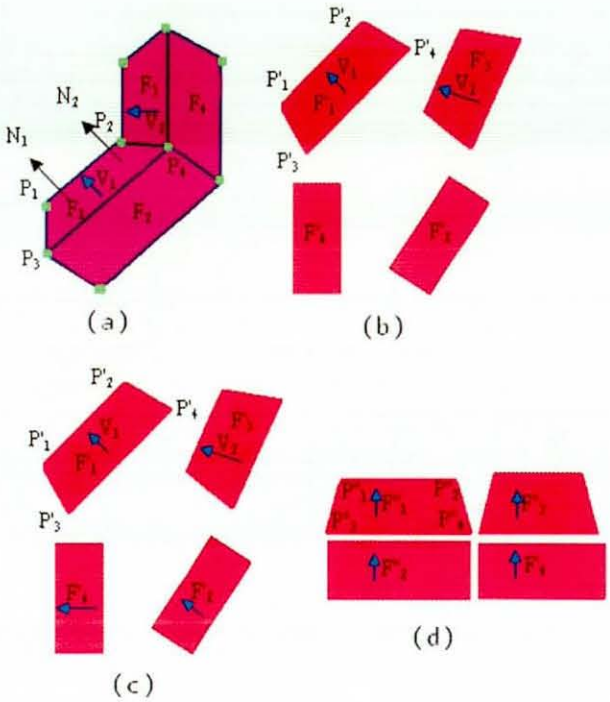


Figure 7.10: 3D faces to 2D faces

User defined vector direction and propagation gives smooth flow of texture patterns on a surface. However in general mismatches of texture direction on individual patches may result in the formation of a seam in the patch overlapping area. To overcome this problem, we have proposed a weighted blending function which changes the blending parameter as the direction is changed. This is described in the following section 7.2.5.

7.2.5 Weighted Edge Blending based on Direction

Blending in 2D texture synthesis is straightforward as compared to the 3D equivalent. When blending texture in 3D, the direction of texture blending should be changed as the direction of the synthesis process is changed. This is considered as a key challenge.

Texture synthesis over 3D faces mostly depends on the user defined vector field and propagation of vectors. For any quadrilateral face there are four directions of blending (see figure 7.11).

Figure 7.11: Four direction of blending

Experiments have shown that obtaining a 100% accurate match as a neighbouring face at all times is rare. Therefore some seams will appear along the patch edges if left unprocessed. To improve blending and to avoid any artefacts, we have designed the following weighted edge blending function.

The general idea of weighted edge blending is based on gradually reducing the luminance contribution made by a neighbouring face to its seams, whilst moving away to an adjoining face across seams. In our proposed algorithm weighted edge blending is carried out in the pixel domain.

Let W represent an empirically defined matrix capable of performing weighted edge blending in the pixel domain, via a matrix multiplication with the overlap area. [Note: the top left hand corner coefficient of W is a calculated value and is an approximation to 0]. The size of W depends on the width of the overlap. In our experiment we have assumed that the width of overlap is eight pixels and have therefore defined W to be of size, $w \times w$ *(w=width).*

$$W = \begin{bmatrix} 0.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/w & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2/w & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \bullet & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \bullet & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \bullet & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (w-2)/w & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (w-1)/w \end{bmatrix} \qquad (7.9)$$

Let the overlapping areas of two adjoining faces, i.e., the edge of the first face be denoted by $Edge_{1olap}$ and matched edge of the second face be denoted by $Edge_{2olap}$. The resulting blended overlap area, $O_{blendolap}$ can be represented by,

$$O_{blendolap} = Edge_{1olap} \times W + Edge_{2olap} \times (I_{ident} - W) \qquad (7.10)$$

Where I_{ident} is an identity matrix. $Edge_{1olap}$, $Edge_{2olap}$ are of size *w x height.*

Figure 7.12: Swap direction of blending

Figure 7.12 illustrates the blending of three matching patches. Let's assume that all faces have already been textured. When considering the blending of texture of patch F_1 to that of patch F_3 we use Eq (7.10) as the blending function, where the area d_4 of patch F_3 overlaps the area d_3 of patch F_1. However when patching F_2 to F_1, the area depicted d_3 overlaps the area depicted d_4 of patch F_1. In this situation it is important to decide whether w should multiply $Edge_{1olap}$ or $Edge_{2olap}$ in eq. 7.10. This decision depends on the face which is to be overlapped, which is decided on the basis of whether the pixel value of the inside edge of an already synthesised texture is to be changed or to be maintained the same. If it is to be changed then we swap the blending direction around and the equation changes to,

$$O_{blendolap} = Edge_{1olap} \times (I_{ident} - W) + Edge_{2olap} \times W \qquad (7.11)$$

This switch over helps in maintaining a smooth texture pattern on the surface. Further the blending function gets swapped around as the blending direction changes, while the surface is being synthesised. Figure 7.13(a) shows the texture synthesised on a

surface without the use of blending and figure 7.13(b) shows the texture synthesised 3D object when weighted edge blending as described above, has been used.

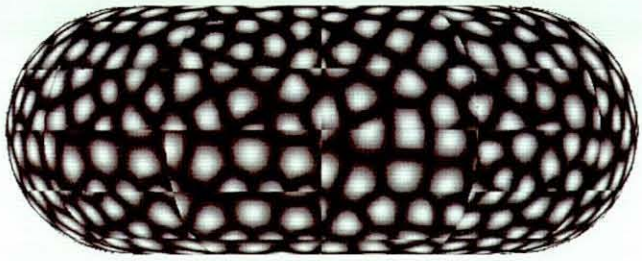


Figure 7.13 (a): Texture without blending

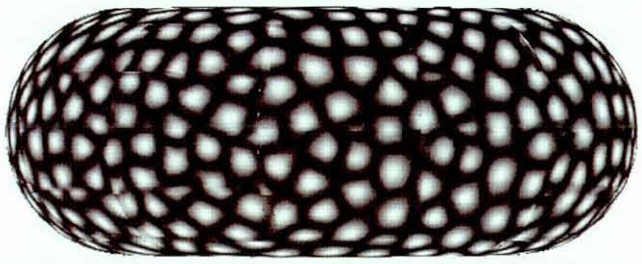


Figure 7.13 (b): Texture with swap blending

7.3 Texture Mapping on Control Polygons

In our implementation we have adopted a strategy in which we initially create propagating seed vector directions, which are subsequently used to smooth the surface vector field. However alternatively a number of other surface vector field generation techniques (Wei-Levoy [60]; Turk [62]; Ying et al., [63]) can be used to replace the approach we have selected above. Once vector fields are assigned to all control polygonal faces, we rotate all the faces according to their tangential vector field and surface normal, thus placing all faces in the same 2D plane. Using a modified version of Soucy et al.'s, 1996 [119] approach, (note: modified from using triangles to using polygons) a texture map T is created. For each face of the control polygon, we calculate the bounding box of the face and then map it to a corresponding face in *T* in a compact form, i.e. with no space being wasted. The faces in T are textured using the corresponding best matching block. The faces in T that we use are of non-uniform size, which are a better fit to the shape and size. It is noted that the resulting texture

can be rendered on the control polygon surfaces at interactive rates. The images illustrated in section-4 (See figure 7.14, 7.15 and 7.16) were rendered in this manner using 256 x 256 textures. The models used in our experiments are composed of smooth surfaces having between 1000 to 20000 faces, whereas the control polygons used consisted of 100 faces to 400 faces (It can be further increased to an arbitrary number of faces.) We have observed that these surfaces render at real-time rates.

7.4 EXPERIMENTAL RESULTS AND ANALYSIS

In order to analyse the performance of the proposed algorithm, and to show that surfaces can be rendered effectively, we have implemented the proposed algorithms using OpenGL, C, and C++. The operating system used was Windows XP(Professional). A Pentium 4 CPU, 2.80 GHz was used with 2.00 GB RAM.

Experiments were performed on a diverse range of texture samples that include regular, near-regular, irregular and stochastic [Lin] textures. Results illustrated in figures 7.14, 7.15, 7.16 and 7.17 clearly indicate the ability of the proposed technique to efficiently map texture with minimal visual artefacts on arbitrary shaped 3D surfaces that include the Utah Teapot, Cup and a number of 3D geometric surfaces such as a Cyclide, Torus and a Sphere.

In particular, figure 7.14 illustrates two examples each of texture synthesis on a 3D surface (i.e. the Cup) for two regular textures. Note that each synthesis on the *Cup* using a given texture relates to a different vector direction. The results illustrated clearly demonstrate that visually appealing textures can be flexibly synthesised via the use of a vector field, when using the proposed approach.

Figure 7.15 illustrates the complete cycle of the proposed texture synthesis process on a Bézier surface. The associated stages are labelled in figure 7.15 (a)-(d).

Figures 7.16 and 7.17 respectively illustrate synthesis results of a wide variety of texture samples on arbitrarily shaped and geometric, 3D surfaces, respectively [the zoomed result is presented in Appendix B]. Tables 7.1 and 7.2 tabulate the number of decomposition levels of the DWT that was used to generate the texture synthesis

results for each 3D object. Note that it is efficient to use the appropriate number of decomposition levels of DWT for different textures as the amount of detail required for good quality texture synthesis differs from texture pattern type to type. For example our experiments revealed that stochastic textures can be effectively synthesised using the sub-bands of the 3rd level of DWT whereas irregular textures require the sub-bands of the 2nd level of transform and near-regular, regular textures require sub-bands of the 1st level of transform (see tables, 7.1 and 7.2). However further analysis of the results tabulated in Table 7.1 and 7.2 revealed that there is no conclusive evidence between the regularity of texture and how many decomposition levels should be used.

Figure 7.14 (a, b): Texture patterns synthesised over the 3D surface, *Cup,* which depends on the direction of the marked tangential vector.

The synthesis speed varies not only based on the number of patches that cover the 3D shape but also on the nature of the texture sample.

Model	Texture Pattern	No of Patches	Level *n*
Teapot	Regular (figure 7.16 (a))	32	2
Teapot	Near-Stochastic (figure 7.16 (b))	32	3
Teapot	Irregular (figure 7.16 (c))	32	2
Teapot	Near Regular (figure 7.16 (d))	32	1
Cup	Irregular (figure 7.16 (e))	26	2
Cup	Stochastic (figure 7.16 (f))	26	3
Cup	Near-Stochastic (figure 7.16 (g))	26	3
Cup	Regular (figure 7.16 (h))	26	2

Table 7.1: Arbitrary surface

Model	Texture Pattern	No of patches	Level *n*
Torus	Near-Stochastic (figure 7.17 (a))	16	3
Torus	Irregular(figure 7.17 (b))	16	2
Torus	Irregular (figure 7.17 (c))	16	2
Torus	Near-Stochastic (figure 7.17 (d))	16	3
Cyclide	Stochastic (figure 7.17 (e))	16	3
Cyclide	Irregular (figure 7.17 (f))	16	2
Sphere	Stochastic (figure 7.17 (g))	8	3
Sphere	Regular (figure 7.17 (h))	8	2

Table 7.2: Geometric surface

To further extend the practical applicability of the proposed method, we have extended our work to progressive texture synthesis on 3D surfaces. We have performed a wide range of experiments. Figure 7.18 and 7.19 illustrate that using the proposed approach, texture can be synthesised at seamlessly different levels of quality on 3D surfaces. Figure 7.18 and 7.19 illustrate the synthesis of an irregular texture of

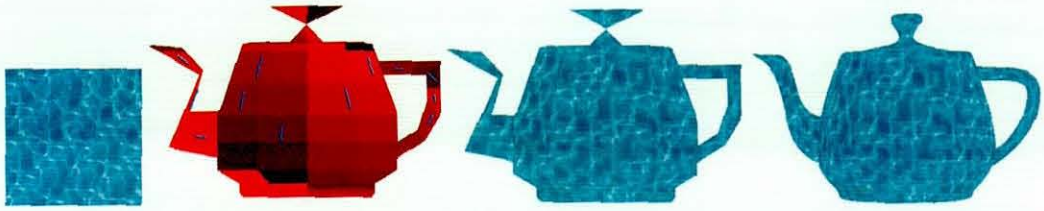
an *animal skin* pattern on the Utah Teapot. It is evident from the results that only 20% of information (i.e. DWT coefficients) from the sample texture is required to create texture with sufficiently good quality, i.e. equivalent to which could be achieved when using all of the texture information. It is observed that by increasing the percentage of the coefficients used in texture synthesis, the quality of the texture can be seamlessly improved. Due to space limitations we have restricted the number of discrete texture results to eight, whereas the proposed method is capable of progressively synthesising texture at seamlessly different number of levels. It should be noted that progressive texture synthesis gives the added advantage of being able to truncate a bit stream representing the sample texture at any intermediate stage still being able to synthesise texture at some intermediate quality level. Further, the texture synthesis results on arbitrarily shaped 3D objects prove that our technique can be extended to many surface topologies.

7.5 SUMMARY AND CONCLUSION

This chapter presented a novel DWT based approach for synthesizing texture on a 3D surface, which is made of an arbitrary surface defined using Bézier patches. The idea is to achieve good quality progressive texture synthesis on 3D surfaces, without being limited by its geometrical shape. None of the previously proposed algorithms have investigated the use of Bézier surfaces for texture synthesis. In other words Bézier surfaces have been widely overlooked by the texture synthesis community. However Bézier surfaces have various positive properties such as, animation flexibility and compact representation of vertices, which required less storage and transmission space. This is the first attempt taken to prove the feasibility of texture synthesis on Bézier surface using control polygonal information. The proposed method has the capability of synthesizing texture at seamlessly different quality settings. This functionality is not possible via existing state-of-the art techniques. The texture synthesis process is reliant on the user interaction, where the user sets few vector fields on the control polygon of the Bézier patches. These set vectors are then used to propagate texture all over the surface. The initial texture is synthesised on a control polygon. The synthesised texture is then projected on a smooth surface. This approach gives further gain in the processing texture on the surface.

The use of visual prioritisation of information in the sample image during texture synthesis allows the task to be carried out at a higher speed but at an equivalent visual quality level. We show that the proposed approach is computationally efficient and results in good quality texture synthesis. Further due the ability to prioritise visually significant coefficients, we have discussed the proposed method's possible use in bandwidth-adaptive/compressed-domain applications such as remote visualization. We have shown that the control polygon strategy used can be extended to cover synthesizing texture on many 3D objects with arbitrary surface topologies.

We have used a Dupin Cyclide [6, 15], a Sphere and a Torus surface as examples to demonstrate texture synthesis on geometric surfaces. These surfaces are widely used in computer graphics and in the development of a number of CAD tools.



(a) (b) (c) (d)

Figure 7.15: Complete cycle of the proposed texture synthesis process, (a) The sample texture (b) User defined, marked vectors (c) Texture synthesis on the control polygons of the 3D model (d) Smoothly projected texture on the actual surface

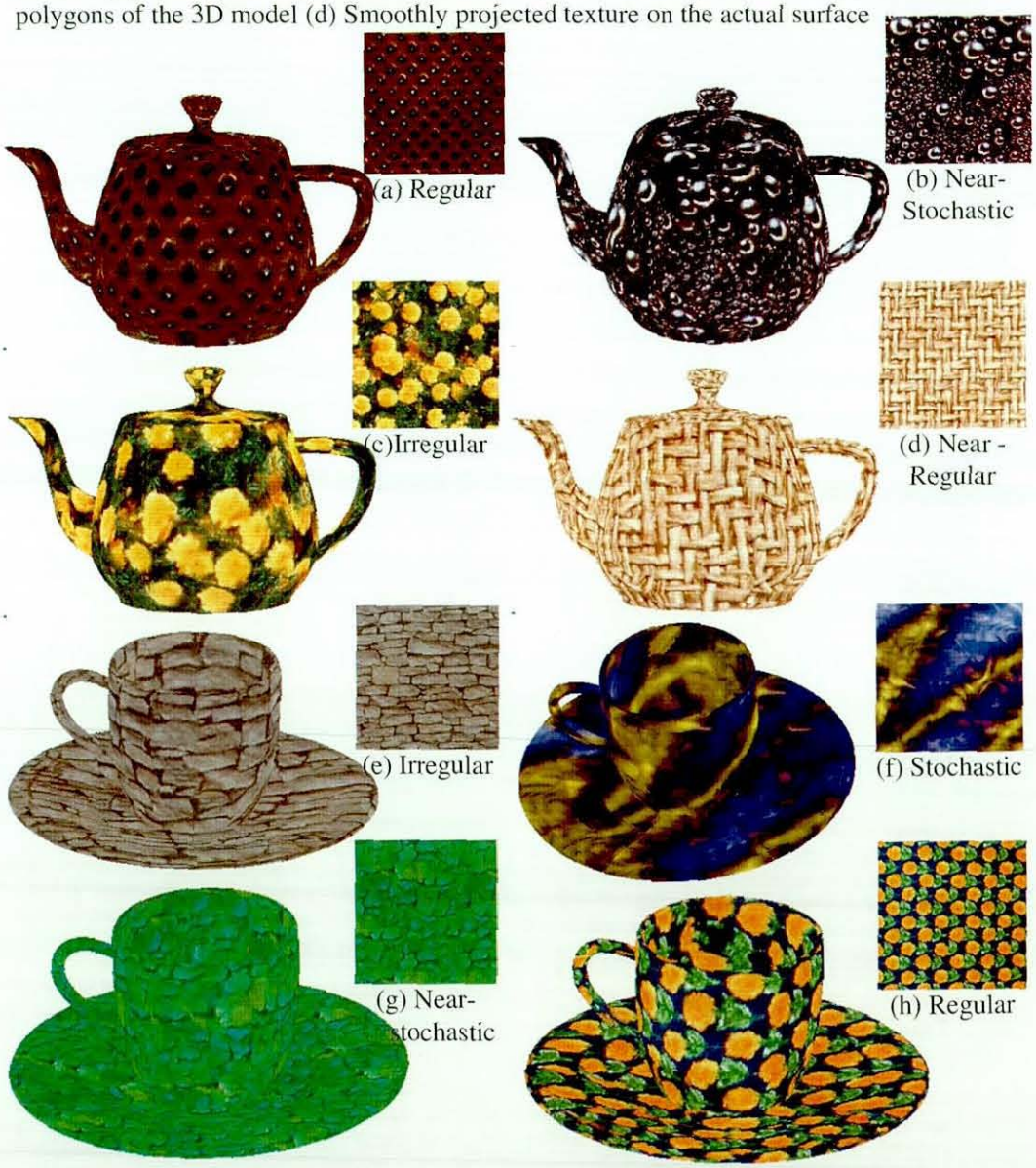
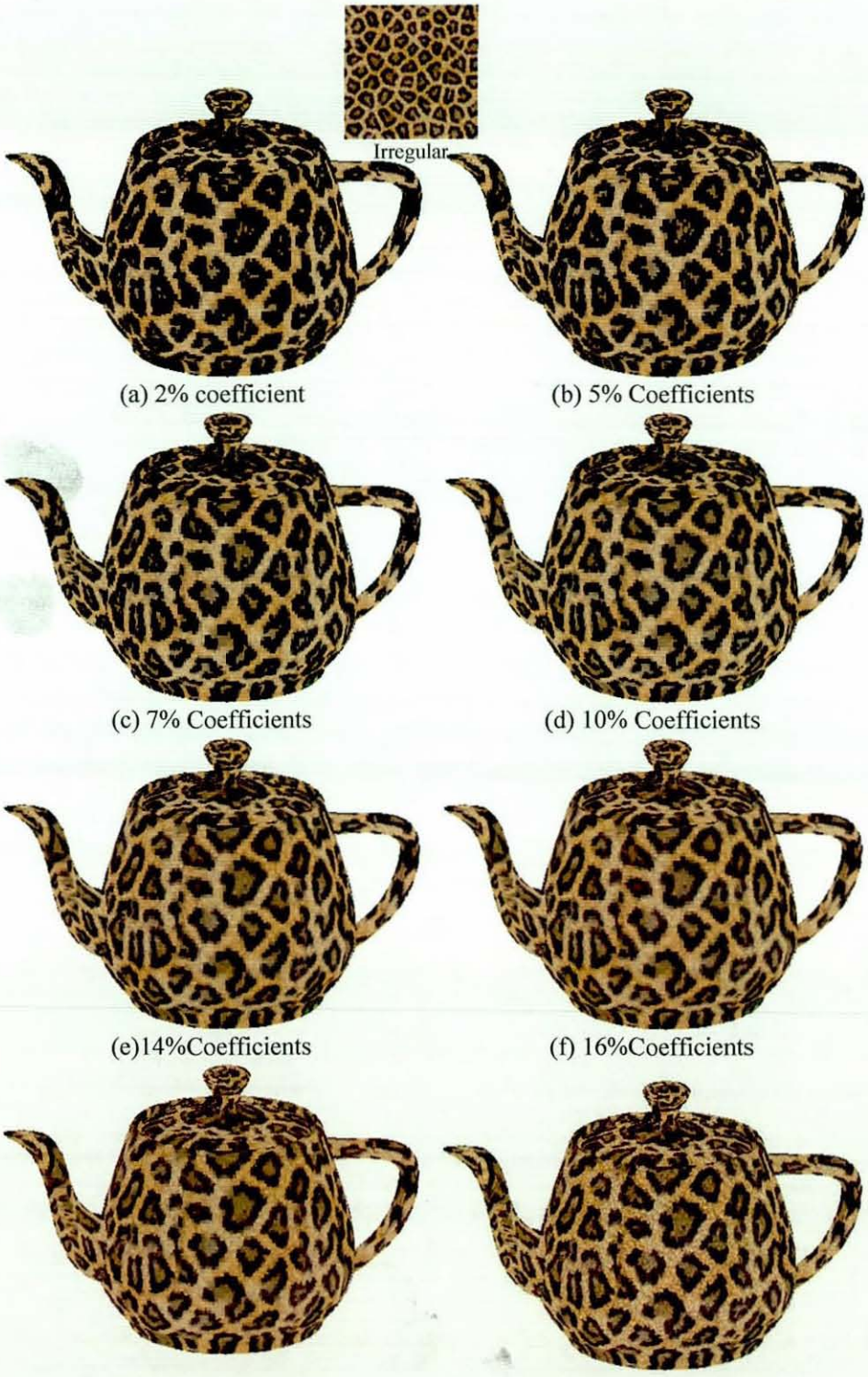


Figure 7.16: Texture synthesis on Teapot and Cup

Figure 7.17: Texture synthesis on a Torus, Cyclide and a Sphere



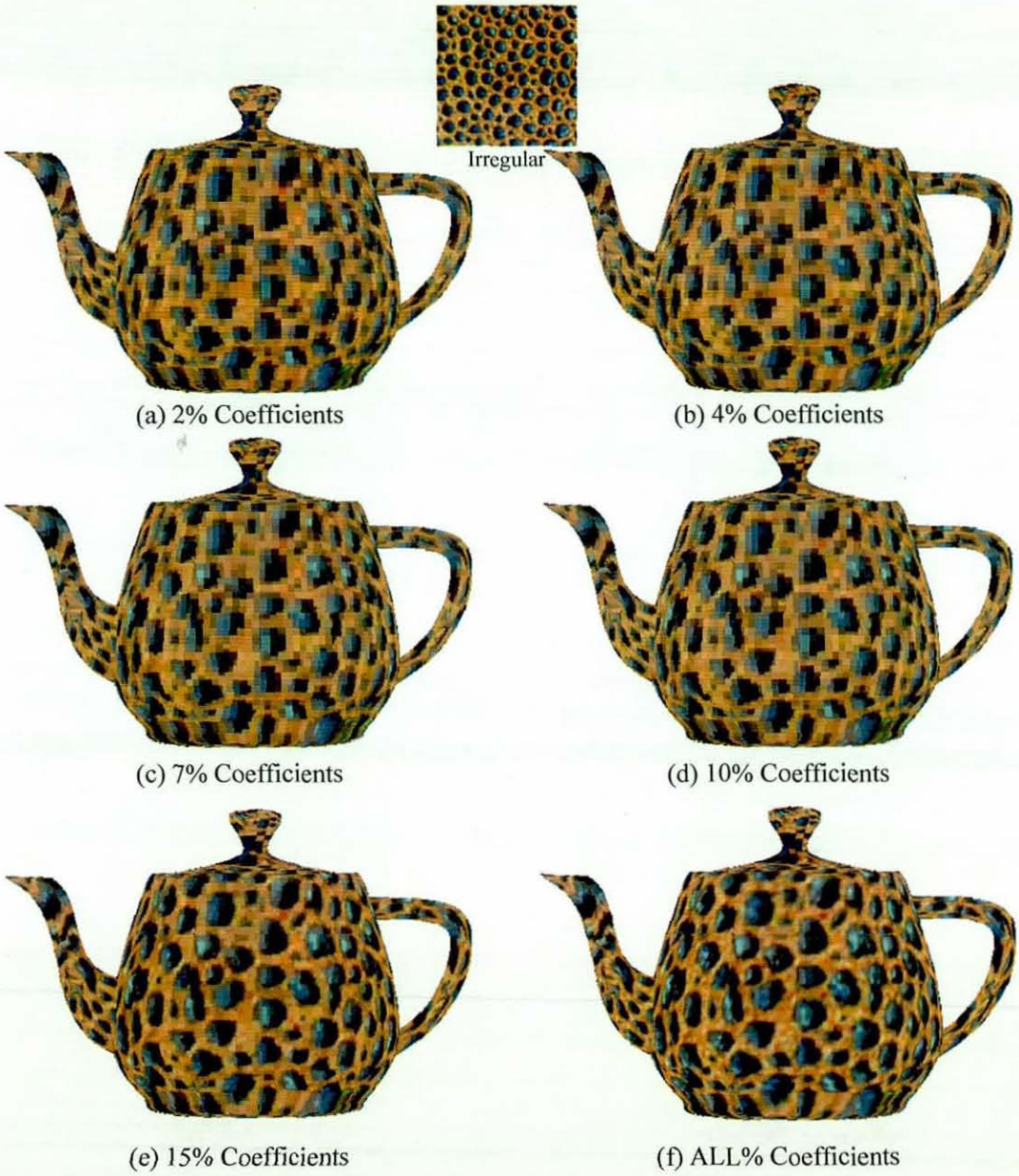
(a) 2% coefficient (b) 5% Coefficients

(c) 7% Coefficients (d) 10% Coefficients

(e)14%Coefficients (f) 16%Coefficients

(g) 18%Coefficients (h) All Coefficients

Figure 7.18: Progressive texture synthesis on the Teapot



(a) 2% Coefficients (b) 4% Coefficients

(c) 7% Coefficients (d) 10% Coefficients

(e) 15% Coefficients (f) ALL% Coefficients

Figure 7.19: Progressive texture synthesis on the Teapot

Chapter 8

Conclusion and Future Research

This chapter summarizes the key ideas presented in chapters 4, 5, 6 and 7, draws conclusions and emphasizes the important contributions made by the research presented in this thesis. It also gives an insight into possible future directions of research, particularly with the intention of further extending the functionality and efficiency of the proposed algorithms.

In summary, the thesis presents five original contributions to the state-of-the-art in texture synthesis. The first contribution was an extension to the pixel domain 2D texture synthesis algorithms of Turk [62], so that it can be used for progressive texture synthesis. The second contribution was an extension to Wickramanayake et al.'s patch based, transform domain texture synthesis approach [5, 39]. In this effort the non-optimal patch quilting approach adopted by Wickramanayake et al. was replaced by a optimal, transform domain extension of the popular max-flow, min-cut graph-cut algorithm[36] effectively exploited in pixel domain in Kwatra et al.'s texture synthesis algorithms on planar (2D) surfaces [36]. It is noted that Turk's, Wickramanayake et al.'s and Kwatra's texture synthesis algorithms have been used as benchmarks to evaluate the performance of the proposed improvements and extensions to the original algorithms. Further, the thesis reports a third original contribution in the design and implementation of a modified and improved version of Wickramanayake et al.'s transform domain texture synthesis idea in texture synthesis on Bezier surfaces. It was noted that the texture synthesis on Bezier surfaces enables the idea to be extended to texture synthesis on any geometric surface giving a compact representation of surface texture and the flexibility of random surface shape changes. With the fourth original contribution this work was further extended to texture synthesis on meshes representing 3D geometric or arbitrarily shaped objects. The final contribution concentrated on texture synthesis on control polygons that were

subsequently matted into the smooth surfaces of the corresponding 3D objects. This further implemented the means of specifying the direction of texture flow by a user.

Chapter 4 presents the first contribution in which G. Turk's popular, pixel domain texture synthesis algorithm was converted into a transform domain texture synthesis algorithm that enables progressive texture synthesis. It was shown that the multi-resolution DWT domain representation of the sample texture and the use of the so-called EZW ideas in prioritisation of DWT coefficients enable effective progressive texture synthesis. The novel algorithm was shown to be capable of synthesizing textures of any progressive texture quality using a embedded bit stream that represents the sample texture, as compared to the fixed number of pre-selected quality levels to which the original Turk's algorithm was limited. Further it was shown that a perceptually equivalent level of output texture quality can be achieved using a lesser amount of textual information (i.e. a subset of perceptually important DWT coefficients) resulting in a faster texture synthesis as compared to the original algorithm. It was discussed that the progressive texture synthesis capability improves the algorithm's potential usability in remote visualization applications, due to the flexible and effective use of bandwidth and required computational power.

Chapter 5 presents the second novel algorithm which was developed based on a thorough analysis of advantages and disadvantages of Wickramanayake et al.'s [5, 39] and Kwatra et al.'s [36] texture synthesis algorithms. The novel algorithm combined the most efficient features of both algorithms; i.e. the fast, transform domain patch matching process of Wickramanayake et al.'s algorithm and the optimal, graph cut based patch quilting approach adopted by Kwatra et al. To enable the seamless use of both algorithms within a single texture synthesis algorithm, the pixel domain graph-cut approach originally adopted by Kwatra in patch quilting, was modified to be effectively used within a multi-resolution DWT domain representation of texture. Further the computational complexity of the graph-cut approach was kept to a minimum by using sub-ordinate and refinement stages in the synthesizing process. The sub-ordinate stage introduces an initial cutting path which was subsequently refined, by using neighbouring wavelet coefficients progressively, thereby reducing algorithmic complexity. Experimental results and an in-depth analysis was provided to justify that the proposed method produces better synthesised texture quality as

compared to the method proposed in [5, 39] and provides similar or better quality results compared to Kwatra et al.'s [36].

Chapter 6 presents the third novel algorithm which is essentially an extension of the planar transform domain texture synthesis algorithm of chapter 5 into texture synthesis on parameterised rational biquadratic patches. It was shown that this algorithm was able to perform good quality texture synthesis at real-time rates. Experimental results were provided to prove that the method was capable of fast texture synthesis on geometric surfaces that can be represented by parameterised rational biquadratic patches, e.g., Dupin Cyclides, Spheres and Toruses.

Finally chapter 7 presents the fourth and fifth novel contributions of the thesis. The fourth contribution focused on a novel approach to texture synthesis on Bezier patches. This algorithm overcame the limitations of the algorithm that was presented in chapter 6, i.e. not having control over the size of texture pattern to be used in synthesis thus extending its applicability to more practical domains. The underlying texture synthesis process though somewhat similar to the algorithm of chapter 6 had two major differences; 1) in the previous approach the texture block size is the same in both the synthesis and embedding processes, where as in this case the block size is picked randomly, 2) instead of synthesizing texture directly on to a surface, it is synthesised on a control polygon. A random size block provides a better fit of a shape as well as size of a control polygon face, which provides flexible control over a texture pattern while being synthesised onto a surface. Further it was shown that synthesizing onto a control polygon mesh optimises the synthesizing process. This basic approach showed that texture can be efficiently synthesised onto a control polygon, which can then be projected onto a smooth Bézier patch. The algorithm was shown to provide a compact representation for texture as well as patches. It was shown that the algorithm can be applied to a wide variety of texture patterns and surfaces. Further the EZW algorithm was utilized to enable progressive texture synthesis capabilities.

The fifth contribution of the thesis presented in Chapter 7, provided an interactive GUI for texture synthesis on arbitrarily shaped surfaces. The underlying texture synthesis approach adopted inherited all positive properties and functionalities of the

novel algorithms presented previously in the thesis. Further it provided added functionality in providing the user with the ability to define tangential vectors indicating required directions of texture synthesis on a given surface. Experimental results were presented on both arbitrary and geometric 3D surfaces, which showed the flexibility and efficiency of the novel approach.

Therefore all objective stated in chapter 1 have been achieved. However there are possibilities to further improve the contributions of this thesis.

In the implementation of the proposed algorithms we have not focused on software optimization. This has made it difficult to quantify the speeds of the proposed algorithms. Further our implementations of the state-of-the-art algorithms, which have been used as benchmarks, have not been optimised. Therefore further work will be required in software optimization, allowing a fair quantitative comparison of the speed of all algorithms.

Work is now in progress on the use of hardware acceleration to further speed up the proposed texture synthesis processes. The effective use of a GPU (Graphics Processing Unit) will give further gains in the synthesizing process. In addition, we are currently looking at extending the idea related to *transform domain texture synthesis* to more complex NURBS surfaces.

The transform domain edge cutting techniques proposed in this thesis can be used for many alternative applications such as, view morphing, image stitching, image restoration etc. Furthermore, the texture synthesis methods discussed in this thesis, and the results of the extensive experiments are useful in video texture synthesis applications.

The use of the Bézier surfaces in texture synthesis provides additional functionality. In particular, the use of the proposed texture synthesis approaches on animated surfaces needs to be further studied. Such a study should address the relationship between the amount of movement in control points and the synthesised texture pattern over a surface, in order to realistically animate the change of the texture pattern on an animated surface. Further incorporating various motions on a control polygon will

reflect the movement of the texture pattern on the surface. Further studies on converting any given 3D model to a Bézier model requires to be done. Such a model will enable simultaneous compression of the mesh and synthesised texture.

Further the proposed texture synthesis algorithms can be implemented within many software tools such as 3D Max, Maya, CAD and even advanced modelling systems in order to further extend their capabilities.

References

[1] *IEEE standard glossary of image processing and pattern recognition terminology*, standard 610.4, IEEE Press, New York, 1990.

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley Publications, 1993.

[3] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989.

[4] W-C. Lin, J.H. Hays, C. Wu, V. Kwatra, and Y. Liu, *A Comparison Study of Four Texture Synthesis Algorithms on Regular and Near-regular Textures*. Tech. report CMU-RI-TR-04-01, Robotics Institute, Carnegie Mellon University, January, 2004.

[5] D. S. Wickramanayake, E. A. Edirisinghe, and H. E. Bez, *Zerotree Wavelet Based Image Quilting for Fast Texture Synthesis*, Proceedings of Pattern Recognition and Image Analysis: Second Iberian Conference, IbPRIA 2005, Estoril, Portugal, Part I , Springer Lecture Notes in Computer Science (LNCS), vol. 3522 (2005), pp 384, ISSN: 0302-9743, ISBN: 3-540-26153-2, June 7-9, 2005.

[6] J. R. Bergen and B. Julesz, *Rapid discrimination of visual patterns*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 13, no. 5, pp 857-863, 1983.

[7] B. Julesz, *Visual pattern discrimination*, IRE transactions on Information Theory, vol. 8, pp 84-92, 1962.

[8] W. K. Pratt, O. D. Faugeras, and A. Gagalowicz, *Visual discrimination of stochastic texture fields*, IEEE Transactions on Systems, Man and Cybernetics, vol. 8, pp 796-804, Nov. 1978.

[9] T. Caelli, B. Julesz, and E. Gilbert, *On perceptual analyzers underlying visual texture discrimination: Part II,* Biological Cybernetics, vol. 29, no. 4, pp 201-214, 1978.

[10] B. Julesz, *Textons, the elements of texture perception, and their interactions*, Nature, vol. 290, pp 91-97, Mar. 1981.

[11] B. Julesz, *A theory of preattentive texture discrimination based on first-order statistics of textons*, Biological Cybernetics, vol. 41, no. 2, pp 131-138, 1981.

[12] D. S. Wickramanayake, *Transform domain texture synthesis*, PhD thesis, Loughborough University, UK, October 2005.

[13] J. M. Dischler and D. Ghazanfarpour, *A Survey of 3D Texturing*, Computers and Graphics, vol. 25, no. 10, pp 135-151, January 2001.

[14] A. Turing, *The chemical basis of morphogenesis. Philosophical Transactions of the Royal Society* (B), pp 237–272, 1952.

[15] J. Bard, *A Model for Generating Aspects of Zebra and Other Mammalian Coat Patterns*, Journal of Theoretical Biology, vol. 93, no. 2, pp 363–385, November 1981.

[16] J. D. Murray, *On Pattern Formation Mechanisms for Lepidopteran Wing Patterns and Mammalian Coat Markings*, Philosophical Transactions of the Royal Society B, vol. 295, pp 473–496.

[17] S. P. Worley, *A cellular texture basis function*, Proceeding of SIGGRAPH 96, ACM SIGGRAPH, pp 291–294, August 1996.

[18] K. Fleischer, D. Laidlaw, B. Currin, and A. Barr. *Cellular texture generation.* Proc. of SIGGRAPH 95, pp 239-248, ISBN 0-201-84776-0. August 1995.

[19] W. Becket, and N. I. Badler, *Imperfection for realistic image synthesis.* Journal of Visualization and Computer Animation vol. 1, no. 1, pp 26–32, August 1990.

[20] G. Miller, *Efficient algorithms for local and global accessibility shading.* In Computer Graphics Proceeding, ACM SIGGRAPH, pp 319–326, 1994.

[21] P. Hanrahan, and W. Krueger, *Reflection from layered surfaces due to subsurface scattering.* In Computer Graphics Proceedings, ACM SIGGRAPH, pp 165–174, 1993.

[22] D. J. Heeger and J. R. Bergen. *Pyramid-Based texture analysis/synthesis.* In R. Cook, editor, SIGGRAPH 95 Conference Proceedings, pp 229–238. ACM SIGGRAPH, AddisonWesley, August 1995.

[23] J. S. De Bonet, *Multiresolution sampling procedure for analysis and synthesis of texture images*, SIGGRAPH 97 Conference Proceeding, pp 361–368. ACM SIGGRAPH, AddisonWesley, August 1997.

[24] R. Paget and I.D. Longstaff. *Texture synthesis via a noncausal nonparametric multiscale Markov random field.* IEEE Transactions on Image Processing, 7(6):925–931, June 1998.

[25] A. Efros and T. Leung. *Texture synthesis by non-parametric sampling.* In International Conference on Computer Vision, volume 2, pp 1033–8, Sep 1999.

[26] Li-Yi. Wei and M. Levoy, *Fast Texture Synthesis Using Tree-Structured Vector Quantization.* In Proceedings of SIGGRAPH 2000, pp 479–488, July 2000.

[27] M. Ashikhmin, *Synthesizing Natural Textures,*The proceedings of ACM Symposium on Interactive 3D Graphics, Research Triangle Park, NorthCarolina March 19-21, pp 217-226, 2001.

[28] S. Zelinka and M. Garland, *Real-Time Texture Synthesis with the Jump Map*, In Proceedings of the Thirteenth Eurographics Workshop on Rendering Techniques, Eurographics Association, pp 99-104, 2002.

[29] M. Tran and A. Datta. *Synthesising textures using variable neighbourhood searching.* In Proc. 7th International Conference on Digital Image Computing: Techniques and Applications, pp 643-652. CSRIO Publishing, 2003.

[30] M. Sabha, P. Peers and P. Dutré, *Texture Synthesis using Exact Neighborhood Matching,* Computer Graphics Forum, Volume 26, Number 2, pp 131-142, June 2007.

[31] Y. Q. Xu, B. N. Guo, and H. Y. Shum, *Chaos Mosaic: Fast and Memory Efficient Texture Synthesis*, Technical Report-2000-32, April 2000.

[32] L. Ling, C E Liu, Y. Xing, B. Guo, and H-Y. Shum. *Real Time Texture synthesis by patch based Sampling*, ACM Transaction on Graphics, Vol 20, No 3, pp 127-150, July 2001.

[33] A. Efros and W. T. Freeman, *Image Quilting for Texture Synthesis and Transfer,* Proceedings of SIGGRAPH 01, pp 341-6, Los Angeles, California, August 2001.

[34] R. Szeliski and H.-Y. Shum. *Creating full view panoramic mosaics and environment maps.* Proceedings of SIGGRAPH 97, pp 251–258, August 1997.

[35] A. Nealen and M. Alexa, *Hybrid Texture Synthesis.* Proceedings of the

Eurographics Symposium on Rendering 2003, 14th Eurographics Workshop on Rendering, pp 97-105, 2003.

[36] V. Kwatra, A. Schödl, I. Essa, G. Turk and A. Bobick, *Graphcut Textures: Image and Video Synthesis Using Graph Cuts.* SIGGRAPH 2003 conference proceeding, ACM Transactions on Graphics, vol. 22, no. 3, pp 277 286, 2003.

[37] Y. Boykov, O. Veksler and R. Zabih, *Fast approximate energy minimization via graph cuts.* In International Conference on Computer Vision, 377–384, 1999.

[38] M. Cohen, J. Shade, S.Hiller and O. Deussan, *Wang Tile for image and texture generatio,* SIGGRAPH 2003 conference proceeding, vol. 22, no. 3, pp 287 – 294, July 2003.

[39] D. S. Wickramanayake, E. A. Edirisinghe and H. E. Bez, *Multiresolution texture synthesis in wavelet transform domain,* The Journal of Imaging Science and Technology, Vol.50, no.1, pp 93-102, ISBN/ISSN:1062-3701, January/February 2006.

[40] D. S. Wickramanayake, E. A. Edirisinghe and H. E.Bez, *Transform Domain Texture Synthesis based on DCT*, In PREP 2005 proceedings (Orals), pp 152-153, March 2005.

[41] D. Salomon, *Curves and Surfaces for Computer Graphics*, Published by Birkhäuser, ISBN 0387241965, 9780387241968, 2006.

[42] G. Gardner, *Simulation of natural scenes using textured quadric surfaces.* Computer Graphics, ACM Siggraph Annual Conference Series; vol. 18, no. 3, pp 11-20, 1984.

[43] G. Gardner, *Visual simulation of clouds.* Computer Graphics, ACM Siggraph Annual Conference Series, vol. 19, no. 3, pp 297-303, 1985.

[44] D. Peachey, *Solid texturing on complex surfaces*. Computer Graphics, ACM Siggraph Annual Conference Series, vol. 19, no. 3, pp 279-286, 1985.

[45] K. Perlin, *An image synthesizer*. Computer Graphics, ACM Siggraph Annual Conference Series, vol. 19, no. 3, pp 287-296, 1985.

[46] D. Ghazanfarpour, J. M. Dischler, *Spectral analysis for automatic 3-D texture generation*. Computers and Graphics, vol. 19, no. 3, pp 413-422, 1995.

[47] D. Ghazanfarpour, J. M. Dischler, *Generation of 3D texture using multiple 2D models analysis*. Computer Graphics Forum (Eurographics 96), vol. 15, no. 3, pp 311-323, 1996.

[48] J. N. Dischler, D. Ghazanfarpour, and R. Freydier, *Anisotropic solid texture synthesis using orthogonal 2D views*. Computer Graphics Forum, vol. 17, no. 3, pp 87- 95, 1998.

[49] D. J. Heeger, J. R. Bergen, *Pyramid-based texture analysis/ synthesis. Computer Graphics*, ACM Siggraph Annual Conference Series, vol 29, pp 229-38, 1995.

[50] J. W. Buchanan. *Simulating wood using a voxel approach*.Computer Graphics Forum, 17(3):103-112, Eurographics 1998.

[51] K. Hirota, Y. Tanoue and T. Kaneko, *Generation of crack patterns with a physical model*. The Visual Computer vol. 14, no.3, pp 126-137, 1998.

[52] G. Turk. *Generating textures on arbitrary surfaces using reaction diffusion*. Computer Graphics, ACM Siggraph Annual Conference Series 1991, vol . 25, no. 4, pp 289-298, 1991.

[53] J. F. Blinn. *Simulation of wrinkled Surfaces*, ACM SIGGRAPH, Annual

Conference proceeding, vol. 11, pp 286-292, 1978.

[54] R. L. Cook, *Shade trees*. ACM SIGGRAPH, Annual Conference proceeding vol 18, no. 3 pp 223-231, 1984.

[55] N. Max, *Horizon mapping: shadows for bump-mapped surfaces*, The Visual computer pp 109-117, 1988.

[56] K. Perlin, *Hypertextures*, Computer Graphics 23(3), ACM Siggraph annual Conference Series, pp 253-262, July 1989.

[57] J. Kajiya and T. Kay, *Rendering Fur with three dimensional textures*, Computer Graphics, ACM Siggraph annual Conference Series, vol. 23, no. 3 pp 271-298 July 1989.

[58] K.W Fleischer, D.H. Laidlaw, B. L. Currin and A. H. Barr, *Cellular Texture Generation,* Computer Graphics, ACM Siggraph annual Conference Series, vol. 29, pp 239-248, 1995.

[59] J. M. Dischler and D. Ghazanfarpour, *Interactive Image-Based Modeling of Macrostructured Textures*, IEEE Computer Graphics and Applications, vol. 19, no. 1, pp 66-74, 1999.

[60] L.-Y. Wei and M. Levoy, *Texture synthesis over arbitrary manifold surfaces*, SIGGRAPH 2001, Computer Graphics Proceeding pp 355-360, 2001.

[61] G. Turk, *Re-tiling polygonal surfaces*, ACM SIGGRAPH, pp 55-64, 1992.

[62] G. Turk, *Texture Synthesis on Surfaces*, ACM SIGGRAPH, pp 347-354, 2001

[63] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin, *Texture and shape synthesis on surfaces*. In 12th Eurographics Workshop on Rendering, pp 301--312, June 25-27 2001.

[64] T. Xin, J. Zhang, L. Liu, X Wang, B. Guo and H-Y. Shum, *Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces*, ACM SIGGRAPH, pp 665–672, 2002.

[65] J. Zhang, K. Zhou, L. Velho, B. Guo and H-Y. Shum, *Synthesis of progressively-variant textures on arbitrary surfaces*, ACM SIGGRAPH, pp 295-302, 2003.

[66] S. Zelinka and M. Garland. *Interactive Texture Synthesis on Surfaces Using Jump Maps*. Eurographics Symposium on Rendering 2003, June 2003.

[67] S. Lefebvre, H. Hoppe *Appearance-Space Texture Synthesis,* International Conference on Computer Graphics and Interactive Techniques archive ACM SIGGRAPH, pp 541 - 548, ISBN: 1-59593-364-6, 2006.

[68] J. Han, K. Zhou, Li-Yi. Wei, M. Gong, H. Bao, X. Zhang and B. Guo, *Fast example-based surface texture synthesis via discrete optimization*, Visual Computer, vol. 22, pp 918-925, 2006.

[69] J. Kopf, C-W. Fu, D. Cohen-Or, O. Deussen and T-T. Wong, *Solid Texture Synthesis from 2D Exemplars,* ACM SIGGRAPH, pp 2:1-2:9, 2007.

[70] F. Neyret and M-P. Cani, *Pattern-based texturing revisited*, Proceedings of SIGGRAPH, pp 235–242, August 1999.

[71] E. Praun, A. Finkelstein and H. Hoppe, *Lapped Textures*, Computer Graphics Proceedings, pp 465–470, 2000.

[72] C. Soler, M.-P. Cani and A. Angelidis, *Hierarchical Pattern Mapping*. SIGGRAPH 2002, Conference Proceedings, pp 673–680, 2002.

[73] L. Wang, X. Gu, K. Mueller and S-T. Yau: *Uniform texture synthesis and*

texture mapping using global Parameterization, ISSN: 0178-2789 (Paper), vol. 21, Numbers 8-10, pp 801 – 810, Special Issues of Pacific Graphics, 2005.

[74] X. Gu, S. T. Yahu, *Global conformal surface parametrization*, In proceeding of Eurographic/SIGGRAPH Symposium on Geometry Processing, Aachen, Germany, pp 127-137, 23-25 June 2003.

[75] L. Piegl and W. Tiller: *The NURBS Book*, Springer-Verlag 1995–1997 (2nd ed.).

[76] F. Bernardini and H. E. Rushmeier: *The 3D Model Acquisition Pipeline*. Computer Graphics Forum 21(2): pp 149-172, 2002.

[77] B. Curless, *From Range Scans to 3D Models*, ACM SIGGRAPH Computer Graphics, vol. 33, no. 4, pp 38-41, Nov 2000.

[78] C. de Boor. *Bicubic spline interpolation*. Journal of Maths and Physics, vol. 41, pp 212-218, 1962.

[79] J. Ferguson. *Multivariable curve interpolation*. Journal of ACM, vol 11, no. 2, pp 221-228, 1964.

[80] W. Gordon. *Free-form surface interpolation through curve networks*. Technical Report GMR-921, General Motors Research Laboratories, 1969.

[81] W. Gordon. *Spline-blended surface interpolation through curve networks*. Journal of Mathematics and Mechanics, vol. 18, no. 10, pp 931-952, 1969.

[82] W. Boehm, *Cubic B-spline curves and surfaces in computer aided geometric design*. Computing, vol. 19, no. 1, pp 29-34, 1977.

[83] G. Chaik, *An algorithm for high speed curve generation*. Computer Graphics

and Image Processing, vol. 3, pp 346-349, 1974.

[84] D. Doo and M. Sabin, *Behaviour of recursive division surfaces near extraordinary points. Computer Aided Design*, vol. 10, no. 6, pp 356-360, 1978.

[85] E. Catmull and J. Clark, *Recursively generated B-spline surfaces on arbitrary topological meshes.* Computer Aided Design, vol. 10, no. 6, pp 350-355, 1978.

[86] R. Sedgewick, *A logorithms in C part 5: Graph Algorithms.* Addison-Welsley, Reading, Massachusetts, 2001.

[87] P.J. Burt, E. H. Adelson, *A multiresolution spline with application to image mosaics.* ACM transactions 2, 217–236. ISSN: 0730-0301, 1983.

[88] H. S. Malavar, *Signal Processing with Lapped Transforms,* Norwood, MA, Artech House, 1992.

[89] S. G. Mallat, *A Theory for Multiresolution Signal Decomposition: The Wavelet Representation*, IEEE Trans. PAMI, vol. 11, no. 7, July 1989, pp 674-693.

[90] S. Gortler, P. Schröder, M. Cohen and P. Hanrahan, *Wavelet Radiosity,* Computer Graphics, Annual Conference Series SIGGRAPH 93, pp 221-230, 1993.

[91] D. Berman, J. Bartell and D. Salesin, *Multiresolution Painting and Compositing,* Computer Graphics, Annual Conference Series SIGGRAPH 94, pp 85-90, 1994.

[92] A. Finkelstein, D. Salesin, *Multiresolution Curve*, Computer Graphics, Annual Conference Series SIGGRAPH 94, pp 261-268, 1994.

[93] M. Eck, T. DeRose, T .Duchamp, H. Hoppe, M. Lounsberry and W. Stuetzle, *Multiresolution Analysis of Arbitrary Meshes,* Computer Graphics, Annual Conference Series SIGGRAPH 95, pp 173-182, 1995.

[94] C. Jacobs, A. Finkelstein and D. Salesin, *Fast Multiresolution Image Querying,* Computer Graphics, Annual Conference Series SIGGRAPH 95, pp 277-286, 1995.

[95] E. J. Stollnitz, T. D. DeRose and D. H. Salesin, *Wavelets for Computer Graphics -Theory and Applications*, Morgan Kaufmann Publishers Inc., San Francisco, California, 1996.

[96] P. Schröder, W. Sweldens, M. Cohen, T. DeRose and D. Salesin, *Wavelets in Computer Graphics*, SIGGRAPH 96 Course Notes, 1996.

[97] C. K. Chui, *An Introduction to Wavelets*, Academic Press Inc., Boston, MA 1992.

[98] I. Daubechies, *Ten Lectures on Wavelets*, Vol. 61 of CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, 1992.

[99] *http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html* , The Wavelet Tutorial by Robi Polikar [Accessed 01/08/2007].

[100] I. Daubechies, *Ten Lectures on Wavelets.* Philadelphia, PA: SIAM, 1992.

[101] G. Strang, *Wavelet Transforms Versus Fourier Transforms,* Bull. Amer. Math. Soc. vol. 28, pp 288-305, 1993.

[102] S. G .Mallat, *A theory of multiresolution signal decomposition: the wavelet representation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp 674-693, 1989.

[103] J. M. Shapiro, *Embedded Image Coding Using Zerotrees of Wavelet Coefficient, IEEE Trans. Signal Processing, December,* vol. 41, no.12, pp 3445-3462, 1993.

[104] B. Waggener. *Pulse Code Modulation Techniques*, 1st ed., New York, NY: Van Nostrand Reinhold. ISBN 0-442-01436-8.

[105] L. Ford and D. Fulkerson, *A Flow in Networks*, Princeton University Press, 1962.

[106] B. Grünbaum and G. C. Shepard, *Tiling and patterns*, W.H. *Freeman and company*, New York, 1986.

[107] E. V. DeNardo, *Dynamic Programming: Models and Applications*, Courier Dover Publications, April 2003.

[108] R. H. Bartels, J.C. Beatty, and B.A Barsky, Bézier Curves Ch. 10 in *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling.* San Francisco, CA: Morgan Kaufmann, pp 211-245, 1998.

[109] G. Farin, *Curves and Surfaces for CAGD*, 5th ed. published by Academic Press. ISBN 1558607374.

[110] H. E. Bez and T.J Wetzel. *Induced rational parametrisations of special curves.* In International Journal of Computer Mathematics, vol. 80, no. 9, pp 1093–1109, 2003.

[111] S. Foufou, L. Garnier, and M. J. Pratt. *Conversion of dupin cyclide patches into rational bi-quadratic bezier form.* In Proc. of the IMA Mathematics of Surfaces XI, pp 201–218, 2005.

[112] H. E. Bez, *Bounded domain, bi-quadratic rational parametrisations of Dupin*

cyclides, Report No. 1090, Dept of CS, Loughborough University, 2006.

[113] D. Dutta, R. R. Martin, and M. J. Pratt. *Cyclides in surface and solid modeling*. IEEE Computer Graphics and Applications, vol. 13, pp 5359, 1993.

[114] R. Martin, J. de Pont, and T. Sharrock, *Cyclide surfaces in computer aided design,* In Proceedings of the In Proceedings of the IMA Conference on the Mathematics of Surfaces, 1986.

[115] M. J. Pratt, *Cyclides in computer aided geometric design*, In Computer Aided Geometric Design, vol. 7, pp 221–242, 1990.

[116] R. Szeliski and H.-Y. Shum, *Creating full view panoramic mosaics and environment maps,* In Proc. of SIGGRAPH 1997, pp 251–258, 1997.

[117] S. Magda, D. Kriegman, *Fast Texture synthesis on Arbitrary Meshes*, Proceedings of the 14th Eurographics workshop on Rendering vol. 44, pp 82 - 89, 2003.

[118] C-W. Fu and M-K. Leung, *Texture tiling on arbitrary topological surfaces*, In Proceedings of Eurographics Symposium on Rendering 2005, pp 99–104, 2005.

[119] M. Soucy, G. Guy and R. Marc, *A Texture-Mapping Approach for the Compression of Colored 3D triangulations,* The Visual Computer, vol. 12, no. 10, pp 503–514, 1996.

[120] http://www.holmes3d.net/graphics/roffview/tools/patchoff/index.html [Accessed 20-Jan-2008].

[121] D. M. Mount. *ANN Programming Manual. University of Maryland, 1998.*

[122] P. F. Preparata, M. I. Shamos, *Computational Geometry: An Introduction,*

Springer-Verlag, ISBN 3-540-96131-3, 1985.

[123] G. Ross, A. Willocks, *Uncertainty in GIS and 3D modelling - approaches to quantify and represent error in interpreted digital datasets*, The 33rd International Geology Congress, 2008.

[124] R. B. Catalan, E. I. Perez and B. Z. Perez, *Evaluation of 3D Scanners to Develop Virtual Reality Applications*, cerma, Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007), pp 551-556, 2007.

[125] F. Bernardini, H. E. Rushmeier: *The 3D Model Acquisition Pipeline*. Comput. Graph. Forum 21(2), pp 149-172, 2002.

[126] G. Farin, *A History of Curves and Surfaces in CAGD*. In: Handbook of CAGD, Elsevier, 2002.

[127] S. Bernstein, *Démonstration du théorème de Weierstrass fondée sur le calcul des probabilities. Comm. Soc. Math. Kharkov* 13, pp 1-2, 1912.

[128] H.E.Bez, *Bounded domain, bi-quadratic rational parametrisations of Dupin cyclids*, International Journal of Computer Mathematics, Vol. 85:7, pp 1097-1111, 2008.

[129] A. Treisman, *Preattentive processing in vision*, Computer Vision, Graphics and Image Processing 31, pp 156-177, 1985.

Appendix A

The weighted computed for the patches are given below (refer chapter 3, section 3.4.4)

Weight Computed as: First patch: $\theta_0 = 0, \theta_1 = \frac{\pi}{2}$, $0, \phi_1 = \frac{\pi}{2}$

$$\begin{aligned} w_{00} &= a - c, & w_{01} &= \frac{a-c}{\sqrt{2}}, & w_{02} &= a, \\ w_{10} &= \frac{a-c}{\sqrt{2}}, & w_{11} &= \frac{a-c}{2}, & w_{12} &= \frac{a}{\sqrt{2}}, \\ w_{20} &= a, & w_{21} &= \frac{a}{\sqrt{2}}, & w_{22} &= a \end{aligned}$$

Second patch: $\theta_0 = 0, \theta_1 = \frac{\pi}{2}$, $\phi_0 = \frac{\pi}{2}, \phi_1 = \pi$

$$\begin{aligned} w_{00} &= a, & w_{01} &= \frac{a+c}{\sqrt{2}}, & w_{02} &= a + c, \\ w_{10} &= \frac{a}{\sqrt{2}}, & w_{11} &= \frac{a+c}{2}, & w_{12} &= \frac{a+c}{\sqrt{2}}, \\ w_{20} &= a, & w_{21} &= \frac{a}{\sqrt{2}}, & w_{22} &= a. \end{aligned}$$

Third patch $\theta_0 = \frac{\pi}{2}, \theta_1 = \pi$, $\phi_0 = 0, \phi_1 = \frac{\pi}{2}$

$$\begin{aligned} w_{00} &= a, & w_{01} &= \frac{a}{\sqrt{2}}, & w_{02} &= a, \\ w_{10} &= \frac{a+c}{\sqrt{2}}, & w_{11} &= \frac{a+c}{2}, & w_{12} &= \frac{a}{\sqrt{2}}, \\ w_{20} &= a + c, & w_{21} &= \frac{a+c}{\sqrt{2}}, & w_{22} &= a. \end{aligned}$$

Fourth patch $\theta_0 = \frac{\pi}{2}, \theta_1 = \pi$, $\phi_0 = \frac{\pi}{2}, \phi_1 = \pi$

$$\begin{aligned} w_{00} &= a, & w_{01} &= \frac{a}{\sqrt{2}}, & w_{02} &= a, \\ w_{10} &= \frac{a}{\sqrt{2}}, & w_{11} &= \frac{a-c}{2}, & w_{12} &= \frac{a-c}{\sqrt{2}}, \\ w_{20} &= a, & w_{21} &= \frac{a-c}{\sqrt{2}}, & w_{22} &= a - c. \end{aligned}$$

. The Bézier vertices of the four patches

1st patch Bézier vertex data

$$
\begin{aligned}
xv_{00} &= \frac{\mu(c-a)+b^2}{a-c} & xv_{01} &= \frac{\mu(c-a)+b^2}{a-c} & xv_{02} &= \frac{\mu c+b^2}{a} \\
xv_{10} &= \frac{\mu(c-a)+b^2}{a-c} & xv_{11} &= \frac{\mu(c-a)+b^2}{a-c} & xv_{12} &= \frac{\mu c+b^2}{a} \\
xv_{20} &= \frac{\mu c}{a} & xv_{21} &= \frac{\mu c}{a} & xv_{22} &= \frac{\mu c}{a}
\end{aligned}
$$

$$
\begin{aligned}
yv_{00} &= 0 & yv_{01} &= 0 & yv_{02} &= 0 \\
yv_{10} &= \frac{b(a-\mu)}{a-c} & yv_{11} &= \frac{b(a-\mu)}{a-c} & yv_{12} &= b \\
yv_{20} &= \frac{b(a-\mu)}{a} & yv_{21} &= \frac{b(a-\mu)}{a} & yv_{22} &= b
\end{aligned}
$$

$$
\begin{aligned}
zv_{00} &= 0 & zv_{01} &= \frac{b(c-\mu)}{a-c} & zv_{02} &= \frac{b(c-\mu)}{a} \\
zv_{10} &= 0 & zv_{11} &= \frac{b(c-\mu)}{a-c} & zv_{12} &= \frac{b(c-\mu)}{a} \\
zv_{20} &= 0 & zv_{21} &= -\frac{b\mu}{a} & zv_{22} &= -\frac{b\mu}{a}
\end{aligned}
$$

2nd patch Bézier vertex data

$$
\begin{aligned}
xv_{00} &= \frac{\mu c+b^2}{a} & xv_{01} &= \frac{\mu(a-c)-b^2}{a+c} & xv_{02} &= \frac{\mu(a+c)+b^2}{a+c} \\
xv_{10} &= \frac{(\mu c+b^2)}{a} & xv_{11} &= \frac{\mu(a-c)-b^2}{a+c} & xv_{12} &= \frac{\mu(a+c)+b^2}{a+c} \\
xv_{20} &= \frac{\mu c}{a} & xv_{21} &= \frac{\mu c}{a} & xv_{22} &= \frac{\mu c}{a}
\end{aligned}
$$

$$
\begin{aligned}
yv_{00} &= 0 & yv_{01} &= 0 & yv_{02} &= 0 \\
yv_{10} &= b & yv_{11} &= \frac{b(a+\mu)}{a+c} & yv_{12} &= \frac{b(a+\mu)}{a+c} \\
yv_{20} &= b & yv_{21} &= \frac{b(a+\mu)}{a} & yv_{22} &= \frac{b(a+\mu)}{a}
\end{aligned}
$$

$$
\begin{aligned}
zv_{00} &= \frac{b(c-\mu)}{a} & zv_{01} &= \frac{b(c-\mu)}{a+c} & zv_{02} &= 0 \\
zv_{10} &= \frac{b(c-\mu)}{a} & zv_{11} &= \frac{b(c-\mu)}{a+c} & zv_{12} &= 0 \\
zv_{20} &= -\frac{b\mu}{a} & zv_{21} &= -\frac{b\mu}{a} & zv_{22} &= 0
\end{aligned}
$$

3rd patch Bézier vertex data

$$
\begin{aligned}
xv_{00} &= \frac{\mu c}{a} & xv_{01} &= \frac{\mu c}{a} & xv_{02} &= \frac{\mu c}{a} \\
xv_{10} &= \frac{\mu(a+c)-b^2}{a+c} & xv_{11} &= \frac{\mu(a+c)-b^2}{a+c} & xv_{12} &= \frac{(\mu c-b^2)}{a} \\
xv_{20} &= \frac{\mu(a+c)-b^2}{a+c} & xv_{21} &= \frac{\mu(a+c)-b^2}{a+c} & xv_{22} &= \frac{\mu c-b^2}{a}
\end{aligned}
$$

$$
\begin{aligned}
yv_{00} &= \frac{b(a-\mu)}{a} & yv_{01} &= \frac{b(a-\mu)}{a} & yv_{02} &= b \\
yv_{10} &= \frac{b(a-\mu)}{a+c} & yv_{11} &= \frac{b(a-\mu)}{a+c} & yv_{12} &= b \\
yv_{20} &= 0 & yv_{21} &= 0 & yv_{22} &= 0
\end{aligned}
$$

$$
\begin{aligned}
zv_{00} &= 0 & zv_{01} &= -\frac{b\mu}{a} & zv_{02} &= -\frac{b\mu}{a} \\
zv_{10} &= 0 & zv_{11} &= -\frac{b(c+\mu)}{a+c} & zv_{12} &= -\frac{b(c+\mu)}{a} \\
zv_{20} &= 0 & zv_{21} &= -\frac{b(c+\mu)}{a+c} & zv_{22} &= -\frac{b(c+\mu)}{a}
\end{aligned}
$$

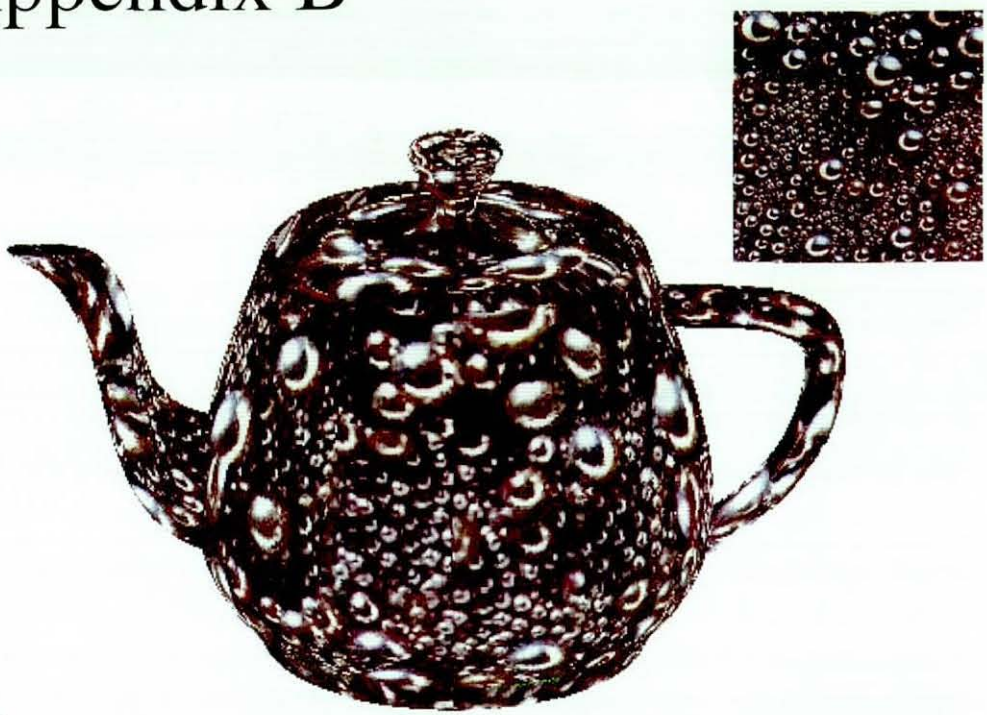
4th patch Bézier vertex data

$$
\begin{aligned}
xv_{00} &= \frac{\mu c}{a} & xv_{01} &= \frac{\mu c}{a} & xv_{02} &= \frac{\mu c}{a} \\
xv_{10} &= \frac{(\mu c-b^2)}{a} & xv_{11} &= \frac{\mu(c-a)-b^2}{a-c} & xv_{12} &= \frac{\mu(c-a)-b^2}{a-c} \\
xv_{20} &= \frac{\mu c-b^2}{a} & xv_{21} &= \frac{\mu(c-a)-b^2}{a-c} & xv_{22} &= \frac{\mu(c-a)-b^2}{a-c}
\end{aligned}
$$

$$
\begin{aligned}
yv_{00} &= b & yv_{01} &= \frac{b(a+\mu)}{a} & yv_{02} &= \frac{b(a+\mu)}{a} \\
yv_{10} &= b & yv_{11} &= \frac{b(a+\mu)}{a-c} & yv_{12} &= \frac{b(a+\mu)}{a-c} \\
yv_{20} &= 0 & yv_{21} &= 0 & yv_{22} &= 0
\end{aligned}
$$

$$
\begin{aligned}
zv_{00} &= -\frac{b\mu}{a} & zv_{01} &= -\frac{b\mu}{a} & zv_{02} &= 0 \\
zv_{10} &= -\frac{b(c+\mu)}{a} & zv_{11} &= -\frac{b(c+\mu)}{a-c} & zv_{12} &= 0 \\
zv_{20} &= -\frac{b(c+\mu)}{a} & zv_{21} &= -\frac{b(c+\mu)}{a-c} & zv_{22} &= 0
\end{aligned}
$$

Appendix B



Teapot

Teapot

Cup

Sphere

Appendix C

Publications

PROGRESSIVE TEXTURE SYNTHESIS ON 3D SURFACES

Rupesh N.Shet, Eran A.Edirisinghe, Helmut E.Bez
Department of Computer Science
Loughborough University, LE11 3TU
United Kingdom
R.Shet@lboro.ac.uk

ABSTRACT
In this paper we propose a novel approach to progressive texture synthesis on 3D surfaces. The method is based on a multiresolution DWT decomposition of a sample texture, which is subsequently progressively transmitted and utilised within a 3D texture synthesis process. We use the Embedded Zerotree Wavelet (EZW) concept originally proposed by Shapiro, in prioritising the DWT coefficients in progressive texture transmission and synthesis. We show that the proposed algorithm has applications in bandwidth and processing power constrained application domains, such as remote visualisation, streaming and in computer games/animations.

KEYWORDS
Texture synthesis, progressive texture, discrete wavelet transform, EZW, rendering

1. INTRODUCTION

Texture synthesis on 2D/3D surfaces, enhances the realism of virtual scenes. At present significant amount of research is focussed on establishing virtual realism of arbitrary shaped 3D surfaces. As a result a number of 'image based' texture synthesis algorithms have been proposed in the past. A typical texture synthesis algorithm starts from a sample image and attempts to produce a larger texture with visual appearance similar to the sample by repeated placement of micro patterns of texture elements on a surface. It does this in a way that when perceived by an observer, the synthesized texture appears to be generated by the same underlying stochastic process. However all texture synthesis algorithms are challenged by the high statistical variability of textures involved in synthesis. Thus a universal solution to texture synthesis yet remains an open problem. Texturing arbitrary shaped 3D surfaces which provide further challenges have attracted much research interest in the recent pass due applications in animated movie production, computer games and virtual productions.

A major proportion of research in the area of texture synthesis has focused on synthesizing texture on planner surfaces. Recently a number of approaches have been proposed for texture synthesis on 3D surfaces [1-8]. These approaches can be broadly classified into two groups, namely, patch based texture synthesis [1,2,3] and pixel based texture synthesis [4-8]. The progressive texture synthesis algorithm proposed in this paper belongs to the latter category, i.e. pixel based texture synthesis. As the name suggests the texture synthesis is performed considering a pixel as the basic unit rather than a complete patch. In [4] Wei & Levoy proposed a 2D texture synthesis method based on tree structured vector quantization and later extended it [5] to arbitrary surfaces using Turk's retiling algorithm for generating mesh pyramids. Correspondingly Turk used a slightly different approach to 3D texture synthesis [6] based on Wei & Levoy's 2D algorithm. Ying [7] proposed two 3D texture synthesis methods based on Wei & Levoy's 2D algorithm and Ashikhmin's [8] algorithm. Her implementation handled multi-resolution sub-division of surfaces and performed well for highly structure patterns.

The motivation to our work comes from the present requirements for progressive texture synthesis on 3D surfaces, which is a result of the extensive use of bandwidth limited transmission media in modern application domains such as remote visualisation, distributed/collaborative gaming etc. To this extent, the use of Greg Turk's popular work on texture synthesis on 3D surfaces (see section 2.1 for details) is limited, due to its ability to only synthesize textures at certain fixed resolution levels determined by the pyramidal decomposition of the sample texture. To overcome this limitation and to further improve its applicability, in this paper we propose a progressive texture synthesis method that replaces the pyramidal decomposition of sample texture adopted by Turk, by an EZW (embedded zero-tree wavelet) decomposition, capable of prioritising the DWT coefficients according to their visual significance. We illustrate that the proposed novel algorithm is capable of creating seamlessly varying quality levels of synthesized texture on 3D surfaces.

For clarity of presentation the paper is organised as follows: Section-2 introduces the reader to the basics of Turk's algorithm, i.e., *texture synthesis on surfaces* [6], multiresolution DWT representation of texture and Shapiro's EZW algorithm [10] for DWT coefficient prioritisation. Section-3 presents our proposed progressive

texture synthesis algorithm. Section- 4 provides experimental results, their analysis and possible application domains of the proposed algorithm. Finally, Section-5 concludes, with an insight to possible improvements and future variations.

2. THEORETICAL BACKGROUND

For the purpose of clarity and ease of reference we summarise the methods/contents of [6], [10] and [11] in this section. Hence readers who are familiar with these concepts can forego reading this section.

2.1 Texture synthesis on surfaces

This algorithm is based on the principle of texture synthesis on surfaces, independent of parameterization and use of neighbourhood search [6]. The method draws upon texture synthesis methods that uses image pyramids [9] for texture decomposition/representation, but uses a mesh hierarchy to serve in place of a pyramidal structure. Firstly a hierarchy of user defined number of sample points from a low to high density are created in a random order, over the given surface (Figure1 (a)). Using a point repulsion method, the points are subsequently repulsed and separated from each other to uniformly distribute over the surface. By connecting these points, a mesh is formed on the surface. Likewise a hierarchy of meshes are created on the surface, resembling an image pyramid representation. Subsequently a user specified vector field that indicates the orientation of texture patterns is created over the surface. Mesh vertices are sorted in such a manner that the vector field will be followed when visiting the points (see Figure1 (b)). Then for each mesh vertex, a local parameterization of the surrounding vertices is established. Using this parameterization a small rectangular neighbourhood with the vertex as the centre is created. Each point is then scanned to determine the colour. The colour of particular point is established by examining the colour of neighbouring points, and finding the best match to a similar pixel neighbourhood in the given sample texture image. A multi-level representation of the synthesised texture on the arbitrary surface is finally generated as shown in Figures, 1 (c), (d), (e), (f).

Note that in figure 1, (c), (d), (e) and (f), the number of points on the surface are, 4K, 16K, 64K and 256K respectively. The limitation of Turk's algorithm is that at the above fixed resolutions of the surface shape, the synthesized textures are also fixed in resolution. In other words the surface texture on the 'Bunny' illustrated in figure 1(f) is fixed, i.e. the one illustrated. However in a bandwidth constraint environment one would have been forced to synthesize a lower quality texture as compared to that of fig 1(f). In the proposed algorithm we provide an effective means for providing this flexibility.

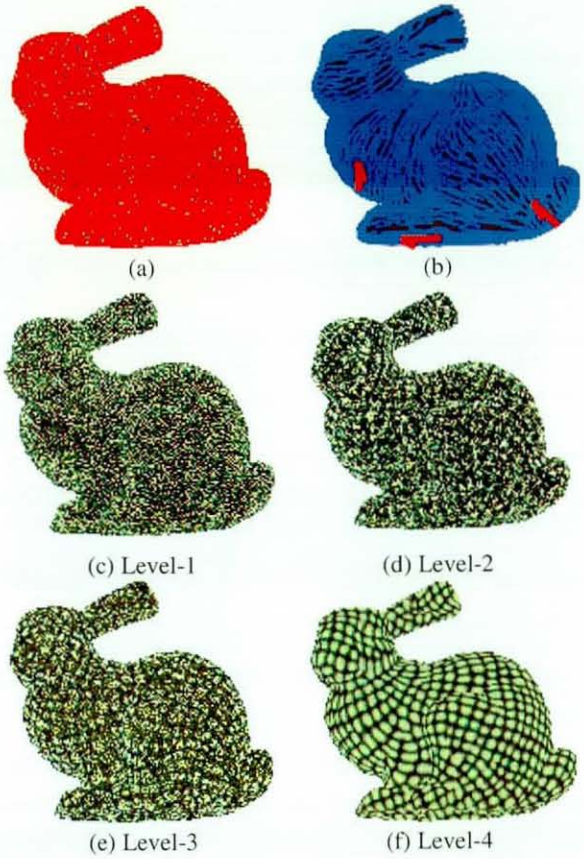


Figure 1: Turk's Method (a): Dense points on surface (b): Vector field creation on the surface (c) (d) (e) (f): Multi-level texture synthesis

2.2 DWT representation of texture

Textured images contain a large amount of perceptual data. Therefore the number of bits required to represent/encode a texture image is high. However typical images consist of a wide range of frequency components spread throughout the human visual frequency band. Some of theses frequency components have a significant effect in human perception while some others have very low significance. Fortunately texture images are often of this type.

The Discrete Wavelet Transforms provide a compact multi resolution representation of an image. It gives a signal representation in correspondence to a narrow band, low frequency range and some of the coefficients represent short data lags corresponding to a wide band, high frequency range. Using the concept of scale, data representing a continuous trade off between space and frequency can be made available for further processing.

In our algorithm we use two-dimensional DWT. To begin with, the texture image is subdivided in to four sub-bands using horizontal and vertical DWT filters over the image pixels. The resulting sub bands labeled LH_1,

HL_1 and HH_1 represent the finest scale wavelet coefficient where as the sub-band labeled LL_1 represents low resolution coefficients. In order to obtain the next level of wavelet sub-bands, the sub band labeled LL_1 is further decomposed and sampled using the vertical and horizontal DWT filters. This process is repeated until the required final decomposition is reached (see figure 2). The coefficients of the sub-bands are then prioritized using the EZW algorithm presented next.

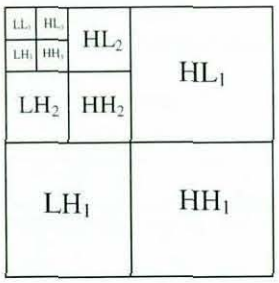


Figure 2: Application of three level discrete wavelet decomposition

2.3 Embedded Zerotree Wavelet (EZW) algorithm

The *Zerotrees of wavelet coefficient* concept was originally introduced by Shapiro [10] in progressive encoding of images. It is based on two important observations:

1. Natural images in general have a low pass spectrum. Therefore when an image is wavelet transformed the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average be smaller in the higher subbands than in the lower subands.
2. Large wavelet coefficients are visually more important than smaller wavelet coefficients.

EZW provides a compact representation of perceptually significant coefficients and multi resolution construction capability of an image. The idea is to organize DWT coefficients of an image (see Figure 2) in a prioritised order of visual significance, depending on their position and magnitude in the DWT decomposition and to subsequently encode the ordered list of coefficients following an embedded coding algorithm. In an embedded coding algorithm the encoder can terminate the encoding at any point there by allowing a target bit rate or target distortion metric to be met exactly. On the other hand, given a bit stream, a decoder can cease decoding at any point in the bit stream. Thus it is capable of producing exactly the same image that would have been encoded at the bit rate corresponding to the truncated bit stream.

In this paper we use the EZW algorithm's initial coefficient prioritisation procedure to prioritise their use within Turk's algorithm. Due to space limitation we refer readers interested in the detail of the EZW coefficient prioritization algorithm to [10]. We show that the visually prioritized availability of coefficients enables seamless progressive texture synthesis capability in Turk's technique, as against its original capability of synthesizing textures at only given discrete quality levels. This is the main contribution of our present work.

3. PROGRESSIVE TEXTURE SYNTHESIS

Figure 3 illustrates the block diagram of the proposed progressive texture synthesis algorithm on surfaces. [Note: the complete, progressive texture coding, transmission and synthesis process is illustrated].

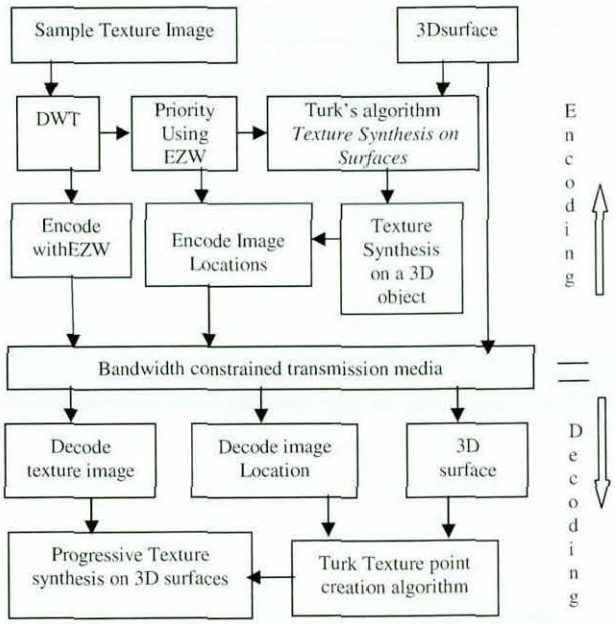


Figure 3: proposed progressive texture transmission algorithm

Note that figure 3 resembles the block diagram of Turk's algorithm [6], though with two important differences. Firstly in our design, the Gaussian pyramid based texture representation used by Turk is replaced by a DWT representation supported by an EZW based coefficient prioritisation scheme as illustrated by the modules named, 'DWT' and 'Priority Using EZW', in figure 3 and detailed in figure 4. This ensures seamless texture representation capability as against discrete quality level texture representation of Turk's original algorithm. In addition texture encoding via EZW enables embedded texture decoding capability allowing almost any intermediate texture quality to be readily reconstructed/ made available at the receiver depending on bandwidth constraints.

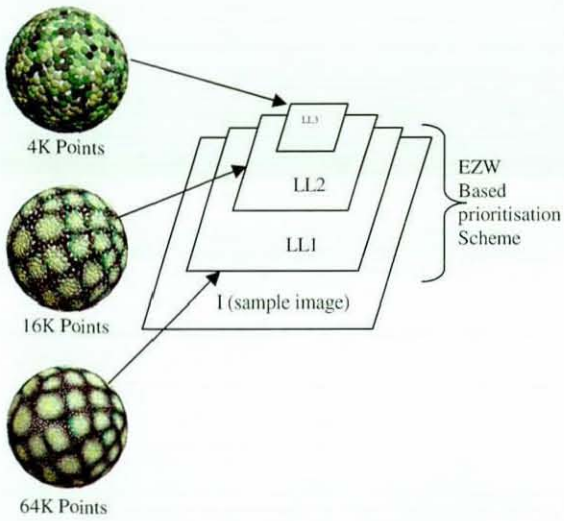


Figure 4: DWT and EZW based prioritisation scheme

Secondly in the proposed method, while synthesising texture using the modified (with EZW) Turk's method, we maintain a record of sample image locations from where each 3D surface point is textured (see section 2.1). These are denoted as *'point image locations'* (see figure 5). These records are maintained for each representation 3D surface shape. In order to avoid any searching at the decoder end and enable direct texture mapping from the received sample image, we propose the suitable coding of the 'point image locations' and their transmission. We have adopted differential pulse code modulation which provides an effective means of coding this high redundancy information. However if one wishes a texture synthesis to be carried out at the decoder as well, this extra information will be redundant and thus do not have to be transmitted. We suggest that the decision on whether or not the 'point image locations' needs to be transmitted to be taken depending on whether the bottle-neck is in the transmission bandwidth or the receiver capability.

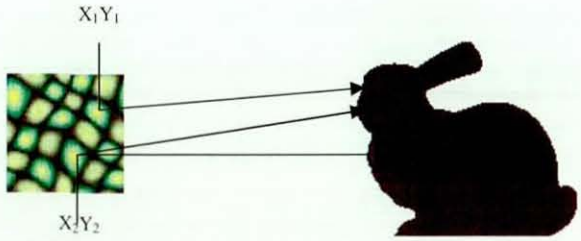
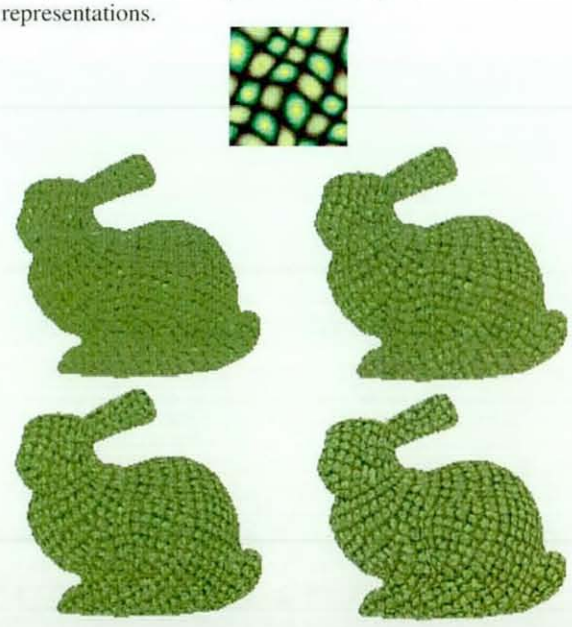


Figure 5: Single and multiple mappings of 'point image locations' between the sample and 3D surface

At the receiver, all texture sample information transmitted using EZW encoding is decoded to generate the texture sample at seamless quality settings. As more bits are received by the decoder better quality textures are produced. At any given instance of time and bit rate, using the re-generated texture sample image (i.e. via EZW decoding of texture), and separately received (and decoded) point image locations and the recovered 3D surface shape, the texture pixel value (colour) of each 3D point can be obtained. When the pixel values of all 3D points are obtained the 3D object will be completely textured with the corresponding instantaneous texture quality at the given 3D surface representation level, i.e. number of vertex points accuracy to which the 3D surface is defined at the given moment. When more bits representing the texture information are received, the available texture quality at the decoder improves and hence the accuracy of texture representation on the 3D surface can be improved. Thus the process ensures seamless quality texture synthesis capability on 3D surfaces.

4. EXPERIMENTAL RESULTS & ANALYSIS

In order to analyse the performance of proposed algorithm experiments were performed on a widely used set of test texture samples and the 3D object bunny. The results illustrated in figure 6 and 7 indicate the ability of the proposed technique to synthesize texture at seamlessly different levels (compare with figure 1). Although for practical reasons we have limited the number of texture synthesis examples to eight the proposed method is capable of progressive texture transmission aimed at seamless texture mapping. Note that our demonstration here is for a set of 256,000 (256K) points on the surface of the Bunny. In general, the proposed algorithm is capable of providing any intermediate texture level from texture-less to the quality of texture depicted in figure-1 for 4K, 16K, 64K, 256K point accuracy of the surface representations.



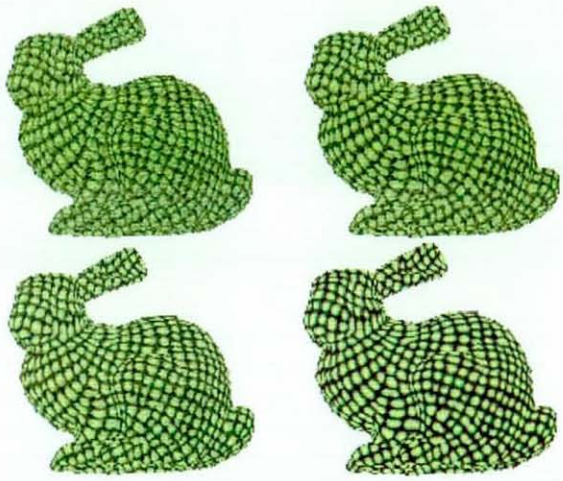


Figure 6: Progressive texture synthesized on the bunny by proposed algorithm using texture sample-1.

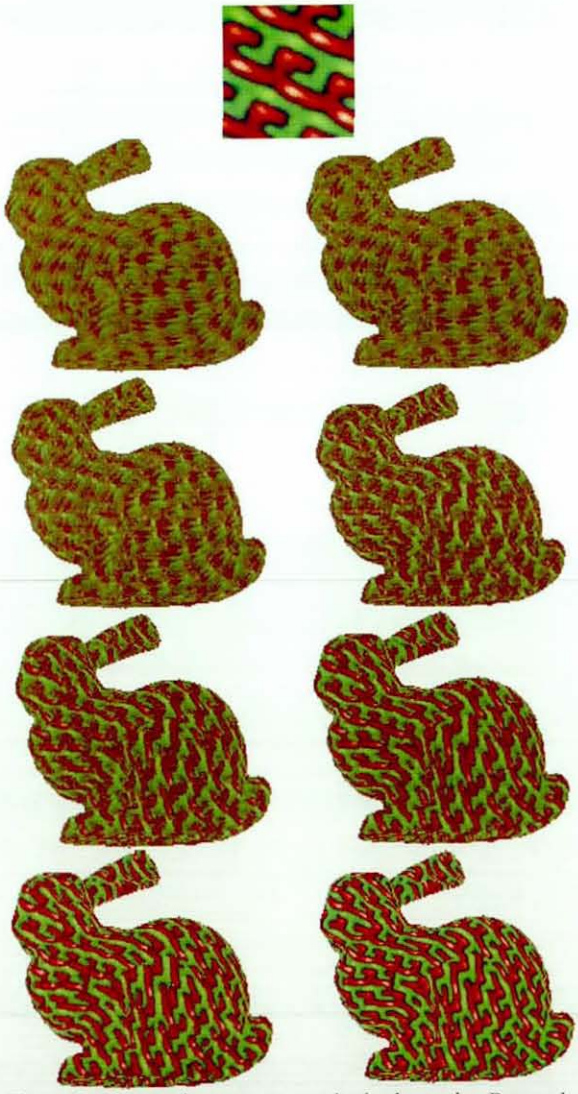


Figure 7: Progressive texture synthesized on the Bunny by proposed algorithm using texture sample-2.

The nature of the design and implementation of the proposed design enables its use in application domains where existing texture synthesis algorithms perform ineffectively due to functional constraints. The following is a summary of applications that could benefit from the specific progressive/ multi-resolution design of the proposed algorithm.

Progressive 3D texture transmission: Within a progressive transmission scenario, data is transmitted according to visual and/or decoding significance. Special design of the proposed texture synthesis algorithm allows DWT coefficient significance based progressive creation, transmission and reconstruction of the synthesized texture.

Texture mapping of progressively transmitted 3D structures: MPEG4 AFX has standardised MESHGRID for progressive transmission of 3D structure. Initially they transmit data sufficient for a coarse representation of the 3D structure/object-shape. Our design can complement this effort by texturing the progressive reconstructed surface with a matching, minimal transmission of texture data. [Note: progressive synthesis of texture has not been standardized by MPEG-4 AFX as yet]. Thus, both the structure as well as the texture can be refined progressively with more data transmission.

Compressed domain texture synthesis: Synthesizing a compressed output texture with the use of a compressed original texture sample. This is useful in fast, on demand applications.

5. CONCLUSION & FUTURE WORK

We have introduced a novel approach to texture synthesis based on the representation and analysis of texture in the DWT domain. We have provided experimental results illustrating the methods functionality and advantages. The proposed method has the capability of synthesizing texture at seamlessly different quality settings, a functionality which was not possible via existing state-of-the-art techniques. The use of visual prioritisation of information in the sample image during texture synthesis allow the tasks to be carried out at a higher speed at equivalent visual quality levels, as compared to other techniques.

We are currently in the process of extending our work towards MESHGRID coding and algorithmic optimisations. In particular the use of the proposed algorithm as a means of providing progressive quality texture to the already standardized progressive shape coding with MESHGRID within MPEG-4 AFX, is being considered.

ACKNOWLEDGMENTS

We wish to thank Greg Turk for providing his texture synthesis on surface source code.

REFERENCES

[1] Neyret F., and Cani, M.P., Pattern-based texturing revisited, *Proc. of SIGGRAPH 99*, 235–242, 1999.

[2] Praun E., Finkelstein, A., and Hoppe, H., Lapped textures, *Proceedings of SIGGRAPH00*, 465–470, 2000.

[3] Soucy, Marc,G Godin and Marc Rioux, A Texture-Mapping Approach for the Compression of Colored 3D triangulations, *The Visual Computer,Vol.12*, No.10,503–514, 1996.

[4] Wei L.-Y., and Levoy, M., Fast texture synthesis using tree-structured vector quantization, *Proc. of SIGGRAPH2000*, 479–488, 2000 (July)

[5] Wei L.-Y., and Levoy, M., Texture synthesis over arbitrary manifold surface, *Proceed. of SIGGRAPH2001*, 355–360, 2001.

[6] Turk, G., Texture synthesis on surfaces, *Proc. of SIGGRAPH2001*, 347–354, 2001(August).

[7] Ying, L., Hertzmann, A., Biermann H., and Zorin, D., 2001, Texture and shape synthesis on surfaces, *Eurographics Rendering Workshop*, 301–312, 2001.

[8] Ashikhmin, M., Synthesizing natural textures, *ACM Symp. on Interactive 3D Graphics (March)*, 217–226, 2001.

[9] Heeger, David J. and James R. Bergen, Pyramid-Based Texture Analysis/Synthesis, *Comp. Graphics Proc., Annual Conf. Series (SIGGRAPH 95)*, Aug., 229–238, 1995.

[10] J. M. Shapiro, Embedded Image Coding Using Zerotrees of Wavelet Coefficient", *IEEE Trans. Signal Processing, December, VOL 41(no.12)*, 3445-3462, 1993.

[11] Wickramanayake, D.S., Edirisinghe, E.A., Bez H.E., A wavelet based image quilting approach to fast, multiresolution texture synthesis, *Proc. of IEE VIE 2005*, 93-100, 2005.

A Wavelets Based Max-Flow/Min-Cut Approach For Texture Synthesis

Rupesh N. Shet*, Helmut E. Bez*, Eran A. Edirisinghe*

*Department of Computer Science,
Loughborough University, UK
rupesh_shet@hotmail.com

Keywords: DWT, Texture Synthesis, Max-Flow/Min-Cut, Wavelets

Abstract

In this paper we have proposed a Discrete Wavelet Transform (DWT) based multi-resolution max-flow/min-cut algorithm that is able to provide the best possible minimum cutting path in patch based texture synthesis. A coarse cutting path is initially generated using the lowest resolution sub-band of the DWT decomposition and subsequently refined using the coefficients of the higher resolution sub-bands. The Embedded Zero-tree Wavelet (EZW) concept has been used to prioritise and use the DWT coefficients according to their impact on visual quality, thus optimising the visual quality at patch boundaries. Throughout the algorithm design, special consideration has been given to minimizing the computational cost. We have shown that the adaptation of max-flow/min-cut in wavelet domain results in an efficient texture synthesis algorithm that is capable of being used in conjunction with modern, band-width adaptive applications. A set of regular to stochastic test textures have been used to prove the effectiveness of proposed algorithm and to compare them with existing state-of-art techniques.

1 Introduction

Texture Synthesis has attracted significant research interest due to its wide range of application areas, such as computer graphics, games, animations and digital movies. As a result many algorithms have been proposed up to date. These algorithms are broadly classified in to two groups namely; pixel based and patch based algorithms. Recently a new trend of research in transform domain texture synthesis has been commenced [12,13,14]. These approaches have applications in the distributive and collaborative gaming and other applications that use limited bandwidth channels, such as 3D mobile graphics, where only a small amount of information is used. The recent key contributions in transform domain texture synthesis have been made by Wickramanayake et al. [12,13,14]. These algorithms perform well in a large variety of textures at high operational speeds. However in synthesizing some regular and near-regular textures which often possess sharp edges, due to the non-optimised weighted edge blending technique used, patch boundaries show blurring artefacts. As an effective solution to this shortcoming, in this paper we have proposed a novel, optimized technique for edge blending, based on the popular max-flow/min-cut idea in graph theory [1,10]. It is similar to the approach used by Kwatra et al. in their work of [7] and improves the visual quality of edges obtainable by Wickramanayake et al.'s recent transform domain texture synthesis algorithms. This improvement enables the work of [12,13] to be used in a much wider range of textures.

Initial work on texture synthesis commenced a decade ago. However, Effro's and Freeman's popular algorithm of [3] is widely accepted as the seminal contribution to the current state-of-the-art, pixel domain, texture synthesis algorithms. The research in [3] introduced the use of block based texture synthesis and a minimum boundary edge cutting (MBEC) technique to optimize the quality at overlapping regions of texture blocks. Since the publication of [3] many extensions and improvements to it has been proposed in pixel domain texture synthesis. The main drawback of Effro's and Freeman's work that attracted attention of [12,13] is the speed of texture synthesis. Wickramanayake et al. implemented block based texture synthesis in transform domain as a solution to increasing the texture synthesis speed. In these techniques Shapiro's embedded zero-tree wavelets (EZW) concept [11] is used for limiting the number of coefficients used in the texture synthesis process, thereby reducing the computational cost of texture synthesis. In addition a simple weighted edge blending technique is used for artefact reduction at block boundaries. The use of visually significant coefficients in selecting blocks for synthesis enables the matching blocks to be found with minimum computational cost. However our detailed experiments revealed that the simple, non-optimised, weighted edge blending technique used to improve the quality at block edges needs improvement, for the overall texture synthesis algorithm's best effectiveness in synthesising a wide variety of textures. In [7], Kwatra et al. proposed the use of their popular graphcut technique of texture synthesis that uses max-flow/min-cut algorithm [1,10] to optimise the quality at patch edges. They demonstrated that the graph cut algorithm is capable of better quality texture synthesis as compared to most other patch/block based texture synthesis techniques, such as [4,6,8]. To speed up the searching process, a FFT based acceleration technique was adopted. It has been shown that transform domain texture synthesis has the ability to perform faster texture synthesis, as only a limited set of transform coefficients can be used in synthesis [12,13]. Extending Kawtra's algorithm directly to transform domain is however very expensive due to repetitive iterations required to refine the texture quality.

We have analyzed both Wickramanayake's [12,13] and Kwatra's [7] algorithms and propose a novel algorithm which combines the merits of both. We use transform domain searching for best block selection in block patching (as in [12,13]) and a novel transform domain max-flow/min-cut technique for edge artefact reduction, which is a transform domain extension to the max-flow/min-cut technique used in Kwatra's work of [7].

For clarity of presentation the paper is organised as follows: Apart from this section which is an introduction to the problem domain to be addressed, section-2 presents the proposed novel algorithm in detail. Section-3 provides experimental results, their analysis and possible applications of the proposed algorithm. Finally, Section-4 concludes, with an insight to possible improvements and future variations.

2 Methodology

This section provides design and implementation details of the proposed transform domain max-flow/min-cut technique which is capable of enhancing the overall synthesized texture quality of [12,13].

In [12,13] a n level DWT is initially applied on the sample texture (see Figure-1) which results in a DWT decomposed image. (see Figure-2 where n=3 decomposition is applied on sample image).

Figure 1: Sample Texture

Figure 2: n level decomposed sample image

Texture synthesis requires patching blocks with seamless boundaries. Once a block has been synthesized, searching for it's best matching block (to be patched as the subsequent block to be synthesized) is carried out by effectively using the coefficients of the LLn and HHn sub-bands [12,13]. The coefficients in the sub-bands can be prioritised according to their visual impact, providing the ability to synthesize textures faster than traditional approaches.

Once a matching pair of adjacent blocks is selected their overlapping boundary should be further refined to remove edge artefacts. In general, given the fact that a certain amount of texture would have been already synthesized, synthesizing a new block will require the consideration of an L-shaped boundary region. For clarity of presentation we have illustrated a typical overlapping region in Figure-3 and its transform domain representation in Figure-4.

The idea of max-flow/min-cut algorithm is to make a cut in the overlapping region that results in minimising any potential seam artefact. In the proposed transform domain extension to the max-flow min-cut algorithm, the process begins with a sub-ordinate pass (see section 2.1) which gives an approximate cutting path. This path is further refined using a subsequent refinement pass (see section 2.2).

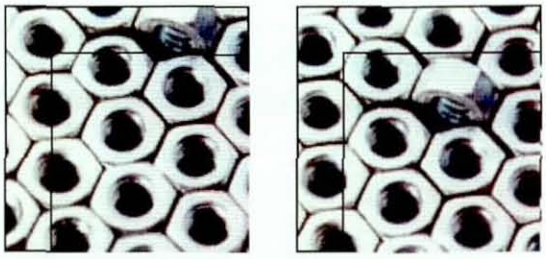


Figure 3: Pixel image of overlapping region.

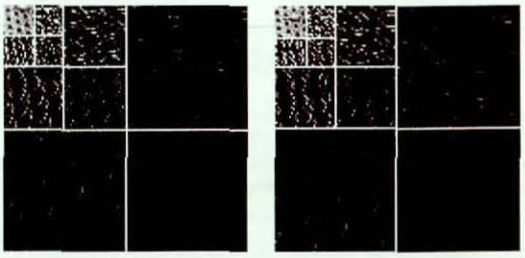
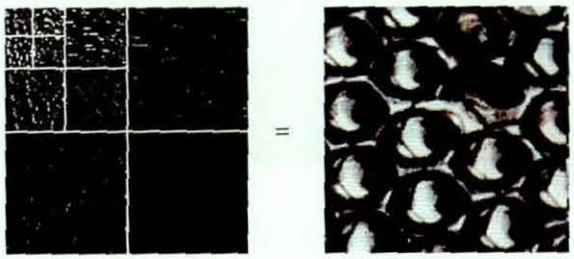


Figure 4: Transform domain overlapping images

2.1 Sub-Ordinate Pass of Patch Fitting

The first step of the sub-ordinate pass is to generate the 'difference block', which is the difference between transform domain overlapping blocks as illustrated in Figure-5(a). For clarity of presentation and explanation, figure-5(b) represents the 'difference block' in pixel domain.



(a) (b)

Figure 5: Difference Image

In our experiments, after a 3-level decomposition, all ten sub-bands are initially extracted. The max-flow/min-cut algorithm is then applied on the overlapping region using the lowest resolution sub-bands, i.e. LL_3 bands to obtain the sub-ordinate cutting path. (see Figure-6). Readers who are interested in the details of the max-flow/min-cut approach are referred to [1,5,7,10].

Figure6: Cut on Low Level Band (Note: LL*n* has been zoomed for clarity).

The cutting path obtained using the sub-ordinate pass is called the 'primary cut'. Considering the 'primary cut' as the cutting path, it is then used to cut the sub-bands, HH_3, LH_3, HL_3. Subsequently using inverse-DWT on sub-bands LL_3, HH_3, LH_3, HL_3, the sub-band LL_2 is obtained. (see Figure 7). For ease of reference, the cut obtained in the LL_2 sub-band is named the 'incorporated path'.

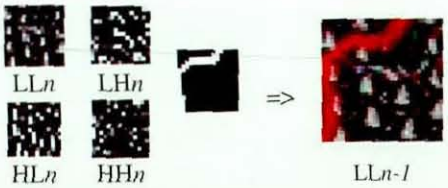


Figure 7: LL_2 with incorporated cut (Note: LL*n*, LL*n*-1 has been zoomed for clarity).

2.2 Refinement Pass of Patch Fitting

The primary cut is further refined by considering only the neighbouring coefficients around it in the subsequent levels of inverse discrete wavelet transform i.e. IDWT, starting from the incorporated path of LL_2. In practice this refinement process is continued through all the IDWT levels. It is noted here that the process of refinement passes through all n levels of decomposition. Therefore the number of refinement steps is n, i.e. n =3 in our experiments.

Figure-8(a1) illustrates that the incorporated cut in LL_{n-1} is first refined by max-flow/min-cut using neighbourhood coefficients (see Figure-8 (a2)). This refined cut is then incorporated in to the three detailed sub-bands of the same resolution level. The incorporated cut at level LL_{n-2}, i.e. LL_1 is obtained via inverse DWT of the four sub-bands (see Fig. 7). The incorporated cut at level LL_1 is illustrated in figure-8(b1). This process is repeated through all the levels until the refined path at level LL_0 (full resolution) is obtained (see Fig-8(c2)).

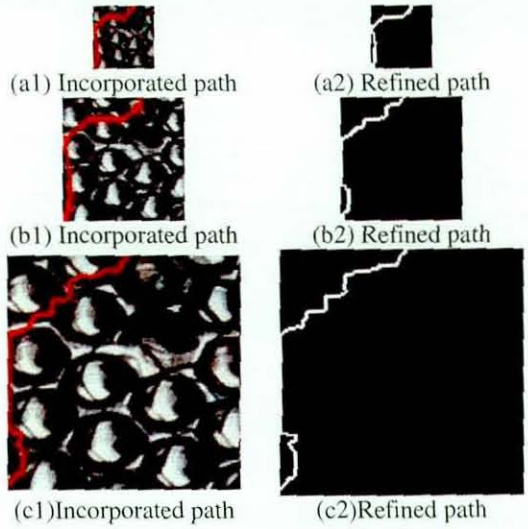


Figure 8: Refinement of primary cut using corresponding bands and neighbouring coefficients.

The cutting path produced by following the above process results in a final blended quality similar to that illustrated by figure 9.

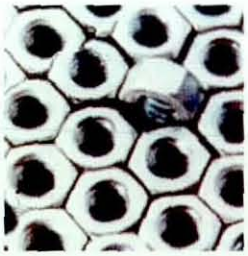


Figure 9: Final blended block

2.3 Feathering

Although the max-flow/min-cut approach produces an optimum seam, the resulting quality is restricted by the quality of the original edge blocks. Therefore it is likely that some artefacts still remain after the application of the max-flow/min-cut approach. To address this problem we used the popular feathering [2] approach that uses suitable weighting factors to combine those pixels at the seams. Feathering results in a further improvement of the seam quality.

3 Experimental Results and Analysis

Experiments were carried out on a large database of commonly used test texture samples representing all textures

on texture spectrum [9] i.e. regular to stochastic. This section discusses experimental results in details.

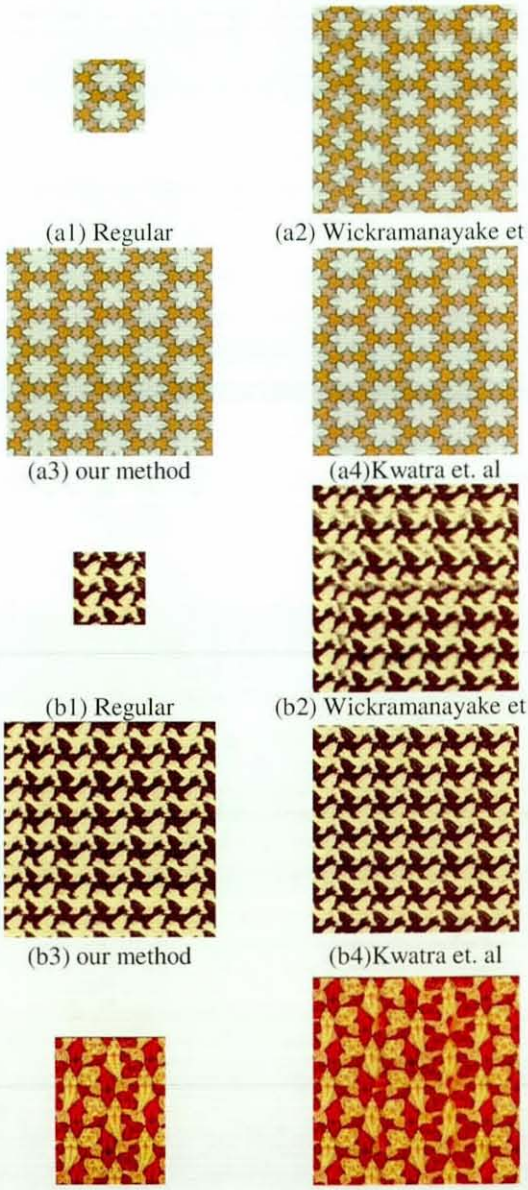
The Figure 10 illustrates a comparison between edge blending approach used in [12,13] and the proposed approach. To clearly illustrate the possible improvements, a block selected for demonstration purposes above was one in which the edge-blending algorithm would perform sub-optimally (note the artefacts at the top and front rows of figure 10(a).

(a)Edge blending (b)Our algorithm

Figure 10: Comparison between edge blending and proposed transform domain max-flow/min-cut algorithm.

For a more comprehensive analysis we compare the results of the proposed algorithm against Kwatra et.al's [7] pixel domain max-flow, min-cut algorithm and Wickramanayake et al's [12,13] transform domain edge-blending algorithm. The results clearly illustrate that the proposed algorithm provides better visual quality at block edges as compared to using the non-optimized edge-blending algorithm adopted by Wickramanayake et al. Though the algorithms of [12,13] provide faster texture synthesis as compared to Kwatra et al.'s [7] algorithm, it was stated previously that the simple edge-blending approach adopted was non-optimum. However Kwatra et.al's algorithm of [7] used the max-flow, min-cut algorithm, which is capable of providing the optimum seam between two matching blocks in the pixel domain. Our extension of this algorithm to the transform domain has enabled its application in Wickramanayake et. al's fast texture synthesis algorithm. The detailed experimental results show that the proposed approach provides equivalent visual quality at block boundaries, at a much lower computational cost, when compared to the pixel domain max-flow, min-cut algorithm adopted by Kwatra.

Figure-11 illustrates the texture synthesis results obtained for 9 texture samples of different nature, when using the proposed, Wickramanayake et al's [12,13] and Kwatra et al's [7] algorithms. Figure-11 (a1, b1) are examples of regular sample textures. Carefully observing the results of texture synthesis obtained when using the proposed technique illustrated in Figure-11(a3,b3)), we can conclude that the proposed technique performs better than Wickramanayake et al's (see Figure-11(a2,b2)) and similar to Kwatra et.al's (see Figure-11(a4,b4)) algorithms. Figure-11(c1, d1, e1, f1) are near-regular sample texture where the proposed algorithm (see Figure-11(c3,d3,e3,f3)) perform better than Wickramanayake et.al's algorithm (see Figure-11(c2,d2,e2,f2)) and similar to Kwatra et.al's algorithm (see Figure-11(c4,f4) and better than Kwatra et.al's algorithm (see Figure-11(d4,e4)) in certain other cases. Note that the rare possibility of an improved quality texture as compared to Kawtra's approach, when using the proposed multi-resolution, transform domain approach, exists due to the non-consideration of noise and unnecessary texture details in matching, when the proposed approach is used. Figure-11(g1, h1) are irregular sample textures where the proposed algorithm (see Figure-11(g3,h3)) performs better than Wickramanayake et al's (see Figure-11(g2,h2)) and similar to Kwatra et.al's (see Figure-11(g4,h4)) algorithms. Figure-11(i1) are stochastic sample texture where the proposed algorithm (see Figure-11(i3)) perform similar to Wickramanayake et al's (see Figure-11(i2)) and to Kwatra et al's (see Figure-11(i4)) algorithms.



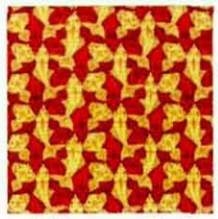
(a1) Regular (a2) Wickramanayake et al.

(a3) our method (a4)Kwatra et. al

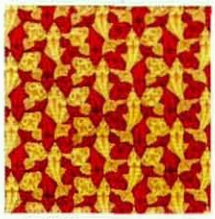
(b1) Regular (b2) Wickramanayake et al.

(b3) our method (b4)Kwatra et. al

(c1) Near-regular (c2) Wickramanayake et al.



(c3) our method



(c4)Kwatra et. al

(d1) Near-regular

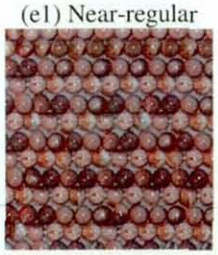
(d2) Wickramanayake et al.

(d3) our method

(d4)Kwatra et. al

(e1) Near-regular

(e2) Wickramanayake et al.



(e3) our method

(e4)Kwatra et. al

(f1) Near-regular

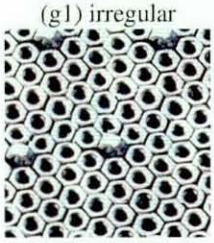
(f2) Wickramanayake et al.

(f3) our method

(f4)Kwatra et. al

(g1) irregular

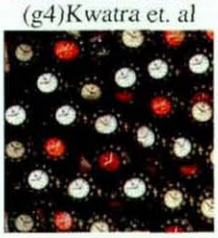
(g2) Wickramanayake et al.



(g3) our method

(g4)Kwatra et. al

(h1) irregular



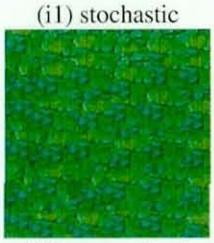
(h2) Wickramanayake et al.

(h3) our method

(h4)Kwatra et. al

(i1) stochastic

(i2) Wickramanayake et al.



(i3) our method

(i4)Kwatra et. al

Figure 11: Comparison of proposed algorithms results with Kwatra et.al's and Wickramanayake et al's algorithms.

The replacement of edge-blending algorithm adopted by Wickramanayake et. al. in [12,13] by the proposed transform domain max-flow, min-cut algorithm, provides a fast and efficient, fully transform domain texture synthesis algorithm with optimum quality at edges. Such a texture synthesis

algorithm has applications in many practical domains where existing algorithms perform ineffectively due to functional constraints. Some examples of suc applications areas are:

Compressed domain texture synthesis: Refers to the syntheses of a compressed output texture with use of a compressed original texture sample. This is a useful feature in fast, on demand applications, with resource/device constraints such as mobile and hand held devices.

Computer Games: Transform domain texture synthesis can be used in generating rendered textures of surroundings, such as walls, background or foreground objects, in which the quality of the rendered texture is kept minimum depending on its required visual significance at any given time.

MPEG4: MPEG4 AFX uses Bézier surfaces to model and animate (describe smooth motion) three dimensional objects. Ttransform domain texture synthesis approaches can be easily adopted to provide progressive, surface rendering to these objects.

4 Conclusion

In this paper we have proposed a transform domain extension to the max-flow/min-cut algorithm, popularly used in edge enhancement of patch based texture synthesis algorithms. We have provided experimental results and detailed analysis to prove that proposed technique is capable of being used to replace the simple non-optimised edge-blending strategy adopted by Wickramanayake et al. in their pioneering work of transform domain texture synthesis, thus further improving its efficiency and extending it's application.

In designing the proposed multi-resolution, transform domain extension to the max-flow/min-cut framework, we have given special attention toward reducing algorithmic complexity by introducing refinement passes in progressively improving the visual quality at block edges. We are currently looking at the application of Wickramanayake et al's transform domain texture synthesis algorithm, enhanced by the proposed max-flow, min-cut algorithm, to the texture synthesis of full, arbitrarily shaped geometric surfaces.

Acknowledgement

Authors wish to thank Vivek Kwatra for providing valuable discussions, the source code and sample textures used in his original work.

References:

[1] Y. Boykov, O. Veksler and R. Zabih, " Fast approximate energy minimization via graph cuts" *In IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 11, pp. 1222-1239, 2001.

[2] P.J. Burt, E. H. Adelson "A multiresolution spline with application to image mosaics". *ACM transactions* 2, 4 (12 1983), 217–236. ISSN: 0730-0301.

[3] A.Efros, W.T. Freeman, "Image quilting for texture synthesis and transfer". *In Proc. of SIGGRAPH* 2001, pp. 341–346.

[4] A.Efros and T.Leung. "Texture synthesis by non-parametric sampling", *International Conference for Computer Vision*, Vol. 2, September 1999, pp. 1033-1038

[5] L. Ford and D. Fulkerson," *A Flow in Networks*" Princeton University Press , 1962

[6] D.J.Heeger and J.R.Bergen. "Pyramid-Based texture analysis/synthesis". *Proc. SIGGRAPH '95*, Aug. 1995, pp. 229-238

[7] Vivek Kwatra , Arno Schödl , Irfan Essa , Greg Turk and Aaron Bobick 2003, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts", *ACM Transactions on Graphics, vol 22, 3* (2003), pp 277 286. (SIGGRAPH 2003 proceedings).

[8] L Ling, C E Liu, Ying Xing, Baining Guo, and Heung-Yeung Shum. "Real Time Texture synthesis by patch based Sampling" *ACM Transaction on Graphics, Vol 20, No 3*, July 2001, pp.127-150

[9] Wen- Cheieh Lin, James Hays, Chenyu Wu, Vivek Kwatra, Yanxi Liu," A comparison study of four texture synthesis algorithm on near-regular textures". *In Tech.Report CMU-RI-TR-04-01, Robotics Institute*, Carnegie Mellon University (2004).

[10] R. Sedgewick , "*A logorithms in C part 5: Graph Algorithms*." Addison-Welsley, Reading, Massachusetts, 2001

[11] J. M. Shapiro, Embedded Image Coding Using Zerotrees of Wavelet Coefficient", *IEEE Trans. Signal Processing, December*, VOL 41(no.12), 3445-3462, 1993.

[12] D.S Wickramanayake, E.A.Edirisinghe, H.E. Bez., "Multiresolution texture synthesis in wavelet transform domain", *The Journal of Imaging Science and Technology*, January/February 2006, Vol.50, no.1, p93-102, ISBN/ISSN:1062-3701

[13] D.S Wickramanayake, E.A.Edirisinghe, H.E. Bez, "A wavelet based image quilting approach to fast, multi-resolution texture synthesis", *IEE International Conference on Visual Information Engineering (VIE 2005)*,(CP509), p.93-100, Glasgow, UK, 4-6 April 2005, ISBN: 0 86341 5075

[14] D.S Wickramanayake, E.A.Edirisinghe, H.E. Bez., "Transform Domain Texture Synthesis based on DCT" *In PREP 2005 proceedings*,pages 152-153, march 2005

PROGRESSIVE TEXTURE SYNTHESIS ON GEOMETRIC SURFACES PARAMETRISED BY BI-QUADRATIC RATIONAL BÉZIER PATCHES

Rupesh N. Shet[1], H. E. Bez[1], E. A. Edirisinghe[1]

[1] Department of Computer Science, Loughborough University, UK

Keywords: Texture Synthesis, Progressive Texture , Discrete Wavelet Transform, EZW, Rendering

Abstract

Progressive texture synthesis can provide an added functional advantage to existing texture synthesis algorithms, which are time consuming and fail to deliver in some application areas. To provide practical solutions to this challenge we have previously proposed a Discrete Wavelet Transform (DWT) based texture synthesis algorithm for 2D surfaces. In this paper we propose the extension of this approach to 3D progressive-texture synthesis. The proposed 3D progressive-texture synthesis algorithm makes use of the EZW (Embedded Zero-tree Wavelet) idea proposed by Shapiro, which is capable of prioritising the coefficients of a DWT decomposed image according to their visual significance. We demonstrate the use of the proposed algorithm on progressive texturing geometric surfaces such as Ring Dupin Cyclides, Toruses, Spheres, parametrised by rational Bézier patches. We provide experimental results to prove the effectiveness of the proposed approach, when synthesising textures of regular, irregular and stochastic nature. Further experimental results are provided to illustrate the practical use of the proposed progressive texture synthesis algorithm in resource constrained application domains.

1 Introduction

Texturing surfaces is the key to enhancing the realism of scenes, particularly in artificially created surfaces. As a result, a number of 'image based' texture synthesis and texture mapping algorithms have been proposed in the past decade. However all texture synthesis algorithms are challenged by the high statistical variability of textures involved in the synthesis. Therefore texture synthesis continues to be an open problem in computer graphics and has attracted much research interest due to popular applications in animated movie production, computer games, education, architecture, artwork and virtual productions.

A major proportion of texture synthesis research has focused on texture synthesis on planar surfaces. However, recently a number of approaches have been proposed for texture synthesis on 3D geometric and arbitrarily shaped surfaces [27, 24, 29, 2, 23, 13, 16, 20, 11, 10, 8, 18, 9]. Existing texture synthesis approaches can be broadly classified into two groups, namely, pixel based texture synthesis [26, 27, 24, 29, 2, 23, 9, 18] approaches and patch based texture synthesis [13, 16, 20, 11, 10, 8] approaches.

Pixel based texture synthesis is performed considering a pixel as the basic unit rather than a complete patch, where as the focus of our current research is on patch based texture synthesis. Patch based texture synthesis approaches attempt to synthesis texture by copying selected regions of pixels from a given sample texture and stitching them together. These approaches overcome some of the limitations of pixel based approaches as the techniques are relatively computationally inexpensive. The early work of Neyret-cani [13] proposed a technique that is based on pre-computed triangular texture samples. However, good synthesis results were obtained only for isotropic textures. Praun [16] introduced the lapped texture synthesis technique where oriented texture patches are placed in an overlapping fashion on a surface. However experiments were limited to a specific class of texture. Soler [20] introduced hierarchical texturing to overcome drawbacks of the previous algorithms. The basic idea was to capture low-frequency patterns while preserving high frequency randomness in texture. The generalised idea is to pick up texture patches from the sample image and map them onto the surface so that they fit with their neighbours. The running time is closely related to the number of resulting patches, which depends on both the mesh geometry and the texture sample. Sebastian et al [11] proposed an algorithm that separated a texture pre-processing step from an actual texture synthesis stage. Pre-processed texture is stored and used when needed. Access of the stored texture data is very slow and thus a look-up-table is used to speed up memory access. In [10] Wang et al proposed an algorithm based on global conformal parameterization of surfaces, where the textures are preserved on surfaces without seams or cracks. Moreover, the parameterization can segment the surface into patches, where each patch is mapped to a planar rectangle. The algorithm adopts a complex procedure in parameterization, and in creating multi-scale samples, leading to an increased computational cost in the synthesis process. Chi-Wing Fu et al [8] proposed a texture tiling mechanism, where a low distortion conformal quad based map is created for the input surface. It produces proper tile orientations on all quad faces so that texture can be laid out on quads and mapped back to the input surface accordingly. This texture synthesis algorithm is based on the *Image quilting* approach[1]

and therefore inherits all the drawbacks of image quilting algorithms such as, random selection of patches, expensive cost of minimum boundary cut, somewhat limited applicability to isotropic textures and computation cost due to the need of generating pre-synthesis textures.

All above texture synthesis techniques use a triangular mesh, which results in seams at edges.The size of triangles used in meshes also varies, which makes prominent the visual artefacts on the surface. Further to this, the graphical object will use extensive bandwidth in transmission since the surface mesh representation depicting the objects shape and texture information depicting its appearance are in uncompressed format. Therefore their applicability in constrained bandwidth channels is limited. Further, animation and deformation of triangular meshes is relatively difficult and costly. To overcome many of the above problems and to be used effectively in bandwidth limited transmission media, we propose an algorithm that enables fast, progressive-texture synthesis on bi-quadratic rational Bézier patches, allowing any geometric surface that can be parametrised by such patches to be completely textured. Within our present research, we have used a Dupin Cyclide [6, 15], a Sphere and a Torus as examples to demonstrate texture synthesis; these surfaces are widely used in computer graphics and in the development of a number of CAD tools. It is noted that the proposed are the first attempts of progressive or non-progressive texture synthesis on geometric surfaces generated using bi-quadratic rational Bézier equations.

The remainder of the paper is organised as follows: sections-2 and 3 summarise the fundamental theory of rational parametrisation and introduce the reader to the parametrisation of the surface of a Dupin cyclide(an example for study) by bi-quadratic rational Bézier patches. Hence readers who are familiar with these concepts can forego reading these sections. Section-4 presents the proposed novel progressive texture synthesis algorithm on surfaces along with detailed discussions on the underlying techniques used, such as multiresolution DWT representation of texture, texture synthesis on a single Bézier patch, using a modified graph-cut technique [25] to improve quality of seams between Bézier patches and the use of Shapiro's EZW algorithm [17] for DWT coefficient prioritisation in providing progressive synthesis of texture. Section-5 provides experimental results and their analysis. Finally, Section-6 concludes, with an insight to possible improvements and future variations.

2 Rational parametrisation

Rational parametrisation is a de-facto standard representation in computer graphics and geometric modeling software, allowing portability across applications and systems. In addition to possessing desirable geometric properties, rational parametrisation:

- requires the evaluation of only polynomial functions,
- gives rise to a compact data-structure,
- facilitates interactive control and animation of shape,
- is complete in the sense that approximation of any shape to a specified tolerance δ can be achieved, and exact parametrisation (i.e., $\delta = 0$) is often possible.

Rational parametrisations of surfaces comprise local atlases, or patches, of the form:

$$\tau(s,t) = \frac{\sum_{k,j=0}^{n,m} b_{n,k}(t) b_{m,j}(s) v^*_{k,j}}{\sum_{k,j=0}^{n,m} b_{n,k}(t) b_{m,j}(s) \omega_{k,j}}, \quad 0 < s,t < 1,$$

where $\omega_{k,j}$ are the weights and $v^*_{k,j}$ are the Bernstein vectors. If all the weights are non-zero this may be expressed as

$$\tau(s,t) = \frac{\sum_{k,j=0}^{n,m} b_{n,k}(t) b_{m,j}(s) \omega_{k,j} v_{k,j}}{\sum_{k,j=0}^{n,m} b_{n,k}(t) b_{m,j}(s) \omega_{k,j}}, \quad 0 < s,t < 1,$$

where $v_{k,j} = \frac{v^*_{k,j}}{\omega_{k,j}}$ are the Bézier vertices. The values of n and m determine the degree of the parametrisation; if $n = m = 2$ the patch is said to be bi-quadratic and if $n = m = 3$ it is bi-cubic.

Many of the desirable geometric properties of rational representation, eg the convex hull property and the existence of Bézier vertices, are lost if negative or zero weights occur - hence, in computer graphics and geometric modeling applications, positive weight parametrisations are always preferred. For computational efficiency, low degree parametrisations are desirable.

3 An Example: The parametrisation of Ring Dupin Cyclides by trigonometric functions

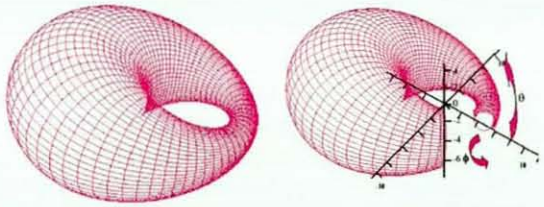
Dupin cyclides may be defined implicitly as:

$$(x^2 + y^2 + z^2 - \mu^2 + b^2)^2 - 4(ax - c\mu)^2 - 4b^2y^2 = 0;$$

where the parameters a, b and c satisfy $c^2 = a^2 - b^2$. The surfaces for which $c < \mu \leq a$ have found application in geometic modelling, see [12] and [14], and are known as ringed cyclides. Ringed cyclides with $c = 0$ are torii , and the ringed cyclides may therefore be regarded as generalised torii. Figure 1 (a) shows a ringed cyclide for which $a = 6$, $b = 4\sqrt{2}$, $(c = 2)$ and $\mu = 3$. Cyclides admit trigonometric parametrisations which may be written as:

$$\tau_u(\theta,\phi) = \frac{1}{a - c\cos\theta\cos\phi} \begin{bmatrix} \mu(c - a\cos\theta\cos\phi) + b^2\cos\theta \\ b(a - \mu\cos\phi)\sin\theta \\ b(c\cos\theta - \mu)\sin\phi \end{bmatrix}$$

for $0 < \theta < 2\pi$, $0 < \phi < 2\pi$. Figure 1 (b) shows the origins and directions of the angular parameters, θ and ϕ, on the surface.



(a) (b)

Figure 1: (a): The surface, (b): The trigonometric parametrisation τ_u of the surface

3.1 The rational parametrisation induced from τ_u

The restriction of the trigonometric parametrisation τ_u to the boundary defined by

$$\theta_0 \leq \theta \leq \theta_1 \quad \phi_0 \leq \phi \leq \phi_1$$

is written as $\tau_{u,\partial}$ and may be used to 'induce' rational parametrisations of the same surface patch as that parametrised by $\tau_{u,\partial}$. In particular it may be used to induce bi-quadratic rational Bézier representations - see [7] and [3]. The induction process discussed in [7] produces a rational bi-quadratic patch, that parametrises the same region of the surface as $\tau_{u,\partial}$, with weights, $w_{ij}, 0 \leq i, j \leq 2$, given by:

$$\begin{aligned}
w_{00} &= a - c\,x_{0\theta}x_{0\phi}, & w_{01} &= aw_{1\phi} - c\,x_{0\theta}x_{1\phi}, \\
w_{02} &= a - c\,x_{0\theta}x_{2\phi} \\
w_{10} &= aw_{1\theta} - c\,x_{1\theta}x_{0\phi}, & w_{11} &= aw_{1\theta}w_{1\phi} - c\,x_{1\theta}x_{1\phi}, \\
w_{12} &= aw_{1\theta} - c\,x_{1\theta}x_{2\phi} \\
w_{20} &= a - c\,x_{2\theta}x_{0\phi}, & w_{21} &= aw_{1\phi} - c\,x_{2\theta}x_{1\phi}, \\
w_{22} &= a - c\,x_{2\theta}x_{2\phi}
\end{aligned}$$

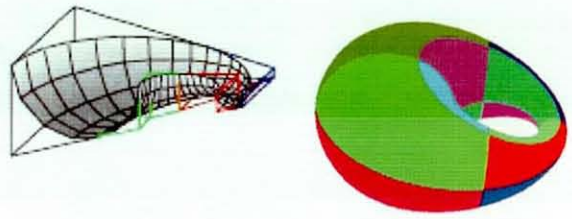
where

$$\begin{aligned}
w_{1\theta} &= \cos(\theta_1 - \theta_0)/2, & w_{1\phi} &= \cos(\phi_1 - \phi_0)/2, \\
x_{0\theta} &= \cos(\theta_0), & x_{0\phi} &= \cos(\phi_0), \\
x_{1\theta} &= \cos(\theta_0 + \theta_1)/2, & x_{1\phi} &= \cos(\phi_0 + \phi_1)/2, \\
x_{2\theta} &= \cos(\theta_1), & x_{2\phi} &= \cos(\phi_1), \\
y_{0\theta} &= \sin(\theta_0), & z_{0\phi} &= \sin(\phi_0), \\
y_{1\theta} &= \sin(\theta_0 + \theta_1)/2, & z_{1\phi} &= \sin(\phi_0 + \phi_1)/2, \\
y_{2\theta} &= \sin(\theta_1), & z_{2\phi} &= \sin(\phi_1).
\end{aligned}$$

It follows from a little analysis of the formulæ for the weights that, for any cyclide (i.e., for any $a, b, (c > 0), \mu$), there exist 16, positive-weight, bi-quadratic Bézier patches that parametrise the entire surface. Further, the 4-way symmetry of the cyclide can be exploited, enabling 12 of the 16 patches to be determined, by vertex transformations, from the 4 patches defined by the following angular displacements:

- $0 \leq \phi \leq \frac{\pi}{2}, 0 \leq \theta \leq \frac{\pi}{2}$
- $\frac{\pi}{2} \leq \phi \leq \pi, 0 \leq \theta \leq \frac{\pi}{2}$
- $0 \leq \phi \leq \frac{\pi}{2}, \frac{\pi}{2} \leq \theta \leq \pi$
- $\frac{\pi}{2} \leq \phi \leq \pi, \frac{\pi}{2} \leq \theta \leq \pi$.

These 4 patches are shown, with their Bézier polygons, in Figure 2. These patches parametrise $\frac{1}{4}$ of the surface and their



(a) (b)

Figure 2: (a)$\frac{1}{4}$-cyclide comprising 4 positive weight quadratic rational Bézier patches,(b) 16 patch NURBS representation obtained from $\frac{1}{4}$ patch

weights the computation of are given in the appendix of the paper. As $0 < c < a$, it is clear that all the patches have positive weights. The Bézier vertices of the four patches are also given in the appendix of the paper.

A 16 Patches cyclide NURBS represenataion which is generated from the $\frac{1}{4}$ patch equation are used in our proposed algorithm. Note that different size of patches can be created for different models. In general the proposed algorithm supports any numebr of patches.

4 Proposed Texture Synthesis Algorithms

4.1 Overview of our approch

A high level block diagram of the proposed texture synthesis algorithm can be illustrated as in Figure 3. The geometric

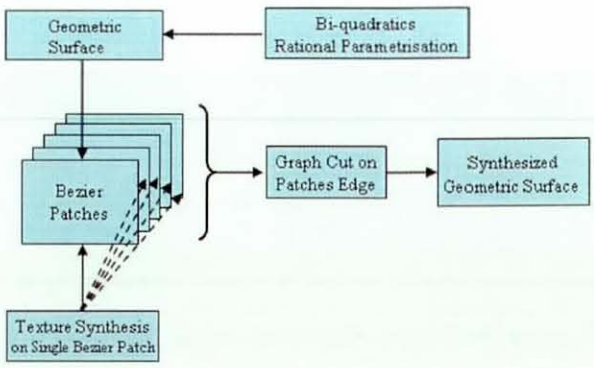


Figure 3: Proposed block diagram for texture synthesis on geometric surfaces

surface (a ring Dupin Cyclide in our experiments) to be textured, is first parametrised into a collection of Bézier patches. Subsequently, following a procedure illustrated in more detail in Figure 4, texture is independently synthesized into individual Bézier patches (see section 4.2 and 4.3). The texture synthesis techniques adopted results in a seamless

texturing within patches. Depending on whether or not seamlessly progressive texture synthesis is required, the use of the EZW algorithm (see section 4.6) is considered for coefficient prioritisation/selection in forming the texture blocks (see Figure 4) of a Bézier patch. The Bézier patches are subsequently embedded into the surface mesh representation obtained using the rational parametrisation approach presented in section 2 and 3. The visibility of seams between Bézier patches is then reduced using a minimum boundary cut technique, i.e. graph-cut(see section 4.4). Figure 4 illustrates the block diagram of the fundamental unit of the proposed overall texture synthesis algorithm of Figure 3, i.e. the block diagram depicting the detailed procedure adopted in texture synthesis on a single bi-quadratic rational Bézier patch. An

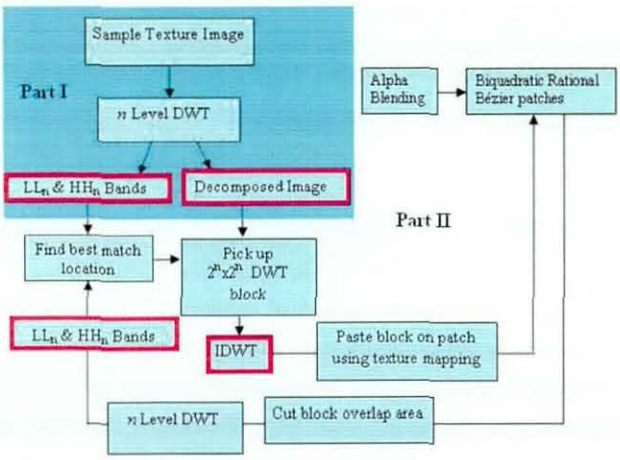


Figure 4: Texture synthesis on a single bi-quadratic rational Bézier patch.

overview of the operation of the fundamental unit of texture synthesis illustrated in Figure 4 can be given as follows: The sample texture is first decomposed using a n-level DWT (see section 4.2). A random decomposed block of this sample is initially picked, converted into pixel domain using a *n*-level IDWT procedure and subsequently embedded (see section 4.3) as a starting, corner block of a patch of the 3D surface (i.e. Dupin Cyclide). An overlap area of this block with the subsequent texture block to be synthesized is selected, and converted into the DWT domain using a *n*-level forward DWT. The LL and HH bands of this overlap area are then used to find a suitably matching block from the decomposed sample texture image (see section 4.2). The best matching block is subsequently converted back into the pixel domain using an n-level IDWT and is placed adjacent to the previously synthesized texture block. A edge-blending algorithm [22] is used to improve the quality of the seam between the two adjacent blocks. The above process is continued until the entire surface of the patch is textured. The operational details of the main building blocks of figures 3 and 4 can be provided as follows:

4.2 Multiresolution, Discrete Wavelet Transforms (DWT) decomposition of the sample texture

Textured images contain a large amount of perceptual data. *Therefore the number of bits required to represent/encode a* texture image may be high. However typical images consist of a wide range of frequency components spanning the human visual frequency band. Some of theses frequency components have a significant effect in human perception while some others have very low significance. Fortunately texture images are often of this type. The DWT, popularly used in image compression to achieve signal compaction, provide a compact multi resolution representation of an image. It gives a signal representation/decomposition in correspondence to a narrow band, low frequency range and a wide band, high frequency range. Using the concept of scale, data representing a continuous trade-off between space and frequency can be made available for further processing. In our algorithm we use a two-dimensional DWT to obtain a multi-resolution decomposition of the texture sample.

Texture synthesis commences with the application of a *n*-level (*n*=3 is used in our experiments) 2D DWT (e.g. Haar Transform) decomposition on the sample texture image, I_{sample} (see Figure 5). The application of a single level 2D DWT on the sample texture divides it into a set of four *component images, i.e. sub-bands, as illustrated in Figure 5.* This can be mathematically expressed as

$$(I_{LL0},\ I_{HL0},\ I_{LH0},\ I_{HH0}) = DWT(I_{sample}) \quad (1)$$

Where $I_{LL0}, I_{HL0}, I_{LH0}, I_{HH0}$ are the image sub-bands corresponding respectively to the low-resolution approximation, vertical details, horizontal details and diagonal details of the sample texture and *DWT()* represent the forward discrete wavelet transform function. Similarly 2nd level and 3rd level decomposition are obtained by applying DWT to the low-resolution sub-bands of previous decomposed level. This can be mathematically represented as follows.

$$(I_{LL1}, I_{HL1}, I_{LH1}, I_{HH1}) = DWT(I_{LL0}) \quad (2)$$

$$(I_{LL2}, I_{HL2}, I_{LH2}, I_{HH2}) = DWT(I_{LL1}) \quad (3)$$

$$(I_{LLn}, I_{HLn}, I_{LHn}, I_{HHn}) = DWT(I_{LLn-1}) \quad (4)$$

From the *n*-level decomposition above, we extract the lowest-resolution and the corresponding diagonal detail band, i.e. I_{LLn-1} and I_{HHn-1}. To simplify the use of notations, we generalise the notation used in equations [1-4]as I_{pl} where $p \in (LL, HL, LH, HH)$ and $l \in (0, 1, 2, ...n-1)$.

Once sample texture is multi-resolution DWT decomposed as described above, the next step is the placement of texture blocks from the sample texture on a Bézier patch that represents a section of the surface of the geometric surface.

4.3 Texture synthesis of Bézier patches

Refer Figure 4 Part II, Let $B_{pl(x,y)}$ represent a general square block of the decomposed sample image located at position

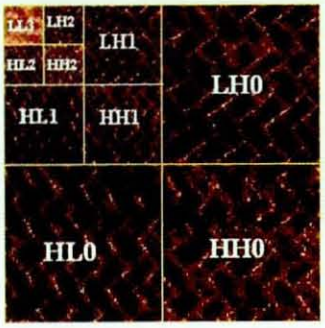


Figure 5: Transforming the sample texture into a multiresolution image representation with three level decomposition.

(x, y) relative to the sub-bands *(p, l)*'s origin. [Note that in our experiments we have set the size of the above block to be $2^{5-l} \times 2^{5-l}$, where $l(= 0, 1, 2)$]. Initially we randomly pick a block , $B_{LLn-1(x,y)}$ from the lowest resolution,*(LL, n-1)* sub-band of decomposed sample image. Subsequently by combining this block with the corresponding blocks of the other sub-bands and performing an *n-IDWT*, the randomly selected block is obtained in the original pixel domain, (i.e. B_{LL}), as follows:

$$B_{LLn-2} = IDWT(B_{LLn-1}, B_{HLn-1}, B_{LHn-1}, B_{HHn-1}) \quad (5)$$

$$B_{LL} = IDWT(B_{LL0}, B_{HL0}, B_{LH0}, B_{HH0}) \quad (6)$$

After randomly picking up the block as described above and converting it into its pixel domain, the next stage is to map the texture block on to a Bézier patch of the 3D surface. It is noted that by following the parametrisation procedure of section 2, the Bézier patches of the surface will be divided into a quadrilateral mesh, where each mesh element is referred to as a surface division. The above mentioned randomly picked texture block is embedded to surface division (see Figure 10) by following the texture embedding method detailed in section 4.5. This procedure is popularly known in literature as texture mapping.

Once the first, randomly selected block is mapped as above, an overlapping area of this block (with the adjacent, potential block to be mapped), is selected to be used in locating the next blocked from the sample texture to be mapped onto the patch. Note that in our experiments this area was 8 pixels wide, and was selected from the left end, bottom end or both above ends of the block (depending on the location of the original texture block on the Bézier patch). The overlap area is subsequently *n*-level decomposed (*n*=3 in our experiments) and used to find a suitable match for the adjacent block from the decomposed sample texture using minimum sum of squared-differences (see figure6). To speed up the block matching we only use all coefficients of the LL_n and HH_n sub-bands for matching. In our experiment where *n*=3 and a sample texture size of 128 $\times$ 128 is used, out of the 16384 possible coefficients, we only use 256 coefficient i.e. 1/64 of total number coefficient in the sample texture. Therefore the matching speed is significantly enhanced.

Although adjacent blocks on a Bézier patch are matched using an overlapping area, block seam mismatches are likely to exist. However due to the efficiency of matching in the DWT domain[28] these seam mismatches are minimal as compared to using an alternative patch based texture synthesis technique and are therefore easily eliminated using an edge-blending approach[22]. The above process of block matching, selection

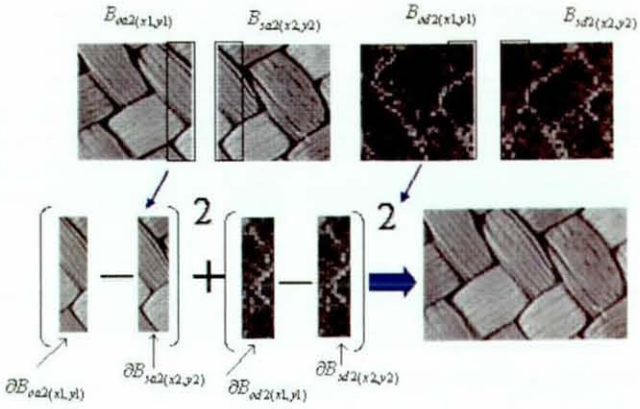


Figure 6: The matching criteria

and placement is continued until the texture of the entire patch is synthesized. As individual Bézier patches are textured independently, it is likely that visible seam boundaries exist between patches (see Figure 7). In the following section we describe the use of a minimum boundary cut technique to minimise these artefacts.

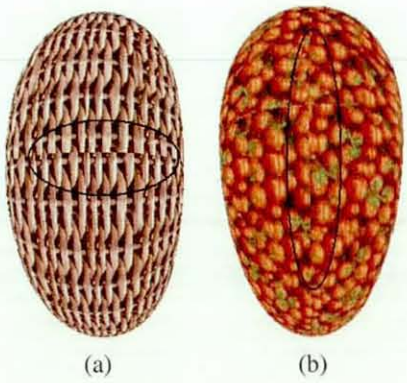


Figure 7: Artifacts at Bézier patch edges.

4.4 Using a modified graphcut technique to improve Bézier patch seams

To remove the edge artefacts between Bézier patches (see Figure 7) we use a modified version of the popular Graphcut

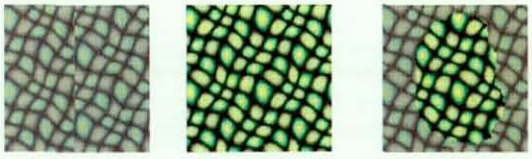
technique proposed in [25]. Its operation can be explained as follows:

4.4.1 Edge block selection and matching

The idea of this stage is to replace B'ezier patch seam regions, with visible straight line artefacts, with matching regions from the sample texture, which can be better blended with the non-seam region texture. To accomplish this task a block of size $2^{5-l} \times 2^{5-l}$ (l=3 in our experiments), which we be called an edge block is first selected from the patch boundary (see Figure 8(a)). It is noted that seams are included within this edge block. Subsequently we apply an n-level (n=3) DWT on the selected edge block to create a DWT decomposed edge block. The LLn and the HHn sub-bands of the decomposed block is then used to search the decomposed sample texture for the best matching block is found using block matching criteria (see Figure 6) . Once the best matching block is found (see Figure 8(b)) then can be combined with the edge block using the graphcut approach in transform domain (see section 4.4.2) to minimise seam artefacts. After an n-level IDWT is performed to obtain a pixel block.

4.4.2 Graphcut Approach

Once the best match to the edge block is found as above, we apply the graphcut technique of [25] which is capable of optimally replacing the seam region of the edge block with regions from the best matching block (see section 4.4.1) to provide an optimal minimisation of boundary artefacts (see Figure 8(c)). We apply Graphcut technique in transform domain to reduce alogrithm complexsity and get best possible seams. Readers who are interested in the details of the graphcut approach are referred to [19, 25]. The above modified edge block is finally embedded back in its original position in the Bézier patch seam area. This process is continued for all edge blocks along the Bézier patch seams. The results illustrated in Figure 9 when compared to the results illustrated in Figure 7 proves the improved quality at Bézier patch seams.

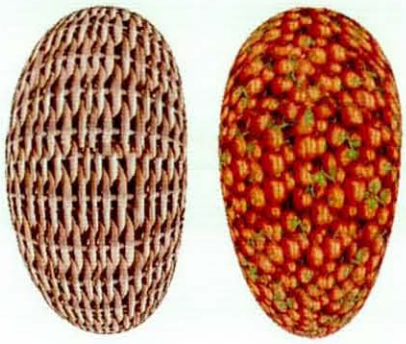


(a)Artifact block (b)Best Block (c)Embedded block

Figure 8: Seam selection

4.4.3 Feathering

Although the grapghcut approach produces an optimum seam, the optimality is restricted by the quality of the original edge blocks. Therefore it is likely that some artefacts still remain



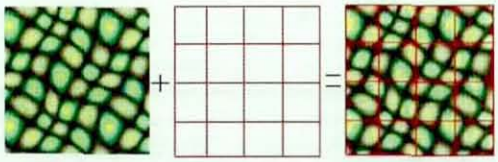
(a) (b)

Figure 9: Refined Bézier patch edges

after the application of the graphcut approach. To address this problem we use the popular feathering [4] approach that uses suitable weighting factors to combine those pixels at the seams. Feathering results in a further improvement of the seam quality.

4.5 Embedding texture on Bézier patches

To embed texture on Bézier patches, we use a modified version of the approach proposed by Soucy et al. [21]. The modification required is due to the use of a quadrilateral mesh in the proposed approach as against a triangular mesh in the original approach of [21]. The texture of each quadrilateral in the original mesh is obtained via a direct mapping [21] from the corresponding quadrilateral in texture space. The texture space referred to above is created as a result of the block placement and matching procedure on Bézier surfaces, described in detail in section 4.3. The quadrilaterals in texture space that we use are rectangle in nature and are uniform in size. It is noted however that it is also possible to use quadrilaterals that are a better fit to the mesh shape and size. It is further noted that the above texture embedding can be performed at interactive (real-time) rates. In our experiments all surfaces are rendered at interactive rates using 1024 $\times$ 1024 sample textures. The surface models are composed of between 1000 and 50,000 quadrilaterals.



(a) Texture Block (b) Parametrisized Mesh (c) Embedded Texture on Mesh

Figure 10: Texture Embedding

4.6 Progressive texture on surfaces using Embedded Zerotree Wavelet (EZW) algorithm

The Zerotrees of wavelet coefficient concept was originally introduced by Shapiro [17] in progressive encoding of images. It is based on two important observations:

1. Natural images in general have a low pass spectrum. Therefore when an image is wavelet transformed the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average be smaller in the higher subbands than in the lower subbands.

2. Large wavelet coefficients are visually more important than smaller wavelet coefficients.

EZW provides a compact representation of perceptually significant coefficients and multiresolution construction capability of an image. The idea is to organize DWT coefficients of an image (see Figure 5) in a prioritized order of visual significance, depending on their position and magnitude in the DWT decomposition and to subsequently encode the ordered list of coefficients following an embedded coding algorithm. In an embedded coding algorithm the encoder can terminate the encoding at any point there by allowing a target bit rate or target distortion metric to be met exactly. On the other hand, given a bit stream, a decoder can cease decoding at any point in the bit stream. Thus it is capable of producing exactly the same image that would have been encoded at the bit rate corresponding to the truncated bit stream.

In this paper we used the EZW algorithm's initial coefficient prioritization procedure to prioritise their use within the proposed texture synthesis algorithm where we have replaced IDWT with EZWIDWT module(see Figure 4). and Equation 4 can be modified to:

$$B_{LLn-2} = EZWIDWT(B_{LLn-1}, B_{HLn-1},$$

$$B_{LHn-1}, B_{HHn-1}, thershold) \quad (7)$$

The thershold can be calculated using the magnitude of wavelet coefficents of the decomposed sample image. Due to space limitation we refer readers interested in the detail of the EZW coefficient prioritization algorithm to [17]. We show that the visually prioritized availability of coefficients and the subsequent embedded coding of the coefficients enable seamless progressive texture synthesis capability on bi-quadratic rational patches. This is an additional key contribution of our present work.

5 Exprimental Results and Analysis

In order to analyse the performance of proposed progressive texture synthesis approach on the surface of a Dupin Cyclide, a Sphere and a Torus. We have carried out a careful and efficient implementation in OpenGL and C++. Experiments were carried out using a range of sample textures, which can be broadly classified as regular, near-regular, irregular, and stochastic [5] in nature. Figure 12 and Figure 13 illustrate the results. Figure 13 proves the effectiveness of the proposed algorithm in the progressive texture synthesis domain. It is noted that the time required to synthesise the textures on the surface was in the order of five seconds.

A closer analysis of the results in Figure 13 reveals the following important facts.

- By varying the threshold selected for DWT coefficient selection, within the EZW scheme, discrete seamless levels of output quality can be obtained see Figure 13 (a1 to a5 and b1 to b5).

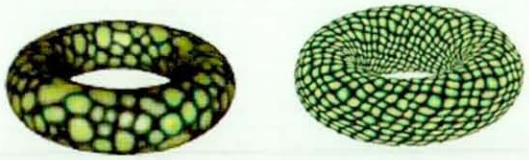
- It is observed that for most types of textures, 15% of the DWT coefficients from sample texture are sufficient for creating synthesis output textures on surface of sufficient visual quality see Figure 13(a4 and b4) . For regular textures this percentage can be significantly lower (7% in our experiments) Figure 13(a3).

- It observe that seams are rarely visible.

Given the above observations, the proposed algorithm can be proved to be beneficial in applications that require fast and accurate progressive texturing capability such distributive and collaborative gaming or low bandwidth transmission application. Although due to space limitations we have restricted the number of progressive texture synthesis results illustrated in Figure 13 to five textures, and five different quality levels, the proposed method is capable of progressive texture synthesis at seamlessly different number of levels and a large number of sample textures covering broad statistical texture properties. Due to the flexibility of the extension of the Bézier patch based rational parametrisation scheme the proposed approach can be further extended to efficient progressive texture synthesis on many object shape.

Our algorithms suffer from two limitation, 1) No control over shape of the geometry of the surface 2) No control over direction of texture element on the surface as it is directed by algorithm.

We have compared our progressive texture synthesis results with similar pixel domain work of Shet et.al [18] for arbitrary surfaces. In his approch he replaced gaussain pyramid by a DWT pyramid and adpoted image pixel location for encoding/decoding surface and texture. It is observed that the surface texture quality and overall speed of our algorithm is improved (see Figure 11). Shet et alś. [18] apporach take about 15 to 20 minute for progressive texture where as our approch is in order of five seconds.



(a) (b)

Figure 11: (a): Shet et.al alogrithm , (b): Our alogrithm

Figure 12: Texture synthesis on Sphere,Torus and Cyclide

Figure 13: Progressive texture synthesis on Cyclide and Torus

6 Conclusions & Future Work

In this paper we have proposed a novel approach to texture synthesis on geometric surfaces. A rational parametrisation stage is initially used to represent the surface as a collection of bi-quadratic Bezier patches onto which the texture is synthesized using a multiresolution DWT approach. We have used the popular EZW idea of prioritizing the DWT coefficients and subsequent embedded coding to introduce progressive texture synthesis. To minimize seam artifacts at the block boundaries within patches, we have successfully used an alpha-blending algorithm, whereas to minimize artifacts at seams between patches we have successfully used the popular random patch cutting algorithm in tarnsform domian, graph-cut, followed by a further feathering algorithm at the exact seams locations. We have provided results to show the effectiveness of the overall surface parametrisation and texture synthesis algorithms. Further experimental results have been provided to prove the effectiveness of the measures taken to minimize seam artifacts, and the flexibility and efficiency of the progressive texturing capability of the proposed algorithm. A number of possible application domains of the proposed algorithm have been identified.

It is possible to show that the rational parametrisation strategy used can be extended to cover many 3D objects with arbitrary surface topology. Hence the proposed texture synthesis approach can be used to progressively synthesize texture onto 3D surfaces with arbitrary shape. We are currently in the process of generalizing the proposed algorithm to address the above issues and to over come the limitaion i.e. control over geometry of the surface and direction of texture elements.

References

[1] A.Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. of SIGGRAPH 2001*, pages 341–346, 2001.

[2] M. Ashikhmin. Synthesizing natural textures. In *ACM Symp. on Interactive 3D Graphics*, page 217226, 2001.

[3] H. E. Bez and T.J Wetzel. Induced rational parametrisations of special curves. In *International Journal of Computer Mathematics*, volume 80,9, pages 1093–1109, 2003.

[4] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 12 1983. ISSN:0730-0301.

[5] Wen chieh. Lin, Hays James, Wu Chenyu, Kwatra Vivek, and Liu Yanxi. A comparison study of four texture synthesis algorithm on near-regular textures. In *Tech. Report CMU-RI-TR-04-01, Robotics Institute, Carnegie Mellon University*, 2004.

[6] D. Dutta, R. R. Martin, and M. J. Pratt. Cyclides in surface and solid modeling. *IEEE Computer Graphics and Applications*, 13:5359, 1993.

[7] S. Foufou, L. Garnier, and M .J. Pratt. Conversion of dupin cyclide patches into rational bi-quadratic bezier form. In *Proc. of the IMA Mathematics of Surfaces XI*, pages 201–218, 2005.

[8] Chi-Wing Fu and Leung Man-Kang. Texture tiling on arbitrary topological surfaces. In *Proceedings of Eurographics Symposium on Rendering 2005 (EGSR 2005)*, pages 99–104, 2005.

[9] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. *ACM Trans. Graph.*, 25(3):541–548, 2006. ISSN:0730-0301.

[10] Wang Lujin, Gu Xianfeng, Mueller Klaus, and Yau Shing-Tung. Uniform texture synthesis and texture mapping using global parameterization. In *Special Issues of Pacific Graphics,ISSN:0178-2789*, volume 21, pages 801–810, 2005.

[11] Sebastian Magda and David Kriegman. Fast texture synthesis on arbitrary meshes. In *Proc. of the 14th Eurographics workshop on Rendering*, volume 44, pages 82–89, 2003.

[12] R. Martin, J. de Pont, and T. Sharrock. Cyclide surfaces in computer aided design. In *Proceedings of the IMA Conference on the Mathematics of Surfaces*, 1986.

[13] F. Neyret and M. P. Cani. Pattern-based texturing revisited. In *Proc. of SIGGRAPH 1999*, page 235242, 1999.

[14] M. J. Pratt. Cyclides in computer aided geometric design. In *Computer Aided Geometric Design*, volume 7, pages 221–242, 1990.

[15] M. J. Pratt. Cyclides in computer aided geometric design ii. In *Computer Aided Geometric Design*, volume 12, page 131152, 1995.

[16] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proc. of SIGGRAPH 2000*, page 465470, 2000.

[17] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficient. *IEEE Trans. Signal Processing*, 41(12):3445–3462, 12 1993.

[18] Rupesh N. Shet, Eran A. Edirisinghe, and Helmut E. Bez. Progressive texture synthesis on 3d surfaces. In *VIIP 2006 : In Proceedings of Sixth IASTED international conference on Visualization, Imaging, and Image Processing*, pages 136–141, 2006. ISBN Hardcopy:0-88986-598-1/CD:0-88986-600-7.

[19] Rupesh N. Shet, Eran A. Edirisinghe, and Helmut E. Bez. A wavelets based max-flow/min-cut approch for texture synthesis. In *VIE2007: In Proceedings of The IEE International conference on Visual Information Engineering: Convergence in Graphic and Vision*, 2007. ISBN CD 978-0-86341-830-3.

[20] C. Soler, M. Cani, and A. Angelidis. Hierarchical pattern mapping. In *Proc. of SIGGRAPH 2002*, volume 21, pages 673–680, 2002.

[21] Soucy, Marc, G. Godin, and M. Rioux. A texture-mapping approach for the compression of colored 3d triangulations. In *The Visual Computer*, volume 12, page 503514, 1996.

[22] R. Szeliski and H.-Y. Shum. Creating full view panoramic mosaics and enviornment maps. In *Proc. of SIGGRAPH 1997*, pages 251–258, 1997.

[23] Xin Tong, Zhang Jingdan, Liu Ligang, Wang Xi, Guo Baining, and Shum Heung-Yeung. Synthesis of bidirectional texture functions on arbitrary surfaces. In *proc. of SIGGRAPH 2002*, page 665 672, 2002.

[24] G. Turk. Texture synthesis on surfaces. In *Proc. of SIGGRAPH 2001*, page 347354, 2001.

[25] Kwatra Vivek, Schdl Arno, Essa Irfan, Turk Greg, and Bobick Aaron. Graphcut textures: image and video synthesis using graph cuts. In *Proc. of SIGGRAPH 2003*, volume 22, pages 277–286, 2003.

[26] L.-Y Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. of SIGGRAPH 2000*, page 479488, 2000.

[27] L.-Y Wei and M. Levoy. Texture synthesis over arbitrary manifold surface. In *Proc. of SIGGRAPH 2001*, page 355360, 2001.

[28] D. S. Wickramanayake, E. A. Edirisinghe, and H. E. Bez. Multiresolution texture synthesis in wavelet transform domain. *The Journal of Imaging Science and Technology*, 50(1):93–102, 1 2006. ISBN / ISSN: 1062-3701.

[29] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin. Texture and shape synthesis on surfaces. In *Eurographics Rendering Workshop*, page 301312, 2001.

Appendix

Weight Computed as: First patch: $\theta_0 = 0, \theta_1 = \frac{\pi}{2}, \quad \phi_0 = 0, \phi_1 = \frac{\pi}{2}$

$$\begin{aligned} w_{00} &= a-c, & w_{01} &= \frac{a-c}{\sqrt{2}}, & w_{02} &= a, \\ w_{10} &= \frac{a-c}{\sqrt{2}}, & w_{11} &= \frac{a-c}{2}, & w_{12} &= \frac{a}{\sqrt{2}}, \\ w_{20} &= a, & w_{21} &= \frac{a}{\sqrt{2}}, & w_{22} &= a \end{aligned}$$

Second patch: $\theta_0 = 0, \theta_1 = \frac{\pi}{2}, \quad \phi_0 = \frac{\pi}{2}, \phi_1 = \pi$

$$\begin{aligned} w_{00} &= a, & w_{01} &= \frac{a+c}{\sqrt{2}}, & w_{02} &= a+c, \\ w_{10} &= \frac{a}{\sqrt{2}}, & w_{11} &= \frac{a+c}{2}, & w_{12} &= \frac{a+c}{\sqrt{2}}, \\ w_{20} &= a, & w_{21} &= \frac{a}{\sqrt{2}}, & w_{22} &= a. \end{aligned}$$

Third patch $\theta_0 = \frac{\pi}{2}, \theta_1 = \pi, \quad \phi_0 = 0, \phi_1 = \frac{\pi}{2}$

$$\begin{aligned} w_{00} &= a, & w_{01} &= \frac{a}{\sqrt{2}}, & w_{02} &= a, \\ w_{10} &= \frac{a+c}{\sqrt{2}}, & w_{11} &= \frac{a+c}{2}, & w_{12} &= \frac{a}{\sqrt{2}}, \\ w_{20} &= a+c, & w_{21} &= \frac{a+c}{\sqrt{2}}, & w_{22} &= a. \end{aligned}$$

Fourth patch $\theta_0 = \frac{\pi}{2}, \theta_1 = \pi, \quad \phi_0 = \frac{\pi}{2}, \phi_1 = \pi$

$$\begin{aligned} w_{00} &= a, & w_{01} &= \frac{a}{\sqrt{2}}, & w_{02} &= a, \\ w_{10} &= \frac{a}{\sqrt{2}}, & w_{11} &= \frac{a-c}{2}, & w_{12} &= \frac{a-c}{\sqrt{2}}, \\ w_{20} &= a, & w_{21} &= \frac{a-c}{\sqrt{2}}, & w_{22} &= a-c. \end{aligned}$$

1st patch Bézier vertex data

$$\begin{aligned} xv_{00} &= \frac{\mu(c-a)+b^2}{a-c} & xv_{01} &= \frac{\mu(c-a)+b^2}{a-c} & xv_{02} &= \frac{\mu c+b^2}{a} \\ xv_{10} &= \frac{\mu(c-a)+b^2}{a-c} & xv_{11} &= \frac{\mu(c-a)+b^2}{a-c} & xv_{12} &= \frac{\mu c+b^2}{a} \\ xv_{20} &= \frac{\mu c}{a} & xv_{21} &= \frac{\mu c}{a} & xv_{22} &= \frac{\mu c}{a} \end{aligned}$$

$$\begin{aligned} yv_{00} &= 0 & yv_{01} &= 0 & yv_{02} &= 0 \\ yv_{10} &= \frac{b(a-\mu)}{a-c} & yv_{11} &= \frac{b(a-\mu)}{a-c} & yv_{12} &= b \\ yv_{20} &= \frac{b(a-\mu)}{a} & yv_{21} &= \frac{b(a-\mu)}{a} & yv_{22} &= b \end{aligned}$$

$$\begin{aligned} zv_{00} &= 0 & zv_{01} &= \frac{b(c-\mu)}{a-c} & zv_{02} &= \frac{b(c-\mu)}{a} \\ zv_{10} &= 0 & zv_{11} &= \frac{b(c-\mu)}{a-c} & zv_{12} &= \frac{b(c-\mu)}{a} \\ zv_{20} &= 0 & zv_{21} &= -\frac{b\mu}{a} & zv_{22} &= -\frac{b\mu}{a} \end{aligned}$$

2nd patch Bézier vertex data

$$\begin{aligned} xv_{00} &= \frac{\mu c+b^2}{a} & xv_{01} &= \frac{\mu(a+c)+b^2}{a+c} & xv_{02} &= \frac{\mu(a+c)+b^2}{a+c} \\ xv_{10} &= \frac{(\mu c+b^2)}{a} & xv_{11} &= \frac{\mu(a+c)+b^2}{a+c} & xv_{12} &= \frac{\mu(a+c)+b^2}{a+c} \\ xv_{20} &= \frac{\mu c}{a} & xv_{21} &= \frac{\mu c}{a} & xv_{22} &= \frac{\mu c}{a} \end{aligned}$$

$$\begin{aligned} yv_{00} &= 0 & yv_{01} &= 0 & yv_{02} &= 0 \\ yv_{10} &= b & yv_{11} &= \frac{b(a+\mu)}{a+c} & yv_{12} &= \frac{b(a+\mu)}{a+c} \\ yv_{20} &= b & yv_{21} &= \frac{b(a+\mu)}{a} & yv_{22} &= \frac{b(a+\mu)}{a} \end{aligned}$$

$$\begin{aligned} zv_{00} &= \frac{b(c-\mu)}{a} & zv_{01} &= \frac{b(c-\mu)}{a+c} & zv_{02} &= 0 \\ zv_{10} &= \frac{b(c-\mu)}{a} & zv_{11} &= \frac{b(c-\mu)}{a+c} & zv_{12} &= 0 \\ zv_{20} &= -\frac{b\mu}{a} & zv_{21} &= -\frac{b\mu}{a} & zv_{22} &= 0 \end{aligned}$$

3rd patch Bézier vertex data

$$\begin{aligned} xv_{00} &= \frac{\mu c}{a} & xv_{01} &= \frac{\mu c}{a} & xv_{02} &= \frac{\mu c}{a} \\ xv_{10} &= \frac{\mu(a+c)-b^2}{a+c} & xv_{11} &= \frac{\mu(a+c)-b^2}{a+c} & xv_{12} &= \frac{(\mu c-b^2)}{a} \\ xv_{20} &= \frac{\mu(a+c)-b^2}{a+c} & xv_{21} &= \frac{\mu(a+c)-b^2}{a+c} & xv_{22} &= \frac{\mu c-b^2}{a} \end{aligned}$$

$$\begin{aligned} yv_{00} &= \frac{b(a-\mu)}{a} & yv_{01} &= \frac{b(a-\mu)}{a} & yv_{02} &= b \\ yv_{10} &= \frac{b(a-\mu)}{a+c} & yv_{11} &= \frac{b(a-\mu)}{a+c} & yv_{12} &= b \\ yv_{20} &= 0 & yv_{21} &= 0 & yv_{22} &= 0 \end{aligned}$$

$$\begin{aligned} zv_{00} &= 0 & zv_{01} &= -\frac{b\mu}{a} & zv_{02} &= -\frac{b\mu}{a} \\ zv_{10} &= 0 & zv_{11} &= -\frac{b(c+\mu)}{a+c} & zv_{12} &= -\frac{b(c+\mu)}{a} \\ zv_{20} &= 0 & zv_{21} &= -\frac{b(c+\mu)}{a+c} & zv_{22} &= -\frac{b(c+\mu)}{a} \end{aligned}$$

4th patch Bézier vertex data

$$\begin{aligned} xv_{00} &= \frac{\mu c}{a} & xv_{01} &= \frac{\mu c}{a} & xv_{02} &= \frac{\mu c}{a} \\ xv_{10} &= \frac{(\mu c-b^2)}{a} & xv_{11} &= \frac{\mu(c-a)-b^2}{a-c} & xv_{12} &= \frac{\mu(c-a)-b^2}{a-c} \\ xv_{20} &= \frac{\mu c-b^2}{a} & xv_{21} &= \frac{\mu(c-a)-b^2}{a-c} & xv_{22} &= \frac{\mu(c-a)-b^2}{a-c} \end{aligned}$$

$$\begin{aligned} yv_{00} &= b & yv_{01} &= \frac{b(a+\mu)}{a} & yv_{02} &= \frac{b(a+\mu)}{a} \\ yv_{10} &= b & yv_{11} &= \frac{b(a+\mu)}{a-c} & yv_{12} &= \frac{b(a+\mu)}{a-c} \\ yv_{20} &= 0 & yv_{21} &= 0 & yv_{22} &= 0 \end{aligned}$$

$$\begin{aligned} zv_{00} &= -\frac{b\mu}{a} & zv_{01} &= -\frac{b\mu}{a} & zv_{02} &= 0 \\ zv_{10} &= -\frac{b(c+\mu)}{a} & zv_{11} &= -\frac{b(c+\mu)}{a-c} & zv_{12} &= 0 \\ zv_{20} &= -\frac{b(c+\mu)}{a} & zv_{21} &= -\frac{b(c+\mu)}{a-c} & zv_{22} &= 0 \end{aligned}$$

CONTROL POLYGON BASED TEXTURE SYNTHESIS ON BIQUADRATIC BĒZIER RATIONAL SURFACES

Rupesh N. Shet, H. E. Bez, E. A. Edirisinghe
Department of Computer Science, Loughborough University, UK
R.Shet@lboro.ac.uk

Keywords: Texture synthesis Discrete Wavelet Transform (DWT), EZW, Rendering, Bézier surfaces.

Abstract: Existing texture synthesis algorithms fail to deliver effectively in application areas where progressive rendering of texture is required. To provide a practical solution to this problem we propose a novel algorithm for progressive-texture synthesis on surfaces, which makes use of the Embedded Zero-tree of Wavelet (EZW) idea proposed by Shapiro et al., 1993 which is capable of prioritising the coefficients of a DWT decomposed image according to their visual significance. We demonstrate the use of the proposed algorithm on texturing a single biquadratic surface and two smoothly joined biquadratic surfaces. It is further shown that the proposed texture synthesis approach on Bézier patches allows the algorithm's general use in texture synthesis on many common surface topologies and can be generalised for arbitrarily shaped surfaces. We provide experimental results to prove the effectiveness of the proposed approach, when synthesising textures of regular, irregular and stochastic nature. Further experimental results are provided to illustrate the practical use of the proposed texture synthesis algorithm in resource constrained application domains.

1 INTRODUCTION

Texture synthesis provides a practical solution for data acquisition and is often used to enhance realism of artificially created scenes. As result, a number of 'image based' texturing algorithms have been proposed in the past decade. A typical texture synthesis algorithm starts from a sample image and attempts to produce a larger texture with a visual appearance similar to the sample, by repeated placement of micro patterns of texture elements. It does this in a way that when perceived by an observer, the synthesized texture appears to be generated by the same underlying stochastic process. However all texture synthesis algorithms are challenged by the high statistical variability of textures involved in synthesis. Thus a universal solution to fast texture synthesis yet remains an open problem. Texturing surfaces provides further challenges and attracted much research interest in the recent past due to applications in computer graphics, animated movie production, computer games, education, architecture, computer art and virtual productions.

A major proportion of research in the area of texture synthesis has focused on synthesizing texture on planner surfaces. Recently a number of approaches have been proposed for texture synthesis on surfaces. These texture synthesis approaches can be broadly classified into two groups, namely, pixel based (Wei & Levoy, 2000, 2001; Turk, 2001;Ying et al., 2001; Ashikhmin et al., 2001; Tong et al., 2002; Shet et al., 2006; Lefebvre and Hoppe, 2006) and patch based (Neyret and Cani, 1999; Praun et al., 2000; Soler et al., 2002; Sebastian et al., 2003; Wang et al., 2005; Wing Fu et al., 2005) approaches. Pixel based approaches consider a pixel as the basic unit in the synthesis process. Patch based approaches are an alternative to pixel based approaches where an attempt is made to synthesis texture by copying selected regions of pixels from the sample texture and stitching them together. This approach overcomes the limitations of the pixel based approaches, i.e. being limited to work with certain types of textures and the lack of computational speed. Neyret-Cani's 1999, technique is based on precomputed triangular texture samples which are mapped non-periodically. However this method is restricted to synthesising isotropic textures. In the lapped texture technique proposed by Praun et al, 2000, the texture patches are first oriented and are subsequently placed in an overlapping fashion on a

surface with a predefined vector field. The method works for a limited set of textures. Soler et al., 2002 introduces hierarchical texturing to overcome drawbacks of previous algorithm. The method is capable of capturing low-frequency pattern while preserving high frequency randomness in the texture. The synthesis time may vary from few minutes to few tens of minutes. Sebastian et al., 2003 separated the texture pre-processing from synthesis and proposed two independent phases. Pre-processed texture is stored on a disk and used when needed. This process is very slow but only needs to be performed once. The pre-processing time vary from minutes to a few tens of minutes. Further storing the texture on a disk is essential. Wang et al., 2005 algorithm is mainly based on global conformal parameterization of surfaces, where the textures are preserved on surfaces without seams or cracks. This algorithm is simple for texture synthesis but parameterization process adopted is time consuming thereby slowing down the overall performance. Wing Fu et.al., 2005 introduced the concept of Wang tiles. Initially a low distortion conformal map is created from the input surface, which forms a quad based geometry. The texture is then laid out on quad surfaces, properly oriented and then mapped back on to the surface. However this approach inherits all drawbacks of the image quilting algorithms.

All the above techniques are applied on irregular shapes of triangular meshes, which results in seams at edges. Size of triangles in the mesh also varies which makes the visual artefacts on the surface, prominent. Further to this, it will also use extensive bandwidth in transmission media as triangular mesh information and texture are in uncompressed format. Further the animation of this triangular mesh is difficult as they are rigid. To overcome many of the above problems we have proposed to use NURBS, a form of surface representation which helps to compress a mesh and thus can be applied in constrained bandwidth environments. NURBS also provides additional facilities to animate the surface.

The inspiration of our work comes from the present requirements for progressive texture synthesis on surfaces, which results in extensive use of transmission media with limited bandwidth for modern application domains such as remote visualisation, distributed/collaborative gaming etc. Current texture synthesis algorithms on surfaces are time consuming and fail to perform in progressive/transform domain. To overcome this problem we propose a progressive texture synthesis algorithm using multiresolution DWT decomposition, coefficient prioritisation using EZW (embedded zero-tree wavelet) algorithm and surface representation using biquadratic rational surfaces which is falls under patch base category. We prove the proposed novel algorithm is capable of creating seamlessly varying quality levels of synthesized texture on surfaces. According to the authors knowledge it is a first attempt that demonstrates progressive texture synthesis on meshes, which utilises control polygons generated from the biquadratic Bézier equations. We show that the proposed work can be generalised to any type of arbitrary mesh.

For clarity of presentation the paper is organised as follows Section-2 introduces the reader to the research background and fundamentals. Section-3 presents the proposed algorithm. Section-4 provides experimental results and a detailed analysis. Finally, Section-5 concludes, with an insight to possible improvements and future variations.

2 RESEARCH BACKGROUND

For the purpose of clarity and ease of reference we have summarized the fundamental techniques used for multiresolution representation of texture (DWT) [Wickramanayake et al., 2005], DWT coefficient prioritization (EZW) [Shapiro et al., 1993] and surface parameterization (biquadratic rational surfaces) in this section. Hence readers who are familiar with these concepts can forgo reading this section.

2.1 DWT Representation of Texture Image

Textured images contain a large amount of perceptual data. Therefore the number of bits required to represent/encode a texture image is high. However typical images consist of a wide range of frequency components spread throughout the human visual frequency band. Some of theses frequency components have a significant effect in human perception while some others have very low significance. Fortunately texture images are often of this type. The Discrete Wavelet Transforms (DWT) provide a compact multi resolution representation of an image. It gives a signal representation in correspondence to a narrow band, low frequency range and some of the coefficients represent short data lags corresponding to a wide band, high frequency range. Using the concept of scale, data representing a continuous trade off between space

and frequency can be made available for further processing.

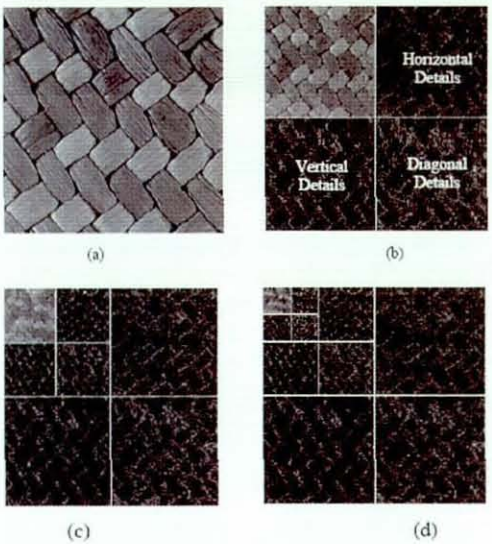


Figure1: Transforming the sample texture into a multi-resolution image representation. (a) Sample texture, (b) single level decomposition (c) two level decomposition, (d) three level decomposition.

In our algorithm we use two-dimensional DWT. To begin with, the texture image is subdivided into four sub-bands using horizontal and vertical DWT filters over the image pixels. The resulting sub bands labeled LH1, HL1 and HH1 represent the finest scale wavelet coefficient whereas the sub-band labeled LL1 represents low resolution coefficients. In order to obtain the next level of wavelet sub-bands, the sub band labeled LL1 is further decomposed and sampled using the vertical and horizontal DWT filters. This process is repeated until the required final decomposition is reached (see Figure 1). The coefficients of the subbands are then used for speeding up searching process and prioritized using the EZW algorithm presented next.

2.2 Embedded Zerotree Wavelet (EZW) Algorithm

The *Zerotrees of wavelet coefficient* concept was originally introduced by Shapiro et al., 1993 in progressive encoding of images. It is based on two important observations:

1. Natural images in general have a low pass spectrum. Therefore when an image is wavelet transformed the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average be smaller in the higher subbands than in the lower subbands.
2. Large wavelet coefficients are visually more important than smaller wavelet coefficients.

EZW provides a compact representation of perceptually significant coefficients and multi resolution construction capability of an image. The idea is to organize DWT coefficients of an image (see Figure 1) in a prioritized order of visual significance, depending on their position and magnitude in the DWT decomposition and to subsequently encode the ordered list of coefficients following an embedded coding algorithm. In an embedded coding algorithm the encoder can terminate the encoding at any point thereby allowing a target bit rate or target distortion metric to be met exactly. On the other hand, given a bit stream, a decoder can cease decoding at any point in the bit stream. Thus it is capable of producing exactly the same image that would have been encoded at the bit rate corresponding to the truncated bit stream.

In this paper we use the EZW algorithm's initial coefficient prioritization procedure to prioritize their use within texture synthesis algorithm. Due to space limitation we refer readers interested in the detail of the EZW coefficient prioritization algorithm to Shapiro et al., 1993. We show that the visually prioritized availability of coefficients enables seamless progressive texture synthesis capability on biquadratic rational surface using control polygon. This is the main contribution of our present work.

2.3 Biquadratics Bézier Surfaces

This section will briefly introduce biquadratic surface patches and construction of simple surface using them. For more details the readers are referred to Bez H.E, 2006.

Rational parametrisation is a de-facto standard representation in computer graphics and geometric modelling software, allowing portability across applications and systems. In addition to possessing desirable geometric properties, rational parametrisation

- requires the evaluation of only polynomial functions,
- gives rise to a compact data-structure,
- facilitates interactive control of shape,
- is complete in the sense that approximation of any shape to a specified tolerance δ can be achieved, and exact parametrisation (i.e. $\delta = 0$) is often possible.

Rational parametrisations of surfaces comprise local atlases, or patches, of the form:

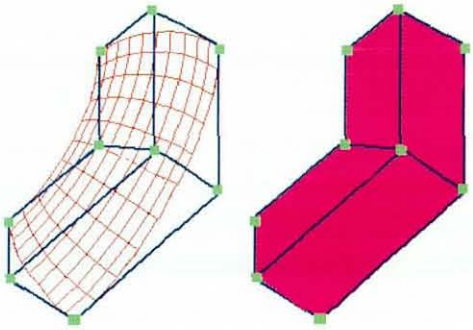
$$\tau(s,t) = \frac{\sum_{k,j=0}^{n,m} b_{n,k}(t) b_{m,j}(s) v_{k,j}^{*}}{\sum_{k,j=0}^{n,m} b_{n,k}(t) b_{m,j}(s) \omega_{k,j}} \rho < s,t < 1, \quad (2.1)$$

where $\omega_{k,j}$ are the weights and $v_{k,j}$ are the Bernstein vectors. If all the weights are non-zero this may be expressed as

$$\tau(s,t) = \frac{\sum_{k,j=0}^{n,m} b_{n,k}(t) b_{m,j}(s) \omega_{k,j} v_{k,j}}{\sum_{k,j=0}^{n,m} b_{n,k}(t) b_{m,j}(s) \omega_{k,j}}, \quad 0 < s,t < 1 \quad (2.2)$$

where $v_{k,j} = \frac{v_{k,j}^{*}}{\omega_{k,j}}$ are the Bézier vertices. The values of n and m determine the degree of the parametrisation; if $n = m = 2$ the patch is said to be biquadratic and if $n = m = 3$ it is bi-cubic.

With given nine control points we compute and draw the biquadratic surface patch defined by them. (see Figure 2)



(a) (b)

Figure2: (a) Green Colour: Biquadratic control polygon point, Red Colour: Smooth surface mesh generated using control polygon (b) Control polygon mesh (for 9 control points generate 4 faces).

Many of the desirable geometric properties of rational representation, e.g. the convex hull property and the existence of bézier vertices, are lost if negative or zero weights occur - hence, in computer graphics and geometric modelling applications, positive weight parametrisations are always preferred. For computational efficiency, low degree parametrisations are desirable.

3 PROPOSED METHDOLOGY

In this section we provide the design details of the proposed texture synthesis algorithm.

3.1 Texture Synthesis on Control Polygon

Figure 3 illustrates the basic block diagram of the proposed algorithm. The texture synthesis process starts by applying a n-level (n=3 used in our experiments) 2D DWT (e.g. Haar Transform) on sample texture image, which is denoted as I_{sample}, The application of single level 2D DWT on the sample texture results in a set of component images i.e. sub-bands, as follows:

$$(I_{LL1}, I_{HL1}, I_{LH1}, I_{HH1}) = DWT(I_{sample}) - (3.1)$$

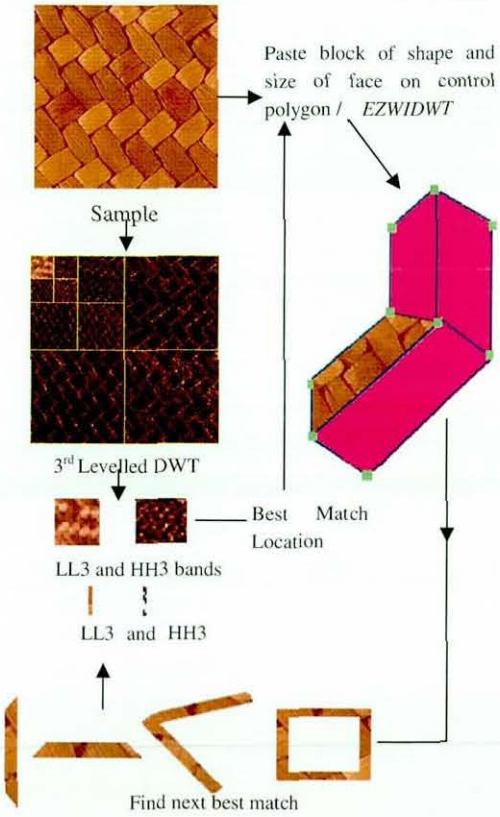


Figure 3: Proposed block diagram for texture synthesis on biquadratic surfaces.

Where $I_{LL1}, I_{HL1}, I_{LH1}, I_{HH1}$ are the texture image sub-bands corresponding respectively to low-resolution approximation, vertical details, horizontal details and diagonal details of sample texture. Similarly 2nd level and 3rd level decomposition are obtained by applying DWT to the low-resolution sub-bands of previous decomposition level. This can be mathematically represented as follows.

$$(I_{LL2}, I_{HL2}, I_{LH2}, I_{HH2}) = DWT(I_{LL1}) - (3.2)$$

$$(I_{LL3}, I_{HL3}, I_{LH3}, I_{HH3}) = DWT(I_{LL2}) - (3.3)$$

$$(I_{LLn+1}, I_{HLn+1}, I_{LHn+1}, I_{HHn+1}) = DWT(I_{LLn}) - (3.4)$$

from n-level we extract low-resolution and diagonal detail bands i.e. I_{LLn} and I_{HHn}.

We generalise the notation used in equation (3.1)-(3.6)as I_{pl} where $p \in \{LL, HL, LH, HH\}$ and $l \in \{0,1,2,..n\}$ where p represent the sub-bands within each decomposition level (LL-low resolution, LH-horizontal, HL- vertical, HH diagonal) and l represent the decomposition level. *DWT* represent the forward discrete wavelet transform.

The basic idea of proposed algorithm is to synthesise texture on control polygons of a given surface. Let $B_{LLn(x,y)}$ represent a general polygonal block of decomposed sample image located at position (x, y) relative to the sub-bands (p, l)'s origin.

Initially we randomly pick block $B_{LLn(x,y)}$ from the sample texture image with identical size and shape to that of the control polygon face. This randomly created texture block is mapped to the control polygon surface shown in figure 3. [Note: the details of the mapping process are described in section 3.2.]

In locating the next block to be synthesized, we cut a 8 pixel wide template of pixels along the edge of already synthesized, neighbouring blocks, apply a 3 level DWT decomposition on the template block and extract low-resolution and diagonal detail bands which are used for searching in sample LL3 and HH3 bands. Best matching block can be found by minimizing the L2 norm. EZW algorithm can be used if coefficient prioritisation is used to further reduce complexity.

In general, if $B_{pl(x1,y1)}$ and $B_{pl(x2,x2)}$ are two randomly shaped blocks to be matched, we say $B_{pl(x1,y1)}$ is the best match for $B_{pl(x2,y2)}$ if $d(B_{pl(x1,y1)}, B_{pl(x2,y2)})$ is minimum for all possible B_{pl} blocks, which is calculated as,

$$d(B_{l(x2,y2)p}, B_{l(x1,y1)}) = \sum_{i \in \partial B} \left[\begin{array}{l} [\ [\ \partial B_{LH(x1,y1)}(i) - \partial B_{LH(x2,y2)}(i)\ ]^2 \\ + [\ \partial B_{HH(x1,y1)}(i) - \partial B_{HH(x2,y2)}(i)\ ]^2\ ] \end{array} \right] - (3.5)$$

Where ∂B_{pl} an edge is zone of block $B_{pl(x,y)}$ and i is an element (coefficient) within the edge zone.

Finally the overlap area of the best matching edge is blended with the overlap area on the original block using alpha bending. The non-overlapping area of the block is picked from the sample texture and subsequently appended to the synthesized texture. This process will continue till all the faces of control polygon are mapped. In some cases we have considered two or more overlapping areas for finding the best match.

3.2 Texture Mapping on Control Polygons

In our implementation initially we create propagating seed vertex directions, which are then used to smooth the surface vector field. However alternatively a number of other surface vector field techniques (Wei-Levoy, 2001; Turk 2002; Ying etal, 2001) can be used to replace the approach we have selected above. Once vector fields are assigned to all control polygon faces, we then rotate all the faces according to tangential vector field and surface normal, thus placing all faces in the same 2D plane. Using a modified version of Soucy et al., 1996, approach (Note: modified from using triangle to using polygon) a texture map. T is created. For each face of the control polygon, we map it to a corresponding face in T in compact form, i.e. with no space being wasted. The faces in T are textured using the corresponding best matching block. The faces in T that we use are of non-uniform size that are a better fit to the shape and size. It is noted that the resulting texture can be rendered on the control polygon surfaces at interactive rates. The images illustrated in Figure 4 were rendered in this manner using 256 x 256 textures. The models used in our experiments are composed of smooth surfaces having between 100 to 1000 faces, whereas the control polygons used consisted of 4 faces to 8 faces (It can be further increase to n faces). We have observed that these surfaces render at real-time rates.

3.3 Projection of Texture from Control Polygons to Biquadratic Rational Surfaces

Firstly we calculate the distance between control points of the control polygon using the standard distance formula between two points in 3D space. Depending on these distances we calculate relative location of projections of these points on the rational surface, parameterised by $0 < s,t < 1$. Using the correspondence between points we then decide on the area projection from control polygonal mesh to the smooth surface. Figure 4(a) illustrates the texture synthesized onto the control polygon using the proposed algorithm and figure 4(b) illustrates the mapped texture onto the smooth rational surface..

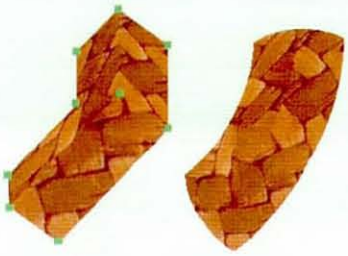


Figure4: (a) Texture on control polygon (b) Projection from control Polygon to smooth surface.

Note that closer the control polygon to the smooth surface representation, lesser distortion in projection will occur and vice –versa.

3.4 Progressive Texture Using EZW

In order to achieve progressive texture synthesis on surfaces, we adopt Shapiro's EZW idea in which coefficient values with magnitudes above a given threshold are considered significant. This threshold (t) is calculated using equation (3.6) based on the magnitude of wavelet coefficients of decomposed sample image.

$$t = 2^{\left\lfloor \log_2 \left(MAX\left(\left| LL_n(x,y) \right| \right) \right) \right\rfloor / K} \quad -(3.6)$$

where *MAX()* means the maximum coefficient value, K is a constant and $LL_n(x,y)$ denotes a general coefficient in LL_n sub-band. By only considering the coefficients of sub-bands, which are larger than the threshold and ignoring all others (i.e. setting to zero), an inverse DWT is calculated to produce the texture at a given progressive texture quality. This can be expressed generally as:

$$(B_{LLn-1}) = EZWIDWT \begin{pmatrix} B_{LLn}, B_{LHn}, \\ B_{HLn}, B_{HHn}, \\ thershold \end{pmatrix} \quad -(3.7)$$

The above equation can produce discrete quality levels of texture depending on threshold or number of coefficients need to be considered. Image quality can be increased by decreasing the threshold and vies-versa. Note that the function *EZWIDWT* above represent an EZW constrained inverse discrete wavelet transform. When progressive texture synthesis is required we replace the normal texture mapping process with the above EZW based approach (see figure 5 & 6).

4 EXPERIMENTAL RESULTS & ANALYSIS

In order to analyse the performance of proposed algorithm and to show that surfaces can be rendered effectively, we have implemented the proposed algorithms in OpenGL, C++.

Experiment were performed on a diverse range of texture samples that include regular, near-regular, irregular and stochastic (Lin et al., 2004) textures. Results illustrated in figure 5 indicate the ability of proposed technique to efficiently map and synthesized texture on surfaces, with minimal artifacts. Further as matching and searching is performed in wavelet domain, the texture synthesis is fast. Textures illustrated in Figure 5 (a), (b), (c) respectively belong to near-regular, regular, and stochastic categories. Similar synthesized quality levels are demonstrated for all three texture categories. Further analysis revealed that the time required to synthesize these texture is in the range of few milliseconds.

To further extend the functionality of the proposed method, we have extended our work to progressive texture synthesis on surfaces. We have preformed a wide range of experiments (see figure 6 & 7) to show that texture can be synthesized at seamlessly different levels of quality on surfaces, without consuming noticeable processing time. Figure 6 illustrates the synthesis of a stochastic texture of a flower. It is evident from the results that only 10% of information from sample texture is sufficient to create a texture with sufficiently rough quality. By increasing the percentage of coefficients further, the quality of the synthesized texture can be seamlessly improved. Further experiments revealed that for this texture, 20% of coefficients was sufficient to synthesize a texture visually equal to the texture that can be synthesized when all coefficients are utilized. Progressive texture synthesis gives the added advantage of being able to truncate a bit stream representing the sample texture at any intermediate stage, still being able to synthesize texture at some intermediate quality level.

To further illustrate the application of the proposed idea, we have extended our approach to synthesizing texture on two smoothly joined biquadratic rational surfaces, shown in figure 7. Figure 7(c) shows two smoothly join biquadratic patches. Figure 7(d) to 7(k) illustrates progressive texture synthesis on this surface. This proves that our technique can be extended to the many geometric topologies. Results in figure 7 further illustrates using regular and near-regular texture

samples that texture variations across patch boundaries are smooth.

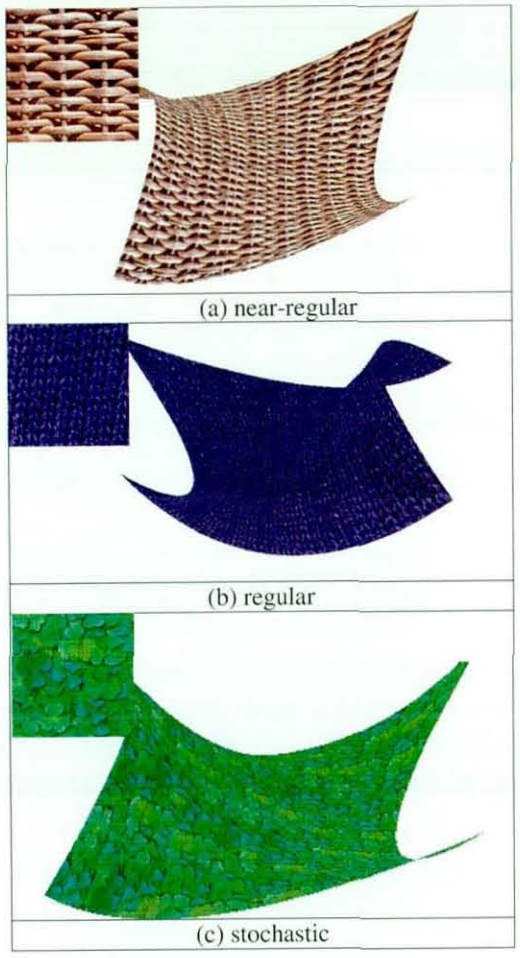


Figure 5: Texture synthesis on biquadratic surface

5 CONCLUSIONS

We have introduced a novel DWT based approach to synthesizing, texture and progressive texture, on biquadratic surfaces. We have presented the methods and algorithms in detail along with possible applications and advantages. The proposed method has the capability of synthesizing texture at seamlessly different quality settings, a functionality which is not possible via existing state-of –the art techniques.

The use of visual prioritisation of information in the sample image during texture synthesis allows the task to be carried out at a higher speed but at an equivalent visual quality level. We show that the proposed approach is computationally efficient, results in good quality texture synthesis, and is applicable in bandwidth-adaptive/compressed-domain applications such as remote visualization. We have shown that the control polygon strategy used can be extended to cover synthesizing texture on many 3D objects with arbitrary surface topology. We are currently in the process of generalizing the proposed algorithm to address this issue.

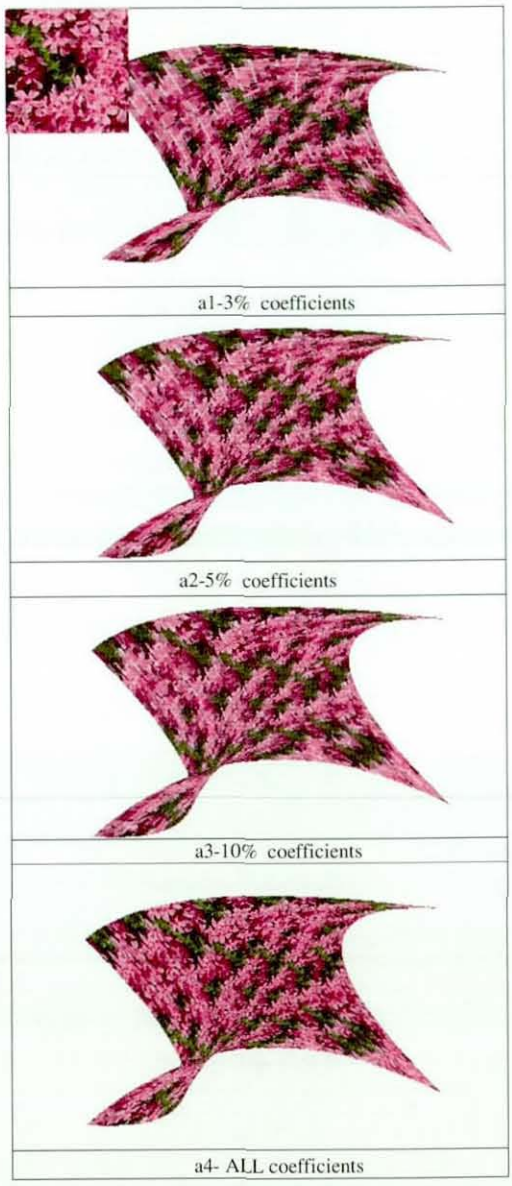


Figure 6: Stochastic progressive texture on biquadratic surface.

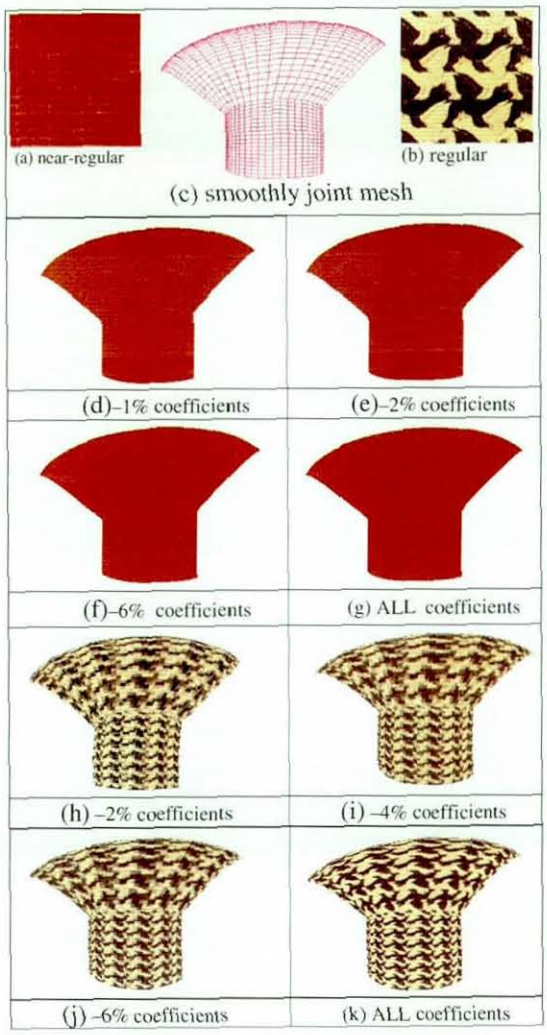


Figure 7: Progressive texture on two-joined biquadratic surface.

REFERENCES

Ashikhmin, M., Synthesizing natural textures, *ACM Symp. on Interactive 3D Graphics (March)*, 217–226, 2001.

Bez H.E., Bounded domain, bi-quadratic rational parametrisations of Dupin cyclides, Report No. 1090, 2006, Dept of CS, Loughborough University.

Chi-Wing Fu and Man-Kang Leung,Texture Tiling on Arbitrary Topological Surfaces *in Proceedings of Eurographics Symposium on Rendering 2005 (EGSR 2005)*, Germany,2005, pp. 99-104.

C. Soler, M.-P. Cani, and A. Angelidis, Hierarchical Pattern Mapping *Proceedings of Siggraph* 2002, *vol. 21, no. 3, pp. 673-680, 2002*

Heeger, David J. and James R. Bergen, Pyramid-Based Texture Analysis/Synthesis, *In proceeding of SIGGRAPH 95*, Aug., 229–238, 1995.

J. M. Shapiro, Embedded Image Coding Using Zerotrees of Wavelet Coefficient", *IEEE Trans. Signal Processing, December, VOL 41(no.12), 3445-3462, 1993.*

Lujin Wang, Xianfeng Gu, Klaus Mueller, Shing-Tung Yau: Uniform texture synthesis and texture mapping using global Parameterization,, *Volume 21, Numbers 8-10, September 2005, Pages: 801 - 810 Special Issues of Pacific Graphics 2005.*

Neyret F., and Cani, M.P., Pattern-based texturing revisited, *Proc. of SIGGRAPH 99, 235–242, 1999.*

Praun E., Finkelstein, A., and Hoppe, H., Lapped textures, *Proceedings of SIGGRAPH 00*, 465–470,

Rupesh N. Shet, Eran A. Edirisinghe, and Helmut E. Bez. Progressive texture synthesis on 3D surfaces. In Proceedings of VIIP06 Sixth international conference, pages 136–141, 2006.

Soucy, Marc, Guy Godin and Marc Rioux, A Texture-Mapping Approach for the Compression of Colored 3D triangulations, *The Visual Computer, Vol. 12, No. 10, 1996, pp. 503–514*

Sebastian Magda, David Kriegman, Fast Texture synthesis on Arbitrary meshes, *Proceedings of the 14th Eurographics workshop on Rendering Vol. 44, pp: 82 - 89 ,2003*

Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. ACM Trans. Graph., 25(3):541–548, 2006. ISSN:0730-0301.

Tong Xin , Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, Heung- Yeung Shum, Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces, *SIGGRAPH 2002, Pages: 665 – 672*

Turk, G., Texture synthesis on surfaces, *Proc. Of SIGGRAPH2001*, 347–354, 2001(August).

Wickramanayake, D.S., Edirisinghe, E.A., Bez H.E., Multiresolution texture synthesis in wavelet transform domain, *The Journal of Imaging Science and Technology* 2005.50(1):93-102, 1 2006.

Wei L.-Y., and Levoy, M., Fast texture synthesis using tree-structured vector quantization, *Proc. of SIGGRAPH 2002*, 479–488,

Wei L.-Y., and Levoy, M., Texture synthesis over arbitrary manifold surface, *Proceed of SIGGRAPH 2001*, 355–360, 2001.

Wen-chieh Lin, James Hays, Chenyu Wu, Vivek Kwatra, Yanxi Liu, A comparison study of four texture synthesis algorithm on near-regular textures. *Tech. Report CMU-RI-TR-04-01, Robotics Institute, Carnegie Mellon University, 2004. Also appeared in Poster Session SIGGRAPH, August 2004.*

Ying, L., Hertzmann, A., Biermann H., and Zorin, D., 2001, Texture and shape synthesis on surfaces, *Eurographics Rendering Workshop*, 301–312, 2001.

Texture Synthesis on Arbitrarily Shaped Surfaces Using Bézier Control Polygons

Rupesh N. Shet[1], Helmut E. Bez[1], Eran A. Edirisinghe[1]

[1]Department of Computer Science,
Loughborough University, UK
E.A.Edirisinghe@lboro.ac.uk

Abstract

This paper presents a novel algorithm for the texture synthesis of arbitrarily shaped surfaces. The texture is initially synthesized onto control polygons using a patch based, multi-resolution, progressive, Discrete Wavelet Transform (DWT) domain texture synthesis algorithm and is subsequently projected onto the arbitrarily shaped, smooth, surface of the given object, which has been represented by Bezier patches. The algorithm has a unique combination of several features that enables its practical use in application scenarios in which existing texture synthesis algorithms fail to deliver. The DWT domain multi-resolution texture synthesis algorithm adopted, supported by the popular Embedded Zerotree Wavelet (EZW) based coefficient prioritization scheme allows fast, multi-resolution, progressive transmission and synthesis capability of texture. The data efficient, Bezier representation of the surfaces to which the texture is projected, results in the possibility of surface animation. The control polygon based texture projection approach enables a flexible approach to texture synthesis of complexly shaped 3D objects. Further the algorithm allows the user definition of vector fields depicting the preferred direction of texture propagation at a given location of the surface. We provide experimental results synthesizing textures of different nature on arbitrarily shaped surfaces of a number of 3D objects. It is shown that efficient, flexible, seamless texture synthesis is possible with most patterns of texture.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation display algorithm; I.3.7 [Computer Graphics]: 3D Graphic and Realism: Texture; J.6 [Computer Graphics]: CAD

1. Introduction

During the past decade the use of user defined vector fields to indicate the texture direction in synthesizing texture on 3D models have gained widespread popularity particularly due to the demands in application areas such as digital cinema creation and 3D computer games. Thus novel approaches to designing vector fields and efficient algorithms for synthesizing texture on 2D/3D surfaces have attracted much attention from the research community. Therefore efficient solutions to both research challenges are vital in achieving good quality texture production on surfaces. Whilst the texture direction can enhance perceived texture quality and thus realism, efficient texture synthesis algorithms enable quick and accurate texturing of surfaces.

A survey of existing approaches to patch based texture synthesis (see section 2) reveals that most of the proposed algorithms use irregular shapes and sizes of triangular meshes, which often result in seams at patch boundaries. Further the varying nature of the size of triangles defining the mesh, results in additional costs in searching for matching texture patches and in their subsequent blending. In addition the inability of representing triangular meshes and the related texture patches in a progressively refined manner, limits their use in applications utilising bandwidth constraint transmission media for data transmission, such as the internet and mobile communication channels. Further the animations of textured surfaces are difficult as they are rigidly defined. Overcoming the above limitations of the existing approaches, in this paper we propose the use of Bézier polygons and an associated progressive, patch based texture synthesis approach.

The paper is organised as follows: Apart from this section which introduces the reader to the problem domain, sections-2 provides an overview of literature. Section-3 presents the proposed texture synthesis algorithm in particular providing details of the underlying techniques used, such as basics of multiresolution DWT representation of texture, texture synthesis on Bézier control polygons, using weighted edge blending technique to improve quality of seams between Bézier polygonal faces and the use of Shapiro's EZW algorithm [Sha93] for DWT coefficient prioritisation in providing progressive synthesis of texture. Section-4 provides experimental results and their analysis. Finally, Section-5 concludes, with an insight to possible improvements and future variations.

2. Related Work

A number approaches have been proposed in literature for texture synthesis on 3D surfaces. Two important objectives of these approaches have been the avoidance of patch boundary artifacts and the minimization of unrealistic

stretching and distortion of the texture pattern, once synthesized on a given surface. These approaches [WL01, Tur01, YHBZ01, Ash01, TJL*02, NC99, PFH00, SCA02, MK03, LXKST05, FMK05, SEB06, LH06] can be broadly classified into two groups, namely, pixel based texture synthesis algorithms [WL00, WL01, Tur01, YHBZ01, Ash01, TJL_02, SEB06, LH06] and patch based texture synthesis approaches [NC99, PFH00, SCA02, MK03, LXKST05, FMK05]. The patch based texture synthesis algorithms have the inherent advantage of being faster and being able to more easily maintain the original texture properties, as compared to pixel based approaches. The focus of our current research is on a novel approach to patch based texture synthesis that aims to avoid a number of limitations of existing patch based texture synthesis approaches (see section 1). Hence only patch based texture synthesis approaches are reviewed in this section.

Patch based texture synthesis approaches attempt to synthesis texture by copying selected regions of pixels from a given sample texture and stitching them together. These approaches overcome some of the limitations of pixel based approaches as the techniques are applicable to any generic texture and are relatively computationally inexpensive. The early work of Neyret and Cani proposed a technique that is based on pre-computed triangular texture samples. However, good synthesis results were obtained only for isotropic textures [NC99]. Praun introduced the lapped texture synthesis technique where oriented texture patches are placed in an overlapping fashion on a surface. However experiments were limited to a specific class of texture [PFH00]. Soler introduced hierarchical texturing to overcome drawbacks of previous algorithms. The basic idea was to capture low frequency patterns while preserving high frequency randomness in texture. The generalised idea is to pick up texture patches from the sample image and map them onto the surface so that they can fit with their neighbours. Although the resulting texture often contains some seams at patch edges, the author demonstrated that the seams can be further reduced by locally readjusting texture coordinates at patch boundaries. The running time is closely related to the number of resulting patches, which depends on both the mesh geometry and the texture sample [SCA02]. Sebastian et al proposed an algorithm that separated a texture pre-processing step from the actual texture synthesis stage. Pre-processed texture is stored and used when needed. Since the access of the stored texture data can be slow a lookup table is used to speed up memory access [MK03]. Wang et al proposed an algorithm based on global conformal parameterization of surfaces, where the textures are preserved on surfaces without seams or cracks. Moreover, the parameterization can segment the surface into patches, where each patch is mapped to a planar rectangle. This method is used to synthesize texture on a 2D rectangle. However non-uniformity of synthesized texture is a shortcoming of the method as a result of using an area stretching factor which is obtained from the conformal factor. The algorithm adopts a complex procedure in parameterization, and in creating multi-scale samples, leading to an increased computational cost in the synthesis process [LXKST05]. Chi-Wing Fu et al proposed a texture tiling mechanism, where a low distortion conformal quad based map is created for the input surface. It produces proper tile orientations on all quad faces so that the texture can be laid out on quads and mapped back to the input surface accordingly. This texture synthesis algorithm is based on the image quilting approach of [AF01] and therefore inherits all the drawbacks of traditional image quilting algorithms such as random selection of patches, expensive cost of minimum boundary cut, somewhat limited applicability to isotropic textures and computation cost due to the need of generating pre-synthesis textures [FMK05].

To overcome some of the shortcomings of the existing patch based texture synthesis approaches, we propose to use a geometric surface representation, namely the Bézier polygons. We show that the use of Bezier polygons enables the progressive compression of the surface representation mesh, thus extending the method's applicability to texture synthesis applications utilising bandwidth and resource constrained transmission media for data transmission. Further the use of the geometric surface based approach provides additional benefits such as the ease of surface animation. In addition the proposed approach uses Discreet Wavelet Transform (DWT) domain texture representation guided by user defined vector fields that enable progressive and realistic refinement of texture on surfaces. We show that the transform domain texture representation considerably increases the synthesis speed. We further discuss that the use of Bézier polygons in surface representation enables high data compression possibility of 3D mesh models and increases the overall texture synthesis speed. Finally we point out that the abovementioned special features extend the proposed methods applicability in distributed and/or collaborative gaming and in digital movie application domains.

The following section introduces the proposed approach giving particular attention to design and practical details.

3. Proposed Algorithm

For the purpose of clarity the explanations are organized into separate sections as follows:

3.1. An overview

The basic logical flow of the proposed algorithm is illustrated in Figure 1. The process begins with the reading of the Bézier control polygonal information from the Bézier Patch file (see section 3.2) of the 3D surface. Subsequently the user interactively marks the required texture directions on the control polygons in the form of vectors (see section 3.3). All faces of the control polygons are then projected on to a 2D plane (see section 3.4). In other words the user defined vector fields representing the expected texture directions along with the polygonal faces gets projected on to the 2D plane. The first marked user defined vector is considered the anchor vector and is used to initiate vector propagation (see section 3.3). It is noted that the time taken for vector propagation depends on the time spent by the user in manually marking the vector fields.

Once the above process has been completed, the texture synthesis commences, where the texture of each Bézier polygon is synthesized (see sections 3.5, 3.6, 3.7). Finally, the synthesized textures on control polygons are projected back on to the actual Bézier patches representing the 3D surface, thus completing the texture synthesis process (see section 3.8).

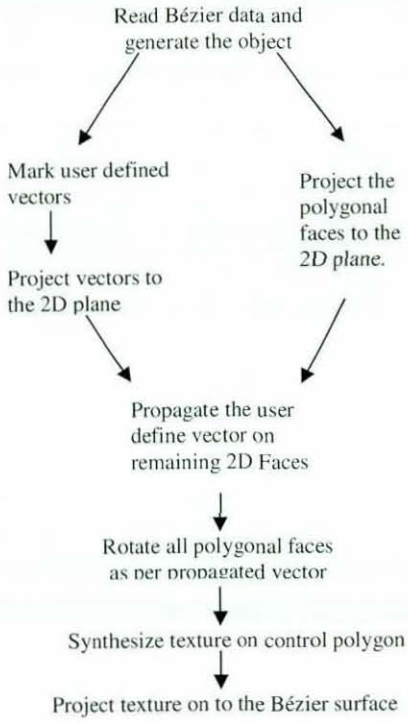


Figure 1: *Logical flow of the proposed algorithms*

The following sections provide further design and implementation details of each sub process of the proposed texture synthesis approach.

3.2. Parametrisation of Bézier Patches

The process begins with the reading of the file that contains the vertex data of the Bézier control polygons (i.e. the coordinates of the Bezier vertices / green dots in figure 2). Each Bezier patch is subsequently parametrised to form a smooth surface defining a part of the surface of the 3D object.

The process of parametrisation of a single Bezier patch can be summarized as follows (for more details readers are referred to [Bez06]): Given a Bezier patch, τ, we first use the vertex data of the nine control points to parametrise the surface (see Figure 2) as,

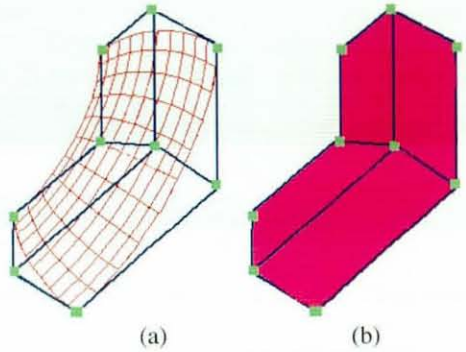


Figure2: (a) Green Colour: Control polygon points of a Biquadratic surface, Red Colour: Smooth surface mesh generated using the control polygon (b) Control polygon mesh (note: for 9 control points 4 faces are defined).

$$\tau(s,t) = \frac{\sum_{k,j=0}^{n,m} b_{n,k}(t)\, b_{m,j}(s)\, \overset{*}{v}_{k,j}}{\sum_{k,j=0}^{n,m} b_{n,k}(t)\, b_{m,j}(s)\, \omega_{k,j}} \quad 0 \le s,t \le 1, \quad (3.1)$$

where $\omega_{k,j}$ are the weights defining the specific shape of the smooth surface, $v^{*}_{k,j}$ are the Bernstein vectors and $b_{n,k}(t)$, $b_{m,j}(s)$ are a Bernstein polynomial calculated as follows:

$$b_{n,i}(u) = \binom{n}{i} u^{i} \, (1-u)^{n-i} \quad (3.2)$$

If all the weights are non-zero this may be expressed as,

$$\tau(s,t) = \frac{\sum_{k,j=0}^{n,m} b_{n,k}(t)\, b_{m,j}(s)\, \omega_{k,j} V_{k,j}}{\sum_{k,j=0}^{n,m} b_{n,k}(t)\, b_{m,j}(s)\, \omega_{k,j}}, \quad 0 \le s,t \le 1 \quad (3.2)$$

where $V_{k,j} = \frac{\overset{*}{v}_{k,j}}{\omega_{k,j}}$ are the Bézier vertices. The values of n and m determine the degree of the parametrisation; if $n = m = 2$ the patch is said to be biquadratic and if $n = m = 3$ it is said to be bi-cubic. Within the context our present research we have assumed a biquadratic parametrisation.

Subsequent to the parametrisation of the Bezier patches, user input/intervention is sought for defining the tangential vectors representing the texture directions on control polygons. This is discussed in detail in the subsequent section.

3.3. User Defined Tangential Vectors and their Propagation

The most frequently used approach in maintaining realism in 3D texture synthesis is to allow the user to interactively specify texture directions (user defined tangential vectors) on selected patches, which are (i.e. the directions) subsequently propagated to provide the texture directions of all other patches defining the surface.

In our approach the user initially marks texture directions on a selected number of faces (note: minimum of one face is required). The algorithm then considers one of the marked faces (usually the one selected first) to be a 'seed face' and will select a neighbouring non-marked face to commence vector propagation.

The vector field (T_f) of a non marked face, c, is calculated as follows:

$$T_{f,c} = (\frac{1}{k})\sum_{i=1}^{k} T_{f,i} \qquad (3.3)$$

where $T_{f,c}$ is the vector field of the non-marked face (i.e. the field being calculated) and $T_{f,i}, T_{f,i+1} \ldots T_{f,n}$ represent the vector fields of the face's, n-connected neighbourhood. (note: in our experiments n=9). Starting from a neighboring block of the seed face we use the above equation to propagate the texture field to cover all non-marked faces of the 3D object.

Once $T_{f,c}$ is obtained using equation 3.3, it may well be placed outside the face. Therefore we move the vector field so that it will be restricted within the face, $T_{f,c}^{*}$, as follows:

$$T_{f,c}^{*} = T_{f,c} * \alpha_1 + T_{f,c} * \alpha_2 \qquad (3.4)$$

where,

$$\alpha_1 = b_1 \bullet T_{f,c} \qquad \alpha_2 = b_2 \bullet T_{f,c} \qquad (3.5)$$

and b_1, b_2 are the face basis vectors.

The vector fields thus obtained are used for the appropriate rotation of the face before being projected to the 2D plane (see section 3.4). This results in minimizing distortion at the patch boundaries.

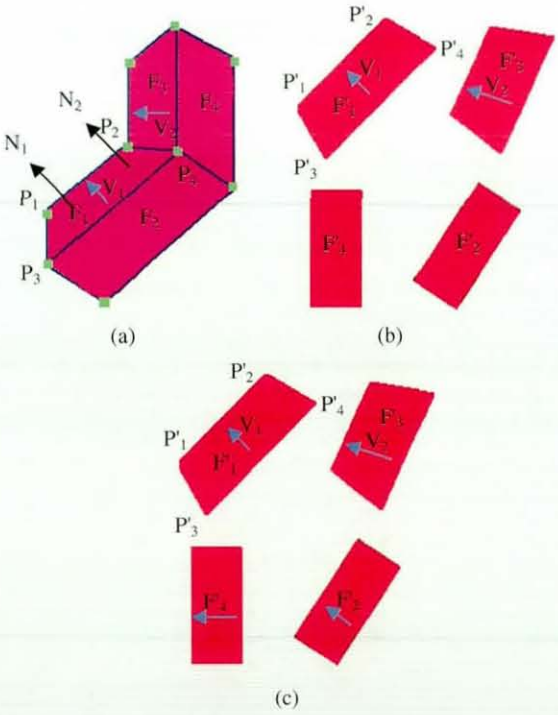
3.4. Project 3D Face on to 2D Plane

To synthesize a texture on a face of the Bézier control polygon, one needs to appropriately align and project the face onto the texture space. The mapping between control polygon faces and planer polygons on the texture space should be isometric, i.e., persevering both angle and distance.

The alignment and projection procedure is illustrated in figure 3. A given arbitrary surface F_1 (P_1, P_2, P_3, P_4) can be assumed to be comprised of two triangles i.e. t_1= Δ (P_1, P_3, P_4) and t_2=Δ (P_1, P_2, P_4) (See Figure 3 (a)), with the surface normals denoted by N_1, N_2. The angle between the normal of each triangle and normal of the 2D plane to which it is to be projected is measured initially. Subsequently the vertices of each triangle are rotated such that their normals are aligned with the normal of the 2D texture space. Finally the adjusted vertices are projected on to the 2D texture space (see Figure 3 (b)) and the two triangles are then combined together (see Figure 3 (d)) carefully such that no overlap occurs (note: additional displacement of vertices of the shared edges of the two triangles may be required). In figure 3, the arbitrary Bezier surface, F_1 is rotated/aligned and projected as F'_1 (P'_1, P'_2, P'_3, P'_4) (See Figure 3 (b)) on the 2D texture space.

In carrying out the above explained projections, if a given Bezier polygon has been assigned a user defined vector field, it is appropriately transferred to its projection in the texture space. Subsequently following the vector field propagation procedure explained in section 3.3, the user defined, projected vector fields are propagated to obtain the vector fields of all projected polygons. All projected faces are finally rotated so that their vector fields (i.e. vectors that define their intended texture direction) are aligned with the y-direction of the texture space (see Figure 3 (c) and (d), i.e., F'_1 is rotated to F''_1 (P''_1, P''_2, P''_3, P''_4)) in order to obtain a desired, realistic texture pattern on surface.

These faces are then moved to the texture space and arranged next to each other as illustrated in Figure 3 (d). Subsequently we assign the texture space to the corresponding control polygon faces. Then we begin with the actual texture synthesis process which is described in the next section.



(d)

Figure 3: *3D faces to 2D faces*

3.5. Texture Synthesis on Control Polygons

3.5.1 Texture Synthesis Approach

Figure 4 illustrates the block diagram of the texture synthesis process on a control polygon. The texture synthesis process starts by applying a n-level (n=1,2,3 used in our experiments) 2D DWT (e.g. Haar Transform) on sample texture image, which is denoted as I_{sample}. The application of single level 2D DWT on the sample texture results in a set of component images i.e. sub-bands, as follows:

$$(I_{LL1}, I_{HL1}, I_{LH1}, I_{HH1}) = DWT(I_{sample}) \qquad (3.6)$$

Where $I_{LL1}, I_{HL1}, I_{LH1}, I_{HH1}$ are the texture image sub-bands corresponding respectively to low-resolution approximation, vertical details, horizontal details and diagonal details of sample texture. Similarly 2nd level and 3rd level decompositions are obtained by applying DWT to the low-resolution sub-bands of previous decomposition level. This can be mathematically represented as follows.

$$(I_{LL2}, I_{HL2}, I_{LH2}, I_{HH2}) = DWT(I_{LL1}) \qquad (3.7)$$

$$(I_{LL3}, I_{HL3}, I_{LH3}, I_{HH3}) = DWT(I_{LL2}) \qquad (3.8)$$

$$(I_{LLn+1}, I_{HLn+1}, I_{LHn+1}, I_{HHn+1}) = DWT(I_{LLn}) \qquad (3.9)$$

from n-level we extract low-resolution and diagonal detail bands i.e. I_{LLn} and I_{HHn}.

We generalize the notation used in equation (3.6)-(3.10) as I_{pl} where $p \in \{LL, HL, LH, HH\}$ and $l \in \{0,1,2,..n\}$ where p represents the sub-bands within each decomposition level (LL-low resolution, LH-horizontal, HL- vertical, HH diagonal) and l represent the decomposition level. *DWT* represent the forward discrete wavelet transform.

The basic idea of proposed algorithm is to synthesize texture on control polygons of a given surface. Let $B_{LLn(x,y)}$ represent a general polygonal block of decomposed sample image located at position (x, y) relative to the sub-bands (p, l)'s origin.

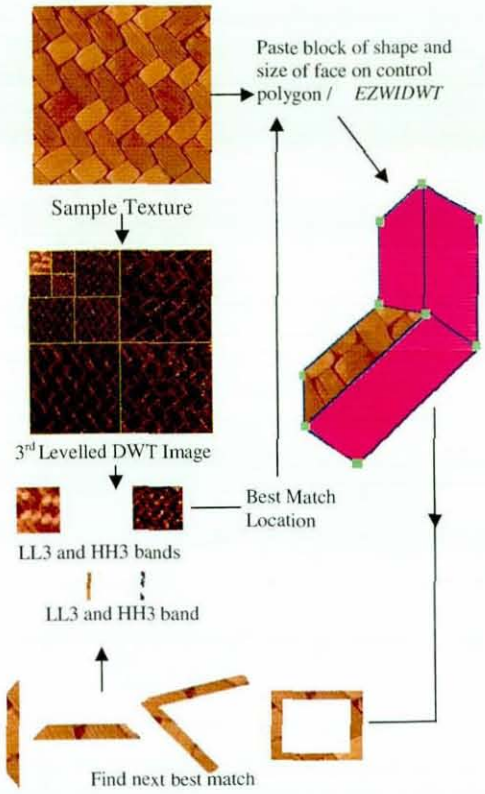


Figure 4: *Proposed block diagram for texture synthesis on control polygon surfaces.*

Initially we randomly pick block $B_{LLn(x,y)}$ from the sample texture image with identical size and shape to that of the control polygon face. This randomly created texture block is mapped to the control polygon surface shown in figure 4. [Note: the details of the mapping process are described in section 3.7.]

In locating the next block to be synthesized, we cut a 8 pixel wide template of pixels along the edge of already synthesized, neighboring blocks, apply a 3 level DWT decomposition on the template block and extract low-resolution and diagonal detail bands which are used for searching in sample LL3 and HH3 bands. Best matching block can be found by minimizing the L2 norm.

To speed up the block matching we only use all coefficients of the LLn and HHn sub-bands for matching. In our experiment where $n=3$ and a sample texture size of 256 × 256 is used, out of the 65536 possible coefficients, we only use 1024 coefficient i.e. 1/64 of total number coefficient in the sample texture. Therefore the matching speed is significantly enhanced. In order to achieve good quality on surface these approach is not always applicable all the time. Depends on the texture pattern these coefficients can be increases or decreases. EZW algorithm can be used for the coefficient prioritisation to further reduce complexity.

In general, if $B_{pl(x1,y1)}$ and $B_{pl(x2,y2)}$ are two randomly shaped blocks to be matched, we say $B_{pl(x1,y1)}$ is the best match for $B_{pl(x2,y2)}$ if $d(B_{pl(x1,y1)}, B_{pl(x2,y2)})$ is minimum for all possible B_{pl} blocks, which is calculated as,

$$d(B_{(x2,y2)}, B_{(x2,y2)}) = \sum_{i,j=0}^{n} [[dB_{B_{(x1,y1)}}(i,j) - dB_{B_{(x2,y2)}}(i,j)]^2 + [dB_{\partial B_{(x1,y1)}}(i,j) - dB_{\partial B_{(x2,y2)}}(i,j)]^2] \quad (3.10)$$

Where ∂B_{pl} an edge is zone of block $B_{pl(x,y)}$ and i, j is an element (coefficient) within the edge zone.

Finally the overlap area of the best matching edge is blended with the overlapped area of the original block using weighted edge bending (see section 3.5.2 for more details). The non-overlapping area of the block is picked from the sample texture and subsequently appended to the synthesized texture. This process is continued till all the faces of control polygon are mapped. In some cases we have considered two or more overlapping areas for finding the best match. The possibility of having three or four edges is rare.

3.5.2 Directional Weighted Edge Blending

Performing texture blending within 2D texture synthesis is much simpler as compared to performing it within 3D texture synthesis. The changes that occur in the direction of texture blending when the direction of texture synthesis changes is considered a key challenge that needs to be faced. Texture synthesis over 3D faces mostly depend on a user defines vector field and propagation of vectors. For any quadrilateral face there are four direction of blending (see figure 4).

Figure 5: *Four direction of blending*

It is observed that to obtain a perfect match between two neighbouring faces, is rare. Therefore to minimize the effects of artifacts at seams of patches, edge blending can be carried out. To this effect we propose the use of a pixel domain, weighted edge blending function. The general idea of weighted edge blending is based on gradually reducing the luminance contribution made by a neighbouring face to its seam, while moving away to the adjoining face across the seam. The

Let W represent an empirically defined matrix capable of performing weighted edge blending in the pixel domain, via matrix multiplication with the overlap area. Note: The size of W is depended on the width of the overlap. In our experiments we have assumed a width of overlap of eight pixels and have therefore defined W to be of size $w \times w$ *(w=width).*

$$W = \begin{bmatrix} 0.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/w & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2/w & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \bullet & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \bullet & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \bullet & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (w-2)/w & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (w-1)/w \end{bmatrix} \quad (3.11)$$

Note that the overlap area of the two adjoining faces are denoted by $Edge_{1olap}$ an $Edge_{2olap}$. The resulting blended overlap area, $O_{blendolap}$ can be represented by,

$$O_{blendolap} = Edge_{1olap} \; x \; W + Edge_{2olap} \; x(I_{ident} - W) \quad (3.12)$$

where I_{ident} is an identity matrix. Note that $Edge_{1olap}$ and $Edge_{2olap}$ are of size *w x height.*

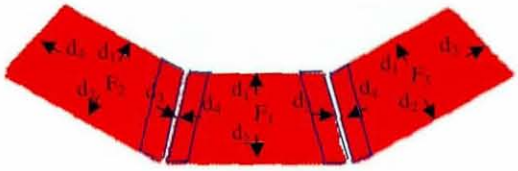


Figure 6: *Swap direction of blending*

Assume that F_1 is already a textured face. Therefore when texturing F_3 we use Eq (3.12) as the blending function, where d_4 is overlapped on d_3 and blended. When texturing F_2, d_3 is overlapped with d_4. In the above case it is important to decide whether w should multiply $Edge_{1olap}$ or $Edge_{2olap}$. It is decided on basis of whether the pixel value of the inside edge of already synthesized textured to undergo a change or would remain same. If it is to change then we swap the blending direction around and equation can be rewritten as follows:

$$O_{blendolap} = Edge_{1olap} \; x \; (I_{ident} - W) + Edge_{2olap} \; x W \quad (3.13)$$

The above blending approach helps in keeping the blended surface smooth, i.e. artifact free.

3.6. Projection of Texture on to a Smooth Surface

In order to project texture from the control polygon to the smooth surface, initially we calculate a length/distance between each of the control points of the control polygon, using the standard distance formula between two control points in 3D space. Subsequently depending on these distances and using the correspondence between points, we calculate the relative location of projections of these points on the smooth surface, parameterized by $0 \le s,t \le 1$. We

then decide on the area of projection from the control polygonal mesh to the smooth surface.

For clarity of presentation the projection of texture on a single dimension (i.e. a Bezier curve) can be presented as follows: Assume that curve c' (see Figure 7) is generated using four control points $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, $\mathbf{P}_4$ via the use of the Bézier curve equation 3.16. Further the lines, $\mathbf{P}_1$ to P2, $\mathbf{P}_2$ to $\mathbf{P}_3$ and $\mathbf{P}_3$ to $\mathbf{P}_4$ are assumed already textured (see Figure 7). We calculate the length between these control points, as say l_1, l_2 and l_3. The total length $l_{T,k}$ between the control points is calculated as the sum,

$$l_{T,k} = \sum_{i=1}^{k} li \qquad (3.14)$$

Where l_i is the distance between two adjacent control points, and k is the total number of adjacent pairs of control points. We normalize all the control point lengths to a unit length l_i^* which is calculated as.

$$l_{T,k}^* = \frac{1}{l_T} \sum_{i=1}^{k} l_i \qquad (3.15)$$

Once the parameterization of the curve has been completed as described above, projecting the texture on to the curve becomes fairly straightforward. It was mentioned that the texture on line segment $\mathbf{P}_1$ to $\mathbf{P}_2$ gets projected on to the smooth curve $\mathbf{P}_1$ to $\mathbf{P'}_2$ where $\mathbf{P'}_2$ can be calculated using the following Bezier curve equation, as;

$$P_i'(l_{T,k}^*) = \frac{\sum_{j=0}^{m} b_{m,j}(l_{T,k}^*)\omega_j v_j^*}{\sum_{j=0}^{m} b_{m,j}(l_{T,k}^*)\omega_j}, \qquad 0 \le l_{T,k}^* \le 1, \quad (3.16)$$

where ω_j are the weights defining the specific shape of the smooth surface, v^*_j are the Bernstein vectors and $b_{m,j}(s)$ is a Bernstein polynomial. The texture projection in general is represented as follows:

$$\text{Texture } (c'(s)) \equiv \text{Texture } (p(s)) \quad 0 \le s \le 1 \qquad (3.17)$$

Therefore in general a patch can be textured using the following equation,

$$\text{Texture } (\textit{smooth patch } (t, s)) \equiv \text{Texture } (\textit{control polygon } (t,s)) \qquad 0 \le s,t \le 1 \qquad (3.18)$$

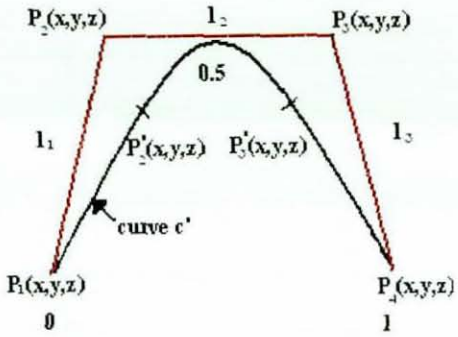


Figure 7: *Basic projection approach*

We used the above projection approach to project texture from the control polygon to a given smooth surface. (Figure 8(a) and 8(b) illustrates the mapping of texture from the control polygons onto the smooth rational surface).

Note that the closer the control polygon is to the smooth surface representation, lesser distortion in projection will occur and vice –versa.

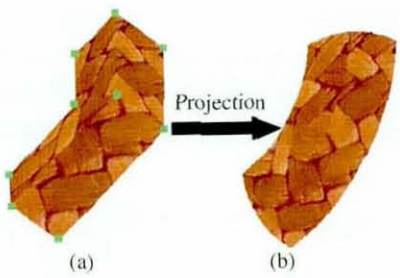


Figure 8: *(a) Texture on control polygon (b) Projection from control Polygon to smooth surface.*

3.7. Progressive Texture Synthesis

In order to achieve progressive nature of texture synthesis on surfaces, we adopted Shapiro's EZW idea [Sha93] which is based on the observation that larger wavelet coefficients are more visually important than smaller coefficients. In our work we have assumed that wavelet coefficient values with magnitudes above a given threshold are considered significant. The threshold provides the minimum number coefficient for the best possible perceptual quality. This threshold (t) is calculated using equation (3.19) based on the magnitude of wavelet coefficients of decomposed sample image.

$$t = 2^{\left\lfloor \log_2 \left(MAX \left(\left| LL_n(x,y) \right| \right) \right) \right\rfloor} \Big/ K - (3.19)$$

where $MAX()$ means the maximum coefficient value, K is a constant and $LL_n(x,y)$ denotes a general coefficient in

LL_n sub-band. A mask is created for the coefficients of sub-bands, which are larger than the threshold and ignoring all others (i.e. setting to zero). This results in perceptually significant frequency components being protected and used in the matching. Finally an inverse DWT is applied on to the masked sub-band decomposition, obtaining the resulting texture in the pixel domain. This can be expressed generally as:

$$(B_{LLn-1}) = EZWIDWT\ (B_{LLn}, B_{LHn}, B_{HLn}, B_{HHn}) - (3.20)$$

The above equation can produce discrete quality levels of texture depending on threshold or number of coefficients needed to be considered. Image quality can be increased by decreasing the threshold i.e. the K constant and vies-versa. Note that the function *EZWIDWT* above represent an EZW constrained inverse discrete wavelet transform. When progressive texture synthesis is required we replace the normal texture mapping process with the above EZW based approach (see figure 12).

3.8. Texture Mapping Approach

In our implementation initially we create so-called, propagating seed vector directions, which are then used to smoothen the surface vector field. However alternatively a number of other surface vector field techniques (Wei-Levoy, 2001; Turk 2002; Ying etal, 2001) can be used to replace the approach we have selected above. Once vector fields are assigned to all control polygon faces, we then rotate all the faces according to tangential vector field and surface normal, thus placing all faces in the same 2D plane. Using a modified version of Soucy et al., 1996, approach (Note: modified from using triangle to using polygon) a texture map. T is created. For each face of the control polygon, we calculate the bounding box of the face and then map it to a corresponding face in *T* in compact form, i.e. with no space being wasted. The faces in T are textured using the corresponding best matching block. The faces in T that we use are of non-uniform size that are a better fit to the shape and size. It is noted that the resulting texture can be rendered on the control polygon surfaces at interactive rates. The images illustrated in section-4 (See figure 9, 10 and 11) were rendered in this manner using 256 x 256 textures. The models used in our experiments are composed of smooth surfaces having between 1000 to 20000 faces, whereas the control polygons used consisted of 100 faces to 400 faces (It can be further increase to *n* faces). We have observed that these surfaces render at real-time rates.

4. Experimental Result and Analysis

In order to analyse the performance of proposed algorithm and to show that surfaces can be rendered effectively, we have implemented the proposed algorithms using OpenGL, C++.

Experiments were performed on a diverse range of texture samples that include regular, near-regular, irregular and stochastic [Lin] textures. Results illustrated in figures 9, 10, 11 and 12 indicate the ability of proposed technique to efficiently map on to arbitrary surfaces such Utah Teapot, Cup and geometric surfaces such as, Cyclide, Torus and Sphere with minimal visual artifact.

Figure 9 illustrates the complete cycle of the proposed texture synthesis process on Bézier surfaces. Figures 10 and 11 show a wide variety of texture samples being textured on surfaces. Table-1 illustrates the level of transform used to generate the result. We have used different levels of wavelet transforms for different textures, such as the stochastic texture can be synthesize using 3rd level of transform where as regular, irregular texture can synthesized in 2nd level of transform and near regular texture on 1st level of transform. Synthesizing speed is varies it depends on the pattern of texture and number of patches. The Table 1 and 2 shows the summary of texture patterns and wavelet transform used for the result shown in Figure 10 and 11.

Model	Texture Pattern	No of Patches	Level *n*
Teapot	Regular	32	2
Teapot	Near-Stochastic	32	3
Teapot	Irregular	32	2
Teapot	Near Regular	32	1
Cup	Irregular	26	2
Cup	Stochastic	26	3
Cup	Near-Stochastic	26	3
Cup	Regular	26	2

Table1: Arbitrary surface

Model	Texture Pattern	No of Patches	Level *n*
Torus	Near-Stochastic	16	3
Torus	Irregular	16	2
Torus	Irregular	16	2
Torus	Near-Stochastic	16	3
Cyclide	Stochastic	16	3
Cyclide	Irregular	16	2
Sphere	Stochastic	8	3
Sphere	Regular	8	2

Table2: Geometric surface

To further extend the functionality of proposed method, we have extended our work to progressive texture synthesis on surfaces. We have performed a wide range of experiments. Figures 12 and 13 shows that texture can be synthesized at seamlessly different levels of quality on surfaces; Figure 12 and 13 illustrates the synthesis of irregular texture of animal skin on Utah teapot. It is evident from the result that 2% of information from sample texture is sufficient to create texture with sufficiently rough quality. By increasing the percentage of coefficient, the quality of the texture can be seamlessly improved. Further experiments reveal that 20% of coefficients was sufficient to synthesise a texture visually equal to the texture that can be synthesized when all coefficient are utilized. Due to space limitation we have restricted the number of discrete texture results to eight, where as the proposed method is capable to of progressively synthesis texture at seamlessly different number of levels. Progressive texture synthesis

give the added advantage of being able to truncate a bit stream representing the sample texture at any intermediate stage still being able to synthesize texture at some intermediate quality level. The results prove that our technique can be extended to many surface topologies.

5. Conclusion and Future Work

We have introduced a novel wavelet based searching approach to synthesize texture and progressive texture, on Bézier surfaces. We have presented the methods and algorithms in detail along with possible applications and advantages. The proposed method has the capability of synthesizing texture at seamlessly different quality settings, a functionality which is not possible via existing state-of – the art techniques.

The use of visual prioritisation of information in the sample image during texture synthesis allows the task to be carried out at a higher speed but at an equivalent visual quality level. We show that the proposed approach is computationally efficient, results in good quality texture synthesis, and is applicable in bandwidth limited channel applications such as remote visualization. We have shown that the control polygon strategy used can be extended to cover synthesizing texture on many 3D objects with arbitrary surface topology.

The popular Wang tile approach for synthesizing texture on surfaces can be pluged-in to our approach of synthesizing texture with minor modification. In future this can be easily pluged-in to OPENGL rendering

Acknowledgements

We would like to thank Ryan Holmes for providing details of the BPT format and models through his website. Further we would like to thank Vivek Kwatra for providing texture samples through his website.

References

[Ash01] ASHIKHMIN M.: Synthesizing natural textures. *In ACM Symp. on Interactive 3D Graphics* (2001),pp. 217- 226.

[BA83] BURT P. J., ADELSON E. H.: A multiresolution spline with application to image mosaics. *ACM TOG* 2, 4 (12 1983), 217–236. ISSN:0730-0301.

[Bez06] BEZ H. E.: Bounded domain, bi-quadratic rational parametrisations of dupin cyclides. *In Research Report No. 1090*, Department of Computer Science, Loughborough University (2006).

[CLJC*04] CHIEH. LIN W., JAMES H., CHENYU W., VIVEK K., YANXI L.: A comparison study of four texture synthesis algorithm on near-regular textures. *In Tech. Report CMU-RI-TR-04-01*, Robotics Institute, Carnegie Mellon University (2004).

[DMPxx] DUTTA D., MARTIN R. R., PRATT M. J.: Cyclides in surface and solid modeling. *IEEE Computer Graphics and Applications* 13.

[EF01] EFROS A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. *In Proc. of SIGGRAPH 2001* (2001), pp. 341–346.

[FGP05] FOUFOU S., GARNIER L., PRATT M. J.: Conversion of dupin cyclide patches into rational Biquadratic bezier form. *In Proc. of the IMA Mathematics of Surfaces XI (2005)*, pp. 201–218.

[FMK05] FU C.-W., MAN-KANG L.: Texture tiling on arbitrary topological surfaces. *In Proceedings of Eurographic Symposium on Rendering 2005* (EGSR 2005) (2005), pp. 99–104.

[LH06] Lefebvre S. and Hoppe H. Appearance-space texture synthesis. *ACM Trans. Graph.*, 25(3):541–548, 2006. ISSN:0730-0301.

[LXKST05] LUJIN W., XIANFENG G., KLAUS M., SHING-TUNG Y.: Uniform texture synthesis and texture mapping using global parameterization. *In Special Issues of Pacific Graphics*, ISSN:"a0178-2789 (2005), vol. 21, pp. 801–810.

[MdPS86] MARTIN R., DE PONT J., SHARROCK T.: Cyclide surfaces in computer aided design. *In Proceedings of the IMA Conference on the Mathematics of Surfaces* (1986).

[MK03] MAGDA S., KRIEGMAN D.: Fast texture synthesis on arbitrary meshes. *In Proc. of the 14th Eurographics workshop on rendering (2003)*, vol. 44, pp. 82–89.

[NC99] NEYRET F., CANI M. P.: Pattern-based texturing revisited. *In Proc. of SIGGRAPH 1999* (1999), p. 235- 242.

[PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: Lapped textures. *In Proc. of SIGGRAPH 2000* (2000), p. 465- 470.

[Pra90] PRATT M. J.: Cyclides in computer aided geometric design. *In Computer Aided Geometric Design (1990)*, vol. 7, pp. 221–242.

[Pra95] PRATT M. J.: Cyclides in computer aided geometric design II. *In Computer Aided Geometric Design (1995)*, vol. 12, p. 131- 152.

[SCA02] SOLER C., CANI M., ANGELIDIS A.: Hierarchical pattern mapping. *In Proc. of SIGGRAPH 2002 (2002)*, vol. 21, pp. 673–680.

[SEB06]Shet R. N., Edirisinghe E.A., and Bez H.E. Progressive texture synthesis on 3d surfaces. In VIIP 2006: *In Proceedings of Sixth IASTED international conference on Visualization, Imaging, and Image Processing*, pages 136–141, 2006. ISBN Hardcopy:0-88986-598-1/CD:0-88986-600-7.

[SEB07]Shet R. N., Edirisinghe E.A., and Bez H, A wavelets based max-flow/min-cut approach for texture synthesis. *In VIE2007: in Proceedings of the IET International conference on Visual Information Engineering: Convergence in Graphic and Vision*, 2007. ISBN CD 978-0-86341-830-3.

[SEB07]Shet R. N., Edirisinghe E.A., and Bez H, Progressive Texture Synthesis on Geometric Surfaces Parametrised by Bi-Quadratic Rational Bezier Patches, *in the Proc. of CVMP07: 4th European Conference on Visual Media Production, November 2007, ISBN CD 978-0-86341-843-3.*

[Sha93] SHAPIRO J. M.: Embedded image coding using zerotrees of wavelet coefficient. *IEEE Trans. Signal Processing* 41, 12 (12 1993), 3445–3462.

[SMGR96] SOUCY, MARC, GODIN G., RIOUX M.: A texture-mapping approach for the compression of colored 3d triangulations. *In The Visual Computer (1996)*, vol. 12, p. 503- 514.

[SS97] SZELISKI R., SHUM H.-Y.: Creating full view panoramic mosaics and environment maps. *In Proc. of SIGGRAPH 1997 (1997)*, pp. 251–258.

[TJL*02] TONG X., JINGDAN Z., LIGANG L., XI W., BAINING G., HEUNG-YEUNG S.: Synthesis of bidirectional texture functions on arbitrary surfaces. *In proc. of SIGGRAPH 2002 (2002)*, p. 665 ˜U 672.

[Tur01] TURK G.: Texture synthesis on surfaces. *In Proc. of SIGGRAPH 2001 (2001)*, p. 347- 354.

[VAI*03] VIVEK K., ARNO S., IRFAN E., GREG T., AARON B.: Graphcut textures: image and video synthesis using graph cuts. *In Proc. of SIGGRAPH 2003* (2003), vol. 22, pp. 277–286. submitted to EUROGRAPHICS 2007.

[WEB06] WICKRAMANAYAKE D. S., EDIRISINGHE E. A., BEZ H. E.: Multiresolution texture synthesis in wavelet transform domain. *The Journal of Imaging Science and Technology* 50, 1 (1 2006), 93–102. ISBN / ISSN: 1062-3701.

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. *In Proc. of SIGGRAPH 2000* (2000), p. 479- 488.

[WL01] WEI L.-Y., LEVOY M.: Texture synthesis over arbitrary manifold surface. *In Proc. of SIGGRAPH 2001* (2001), p. 355- 360.

[YHBZ01] YING L., HERTZMANN A., BIERMANN H., ZORIN D.: Texture and shape synthesis on surfaces. *In Eurographics Rendering Workshop* (2001), p. 301-312.

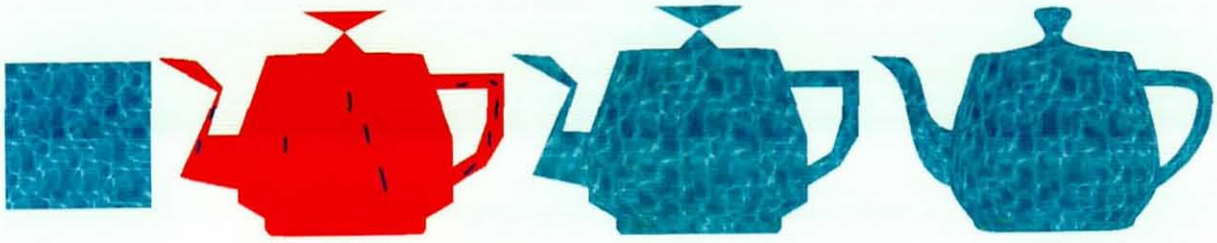


Figure 9: A complete cycle of the proposed texture synthesis process

Figure 10: Texture synthesis on the Teapot and Cup

Figure 11: Texture synthesis on a Torus, Cyclide and a Sphere

Figure 12: Illustrating progressive texture synthesis on the Teapot

Figure 13: Illustrating progressive texture synthesis on the Teapot