University Library

Author/Filing Title WICKRAMANAYAKE, D.

Class Mark T

Please note that fines are charged on ALL
overdue items.

FOR REFERENCE ONLY

0403177073

TRANSFORM DOMAIN TEXTURE SYNTHESIS

by

Dhammike Saranath Wickramanayake

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

Doctor of Philosophy

Department of Computer Science

Loughborough University

December 2005

© by Dhammike Saranath Wickramanayake 2005

Supervisors: Dr. Eran Edirisinghe, Dr. Helmut Bez

Director of Research: Dr. Chris Hinde

Loughborough University Pilkington Library
Date	JAN 2006
Class	T
Acc Nb.	0403177073

ABSTRACT

Transform Domain Texture Synthesis

D.S.Wickramanayake, October 2005.

Developments in digital cinema and computer games industry have required for fast texture synthesis algorithms that are capable of synthesizing better quality textures. Unfortunately most of the existing state-of-the-art algorithms suffer from excessive complexity related issues resulting in high synthesis times. This thesis proposes seven novel algorithms in order to address the existing problems in sample based texture synthesis. All the proposed algorithms use visually significant transform coefficients within texture synthesis and thus effectively exploit the limitations of the human visual system (HVS).

This thesis proposes four algorithms that use the Discrete Wavelet Transform (DWT) for extracting visually significant information within the texture synthesis process. This algorithm stitches together the spatio-frequency components of the input texture to form the corresponding components of the synthesised output texture. An automatically generated threshold determines the significant coefficients, which act as elements of a matching template used in the texture quilting process. The thesis 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. Further, the use of wavelet transforms adds texture compaction and multi-resolution texture synthesis capabilities to the proposed algorithms which can be considered as useful additional features.

Further a Discrete Cosine Transform (DCT) based texture synthesis algorithm that opens a new paradigm in texture synthesis is also proposed in this thesis. We show that it can be carried out in the transform/compressed domain without having to convert the sample texture to the pixel domain, synthesizing the texture and then having to

convert the synthesized texture back to the quantized transform domain for efficient storage/transmission. Further, we show that this enables the potential use of the technique in new application domains.

Further an extension of the proposed DWT based 2D texture synthesis algorithm to 3D texture mapping is proposed. Though within the research context of this thesis it is presented as a proof of concept, we show that its further development can provide solutions to the complexity related issues that are prominent in present 3D texture mapping algorithms.

The thesis also proposes two weighted edge blending strategies in DCT and DWT domains that are capable of removing artefacts across block seams, improving the overall quality of the synthesized textures, further. We show that the use of only visually significant information according to the HVS model in edge smoothing, saves computational cost compared to the popular edge cutting techniques. A popular set of test textures are used to compare the results of the proposed techniques with the existing state-of-the-art techniques. Some application scenarios of the proposed algorithms are also discussed.

ACKNOWLEDGEMENTS

I am indebted to my supervisors, Eran Edirisinghe and Helmut Bez for their exceptional support and guidance. They gave me the freedom to explore research ideas at will, but were always there to educate and guide me. I thank my director of research, Dr. Chris Hinde for his time, effort and suggestions.

My thanks are also due to the technical and clerical staff members of Department of Computer Science. I am grateful to my friends and colleagues for making the last three years at Loughborough University immensely enjoyable. My special thanks go to Murali, Mohamad, Nijad, Ken and Bardia for the useful discussions and for their friendship.

A very special thank you goes to Eran, Gayathri, Avishka and Danushka for their support since our arrival to United Kingdom by providing guidance and extended friendship in every way.

I have a deep regard for my wife, Arundathie who has always been supportive of me and provided me with immeasurable love and inspiration. Above all I must thank 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 – my mother's determination and father's vision have been of great significance in my life. I thank my late grandmother for her blessings that have been with me at every step. I would like to thank my brother, sisters and brothers-in-law, for their love, encouragement, help and advice that words cannot describe. I should express my sincere gratitude for my parents-in-law and the rest of the family for their extensive love, support and friendship.

Dhammike Saranath Wickramanayake

01st October 2005

TABLE OF CONTENTS

ABSTRACT	I
ACKNOWLEDGEMENTS.....	III
TABLE OF CONTENTS.....	IV
LIST OF FIGURES AND TABLES.....	IX
ABBREVIATIONS AND NOTATIONS.....	XII
CHAPTER 1	1
AN OVERVIEW	1
1.0 INTRODUCTION	1
1.1 WHAT IS TEXTURE AND TEXTURE SYNTHESIS?	2
1.1.2 Texture	2
1.1.2 Texture Synthesis.....	4
1.2 APPLICATIONS	5
1.3 MOTIVATION FOR THE RESEARCH.....	7
1.4 RESEARCH OBJECTIVES CONTRIBUTIONS OF THE THESIS.....	8
1.4.1 Objectives	8
1.4.2 Contributions of the Research.....	9
1.4.3 Scholarly Contributions	11
1.5 ORGANIZATION OF THE THESIS	13
CHAPTER 2.....	14
LITERATURE REVIEW	14
2.1 OVERVIEW	14
2.2 PARAMETRIC MODEL BASED TEXTURE SYNTHESIS APPROACHES	14
2.2.1 Reaction-Diffusion.....	15
2.2.2 Cellular Texturing.....	16
2.2.3 Weathering.....	17

2.3	NON-PARAMETRIC MODEL BASED TEXTURE SYNTHESIS APPROACHES	18
2.3.1	Pixel Based approaches.....	18
2.3.2	Patch Based approaches.....	25
2.4	SURFACE TEXTURE SYNTHESIS AND TEXTURE-MAPPING [79, 80].....	27
2.5	SUMMARY.....	30
CHAPTER 3.....		32
FUNDAMENTAL CONCEPTS.....		32
3.1	OVERVIEW	32
3.2	HUMAN VISUAL SYSTEM (HVS).....	33
3.3	MULTI-RESOLUTION ANALYSIS (MRA)	35
3.4	WAVELET TRANSFORM.....	36
3.4.1	The Continuous Wavelet Transform (CWT)	37
3.4.2	The Discrete Wavelet Transform (DWT)	37
3.4.4	Wavelet Families	40
3.4.5	HVS Model in DWT Domain.....	41
3.5	DISCRETE COSINE TRANSFORM.....	43
3.5.1	HVS Model in DCT Domain	46
3.5.2	DCT Coefficient Quantization.....	47
3.6	SUMMARY.....	48
CHAPTER 4.....		49
PIXEL DOMAIN IMAGE TEXTURE SYNTHESIS.....		49
4.1	OVERVIEW	49
4.2	BENCHMARK ALGORITHMS.....	50
4.2.1	Image Quilting for Texture Synthesis and Transfer [56].....	50
4.2.2	Graphcut Textures [57].....	53
4.3	PROPOSED IMPROVEMENT FOR THE BEST PATCH SELECTION CRITERION	56
4.3.1	Picking the minimum cost patch.....	56

4.4	A PROPOSAL FOR IMPROVED PATH CUTTING IN [56]	58
4.4.1	Improved Path Finding and Cutting.....	58
4.4.2	Experimental Results and Analysis	60
4.5	IMPROVEMENTS FOR CALCULATING THE SUMMATION OF OVERLAP	61
4.5.1	Independent Overlaps	61
4.5.2	Experimental Results	62
4.6	USE OF NEIGHBOURING BLOCKS IN MATCHING.....	63
4.6.1	Block Matching.....	63
4.6.2	Experimental Results	64
4.7	USE OF WHOLE SAMPLE IN THE SYNTHESIS PROCESS INSTEAD OF BLOCKS..	65
4.7.1	Sample Quilting Algorithm.....	65
4.7.2	Location Selection	66
4.7.3	Edge Handling	68
4.7.4	Experimental Results & Analysis	68
4.8	CONCLUSION.....	68
CHAPTER 5		69
WAVELET TRANSFORM DOMAIN PATCH BASED TEXTURE SYNTHESIS .		69
5.1	OVERVIEW	69
5.2	WHY DWT?	70
5.3	THE BASIC TECHNOLOGY	71
5.3.1	Application of DWT on an Image and Inter-subband Relationship	71
5.3.2	Texture Synthesis Process.....	73
5.4	ALGORITHM I & II (LOW FREQUENCY COMPONENT BASED)	74
5.4.1	Design Details.....	74
5.4.2	Algorithm I.....	78
5.4.3	Algorithm II	81
5.4.4	Experimental Results and Analysis	82
5.4.5	Comparison with Benchmark	91
5.5	ALGORITHM III	94

5.5.1	Design Details.....	95
5.5.2	Proposed Multiresolution Framework	96
5.5.3	Determining the Threshold	99
5.5.4	Mask Creation and Best Match Selection.....	99
5.6	MULTIRESOLUTION WEIGHTED EDGE BLENDING TECHNIQUE	100
5.6.1	Algorithm IV	100
5.7	EXPERIMENTAL RESULTS AND ANALYSIS	104
5.7.1	Experimental results.....	104
5.7.2	Detailed Analysis	107
5.8	COMPARISON WITH BENCHMARK ALGORITHMS AND COMPUTATIONAL COMPLEXITY ANALYSIS	109
5.9	APPLICATIONS	113
5.10	CONCLUSION.....	114
CHAPTER 6.....		115
2D TEXTURE SYNTHESIS IN DISCRETE COSINE TRANSFORM DOMAIN.		115
6.1	OVERVIEW	115
6.2	WHY DCT?.....	116
6.3	BASIC TECHNOLOGY.....	116
6.3.1	Application of DCT to an image.....	117
6.3.2	Design Details	117
6.4	TILE QUILTING ALGORITHM	121
6.5	WEIGHTED EDGE BLENDING ALGORITHM	124
6.6	EXPERIMENTAL RESULTS AND ANALYSIS	129
6.7	DETAILED EXPERIMENTS.....	130
6.7.1	Number of Candidate Tiles Available for Matching	130
6.7.2	Use of different coefficients in tile quilting:.....	131
6.8	COMPARISON WITH OTHER ALGORITHMS AND COMPUTATIONAL COMPLEXITY ANALYSIS	135

6.9	APPLICATIONS	139
6.10	CONCLUSION.....	139
CHAPTER 7.....		141
TEXTURE MAPPING ON 3D SURFACES		141
7.1	OVERVIEW	141
7.2	BACKGROUND INFORMATION	142
7.3	PRE-PROCESSING OF POLYGONAL INFORMATION	143
7.3.1	PLY File Format [163].....	143
7.3.2	Triangulation of Faces	145
7.3.3	Adjacency Information Extraction.....	145
7.4	TEXTURE-MAPPING WITH TRIANGULAR PATCHES.....	145
7.5	IMPLEMENTATION ISSUES	147
7.5.1	Mask creation and best match selection.....	147
7.5.2	Search Window Creation.....	150
7.5.3	Texture Storage and Rendering	151
7.6	DWT DOMAIN PATCH BASED MAPPING ALGORITHM	152
7.7	EXPERIMENTAL RESULTS AND ANALYSIS	153
7.8	APPLICATIONS	155
7.9	CONCLUSION.....	156
CHAPTER 8.....		158
CONCLUSIONS AND FUTURE RESEARCH		158
REFERENCES.....		165

LIST OF FIGURES AND TABLES

FIGURES

Figure 1.1: Different classes of textures:	3
Figure 1.2: Texture synthesis (a) sample texture (b) Synthesized texture.....	4
Figure 3.1 Contrast sensitivity function.....	34
Figure 3.2 Three level wavelet decomposition tree.....	38
Figure 3.3 Three-level wavelet reconstruction tree	39
Figure 3.4: Wavelet families.....	44
Figure 3.5: step by step application of 3-level DCT to the sample.....	42
Figure 3.6: HVS model in DWT domain using four-level image decomposition.....	43
Figure 3.7: DCT basis images of order, $N = 4$	45
Figure 3.8: HVS model in DCT domain for an 8×8 image.....	47
Figure 4.1: Pixel quilting algorithm.....	51
Figure 4.2: Image texture synthesis by placing small patches at various offsets	53
Figure 4.3: The process of synthesizing a larger texture from an example input texture.	54
Figure 4.4 performance evaluation of the proposed extension.....	57
Figure 4.5: Improved path finding.....	59
Figure 4.6: A comparison of path cutting using a sample image 128×128 , block size 60×60 and overlap $1/3$	59
Figure 4.7. A comparison of cutting methods using only the minimum SSD block. ...	60
Figure 4.8: Effect of independent overlaps.....	62
Figure 4.9. A diagram of neighbouring block areas used for comparison between the sample and output image.	63
Figure 4.10. Comparison of the Block Matching and original Image Quilting procedures, using a sample image 128×128 and a Block Size 20×20	64
Figure 4.11: Finding the best position for the second block.....	66
Figure 4.12, Experimental results	67
Figure 5.1: (a) application of 1 level discrete wavelet decomposition.	72

Figure 5.2: Reconstruction of “Brick” image using only 3rd level wavelet coefficient information.....	76
Figure 5.3: Reconstruction of “mat” image using only 3rd level wavelet coefficient information (a) Original texture image, reconstructed;	77
Figure 5.4: Construction of the output texture.....	79
Figure 5.5: Changing the block size adaptively.....	82
Figure 5.6 (I): Multiresolution image quilting results. Synthesized textures using proposed algorithm	83
Figure 5.7: Texture Synthesis results with adaptive block sizes	84
Figure 5.8: Performance comparison:.....	85
Figure 5.9: Four adjacent blocks of block ‘f’ for MSDS.....	86
Figure 5.10 Effect of addition of sub-bands, left: texture synthesized with only <i>LL3</i> sub band, Right: texture synthesized with all sub-bands of level 3.	88
Figure 5.11 comparison with varying block size.....	89
Figure 5.12: Comparison of performance of the purposed algorithm with Efros’s and Freeman’s algorithm..	92
Figure 5.13, Effect of global thresholding.	96
Figure 5.14: Construction of the output texture.....	98
Figure 5.15. (a) Already selected adjacent blocks showing 3-level wavelet decomposed edge zones.....	101
Figure 5.16 above: joining without blending, below: Joined after blending.....	103
Figure 5.17 comparisons with results of algorithm I; left: results of Algorithm I, Right: results of algorithm III	104
Figure 5.18(i) Synthesized (large) texture samples using the proposed algorithm. The corresponding small textures show the original texture samples.....	109
Figure 5.18(ii) Synthesized (large) texture samples using the proposed algorithm	110
Figure 5.19: Comparison of results; left: Efros and Freeman’s algorithm; Middle: Kwatra [57], Right: Proposed algorithm.....	107
Figure 5.20: Input size (n) Vs Synthesis Time	113
Figure 6.1: DCT basis images of order, $N=8$	117
Figure 6.2 (i): Effect of selecting only low frequency DCT coefficients	118

Figure 6.2 (ii): Effect of selecting only low frequency DCT coefficients,	119
Figure 6.3: subjecting an image block to JPEG/ Inverse JPEG process	120
Figure 6.4: (a) Placing tiles in the output representation.	123
Figure 6.5(i) Synthesized texture samples using the proposed algorithm.	126
Figure 6.5 (ii) Synthesized (large) texture samples using the proposed algorithm. ..	127
Figure 6.6: Comparison of texture samples, synthesized using different algorithms	128
Figure 6.7: Comparison of effect of blending.....	130
Figure 6.8 (I): Reconstructed Image	132
Figure 6.8 (II): Reconstructed image 2	133
Figure 6.9: Sample size Vs synthesis time.....	137
Figure 7.1: Mapping the face in 3D onto the 2D texture space	146
Figure 7.2: Assigning texture coordinates to the first triangular face [164].....	147
Figure 7.3: Creating a mask for a face having two already textured neighbours.	148
Figure 7.4: Face to be textured with the neighbourhood	149
Figure 7.5, neighbourhood mask of two already textured neighbours	150
Figure 7.6: Search for the best match.	151
Figure 7.7: wavelet transformed sample and coefficient triangles corresponding to a face.....	153
Figure 7.8: Texture mapped 3D models	154
Figure 7.9: Multi-resolution texture mapping:.....	155

TABLES

Table 5.1: MSDS comparison.....	93
Table 6.1. Texture synthesis times.....	138

ABBREVIATIONS AND NOTATIONS

1D	ONE DIMENSIONAL
2D	TWO DIMENSIONAL
3D	THREE DIMENSIONAL
3DTV	THREE DIMENSIONAL TELEVISION
AFX	ANIMATION FRAMEWORK eXTENSION
CS	CONTRAST SENSITIVITY
CSF	CONTRAST SENSITIVITY FUNCTION
DCT	DISCRETE COSINE TRANSFORM
DSP	DIGITAL SIGNAL PROCESSING
DWT	DISCRETE WAVELET TRANSFORM
EZW	EMBEDDED ZERO-TREE WAVELETS
FFT	FAST FOURIER TRANSFORM
FT	FOURIER TRANSFORM
GFX	GRAPHICS FRAMEWORK eXTENSION
HH	HIGH-HIGH
HL	HIGH-LOW
HVS	HUMAN VISUAL SYSTEM
IDCT	INVERSE DISCRETE COSINE TRANSFORM
IDWT	INVERSE DISCRETE WAVELET TRANSFORM
JND	JUST NOTICEABLE DIFFERENCE
JPEG	JOINT PICTURE EXPERTS GROUP
LH	LOW-HIGH
LL	LOW-LOW
MEBC	MINIMUM ERROR BOUNDARY CUT
MPEG	MOVING PICTURE EXPERTS GROUP
MRA	MULTI RESOLUTION ANALYSIS
MRF	MARKOV RANDOM FIELDS
MSDS	MEAN SQUARED DIFFERENCE OF SLOPES
PDF	PROBABILITY DENSITY FUNCTION
QMF	QUADRATE MIRROR FILTERS

SAD	SUM OF ABSOLUTE DIFFERENCES
SAT	SUMMED AREA TABLE
SSD	SUM OF SQUARED DIFFERENCES
SSE	SUM OF SQUARED ERROR
STFT	SHORT TIME FOURIER TRANSFORM
TDWEB	TRANSFORM DOMAIN WEIGHTED EDGE BLENDING
TSVQ	TREE STRUCTURED VECTOR QUANTISATION
VSI	VISUALLY SIGNIFICANT INFORMATION
VTC	VISUAL TEXTURE CODING
WEB	WEIGHTED EDGE BLENDING

Chapter 1

AN OVERVIEW

1.0 INTRODUCTION

Digital technology has already taken over much of the entertainment market with new developments in 3D television, digital films and modern video games. Television production is increasingly making use of 3D models, in applications including animation and virtual production. With the ever-increasing power of 3D graphics processors in home personal computers and new developments in 3D display technology researchers are working hard to maintain the content in its 3D form all the way through the program chain. Virtual production techniques have been developed to a level where they can be used to replace some or all of the scenery in conventional television production. The 3D virtual set, background and animated characters of a scene, is therefore the starting point along the road to 3D television. However the central elements in broadcast program content are actors and their interaction with the world around them. Hence in order to achieve 3D television production these elements must also be 'virtual' (the actors must be realistically modelled in 3D and placed into the virtual set). The entire animated 3D scene must be represented in an efficient manner and delivered to the home receiver. There are efforts to replace human actors by animated actors. This requires further improvements of visual appearance and dynamics of animated actors.

Modelling and rendering every minute detail in the computer generated models in order to make them more real looking is a tedious task and requires enormous amounts of computational and manpower. Mapping natural textures into computer-generated models creates high quality and realism into graphics applications. This is one of the important motivations for texture synthesis that comes from texture mapping. Apart from this there are many other applications for texture synthesis (see section 1.2). It is important to understand the context of texture and texture synthesis before further applications are discussed.

1.1 WHAT IS TEXTURE AND TEXTURE SYNTHESIS?

1.1.2 Texture

The word “Texture” is given with various meanings in the computer graphics literature. For example, definitions given in four of the literatures are 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. The 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 quest “ What features and their 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” texture is defined as an equivalence class of images in 2D that share identical features.*

In texture synthesis what is usually referred to as “texture” are the images containing repeating patterns, with the possibility of random variations, a meaning closer to the definition given by A.K. Jain [3]. This definition of texture is more restricted than the notion of texture in graphics. However, since a majority of natural surfaces consist of repeating elements; this narrower definition of texture is still powerful enough to describe many surface properties. This definition of texture is also widely adopted in the computer vision and image processing communities.

Figure 1.1: Different classes of textures: (a), (b) & (c) regular, (d), (e) & (f) near-regular, (g), (h) & (i) irregular, (j), (k) & (l) stochastic

Textures describe a wide variety of surfaces such as terrain, plants, minerals, walls, fur or skin, etc. and some natural phenomena like weathering and corrosion. It exists almost everywhere and in everything we see. Textures have been traditionally classified as either regular or stochastic. Almost all the real world textures lie in between these two extremes. Natural examples of such textures include fur of animals, patterns of flowers, bark on a tree etc., whereas fabric patterns, stone patterns on walls are examples of man-made textures. Some texture synthesis algorithms work better for certain types of textures in between these two ends. Therefore in this thesis textures in between these extremes are further classified into near-regular and irregular [4] figure 1.1 shows examples of each class.

1.1.2 Texture Synthesis

Whether it is a computer game, a digital movie or a 3D television production, texture has become an inevitable feature in any computer graphics application. Textures for these applications can be obtained from a variety of sources such as hand-drawn pictures, photographs or artificially generated textures. Hand-drawn pictures can be aesthetically pleasing, but it is hard to make them photo-realistic. Certain computer graphics and image processing applications requires creation of large areas of texture. Synthetic textures can be made of any size without visual repetition. But most scanned images, however, are of inadequate size and can lead to visible seams or repetition if they are directly tiled and would create distortion if they are directly wrapped around 3D models. This is where the necessity of a texture synthesis algorithm, an alternative way to create textures, evolves.

Figure 1.2: Texture synthesis (a) sample texture (b) Synthesized texture

A texture synthesis method starts from a sample image and attempts to produce a texture with a visual appearance similar to that sample by repeated placement of micro patterns of texture elements on a surface. When perceived by a human observer

the resulting texture appears to be generated by the same underlying stochastic process. Texture synthesis can also produce tileable images by properly handling the boundary conditions. Unfortunately, creating a real-time, robust and general algorithm has been considered difficult.

Texture synthesis is a very active and relatively new research area in graphics. Its roots might be traced back to the 80's, but the "seminal papers" which will be discussed in chapter 2 are less than a decade old. The applications and the areas of interest of texture synthesis are various and significant. In the last years, much work has been done in order to optimize the synthesis process, speeding up the methods and minimizing the processing errors. Recent efforts in this area are concentrated on producing fast flexible algorithms and on introducing tools, which augment the texture with artistic effects.

Textures can be synthesized by different techniques such as fractals, random fields, reaction-diffusion, morphology, Gabor filters, Eigen-patterns, steerable pyramids, wavelets, tiling and co-occurrence methods. Each method can generate only a particular subset of texture patterns and have their own advantages and disadvantages. These methods will be discussed briefly in chapter 2.

1.2 APPLICATIONS

Texture synthesis can be useful in many applications in computer graphics, image processing, and computer vision. The following are some examples of applications of texture synthesis.

Rendering

In rendering, textures can mimic the surface details of real objects, ranging from varying the surface's colour, perturbing the surface normals (bump mapping), to actually deforming the surface geometry (displacement mapping). In pen and ink style illustrations, textures (hatches) can delineate the tone, shade, and pattern of objects [5-6].

Animation

Computer generated animations often contain scripted events and random motions. Scripted events are non-repetitive actions such as opening a door or picking up an object, and are usually rendered under direct control. On the contrary, random motions are repetitive background movements such as ocean waves, rising smoke, or a burning fire. These kinds of motions have indeterminate extent both in space and time, and are often referred to as *temporal textures* [7-10]. These temporal textures are often difficult to render using traditional techniques based on physical modelling, since different textures are often generated by very different underlying physical processes. By treating them as textures, we can model and synthesize them using a single texture synthesis algorithm. In addition to temporal textures, certain motions such as joint angles of articulated motions could also be modelled as one-dimensional textures. These textures can be synthesized on the fly to simulate delicate motions such as eye blinking or human walking.

Compression

Images depicting natural scenes often contain large textures regions, such as grassland, a forest, or a sand beach. Because textures often contain significant high frequency information, they are not well compressed by transform-based techniques such as JPEG. By segmenting out these textured regions in a pre-processing step, they may be compressible by a texture synthesis technique. In addition to image compression, texture synthesis can also be employed for synthetic scenes containing large amounts of textures [11].

Restoration and Editing

Photographs, films and images often contain regions that are in some sense flawed. A flaw can be a scrambled region on a scanned photograph, scratches on an old film, wires or props in a movie film frame, or simply an undesirable object in an image. Since the processes causing these flaws are often irreversible, an algorithm that can fix these flaws, a process known as error concealment amongst the image and video

compression community, is desirable. Often, the flawed portion is contained within a region of texture, and can be replaced by texture synthesis [12-15].

Computer Vision

Several computer vision tasks such as segmentation [16, 17], recognition [18, 19], and classification [20, 21] use textures. These tasks can benefit from a texture model, which could be derived from a successful texture synthesis algorithm.

1.3 MOTIVATION FOR THE RESEARCH

During the past few years many texture synthesis algorithms have evolved. Researchers have mainly concentrated on improving the quality of synthesis and making algorithms more general, i.e. applicable to most of the texture classes. As a result there has been a significant improvement in quality and generality in present texture synthesis algorithms, although the synthesis time is considerable. With the rapid growth of the internet and digital movie industry, requirement of real-time texture synthesis is becoming inevitable. Even with the ever increasing power of computers, researchers have not yet been able to achieve this goal. This has been one main motivation for this thesis.

A texture image contains very large amounts of perceptual data. A texture synthesis algorithm starts with a texture with good resolution and does a lot of computations with this huge set of data in the synthesis process. Due to these reasons existing texture synthesis algorithms producing *near photorealistic texture* demand high computational power and often take a lot of time to synthesize small areas of texture. This implies the fact that if the speed needs to be increased, a texture sample with the least amount of spatial data e.g., a low resolution sample of original texture, should be used. However this accounts for the degradation of the quality of synthesis. This is because they are based on a texture analysis in a lower resolution level in the pixel domain. The best way to speed up these algorithms is to identify the perceptually significant information and use only that information in the synthesis process. Good

human perception based texture discrimination, and analysis techniques [22-27] can be helpful in this regard.

Approaches motivated by the Human visual system (HSV) are commonly used within image and video coding for achieving data compression while still maintaining the perceived quality at an acceptable level. Research in image compression technologies [28-33], has proven that it is possible to produce a texture of good perceptual quality with only about 10-20 percent of the total image data. Unfortunately, identifying this significant data in the pixel domain is difficult.

However, images consist of a wide range of frequency components spread throughout the human visual frequency band. Further, it is a known fact that the human eye is more sensitive to low frequencies. Therefore one may represent an image with only a few low frequency components. Such a representation of an image can be obtained with Discrete Wavelet Transforms (DWT) or Discrete Cosine Transforms (DCT). As the resulting representations are organized in order of spatio-frequency detail, identifying the visually significant information is easy in the transform domain.

Motivated by these facts, several perceptually based texture synthesis approaches using only visually significant information in the transform domain, are proposed in this thesis.

1.4 RESEARCH OBJECTIVES AND CONTRIBUTIONS OF THE THESIS

1.4.1 Objectives

The general research objective is to “Design and implement novel algorithms for high quality, high speed surface texture synthesis and mapping”. Specific research objectives are as follows,

- To investigate possible improvements in pixel domain block based texture synthesis algorithms.

- To design novel approaches to transform domain block based texture synthesis.
- To suggest how the principles of transform domain image texture synthesis can be applied to texture mapping.
- To investigate a possibility of a low cost solution for edge (seam) artefact reduction in block based texture synthesis algorithms.
- To suggest possible extensions of proposed texture synthesis techniques to progressive 3D object transmission systems.
- To identify future directions of research

1.4.2 Contributions of the Research

The following original contributions have been made in this research. The resulting conference and journal papers are included in the section 1.4.3, and are referred as A1, A2 ...A9.

1 Developing Pixel domain based speed and quality improvements in state-of-the-art algorithm, “Image quilting for texture synthesis and transfer”

Performance of the algorithm proposed by Efros and Freeman can be further improved in the pixel domain by addressing some drawbacks suffering due to its design. Contributions proposed resulted in improvements in quality and speed.

2 Discrete Wavelet Transform (DWT) based algorithm for transform domain texture synthesis.

While wavelet based techniques were attempted in parametric texture synthesis they had never been attempted in block-based texture synthesis. The use of visually significant DWT coefficients within texture synthesis, in the proposed algorithm effectively exploits the limitations of the human visual system (HVS), a task which is not possible when using traditional texture synthesis algorithms that perform texture synthesis in the pixel domain. Adaptation of the multiresolution techniques in the proposed algorithm facilitates its use in progressive transmission systems. In addition

performing texture synthesis in the DWT domain allows compressed domain texture synthesis, which is an advantage in bandwidth/resource constrained application areas.

3 Wavelet based weighted edge blending algorithm for block artefact reduction

Though alpha blending and dynamic programming based minimum error boundary cut techniques are implemented in most of the pixel domain texture synthesis algorithms to avoid edge artefacts, they are either computationally costly and/or do not fully exploit limitations of the HVS. Performing weighted edge blending in the DWT domain has specific advantages generally and also due to the nature of the design and application scenarios of our algorithm. The proposed algorithm weights the frequency components of all sub-bands and blends together to form a smooth visual transition across the edges.

4 Discrete Cosine Transform (DCT) based texture synthesis algorithm

The proposed algorithm performs texture synthesis in the DCT domain. By doing this, extra computational speedup is achieved as only the visually significant coefficients are considered in the tile quilting process. DCT domain texture synthesis opens a new paradigm in texture synthesis. It can be carried out in the transform/compressed domain without having to convert sample texture to the pixel domain resulting efficient storage/transmission. This enables the potential use of the technique in new application domains. Three visually significant information selection techniques proposed for this algorithm fully exploit the limitations of the HVS.

5 DCT based edge smoothening technique

In this novel DCT based weighted frequency blending technique, unlike in the DWT based edge blending technique, only a limited number of visually significant coefficients are blended increasing the speed of blending while exploiting the limitations of HVS. In the above mentioned DCT based texture synthesis algorithm (4), one would not want to perform weighted edge blending in the pixel domain as

only then an unnecessary inverse transform would have to be performed to complete the synthesis process. Rather with the proposed technique, texture synthesis can be completed in the DCT domain without incurring additional cost if weighted edge blending is done in the same domain.

6 Transform domain texture mapping algorithm

The proposed texture mapping approach extends the DWT based algorithm into 3D. This tackles a difficult problem of triangular patch matching where one, two or three trapezoidal overlap areas are combined together for creating the matching mask. Acceleration of the proposed mapping algorithm is achieved by selecting only a few visually significant coefficients for the mask.

1.4.3 Scholarly Contributions

Refereed Journal Publications

1. **D.S.Wickramanayake, H.E.Bez, E.A.Edirisinghe,(2005)** "Multiresolution texture synthesis in wavelet transform domain", Accepted for publication in The Journal of Imaging Science and Technology.

Refereed Conference Proceedings

2. **D.S .Wickramanayake, E.A.Edirisinghe, H.E.Bez,** "A wavelet based image quilting approach to Fast multiresolution texture synthesis", Proceedings of IEE international conference on visual information engineering (VIE2005), pages 93-100, April 2005.
3. **D.S. Wickramanayake, Eran A. Edirisinghe, Helmut 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, June 7-9, 2005, Part I , Springer Lecture Notes in Computer Science (LNCS), Vol. 3522 (2005), Page-384, ISSN: 0302-9743, ISBN: 3-540-26153-2

4. **D.S. Wickramanayake**, E.A.Edirisinghe, H.E.Bez, "Transform Domain Texture Synthesis based on DCT" In PREP 2005 proceedings (Orals), pages 152-153, March 2005,
5. **D.S.Wickramanayake**, E.A.Edirisinghe, H.E.Bez, "Fast, Wavelet transform domain texture synthesis" Proceedings of SPIE -- Volume 5308, Visual Communications and Image Processing 2004, Sethuraman Panchanathan, Bhaskaran Vasudev, Editors, January 2004, pp. 979-987,
6. J.T.O'Brien, **D.S.Wickramanayake**, E.A.Edirisinghe, H.E.Bez, "Image Quilting for Texture Synthesis and Transfer: A Revisit & A Variation" Proceedings of Fourth International Conference on Information, Communications & Signal Processing (ICICS-PCM2003) (Fourth IEEE Pacific-Rim Conference On Multimedia), Dec. 2003.
7. **D.S.Wickramanayake**, E.A.Edirisinghe, H.E.Bez "Multiresolution Image quilting for texture synthesis", Proceedings of the third IASTED (International Association of Science and Technology for Development), international Conference on Visualization, imaging and image processing (VIIP2003), Sept 2003, Vol 3, pp . ISBN: 0-88986-382-2, ISSN:14827921

Submitted Publications for Refereed Journals

8. **D.S.Wickramanayake**, E.A.Edirisinghe, H.E.Bez, "Discrete Cosine Transform Domain Texture Synthesis", IEEE Transactions on Visualization and Computer Graphics. (on second review)
9. **D.S.Wickramanayake**, E.A.Edirisinghe, H.E.Bez,"Multiresolution Texture Synthesis for Embedded Image/Video Coding Applications", IEEE Transactions on Image Processing.

Publications to be submitted for Refereed Conferences

10. **D.S.Wickramanayake**, E.A.Edirisinghe, H.E.Bez, "Discrete Wavelet Transform (DWT) domain Multiresolution texture mapping" which is to be

submitted to the international conference on visual information engineering (VIE2006).

1.5 ORGANIZATION OF THE THESIS

Remainder of the thesis is organized into seven chapters.

Chapter 2: This chapter starts with providing an overview of the research work carried out in the field of texture synthesis and mapping. It further includes a comprehensive survey of image texture synthesis and surface texture mapping.

Chapter 3: In order to facilitate the reader in understanding the novel texture synthesis and mapping techniques introduced in this thesis, the related fundamental concepts are described in this chapter. This includes an introduction to the human visual system and basic concepts of frequency transforms used in this research work.

Chapter 4, 5, 6 and 7: Include information on original contributions made by the author to the field of image texture synthesis and texture mapping. Each of these chapters concludes with a discussion on the contributions made by the proposed algorithm and possible improvements.

Chapter 8: concludes with an insight into the future directions of research in image and surface texture synthesis as well as in progressive 3D model transmission.

Chapter 2

LITERATURE REVIEW

2.1 OVERVIEW

Numerous approaches have been proposed for texture synthesis in the past years. With the process of standardization of 3D content in MPEG-4, Animation Framework eXtension (AFX) [34] and Graphical Framework eXtension (GFX) [35], more research focus is expected on texture synthesis and mapping. In this chapter a brief overview of important previous work is described. Due to the large volume of existing literature, providing a complete survey on texture synthesis and mapping algorithms is out of the scope of this thesis. Readers are referred to [13, 36-39] for more complete surveys.

Texture synthesis algorithms can be classified in different ways. In this thesis they are categorized into parametric model based approaches (section 2.2) and non-parametric model based approaches (section 2.3). The parametric model based approaches use a number of parameters to simulate a variety of texture generation processes [40-55]. The non-parametric model based approaches, or 'example based methods' generate textures by considering properties such as image histogram, pixel colour, PDF etc. from input textures [12, 56, 57]. Important areas covered in the rest of the chapter include: surface texture synthesis and texture mapping techniques (section 2.4) and finally a brief overview on the synthesis algorithms discussed in this chapter (section 2.5).

2.2 PARAMETRIC MODEL BASED TEXTURE SYNTHESIS APPROACHES

Parametric model based algorithms are as old as texture synthesis itself. In fact, when the very first 3D rendering software came out, it wasn't possible to use photographs to generate the objects' texture; therefore, mathematical functions were used to simulate the texture patterns such as in wood, stone or of other material. With time, the

creation of such '*shaders*' has become extremely complex. The advantages of this approach, whether used on 3D objects or for video compositing, come from their purely mathematical nature. Since they are only defined by formulae, most of the time they can be scaled without any loss of quality. Their storage requires little memory, and therefore transmission requires little bandwidth. But only a limited set of textures which can be modelled accurately with mathematical equations can be synthesized using this approach.

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

2.2.1 Reaction-Diffusion

This is a biologically motivated method of texture synthesis. Reaction-diffusion is a process in which two or more chemicals diffuse at unequal rates over a surface and react with one another to form stable patterns such as spots and stripes. Biologists and mathematicians have explored the patterns made by several reaction-diffusion systems. This mechanism was first described by Alan Turing [40]. A number of researchers have shown how simple patterns of spots and stripes can be created by reaction-diffusion systems [41-44].

J. Bard [41], J. Murray [42]

In 1981 both Jonathan Bard and James Murray published independent papers suggesting that reaction-diffusion mechanisms could explain the patterns on coats of mammals. Bard's work concentrated on showing that a variety of spot and stripe patterns could result from such systems [41]. He showed that a reaction-diffusion system can produce the small white spots on a deer or the large dark spots on a giraffe. He demonstrated that a wave of activation that sweeps over a surface of cells might account for creating striped patterns. These stripes can be either parallel or perpendicular to the wave of activation depending on the assumptions of the model. Murray's work showed that a single reaction-diffusion system might produce rather different patterns of light and dark on a mammalian coat depending on the size of the

animal [42]. For example, the pattern created on a small animal may be dark regions at the head and tail of the animal separated by a white region in the center. The same mechanism may produce several irregularly shaped dark regions on a larger animal. On an even larger animal, the mechanism might produce a large number of small dark spots. This same paper also demonstrated that a reaction-diffusion mechanism may account for some patterns found on butterfly wings.

A. Witkin, M. Kassy [44]

In this paper, Witkin and Kassy investigate a class of patterns that arise from local, nonlinear interactions of excitation and inhibition. Their work is based on a chemical mechanism that was first proposed by A. Turing [41] to account for pattern formation in biological morphogenesis. They extend traditional reaction-diffusion systems by allowing an isotropic and spatially non-uniform diffusion, as well as multiple competing directions of diffusion. They adapt reaction-diffusion systems to the needs of computer graphics by presenting a method to synthesize patterns, which compensates for the effects of non-uniform surface parameterization. Finally, they develop algorithms for simulating reaction diffusion systems and display a collection of resulting textures using standard texture and displacement-mapping techniques.

These algorithms can create only specific type of biological texture patterns such as butterfly wings, stripe patterns of zebra, spots on cows etc., despite of the advantages such as low memory consumption, low band-width requirement in transmission etc.

2.2.2 Cellular Texturing

Cellular patterns are all around us, in masonry, tiling, shingles, and many other materials. Such patterns are influenced by geometric features of the underlying shape. Numbers of researchers have used schemes, which divide texture space into cells and interpret each cell position to create branching patterns called cellular textures [55].

K. Perlin [46]

Perlin's work is inspired by 'Cook's Shade Trees, [58]. This algorithm is based on a fractal noise basis function which does not need any storage and pre-computation and returns values that are defined over 3D space. It provides a high level language for describing shading functions at each pixel. The scalar value can be directly mapped onto a colour, but it is often used for spatially perturbing regular patterns.

S. Worley [47]

This paper is similar to Ken Perlin's paper [46], and it proposes a new set of related texture basis functions that are based on scattering "feature points" throughout the 3D space and builds a scalar function based on the distribution of the local points. They have used this new basis function based on noise to produce textured surfaces resembling flagstone-like tiled areas, organic crusty skin, crumpled paper, ice, rock, mountain ranges, and craters. The new basis functions are computed efficiently without the need for pre-calculation or table storage.

2.2.3 Weathering

All materials have an inherent tendency to change in appearance or composition when exposed to the physical and chemical conditions of the surrounding environment. The deterioration, decay, and change in appearance of materials due to the effects of the surrounding environment are generally termed weathering. Specific examples of weathering include the corrosion of metals, efflorescence on stone and brick, fungal attack on organic materials, and the wear and tear of everyday life. With the advancement of realistic image synthesis several different approaches to modelling and/or rendering surface weathering effects have been introduced [50-53].

J. Dorsey, A. Edelman, [59]

This paper presents an approach for the modelling and rendering of changes in the shape and appearance of stone. Their work is closely related to Dorsey and Hanrahan's work [54]. To represent stone, they introduce a slab data structure, which

is a surface-aligned volume confined to a narrow region around the boundary of the stone. Their weathering model employs a simulation of the flow of moisture and the transport, dissolution, and re-crystallization of minerals within the porous stone volume. In addition, this model governs the erosion of material from the surface. To render the optical effects of translucency and colouration due to the composition of minerals near the surface, they simulate the scattering of light inside the stone using a general subsurface Monte Carlo ray tracer. These techniques can capture many aspects of the time-dependent appearance of stone. They demonstrate the approach with models of granite and marble statues, as well as a sandstone column.

2.3 NON-PARAMETRIC MODEL BASED TEXTURE SYNTHESIS APPROACHES

A straightforward way to produce realistic textures is to use real world images as texture resources. This is also called example based texture synthesis as the pixels or the blocks of the sample are used as examples in choosing the matching candidates in the synthesis process. These algorithms can be divided mainly into two categories, namely; pixel based approaches and patch based approaches depending on the unit of construction used in the synthesis. These approaches can be used to create a complete texture map covering the entire scene being textured using multiple images as texture sources. The following are some non-parametric model based approaches, presented in the order they have evolved.

2.3.1 Pixel Based approaches

As the name suggests, pixel based approaches gradually synthesize the surface pixel by pixel consuming much time. The properties such as probability density function, image histogram or colour of neighbouring pixels from the sample texture are considered as examples in synthesis. Higher time consumption and the inability to capture the global structures of these approaches are the main drawbacks. Due to these reasons pixel based methods are not applicable to all types of textures, especially those with large micro patterns.

K. Popat, R.W. Picard. [60]

This is a non-parametric Markov chain synthesis algorithm. This algorithm initially orders the pixels and then synthesizes a new pixel from a non-parametric representation of the conditional probability function derived from samples of the input texture. Popat proposed to use stochastic sampling of the conditional probability function and also to compress the conditional probability function via a set of Gaussian kernels. Although this compression limited the neighbourhood order that could be successfully modelled, it allowed for fast look ups.

This synthesis was performed in a sequential order starting from a "seed" and gradually moved further away. The problem with this type of approach is that if the past pixels start to deviate too far from those seen in the input image, the synthesis algorithm can diverge in space thus producing garbage. This severely limits the algorithm for synthesising large areas of texture.

D. J. Heeger, J. R. Bergen. [61]

Heegar and Bergen focuses on the synthesis of stochastic textures. The method described here synthesizes textures by matching distributions (or histograms) of filter outputs. This is one of the first attempts to show the possibility of synthesizing textures in colour.

This approach depends on the principle (though not entirely correct) that all of 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 algorithm involves a sequence of simple image processing operations: convolution, subsampling, up sampling, histogramming and nonlinear transformations using small lookup tables. These operations are fast, simple to implement, and amenable to special purpose hardware implementations (e.g., using DSP chips). The technique starts with a digitized image and analyses it to compute a number of texture parameter values. Those parameter values are then used to synthesize a new image that looks (in its colour and texture properties) like the original. The analysis phase is inherently two-

dimensional since the input digitized images are 2D. The synthesis phase, however, may be either two- or three-dimensional. For the 3D case, the output is a solid texture such that planar slices through the solid look like the original scanned image. In either case (2D or 3D), the texture is synthesized so that it tiles seamlessly. Nevertheless, this model (though incomplete) captures an interesting set of texture properties. Computational efficiency is one of the advantages of this approach compared with many of the previous texture analysis/ synthesis systems.

R. Paget, I.D. Longstaff [62]

The problem with the sequential approach of Popat's algorithm is solved here. 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 overcame a lot of problems inherent in a sequential approach which was prone to having the synthesis algorithm wander into a non recoverable "no man's land".

This algorithm uses a different way of classification of textures. It catches 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 texture visually indistinguishable from its training texture. Based on this MRF model, they propose an algorithm with 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 the multi-scale approach with local annealing improves the implementation of the MRF model, the speed is still very low. As the nonparametric MRF model implemented in this algorithm captures sufficient high order statistical characteristics of texture it can even be used for synthesizing natural textures.

J. S. De Bonet [63]

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 sampled. De Bonet's method could be considered a variant of Heeger and

Bergen's [61] pyramid based texture analysis/synthesis method. It overcomes the iterative requirement of Heeger and Bergen's method by enforcing a top down philosophy akin to Paget and Longstaff's [62] approach. In a two-phase process, the input texture is first analyzed by measuring the joint occurrence of texture discrimination features at multiple resolutions. In the second phase, a new texture is synthesized by sampling successive spatial frequency bands from the input texture, conditioned on the similar joint occurrence of features at lower spatial frequencies. The sampling methodology is based on the hypothesis that texture images differ from typical images in that there are regions within the image, which, to some set of feature detectors, are less discriminable at certain resolutions than at others. By rearranging textural components at locations and resolutions where the discriminability is below threshold, new texture samples are generated which have similar visual characteristics. De Bonet's method is very sensitive to the choice of these threshold parameters, which if chosen incorrectly detrimentally affected the output texture.

S. Zhu, Y. Wu, D. Mumford[64]

This paper presents a statistical theory for texture modelling. Their algorithm combines filtering theory and Markov random field modelling through the maximum entropy principle, and interprets and clarifies many previous concepts and methods for texture analysis and synthesis from a unified point of view. This theory characterizes the ensemble of images, with the same texture appearance by a probability distribution on a random field. Their model consists of two steps. (1) A set of filters is selected from a general filter bank to capture features of the texture. These filters are applied to observed texture images, and the histograms of the filtered images are extracted. These histograms are estimates of the marginal distributions of probability distribution functions. This step is called feature extraction. (2) The maximum entropy principle is employed to derive a distribution, which is restricted to having the same marginal distributions as those in (1). This is considered as an estimate. This step is called feature fusion. A stepwise algorithm is proposed to choose filters from a general filter bank. The resulting model is a Markov random field (MRF) model named as FRAME (Filters, Random fields And Maximum Entropy). All this avoided the messy process of trying to reconstruct a texture from

arbitrary filter responses and wrapped it all in some nice clean mathematics; however the synthesis/modelling process is very slow.

E. Simoncelli, J. Portilla [65]

This is a similar technique to that of Heeger and Bergen [61]. They present a parametric statistical characterization of texture images in the context of an over complete complex wavelet frame. The characterization consists of the local autocorrelation of the coefficients in each sub-band, the local autocorrelation of the coefficient magnitudes, and the cross correlation of coefficient magnitudes at all orientations and adjacent spatial scales. They develop an algorithm for sampling from an implicit probability density conforming to these statistics, and demonstrate its effectiveness in synthesizing artificial and natural texture images. The resulting images show that some textural aspects are missing and there are quality problems.

A. Efros, T. Leung [12]

Their work is similar to Popat's [60] work. However they assume a Markov random field model and synthesize a texture by repeatedly matching the neighbourhood around the target pixel in the synthesis result with the neighbourhood around all pixels in the input texture, starting from a seed pixel and growing outwards. For each to-be synthesized pixel and its neighbourhood of already synthesized pixels, an approximation to the conditional probability is constructed for each input pixel. This is achieved by computing a Gaussian weighted normalized sum of square differences (SSD) between already synthesized pixels and the pixel neighbourhoods of each candidate in the input texture. The degree of randomness is controlled by a single parameter. They also initialized the synthetic texture with an arbitrarily placed patch of texture. However this just changed the order in which the pixels were synthesised, and did not change the fact that it was a sequential nonparametric Markov chain type of approach as proposed by Popat [60], which had inherent stability problems.

L.Y. Wei, M. Levoy [166]

This also is similar to Papat's [60] approach, but with some significant changes to enhance both quality and speed. However they did not change the synthesis order because of speed. Keeping the synthesis order as proposed by Papat, meant that the Markov neighbourhood was fairly consistent over the whole synthesis process. This allowed for data compression. Papat had used Gaussian kernels to define a PDF, Wei and Levoy pre-processed the texture using tree-structured vector quantization (TSVQ) to quickly search for the nearest neighbour. This pre-processing results in logarithmic complexity for each best-pixel-search and an overall speedup by two orders of magnitude compared to Efros and Leung's [12] algorithm, at the price of some artefacts. The algorithm is furthermore extended to a multiresolution synthesis pyramid, progressing from coarse to fine. This results in smaller search neighbourhoods, with synthesis quality comparable to using larger neighbourhoods with single resolution synthesis.

S.C. Zhu, X.W. Liu, Y.N. Wu [67]

This is a cut down version of their previous work, FRAME [64]: Filters, random fields and maximum entropy towards a unified theory for texture modelling. Here Zhu and Liu propose a slightly faster algorithm for synthesis, one that does not need to estimate the model parameters. Instead of creating a PDF for a Gibbs ensemble, they directly use the statistics from the filters to do the sampling via the Markov chain Monte Carlo algorithm.

M. Ashikhmin [68]

Ashikhmin provides a solution to the time consuming procedure of exhaustive nearest neighbour searching, without loss of quality as in Wei and Levoy's approach. In fact Ashikhmin's method actually gives both an increase in synthesis quality and speed. In his seminal paper, he proposes a new measure of nearest neighbour instead of either the Manhattan or Euclidean distance, as he suggests that these may not be the best measures to test for perceptual similarity. He notes that if we are only taking pixel

colours from the input image (and not sampling from a larger distribution), then when we synthesise a colour for a pixel, we can be assured that each of its defined neighbours corresponds to a pixel within the input image. He proposes, instead of doing an exhaustive search, that if only pixels with a corresponding neighbour are considered a substantial speedup can be gained. Unfortunately he applied his new technique to Wei and Levoy's algorithm, which meant that his results were not as good as they should have been.

Hertzmann et al. [69]

Even though Ashikhmin's technique works very well for natural textures sharp discontinuities can occur with textures that contain a high degree of structure. Merging both Ashikhmin and Wei/Levoy synthesis into a framework, Hertzmann introduced the problem statement titled "Image Analogies, a framework based on a simple multiscale auto regression". It involves two stages: a *design phase*, in which a pair of images, with one image purported to be a "filtered" version of the other, is presented as "training data"; and an *application phase*, in which the learned filter is applied to some new target image in order to create an "analogous" filtered result. Hertzmann et al. recognised that the Euclidean distance combined with exhaustive nearest neighbour searching gives a smoother transition between these discontinuities.

Zelinka and Garland [70]

Zelinka and Garland attempt to avoid nearest neighbour comparisons during synthesis by creating a k nearest neighbour lookup table as part of an input texture analysis stage. They then use this table to make random jumps (like as in Schödl et al.'s video textures [8]) during their sequential texture synthesis stage. No neighbourhood comparisons are done during synthesis, which makes the algorithm fast.

Tong et al. [71]

Tong et al. also used a k nearest neighbour lookup table, but instead of defining random jump paths like Zelinka and Garland, they simply use the list of k nearest

neighbours as a sample base from which to choose the neighbourhood that gives the minimal Euclidean distance measure. When k equals one, this method is comparable to Ashikhmin's method. Tong et al. noted that k should be set depending on the type of texture being synthesised. For natural textures where the high frequency component is desired, a low k should be used, and for other textures where better blending is required, then a relatively high k should be used.

2.3.2 Patch Based approaches

This is a fairly new approach to the texture synthesis; first research paper came out in 2002. Since then several algorithms have been introduced to the research community. The main advantages over pixel based approaches are the increased speed, as they synthesizes a larger area after one computation step and their ability to capture the global structure of the texture. Seminal papers introduced with this idea are discussed in this section.

Y.Q. Xu, B.N. Guo, H.Y. Shum [72]

Guo and Shum introduced patch-based texture synthesis to the graphics community through this paper. Instead of copying one pixel at a time from an input image, they copy whole patches. In this case, these algorithms randomly distribute patches from an input texture onto an output lattice, smoothing the resulting edges between overlapping patches with simple cross-edge filtering. This algorithm worked well only for a very limited set of stochastic textures.

L. Ling et al. [73] and Efros and Freeman [56]

These two algorithms were developed concurrently. They have taken patch based texture synthesis a few steps further. Both these algorithms use patch based sampling. Ling addresses the problem of constrained texture synthesis and by using "feathering" [74] at edge boundaries while Efros uses minimum error boundary cut at edges. These algorithms produce reasonably good quality results with less

computation cost compared to the other algorithms. They are explained in detail as our bench mark algorithms, in section 4.2.1.

A. Nealen, M. Alexa, [75]

This is a recent algorithm which adds a further modification to Ling et al.'s patch-based texture synthesis based on the idea that the use of large patches improves the reproduction of global structure. This work was also inspired by Soler, Cani's hierarchical pattern mapping [92]. Nealen and Alexa's hybrid algorithm uses adaptive patches to fill a lattice. These patches are chosen with respect to a minimum border error. Mismatches in the overlapping border above a certain threshold are re-synthesised.

V. Kwatra et al., [57]

This is another algorithm based on Efros and Freeman's image quilting concept. Kwatra et al. use one major pass for initially filling the output and several other refine passes in their algorithm. It uses a graph cut algorithm named "min-cut or max-flow" [76]. In addition they use FFT-based acceleration of patch searching via sum-of-squared differences. Their algorithm is also adapted for video texture synthesis. This algorithm will also be explained further in this thesis (see section 4.2.2) as it is selected as a bench mark to evaluate the performance of the proposed 2D texture synthesis algorithms.

M. Cohen, J. Shade, S.Hiller, O. Deussan [77]

Cohen et al. use principles of provably non-periodic tiling of the plane to generate arbitrary amounts of non-repetitive texture [78]. They use patches called 'Wang Tiles' that are squares whose edges are each assigned a color 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 [56]). 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.

2.4 SURFACE TEXTURE SYNTHESIS AND TEXTURE-MAPPING [79, 80]

A problem inherent to 2D texture mapping is the lack of the “third” dimension information. In fact, it is not possible to simulate correctly a rough or bumpy surface, such as roughcast, using only colour variations because roughness is related to geometry. In order to realistically simulate wrinkled surfaces, Blinn [81] introduced a normal perturbation technique, called bump mapping. This technique, which consists of modifying the surface normal vectors allows for the geometric appearance to be retrieved. However, in some cases, bump mapping remains insufficient because the surface is not “really” deformed.

Other methods [81, 82] were proposed to resolve the limitations of bump mapping. Displacement mapping [82] is a method that consists of “really” displacing surface points along their normal. Hence, all problems of bump mapping can be correctly resolved but with higher computation times. Max [82] proposed a less expensive method but limited to self-shadows. However, even these extensions of bump mapping cannot resolve the problem of highly complex structured surfaces like cotton or fur. Therefore, other kinds of approaches that we called geometric texturing, closer to geometric modelling, appeared later in the form of hypertextures [83] and texel maps [84]. Texel mapping and hypertexturing turn out to be particularly useful for “soft” (translucent) features. However, these approaches are extremely time consuming.

Another form of texture-mapping close to geometric modelling consists of directly distributing small geometric elements on surfaces, as for cellular textures [85] and macrostructured textures [86]. All of these 3D texturing approaches (displacement

maps, hypertextures, texels, cellular textures, etc.), can be considered as “geometric” techniques, since they add some “real” geometric components to the surfaces.

The following section discusses several state-of-art algorithms in surface texture synthesis and mapping.

G. Turk [45]

This is an extension of the work done by Andrew and Michel [44] (see section 2.2.1). The paper proposes an algorithm for demonstrating how textures can be generated in a manner that directly matches the geometry of a given surface. He extends the range of textures that have previously been generated by using a cascade of multiple reaction-diffusion systems in which one system lays down an initial pattern and then one or more later systems refine the pattern. Examples of patterns generated by such a cascade process include the clusters of spots on leopards known as rosettes and the web-like patterns found on giraffes. In addition, the paper introduces a method by which reaction-diffusion textures are created to match the geometry of an arbitrary polyhedral surface. This is accomplished by creating a mesh over a given surface and then simulating the reaction diffusion process directly on this mesh. This avoids the often-difficult task of assigning texture coordinates to a complex surface. A mesh is generated by evenly distributing points over the model using relaxation and then determining which points are adjacent by constructing their voronoi regions [168]. Textures are rendered directly from the mesh by using a weighted sum of mesh values to compute surface colour at a given position. Such textures can also be used as bump maps.

K. W. Fleischer et al. [48]

Fleischer et al. combine properties of particle systems, developmental models, and cellular-texturing methods into one system for modelling surface details of 3D objects. The textures they model are formed from many interacting geometric elements. Actual fur, scales, and thorns may be formed from single cells or multiple cells [49]. In either case, they have assumed that the texture patterns arise from the interactions of discrete elements capable of movement and orientation change, and

modelled each of these elements as one cell. The patterns are formed as the cells experience physical processes of collision, adhesion, and other local interactions. They have written a cellular particle simulator, which uses differential equations to place the cells. The constraints are specified such that energy functions are to be minimized. Then they have made use of a parameterized particle-to-geometry converter to convert cell parameters to tiny textured polygons. Finally texture is synthesized by rendering lots of polygons.

G.Turk [87] and Wei and Levoy [88]

These surface synthesis methods both densely tessellate the input mesh using Turk's re-tiller [89] and then perform a per-vertex colour synthesis. These two approaches are very similar (both use a multiresolution mesh hierarchy), yet have three quite significant differences. (1) Wei and Levoy use both random and symmetric vector fields, whereas Turk always uses a user defined smooth vector field. (2) Turk uses a sweeping order derived from the smooth vector field for vertex traversal, Wei and Levoy visit the mesh vertices in random order. (3) Turk uses surface marching to construct the mesh neighbourhood, while Wei and Levoy perform flattening and re-sampling of the mesh. Results of the two methods are comparable in quality.

L. Ying et al. [90]

This algorithm is proposed in order to overcome the following drawbacks of surface marching methods. (1) Sampling pattern is not guaranteed to be even in the presence of irregular geometry, (2) sampling is numerically unstable, as small surface variations can cause large variations in the pattern and (3) the method is slow due to many geometric intersection and projection operations. In their approach, texture is synthesized on surfaces per-textel using a texture atlas of the polygonal mesh (a collection of rectangular domains on which the surface is smoothly parameterized) and a common planar domain from which neighbourhood sample positions in the rectangular domains are gathered. These positions in the rectangular domain then correspond to the neighbourhood on the original surface along a previously defined,

orthogonal unit tangent vector field. They apply both Wei/Levoy and Ashikhmin per-pixel synthesis strategies with convincing results.

E. Praun et al. [91]

This algorithm extends the chaos mosaic [72] to surfaces with a pre-computed vector field to direct anisotropy. In their system, the user specifies a tangential vector field 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 and parameterizing it in texture space. The parameterization is optimized (by solving a sparse linear system) such that the vector field aligns with the frame of the texture patch. They render the resulting model both with a generated texture atlas and by runtime-pasting, the latter significantly reducing texture memory at the cost of rendering some faces multiple times.

C. Soler et al. [92]

This algorithm shows how a mesh can be seamlessly textured with only the input texture and a set of texture coordinates for each vertex. They set up a hierarchy of face clusters for the input mesh and then each cluster is flattened (if distortion is too high, the cluster is subdivided for later processing). Subsequently they find a texture patch in the example texture for the flattened cluster which best matches already textured neighbours (if the error due to texture discontinuities with the existing neighbours is too high, the cluster is subdivided for later processing), if (1+2) pass, map texture coordinates onto all polygons in the cluster. Unlike previous methods, their method does not use a vector field; instead they let possible texture anisotropy propagate itself. Similar to Lapped Textures runtime-pasting [91], the texture memory overhead for rendering is minimal, and additionally, all faces are rendered only once.

2.5 SUMMARY

A few of the important algorithms out of the numerous texture synthesis approaches that were introduced to the computer graphics community during the past years were

discussed in the above sections. Each of the categories mentioned above has own advantages and disadvantages.

Parametric model based approaches which marked the birth of texture synthesis are still being used in computer graphics to simulate certain types of texture patterns due to the advantages of being able to scale without any loss of quality, low memory space requirement and ease of transmission. But problems such as requirement of enormous computational power when representing complex textures and applicability of this method only to a limited set of texture types limit its usage.

Non-parametric based texturing methods using real world images as texture resources on the other hand are applicable for a wide variety of texture types as they use the similarity of the sample. Amongst the two main types of non-parametric synthesis methods, namely pixel and patch based approaches, the main drawback in pixel based approaches are relative slowness. Most of the acceleration methods, attempting to speed up the process affect the quality. Whereas speed and quality patch based approaches are better due to larger unit of construction which enhance the ability to capture the global structure.

Among the patch based approaches introduced so far, Efros and Freeman [56] and Kwatra et al. [57] are most successful due to their applicability for large range of texture types and the quality. But both are having speed constraints. Kwatra et al. [57] uses a FFT based acceleration as their algorithm consumes more computational power compared to [56]. These two algorithms were selected as bench marks for the thesis due to the following three reasons; (a) they synthesize textures with better quality, (b) the speed of the synthesis is comparably high, (c) they work well with most of the texture types. These two algorithms are analysed in detail in chapter 4.

Among the surface texture synthesis and mapping techniques introduced, the patch based approaches have relatively better speeds than others. Though Praun et al.'s [91] and Neyret's [93] algorithms are capable of synthesizing texture at higher speeds, they are applicable only for a limited set of textures. Solar's [92] method is applicable for a variety of textures and the speed is also acceptable. Therefore the proposed texture mapping approach (Chapter 7) adopts a method similar to [92].

Chapter 3

FUNDAMENTAL CONCEPTS

3.1 OVERVIEW

Algorithms proposed in this thesis aim to effectively exploit the limitations of the Human Visual System (HVS). Thus the Visually Significant Information (VSI) of the sample texture images is specifically considered within the proposed texture analysis and synthesis algorithm. As explained in chapter 1 extraction of VSI, is less complicated in the frequency domain than in the pixel domain. Amongst the techniques available for transforming images into the frequency domain, Discrete Wavelet Transform (DWT) and Discrete Cosine Transform (DCT) are the most popular within the image processing community. Among these two transforms DWT provides a spatio-frequency representation of the image data whereas DCT gives a frequency representation of the original pixel data. Even though wavelet based methods have been previously used for texture analysis, classification, segmentation and in parametric texture synthesis [94-102], as per the author's knowledge, this is the first attempt to use it in patch based synthesis. In addition, there have not been any previous attempts in DCT based texture synthesis.

For the above reasons, in this chapter the reader is introduced to DWT and DCT concepts and their importance in this respect. In order to facilitate understanding the extraction process of VSI, some important aspects of HVS are discussed initially in section 3.2. An insight into multiresolution analysis is given in section 3.3 while the wavelet transform is introduced in section 3.4. Section 3.5 introduces the basic concepts of DCT and HVS as applied to DCT, which is the basis of the algorithms introduced in chapter 6. The chapter concludes in section 3.6 with a brief summary.

3.2 HUMAN VISUAL SYSTEM (HVS)

The HVS is a very complex system and it has the ability to deal with a huge amount of information at the same time. Roughly speaking, it is composed of a receiver with a pre-processing stage; the eye and the retina, a transmission channel; the optic nerve, and a processing engine; the visual cortex. Mainly because of our lack of advanced knowledge about brain behaviour, the enormous effort put into understanding and modelling the HVS behaviour has partly remained fruitless. The aim of this section is to provide an overview of the HVS. For complete discussion on HVS models and more specific details, the reader is referred to existing literature [103-106]. Here, we only attempt to disclose, in a synthetic way and from an engineering perspective, the HVS features, e.g.: sensitivity and contrast handling.

A lot of work has gone into understanding the HVS and applying this knowledge to image processing/coding applications. In image compression algorithms, there has been a need for a good matrix for image quality that incorporates properties of the HVS. This matrix can be used for deriving a good quantization matrix to compress images with better visual quality and higher compression ratio, in image compression. The design of a quantization matrix that provides an optimal quality for a given bit rate depends upon the visibility thresholds of visual signals. This is because the error caused by quantization will not be visible if it is less than the corresponding visibility thresholds of a visual signal. Extensive psycho-visual measurements have been performed in order to determine these thresholds [104-107]. These measurements were performed on sinusoidal gratings with various spatial frequencies and orientations under given viewing conditions. The goal was to determine the contrast thresholds of gratings for the given frequency and orientation. Contrast, as a measure of relative variation of luminance, for periodic patterns such as a sinusoidal grating is given by the equation,

$$C = \frac{L_{\max} - L_{\min}}{L_{\max} + L_{\min}} \quad (3.1)$$

Where $L_{\max}$ and $L_{\min}$ are maximal and minimal luminance of a grating. Reciprocal values of contrast thresholds express the Contrast Sensitivity (CS). Contrast sensitivity is sometimes called visual acuity [107, 108] as well. Mannos and Sakrison [109] proposed a model of the human Contrast Sensitivity Function (CSF). The contrast sensitivity function tells us how sensitive we are to the various frequencies of visual stimuli. If the frequency of visual stimuli is too high we will not be able to recognize the stimuli pattern any more. Imagine an image consisting of vertical black and white stripes. If the stripes are very thin (i.e. a few thousands per millimetre) we will be unable to see individual stripes. All that we will see is a grey image. If the stripes then become wider and wider, there is a threshold width, from which we are able to distinguish the stripes. The contrast sensitivity function proposed by Manos and Sakrison [109] is defined as follows

$$CSF(f) = 2.6 \cdot (0.0192 + 0.144f) \cdot e^{[-(0.114f)^{1.1}]} \quad (3.2)$$

Where f is the spatial frequency in cycles/degree of visual angle. The CSF curve in Fig. 3.1 indicates that HVS is most sensitive to spatial frequencies between 5 and 10 cycles per degree and less sensitive to very low and very high frequencies. This fact can be used to develop a simple image independent HVS model.

Figure 3.1 Contrast sensitivity function

The reason why we can not distinguish patterns with high frequencies is the limited number of photoreceptors in our eye. There are several other functions proposed by other authors, but we choose the above function [109] because it has a simple analytical description.

Common HVS models are composed of image dependent or independent Just Noticeable Difference (JND) thresholds, i.e. steps below which the signal is considered as insignificant or imperceptible. Most of the models incorporate three basic properties of HVS: frequency sensitivity, luminance sensitivity and contrast masking.

Frequency and Color [169]

The above experiments focused on contrast using achromatic sinusoidal patterns. As we are focusing on digital images it is interesting to see how this property of the eye is modified when we use color stimuli. Precisely, as images are represented in color spaces (such as RGB, YCbCr, Lab, etc...) we are interested in knowing what are the thresholds for detecting contrast in each component of the color spaces.

Experiments supporting the opponent color channels theory have shown that for high spatial frequency the eye is only sensitive to variation in the luminance (this is the achromatic channel). In the two other channels, namely the Red-Green channel and the Blue-Yellow channel the spatial frequency is much lower.

Section 3.4.5 and 3.5.1 describes how this HVS model can be applied in extracting the visually significant coefficients in DWT and DCT domains.

3.3 MULTI-RESOLUTION ANALYSIS (MRA)

According to the Heisenberg uncertainty principle obtaining an exact time frequency analysis of a signal is almost 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 times. 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 analyze any signal by using a popular approach called multiresolution analysis (MRA). MRA, as implied by its name, analyzes 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 relationship of images in addition. MRA makes sense especially when the signal at hand has high frequency components for short durations and low frequency components for long durations. Fortunately, images are often of this type. The DWT provides a multiresolution representation of an image. DWT gives a signal representation in which some of the coefficients represent long data lags corresponding 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 are available.

3.4 WAVELET TRANSFORM

In the recent years, wavelet based methods have gained importance in many different fields of computer graphics including wavelet radiosity [110], multi-resolution painting [111], curve design [112], mesh optimization [113], volume visualization [114], image searching [115], variational modeling [116], and object compression [117]. Overviews of wavelet based algorithms for computer graphics can be found in [118, 119]. This section contains only a brief introduction to the theory of wavelets, more details of the mathematical definitions can be found in [120, 121].

The Wavelet Transform [122-126] provides a time-frequency representation of a signal. It was developed to overcome the short comings of the Short Time Fourier Transform (STFT), which can also be used to analyze non-stationary signals [126]. While STFT gives a constant resolution at all frequencies, the Wavelet Transform uses a multi-resolution technique by which different frequencies are analyzed with different resolutions.

The wavelet analysis is done similar to the STFT analysis. The signal to be analyzed is multiplied with a wavelet function just as it is multiplied with a window function in STFT, and then the transform is computed for each segment generated. However, unlike STFT, in Wavelet Transforms, the width of the wavelet function (window size) changes with each spectral component.

3.4.1 The Continuous Wavelet Transform (CWT)

The Continuous Wavelet Transform (CWT) is provided by equation 3.3, where $x(t)$ is the signal to be analyzed and $\psi(t)$ is the mother wavelet or the basis function. All the wavelet functions used in the transformation are derived from the mother wavelet through translation (shifting) and scaling (dilation or compression).

$$X_{WT}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t) \bullet \psi^* \left(\frac{t - \tau}{s} \right) dt \quad (3.3)$$

The mother wavelet used to generate all the basis functions is designed based on some desired characteristics associated with that function. The translation parameter τ relates to the location of the wavelet function as it is shifted through the signal. Thus, it corresponds to the time information in the Wavelet Transform. The scale parameter s is defined as $|1/\text{frequency}|$ and corresponds to frequency information. Scaling either dilates (expands) or compresses a signal. Large scales (low frequencies) dilate the signal and provide detailed information hidden in the signal, while small scales (high frequencies) compress the signal and provide global information about the signal.

3.4.2 The Discrete Wavelet Transform (DWT)

The foundations of DWT go back to 1976 when techniques to decompose discrete time signals were devised [122]. In CWT, the signals are analyzed using a set of basis functions which relate to each other by simple scaling and translation. In the case of DWT, a time-scale representation of the digital signal is obtained using digital filtering techniques. The signal to be analyzed is passed through filters with different cut-off frequencies at different scales.

3.4.2.1 Multi-Resolution Analysis using Filter Banks

Filters are one of the most widely used signal processing functions. Wavelets can be realized by iteration of filters with rescaling. The resolution of the signal, which is a measure of the amount of detail information in the signal, is determined by the filtering operations, and the scale is determined by up-sampling and down-sampling (sub-sampling) operations [122].

Figure 3.2 Three level wavelet decomposition tree

The DWT is computed by successive low-pass and high-pass filtering of the discrete time-domain signal as shown in figure 3.2. This is called the Mallat algorithm or Mallat-tree decomposition [127]. Its significance is in the manner it connects the continuous-time multiresolution to discrete-time filters. In figure 3.2, the signal is denoted by the sequence $X[n]$, where n is an integer. The low pass filter is denoted by G_0 while the high pass filter is denoted by H_0 . At each level, the high pass filter produces detailed information; $d[n]$, while the low pass filter associated with scaling 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 reduced by half. In accordance with Nyquist's rule if the original signal has a highest frequency of ω , which requires a sampling frequency of 2ω radians, then it now has a highest frequency of $\omega/2$ radians. It can now be sampled at a frequency of ω radians thus discarding half the samples with no loss of information. This decimation by 2 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 decimation by 2 doubles the scale.

With this approach, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. The filtering and decimation 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.

Figure 3.3 Three-level wavelet reconstruction tree

Figure 3.3 shows the reconstruction of the original signal from the wavelet coefficients. Basically, the reconstruction is the reverse process of decomposition. The approximation and detail coefficients at every level are up-sampled by two, passed through the low pass and high pass synthesis filters and then added. This process is continued through the same number of levels as in the decomposition

process to obtain the original signal. The Mallat algorithm works equally well if the analysis filters, G_0 and H_0 , are exchanged with the synthesis filters, G_1 and H_1 .

3.4.4 Wavelet Families

There are a number of basis functions that can be used as the mother wavelet for a Wavelet Transformation. Since the mother wavelet produces all wavelet functions 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 3.4 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. These are also called Maxflat wavelets as their frequency responses have maximum flatness at frequencies 0 and π . This is a very desirable property in some applications.

Figure 3.4: Wavelet families (a) Haar (b) Daubechies4 (c) Coiflet1 (d) Symlet2 (e) Mayer (f) Morlet (g) Mexican Hat

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. Wavelets are chosen based on their shape and their ability to analyze the signal in a particular application.

3.4.5 HVS Model in DWT Domain

As described above the DWT decomposes a signal into a low-pass and a high-pass signal by using a pair filters. This process can be further repeated. Two-dimensional signals such as images are decomposed by DWT into sub-bands that vary in spatial frequency and orientation. Here we can distinguish between approximation, horizontal, vertical, and diagonal details. Figure 3.5 shows the separation of details of an image due to application of DWT.

In image compression algorithms based on DWT there has been a need for a good method that would provide better quality of compressed images with higher compression ratio. A good quantization matrix is important in this respect. Such a quantization matrix should provide a high compression ratio and at the same time a very good perceptual quality of a compressed image. For constructing that matrix, the knowledge of detection thresholds for DWT coefficients in various sub-bands and various levels of decomposition is needed. These visibility or detection thresholds $T_{L,O}$ in various sub-bands and on various decomposition levels for 9/7 bi-orthogonal wavelets were determined via psychological experiments [128] and they can be expressed by the following equation

$$\log T_{L,O} = \log T_{\min} + K(\log f_L - \log f_0 g_0)^2 \quad (3.4)$$

Where L is the decomposition level, O denotes orientation (1, 2, 3, and 4 for approximation, horizontal, diagonal, and vertical details), $T_{\min}$ is the minimum threshold occurs at spatial frequency $f_0 g_0$, f_L is the spatial frequency of level L, and g_0 shifts the minimum thresholds by an amount that is a function of orientation. An

example of visibility thresholds for DWT coefficients in a 256×256 image viewed from a viewing distance of 6 image heights and by using 4 level DWT is shown in Fig. 3.6 [129]. These visibility thresholds determine the basic image independent HVS model in the DWT domain and express the frequency sensitivity of the human eye to various frequency bands.

Careful observation of figure 3.6 reveals that the approximation band contains the most visually important information, leading to the lowest threshold. Next the vertical and horizontal details of the highest level of decomposition and so on.

Figure 3.5: Step by step application of 3-level DWT to the sample. (a) sample, (b) DWT transformed sample (1-level), (c) result of applying DWT to the approximation of (b) (2-level decomposition), (d) repeating the procedure for (c) (3-level decomposition)

Figure 3.6: HVS model in DWT domain using four-level image decomposition

3.5 DISCRETE COSINE TRANSFORM

The discrete cosine transform (DCT) is another important transform in 2D signal processing. It is known to be close to optimal in terms of its energy compaction capabilities and can be computed via a fast algorithm. The DCT is used in international image/video compression standards of Joint Photographic Experts Group (e.g. Baseline JPEG), and Motion Picture Experts Group (e.g. MPEG-1, MPEG-2, MPEG-4 VTC) [130, 131].

The one dimensional (1D) DCT, $X[k]$ of a sequence $x[n]$ of length N is defined by equation 3.5,

$$X[k] = \alpha[k] \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad (3.5)$$

for $k = 0, 1, \dots, N-1$

The inverse DCT (IDCT) is defined by equation 3.6,

$$x[n] = \sum_{k=0}^{N-1} \alpha[k] X[k] \cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad (3.6)$$

for $n = 0, 1, \dots, N-1$

Where in both Equations (3.5) and (3.6), $\alpha[k]$ is defined as

$$\alpha[k] = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } k = 0 \\ \sqrt{\frac{2}{N}} & \text{for } k = 1, 2, 3, \dots, N-1 \end{cases} \quad (3.7)$$

The basis sequences of the 1D DCT are real, discrete-time sinusoids defined by

$$c_N[n, 0] = \cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad (3.8)$$

The DCT basis consists of the following N real sequences.

$$\{c_N[n, 0], c_N[n, 1], \dots, c_N[n, N-1]\}$$

As in the case of the familiar Fourier transform, any square integrable real sequence may be represented by a superposition of basis sequences $c_N[n, k]$.

The series for the 2D discrete cosine transform (2D DCT) pair of formulas is

$$X[k_1, k_2] = \alpha[k_1]\alpha[k_2] \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] \cos\left(\frac{\pi(2n_1+1)k_1}{2N_1}\right) \cos\left(\frac{\pi(2n_2+1)k_2}{2N_2}\right) \quad (3.9)$$

$$\text{for } k_1 = 0, 1, \dots, N_1-1 \text{ and } k_2 = 0, 1, \dots, N_2-1$$

$$x[n_1, n_2] = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \alpha[k_1]\alpha[k_2] X[k_1, k_2] \cos\left(\frac{\pi(2n_1+1)k_1}{2N_1}\right) \cos\left(\frac{\pi(2n_2+1)k_2}{2N_2}\right) \quad (3.10)$$

$$\text{for } n_1 = 0, 1, \dots, N_1-1 \text{ and } n_2 = 0, 1, \dots, N_2-1$$

With $\alpha[k]$ defined as in Equation (3.7), Equation (3.9) is called the analysis formula or the forward transform, while Equation (3.10) is the synthesis formula or inverse transform. The DCT is separable and so may be computed using 1D DCT row and column operations. By analogy with the 1D case, any finite length sequence $x[n, m]$ may be represented by a superposition of N^2 distinct, separable, bi-variate

basis images. Assuming the same dimension N for each of the two spatial variables m and n , we obtain the following family of sequences:

$$C = \begin{pmatrix} 1 & c_N[n_1, 1] & \cdot & c_N[n_1, N-1] \\ c_N[n_2, 1] & c_N[n_1, 1] & c_N[n_2, 1] & \cdot & c_N[n_1, N-1] & c_N[n_2, 1] \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ c_N[n_2, N-1] & c_N[n_1, 1] & c_N[n_2, N-1] & \cdot & c_N[n_1, N-1] & c_N[n_2, N-1] \end{pmatrix} \quad (3.11)$$

A plot of all the 2D DCT basis images of order $N = 4$ is shown in figure 3.7.

Figure 3.7: DCT basis images of order $N = 4$

The 2D DCT may be computed using a sequence of row and column 1D transformations (the transform is separable). Thus the analysis and synthesis equations may be compactly written as

$$x_c = C.X.C^T$$

$$X = C^T.x_c.C$$

Where X is an image and x_c is the 2D DCT transform of X .

Application of a DCT for an 8×8 pixel block would result in 64 coefficients, organized in a zig-zag pattern, starting from the zero frequency DC coefficient at the top left corner and ending with the highest frequency component at the bottom right. This representation enables us to suppress less visually important information (low energy, high frequency coefficients). This is the characteristic, which make the DCT transform ideal for texture synthesis algorithms. In the baseline-JPEG image compression standards each DCT block is divided by a quantization matrix (see fig.

6.3 (c)) to filter out visually insignificant frequency components. Only remaining large frequency components (see fig 6.3 (d)) are used for image representation.

3.5.1 HVS Model in the DCT Domain

In image compression algorithms based on DCT there has been a need for a good quantization matrix that would provide better quality of compressed images with higher compression ratio. The design of a quantization matrix that provides an optimal quality for a given bit rate depends on the visibility of the DCT basis functions, since the error caused by quantization of a particular DCT coefficient will not be visible if its quantization error is less than the corresponding Just Noticeable Difference (JND) threshold. The HVS model in the DCT domain can be obtained by approximation of experimentally measured detection thresholds for individual DCT basis functions. These JND thresholds $T(u, v)$ can be expressed in the following form

$$T(u, v) = \frac{T_{\min} f_{u,v}^4}{C(u).C(v).(f_{u,v}^4 - 4(1-r)f_{u,0}^2 f_{0,v}^2)} 10^{K(\log \sqrt{f_{u,0}^2 + f_{0,v}^2} - \log f_{\min})^2} \quad (3.12)$$

Where $f_{0,v}$, $f_{u,0}$ are horizontal and vertical spatial frequency, respectively, the minimum threshold $T_{\min}$ occurs at spatial frequency $f_{\min}$, K determines the steepness of the parabola, and r is the model's parameter [132]. An example of this HVS model for an 8×8 block of DCT coefficients of a 256×256 image viewed from a viewing distance of 8 times the image heights is depicted in Fig. 3.8.

Figure 3.8: HVS model in DCT domain for an 8×8 image

These thresholds determine the frequency sensitivity of the human visual system, which describes the human eye's sensitivity to sine wave gratings at various frequencies. The thresholds are image independent and represent the basic HVS model that depends only on viewing conditions. They are measured by a constant viewing distance and background (or mean) luminance. If the background luminance is changed then the thresholds change, too. A more complete perceptual model can be achieved by considering luminance sensitivity in finding the JND for each coefficient. By close observation of Fig. 3.8 it is clear that HVS is more sensitive to low frequency components. This principle is used in the creation of quantization matrices in DCT based image compression techniques.

3.5.2 DCT Coefficient Quantization

The principle used behind JPEG compression lies with the fact that the human eye is more sensitive to low frequency information than to high frequency information in images [130]. After extensive experiments carried out according to HVS models in the JPEG standardization process, they have come up with a scalar quantization

matrix $Q(i, j)$ for organizing the DCT transformed coefficient magnitudes according to their visual significance. This matrix can be made use of in DCT based texture synthesis algorithm.

Understanding the JPEG quantization process is helpful in explaining its usability in texture synthesis. The quantization and de-quantization formulae are as follows.

$$QDCT(i, j) = \text{round}[DCT(i, j) / Q(i, j)] \quad (3.13)$$

$$IQDCT(i, j) = Q(i, j) \times DCT(i, j) \quad (3.14)$$

Where $QDCT(i, j)$ is the quantized transform matrix and $IQDCT(i, j)$ is the inverse quantized transform matrix.

The effect of the DCT and the scalar quantization stages of a typical image block are best explained by the example [133] in section 6.3.2.

3.6 SUMMARY

This chapter introduced the user to the basic concepts which will be useful in understanding the algorithms proposed in this thesis. It explained the basics of MRA, DCT, DWT and HVS models as applied to these two frequency domains. In addition, the principles behind the creation of quantization matrices in DCT are also discussed.

The next chapter introduces the reader to the benchmark algorithms and some experiments performed in pixel domain.

Chapter 4

PIXEL DOMAIN IMAGE TEXTURE SYNTHESIS

4.1 OVERVIEW

The research work was started in December 2002 with the implementation of the first benchmark algorithm, 'Image Quilting for Texture Synthesis and Transfer' [56]. This algorithm was implemented in the MATLAB platform [134-136] and its performance was tested for a large set of textures [137-140] widely used within the texture synthesis community. After the above platform was established, the initial research work was carried out based on the implementation. The basic aim of the research introduced in this chapter is to improve the quality and/or speed of the pixel domain texture synthesis.

This chapter starts with introducing the reader to the benchmark algorithms (see section 4.2), Efros & Freeman's "Image quilting for texture synthesis and transfer" [56] and Kwatra et al.'s "Graphcut Textures" [57], which are used to assess the performance of the novel techniques proposed in this thesis. Experiments carried out in the pixel domain to evaluate the detailed performance of these algorithms are also discussed. In addition several extensions to the benchmark algorithms are proposed in later sections. The benchmark algorithm, all proposed algorithms and amendments are implemented using MATLAB 7 and all comparisons are done in a Pentium IV, 1.6 MHz PC under equal conditions for a fair comparison.

Section 4.3 proposes an improvement to the selection criterion of the most appropriate patch out of the already grouped minimum error patches in [56]. As the horizontal movement of the edge cutting technique used in [56] is restricted, in certain cases, the cutting path may not be visually optimal. Section 4.4 addresses this problem. Section 4.5 proposes an improvement to the overlap error calculation technique adopted in [56]. Section 4.6 proposes a novel method for block matching based on neighbouring blocks which is shown to be more efficient than the technique adopted in [56]. Section 4.7 proposes a further extension to the benchmark algorithm of [56] which uses the whole sample as the unit of construction within the synthesis process. It

further adopts a less computationally expensive method for edge artefact reduction. Section 4.8 concludes this chapter based on the experimental evidence of previous sections.

4.2 BENCHMARK ALGORITHMS

4.2.1 Image Quilting for Texture Synthesis and Transfer [56]

This algorithm is one of the two algorithms used as a benchmark to assess the performance of the transform domain texture synthesis algorithms proposed in this thesis. This block based algorithm was introduced by Efros & Freeman [56] in 2001 and has been widely quoted in subsequent research work. It has been used as a popular benchmark to test most of the texture synthesis techniques introduced by various authors in the recent past. This algorithm generally works well for most of the texture types. The speed of this algorithm is high compared to pixel based algorithms proposed previously.

4.2.1.1 *Image quilting stage*

In [56] the output texture is formed by selectively transferring randomly selected blocks of a predefined size from the input texture image. This is done in two steps satisfying certain pre-defined criteria. Firstly, given that a corner block of the output image has been appropriately formed, a subset of blocks from which a good candidate for the block to its right (assuming a raster scanned order) could be found as follows: All possible blocks of the same block size from the input image are matched to the first block (top left hand corner) of the output image, under a certain overlap. The algorithm can be summarized as follows:

- Go through the image to be synthesized in raster scan order in steps of one block (minus the overlap). (See fig. 4.1 (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 fig. 4.1 (b) & (c))
- Compute the error surface between the newly chosen block and the old blocks at the overlap region. Find the minimum cost path along this surface and make that the boundary of the new block. Paste the block onto the texture. (See fig. 4.1 (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:

$$SSE = \sum_{p \in O} [I_1(p) - I_2(p)]^2 \quad (4.1)$$

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

Figure 4.1: 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 typical block size used in this algorithm is 64×64 while overlap is $1/6^{\text{th}}$ the width of the block. The block having the best matching overlap, and all other blocks whose matching error is within 10% of that of the best is selected as the sub-set from which subsequently a block is picked up randomly. This block will be used to patch the output image at the location to the right of the previously patched block (in this instance, the top left hand corner 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 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, rather than the more obvious straight edge cut.

4.2.1.2 Minimum Error Boundary Cut

This is done with dynamic programming (Dijkstra's algorithm is used [141]). The minimal cost path through the error surface is computed in the following manner. If B_1 and B_2 are two blocks that overlap along their vertical edge with the regions of overlap B_1^{ov} and B_2^{ov} (see fig. 4.1(c)), respectively, the error surface is defined as, $e = (B_1^{ov} - B_2^{ov})^2$. In order to find the minimal vertical cut through this surface e ($i = 2 \dots N$) is traversed and the cumulative minimum error E_{ij} for all paths i,j is calculated as:

$$E_{ij} = e_{ij} + \min (E_{i-1, j-1}, E_{i-1, j}, E_{i-1, j+1}) \quad (4.2)$$

A similar procedure is applied to horizontal overlaps. When there is both a vertical and a horizontal overlap, the minimal paths meet in the middle and the overall minimum is chosen for the cut.

4.2.1.3 Shortcomings of the Algorithm

Unfortunately this algorithm cannot be used for real time texture synthesis, as its computational efficiency is relatively low. The use of exhaustive searching in choosing the best match causes computational power to be wasted. Due to the use of a random picking technique in selecting the final block to be patched with the preceding block, often the seam between the two adjacent blocks is quite visible. Even though a minimum error boundary cutting technique is used to smooth off these sudden changes in texture, it involves computationally expensive methodologies such as dynamic programming and thus would not be suitable for real time applications.

4.2.2 Graphcut Textures [57]

This algorithm was introduced to the research community in August 2003. As mentioned earlier this benchmark algorithm is based on the image quilting concepts introduced by Efros and Freeman's algorithm. However a new edge cutting technique is adopted instead of dynamic programming. The initial patch stitching method used is similar to extensions we propose to the patch selection of [56] in section 4.3. The new texture is synthesized by copying irregularly shaped patches from the sample image into the output image. The patch copying process is performed in two stages. First a candidate rectangular patch (or patch offset) is selected by performing a comparison between the candidate patch and the pixels already in the output image. Second, an optimal (irregularly shaped) portion of this rectangle is computed and only these pixels are copied to the output image (fig. 4.2).

This process is repeated iteratively to improve the quality of the synthesized texture. The portion of the patch to be copied is determined by using a graph cut algorithm, which is the heart of the synthesis technique.

Figure 4.2: 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.

4.2.2.1 Patch Placement & Matching

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

previously unused offsets. The three patch selection algorithms used are:

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 piece of this patch to lay down into the output image, and then the process is repeated. This is a direct application of Guo & Shum’s [72] algorithm with graphcut based edge cutting. This method provides better results only for a few stochastic textures. Therefore only the following two patch placement methods are used in the general algorithm.

Figure 4.3: The process of synthesizing a larger texture from an example input texture. After initialisation, new patch locations are found depending on the seam costs in the refinement passes (bottom left).

Entire patch matching:

This step, named as the initialization step, starts with 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 synthesized 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 (eqn. 4.3) is used for selecting the best patch amongst them.

$$P(t) \propto e^{-\frac{C(t)}{k\sigma^2}} \quad (4.3)$$

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

Sub-patch matching:

This is the step after entire patch matching. A small sub-patch (which is usually significantly smaller than the input texture) in the output texture is initially picked. The selection is based on the seam cost in the previous steps (fig 4.3 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 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. 4.3.

4.2.2.2 Seam Cutting

In connecting patches the graph cut method is used to cut through the optimum path. In order to cast this into a graph cut problem, a matching quality measure is first selected for pixels from the old and new patch (of two overlapping patches). In the graph cut version of this problem, the selected path will run between pairs of pixels. The simplest quality measure, then, will be a measure of colour difference between the pairs of pixels. This matching cost is used in the graph cut technique proposed in [76, 142] to solve the path finding problem.

4.2.2.3 *Shortcomings of the Algorithm*

With the initial straightforward implementation, the complexity of the graphcut textures of Kwatra et al. algorithm was $O(n^2)$. But a FFT based acceleration technique [78, 143], which uses Summed Area Tables (SAT) and convolutions has been subsequently adopted for reducing the complexity to $O(n \log n)$. However the algorithm still 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 artifacts. Kwatra et al.'s algorithm uses several refinement passes in order to minimize these artifacts. However further improvements are required.

4.3 PROPOSED IMPROVEMENT FOR THE BEST PATCH SELECTION CRITERION OF [56]

Effros and Freeman select a random block from a group of good matches which would subsequently be quilted to the already synthesized texture. This is done in order to remove the risk of repetition of the sample texture as a whole or as a part on the synthesized texture. However the negative effect of this random selection method is the matching of sub-optimal blocks, rather than matching the optimal. Even if the blocks were finally quilted along the available minimum cost patch, there can be instances where visual artefacts considerably reduce the perceptual quality of the synthesized texture. Therefore a modification is proposed for the selection of the final patch to be quilted.

4.3.1 Picking the minimum cost patch

In the proposed extension, the initial block selection method which uses SSD for calculating the overlap errors in order to extract the subset of blocks with minimum overlap errors is not changed and the amendment is suggested in picking a block out of this subset. For each block in this subset, the minimum cost cut path is calculated

and the block with the lowest cost is selected as the most suitable block for the synthesis. The final quilting is done along the said minimum cost patch.

Figure 4.4 Performance evaluation of the proposed extension, (a) sample texture, (b) worst quality (c) best quality of the original algorithm in 10 random runs, (d) results with the proposed improvements

Extensive experiments carried out indicate that the optimal minimum cost patch does not necessarily belong to the candidate block that produced the minimum overlap SSD with the previously quilted block. Thus it is highly unlikely that the proposed extension would introduce repetitions to the resulting structure. Even though this approach may appear to be computationally expensive as compared to the technique adapted by the original algorithm, it is implemented efficiently by reusing summations done in the minimum patch calculation process.

4.3.2 Experimental Results and Analysis

In order to evaluate the performance of the proposed extension several runs of the algorithm were executed. Initially Efros and Freeman's [56] original algorithm was run 10 times for each texture and the perceptually best and worst outputs were selected for a fair comparison. The consequently amended algorithm was run for all the textures. Results are illustrated in fig. 4.4. Both methods used a sample size of 128×128 , block size of 60×60 and overlap $\frac{1}{3}$. The running times of the algorithm with the proposed modification were comparable to the original, with little or no difference. The results show no sign of extensive repetition, and the new procedure consistently performs equally or better than the original procedure.

4.4 A PROPOSAL FOR IMPROVED PATH CUTTING IN [56]

Efros and Freeman use a dynamic programming [144] approach as a compromise to finding the minimum cost path in the overlap region. However their approach prevents the minimum cost path traveling in the horizontal directions. This section proposes a simple extension to their algorithm, which allows travel along rows in addition. The idea is to allow the algorithm to check the neighbors to its right and left, before proceeding to check a pixel in a subsequent row.

4.4.1 Improved Path Finding and Cutting

Due to the constraints placed on the minimum path and the way this path is used to govern the starting points of rows or columns of pixels, the following simple

extension can be applied. After the cost of reaching each element in a row is found, the row is then checked to see if neighbouring elements in the row provide a cheaper route, if this is found then an element simply inherits the row index stored by its neighbour and calculates a new cost, as shown below in fig. 4.5. This adds very little to the computational expense of the original cutting algorithm but produces improvements as compared to the synthesized textures obtained from the original algorithm.

Figure 4.5: Improved path finding.

Figure 4.6: A comparison of path cutting using a sample image 128x128, block size 60x60 and overlap 1/3.

Figure 4.7. A comparison of cutting methods using only the minimum SSD block. The Image Quilting procedure was configured with a sample image 128x128, block size 60x60 and overlap 1/3.

4.4.2 Experimental Results and Analysis

Fig. 4.6 illustrates a comparison of the two path cutting algorithms in which the blocks are spaced out in order to observe the exact path cut by each algorithm. Close

examination of the edge cuts reveals vertical and horizontal lines are as expected with the new cutting algorithm.

Although the new algorithm makes only small changes in the block boundaries as shown in fig. 4.6, it has a considerable effect on the overall appearance of the synthesized texture. This can be seen from fig. 4.7. This extension has resulted in reducing the number of malformed and incomplete pixels present in the synthesized texture.

4.5 IMPROVEMENTS FOR CALCULATING THE SUMMATION OF OVERLAP ERRORS IN [56]

Efros and Freeman [56] have used simple summation of horizontal and vertical overlap errors. This in certain instances can make the algorithm pick the wrong block as the best match. For example, let's assume, there are two candidate blocks b_1 and b_2 with horizontal overlap errors b_1^H & b_2^H and vertical overlap errors b_1^V & b_2^V . If $b_1^H = 20, b_1^V = 20, b_2^H = 10, b_2^V = 30$ and if Efros & Freeman's method is used the overlap errors of both b_1 and b_2 will be the same. Whereas selection of b_2 can lead to a visible artifact across the vertical overlap edge. Selection of b_1 would be more perceptually desired in a texture synthesis algorithm.

4.5.1 Independent Overlaps

In this section independent consideration of vertical and horizontal overlap errors is proposed as an improvement. In the initial phase vertical and horizontal errors were stored against the block number. Then it was sorted in terms of horizontal error and in general the highest 70% of the blocks were discarded. Finally the list is sorted by vertical error the higher two-thirds were discarded. A random block is selected from the remaining blocks.

4.5.2 Experimental Results

Figure 4.8 compares the performance of the original and the proposed method. In order to illustrate the difference, edge cutting is not used in both cases. A slight improvement in the quality is visible in the proposed algorithm. When edge cutting is

Figure 4.8: Effect of independent overlaps

used the proposed technique provides better results than considering vertical and horizontal overlap errors together. This is due to the fact that when overlap error across an edge is high it becomes increasingly difficult for the minimum error boundary cut technique to smooth out the artifacts.

4.6 USE OF NEIGHBOURING BLOCKS IN MATCHING

The overlap area used in the matching process is one of the limiting factors in block based texture synthesis. Efros and Freeman [56] use an overlap size of $\frac{1}{3}$ rd the block size. For certain textures this is not large enough in order to capture the repeating micro patterns of the texture. The extension proposes considering neighbourhood blocks instead of selecting a small rectangular area as the overlap. This is an application of the original ‘pioneering block’ ideas proposed by J.Jing and E.A.Edirisinghe [145].

4.6.1 Block Matching

Though J.Jing and E.A.Edirisinghe [145] proposed their pioneering block concept for stereo image compression, it can be easily adopted for texture synthesis. In their algorithm in order to find a matching block for a particular position, the neighbouring blocks to its left and at the top are considered as matching templates. Therefore in the proposed technique similar to [145], instead of employing the small overlap areas, neighbouring blocks *v* and *h* govern the placement of a block *b* in the output image, shown in fig. 4.9. These blocks *v* and *h* are compared using SSD to neighbouring blocks of a candidate block in the sample image. The block that offers the minimal SSD neighbouring block pair is chosen.

Figure 4.9. A diagram of neighbouring block areas used for comparison between the sample and output image.

4.6.2 Experimental Results

As with Image Quilting the procedure replicates the sample image into the top left corner of the output texture until the block position exceeds the dimensions of maxX and maxY in the sample image. The process achieves better block alignment than that of the original image quilting process with equivalent block size and overlap of $\frac{1}{2}$ the block size, as shown in fig. 4.10. No method exists currently to remove block seams though when blocks are tiled side by side. The process is also more computationally expensive due to the large areas being used as a comparison, consequently the procedure takes up to twice as long to execute as the original, for the equivalent block size.

Figure 4.10. Comparison of the Block Matching and original Image Quilting procedures, using a sample image 128x128 and a Block Size 20x20.

4.7 USE OF WHOLE SAMPLE IN THE SYNTHESIS PROCESS INSTEAD OF BLOCKS

Efros & Freeman use blocks which are smaller in size than the sample as the unit of construction in the synthesis process. In order to avoid artefacts across the boundary and to have a good continuity throughout the texture, the block selected should be large enough so that it can capture the micro patterns properly (see section 5.4.4.1.2). At the same time the size of the overlap also plays a major role in quality (see section 5.7.2.2). Therefore small block size and overlap has become a limiting factor in their algorithm. Based on the ideas proposed by Kwatra et al. [57] and further simplification, the following extension is proposed in this regard.

4.7.1 Sample Quilting Algorithm

The proposed approach starts with placing the whole input texture sample in the top left hand corner of the output representation. Subsequently a copy of the sample is again picked and placed on the output representation, overlapped with the already placed sample. SSE of the overlap area is then calculated using the same equation (eqn. 4.2) as in [56]. The bottom left corner of the sample is moved horizontally and vertically for every possible location inside a window of height h and width w (see fig. 4.11). For each location inside the window the overlap error divided by the overlap area is calculated in order to normalize the values as the areas differ in each move inside the window, and locations having minimum 10% error are recorded,. Then the block of the output representation (sample texture) is placed at a position selected randomly from the recorded locations. The first row of blocks in the output representation is synthesized in a similar way by horizontal movement of the window along a line (let's call this the 'Base line'). The position on this base line is determined in the run depending on the end position of the previously placed block.

Figure 4.11: Finding the best position for the second block by moving the bottom left corner of the sample in a search window of size $h \times w$

In order to synthesize the second row of blocks, the base-line is shifted down a distance less than the height of the sample. For the first block of the second row the window is placed just outside the vertical margin of the output representation. The rest of the texture is synthesized in raster scan order by repeating the above procedure.

4.7.2 Location Selection

In the proposed extension what is basically done in synthesizing the first row of blocks of the output representation is matching the right hand side of the sample with its left hand side. Therefore the algorithm gives the same position as the best matching location which in turn leads to visible repetition. The same scenario happens in the first column of blocks of the output representation by having to match the top region of the blocks with the bottom region. Selecting a random position out of the locations with minimum 10% errors provides a solution to this problem. When synthesizing blocks for other locations, this problem is not that frequent due to having different combinations of edges as overlap. This problem can be further addressed through the edge handling technique discussed in the next section.

Figure 4.12, Experimental results

4.7.3 Edge Handling

Unlike in Efros & Freeman's approach, the proposed extension uses a larger overlap area. Thus if the minimum error boundary cut algorithm, used in [56] is used, large amounts of computation are required. Therefore this extension adopts a simple weighted edge blending technique similar to '*feathering*' proposed in [74], where the pixel values are weighted and blended according to the distance from the mid line of the overlap area.

4.7.4 Experimental Results & Analysis

The experimental results (see fig. 4.12) show that the proposed technique works well for regular textures and has problems when synthesizing irregular and stochastic. In certain textures a smudged effect is visible in the synthesized texture. This is mainly due to the problems in the edge handling approach as the algorithm does not memorize the previously blended areas.

The synthesis time of this algorithm is comparatively higher than that of Efros and Freeman's algorithm. This is due to the larger overlap areas used, that increases the number of computations involved.

4.8 CONCLUSION

The experiments carried out in this chapter implies that the methodology behind the process of Image Quilting as defined by Efros and Freeman [56] and Kwatra et al.[57] were found to be fairly strong, each stage having some advantages, such as the error boundary condition and removing repetition in synthesised texture. However some drawbacks were shown to exist in their implementation and basic idea.

The minor extensions proposed in sections 4.3 to 4.6 were able to achieve slight but nevertheless useful visual improvements. The approach proposed in section 4.7 provides promising results but requires further improvements in edge handling.

Chapter 5

WAVELET TRANSFORM DOMAIN PATCH BASED TEXTURE SYNTHESIS

5.1 OVERVIEW

Wavelet based technologies have been widely used in transform domain coding of still images (e.g. JPEG2000 [146]) and video (e.g. MPEG-4 [33]) sequences and high compression ratios were achieved. As discussed in section 1.3 and section 3.4 these techniques can be used in visually significant coefficient extraction in texture synthesis.

This chapter introduces a DWT based multi-resolution texture-synthesis algorithm which exploits the limitations of the HVS in its design. The idea was to reduce the number of computations in the synthesis process while improving the quality and to use these multiresolution techniques in image texture synthesis. Adaptation of the multi-resolution approach resulted in a fast, cost-effective, flexible texture synthesis algorithm that is capable of being used in conjunction with modern, bandwidth-adaptive, real-time imaging applications. Four algorithms to this effect are proposed. In the first three algorithms coefficient-blocks of the spatio-frequency components of the input texture are efficiently stitched together (i.e. ‘quilted’) to form the corresponding components of the synthesised output texture. The fourth algorithm introduces a DWT based edge artefact reduction technique.

For clarity of presentation this chapter is divided into several sections. Section 5.2 provides the reasons for using DWT in the proposed algorithm; section 5.3 introduces the basic technology used within the design of the three novel algorithms proposed in this chapter. Section 5.4 includes the design, implementation, experimental results and the analysis of the first and second algorithms, where only the low frequency coefficients of the lowest decomposition level of the DCT transformed image are used

in the synthesis. In the first algorithm a constant block size is used throughout. In the second algorithm, blocks are adaptively split so as to use the largest possible block size in order to preserve the global structure, while maintaining the mismatched error of the overlapped boundaries below a certain error tolerance. Section 5.5 discusses the design, implementation and testing of the third algorithm where visually significant coefficients spread throughout all the sub-bands are used for improved matching. Experiments are carried out to show 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 more realistic texture synthesis. Section 5.6 introduces a novel blending technique, which weights the frequency components of the overlapped area of joined blocks and removes remaining artefacts across edges in order to further improve the quality of the synthesized texture. Section 5.7 provides further analysis of the three algorithms. In addition a collection of regular and stochastic test textures is used to prove the effectiveness of the algorithm proposed in this section. The computational complexities of these algorithms are discussed in section 5.8. Finally section 5.10 concludes after discussing the applications of the proposed algorithms in section 5.9.

5.2 WHY DWT?

As explained in earlier sections one main reason for selecting DWT for the proposed algorithm is for speeding up the texture synthesis process while maintaining high quality by selecting visually significant information. This is possibly due to its very good energy compaction capabilities and spatio-frequency decorrelation.

The adoption of DWT results in a framework which provides a compact multi-resolution representation of the texture which in turn provides a multiresolution construction capability of synthesized texture. Another advantage is that the proposed algorithm can be used in bandwidth adaptive systems requiring dynamic quality/speed adjustments while providing fast mapping in hardware with low processing power. The last important reason for selecting DWT is that it had never been tried in block based texture synthesis algorithms before.

5.3 THE BASIC TECHNOLOGY

The specific design features adapted in the four novel algorithms proposed in this chapter are different. However they are based on DWT technology and a patch based texture synthesis architecture. The initial step in all three proposed algorithms involves application of the discrete wavelet transform to the sample texture image. Therefore the basics of DWT technology as applied to an image are discussed initially.

5.3.1 Application of DWT on an Image and Inter-subband Relationship

An image can be decomposed into different decomposition levels using the DWT as explained in chapter 3. Each decomposition level contains a number of subbands, which consist of coefficients that contain horizontal, vertical and diagonal spatial frequency characteristics of the original image. A so-called '*Power of two*' decomposition is allowed in the form of dyadic decomposition, as shown in fig. 5.1. Each image is decomposed into a collection of $3l + 1$ subband images, where l is the number of levels. Two filters, a one-dimensional low pass filter(L) and a one-dimensional high pass filter(H), based on a mother wavelet (section 3.4.4) are applied horizontally and vertically to the original image and the outputs are sub sampled by a factor of two, to create four sub band images.

In the proposed algorithm three-level decomposition is used (see section 5.4.1). The first DWT stage (level 1) decomposes the image into four sub bands, denoted by $LL1$ (represents low resolution version of original image), $LH1$ (horizontal features of the image), $HL1$ (vertical features of the image), and $HH1$ (diagonal features of the image) (see fig. 5.1(a)). In order to obtain the next level of wavelet coefficients, the sub-band $LL1$ is further decomposed and sampled using vertical and horizontal filters into four more sub bands, denoted by $LL2, LH2, HL2$ and $HH2$ (see Fig. 5.1 (b)). Lastly at level 3 the $LL2$ sub band is decomposed further into four further sub bands, denoted by $LL3, LH3, HL3$ and $HH3$. In general, for higher levels this process continues for a number of levels, l , producing a total of $3l + 1$.

Figure 5.1: (a) Application of 1 level discrete wavelet decomposition (b) 2nd level decomposition (c) Parent-child relationship between coefficients (d) Family tree corresponding to a wavelet coefficient in $LL3$ sub-band in a 3 level wavelet decomposition image.

By observing the sub-bands we can note that for each coarser scale, the coefficients represent a larger spatial area of the texture but a narrower band of frequencies. There are three sub-bands at each scale and the remaining, lowest frequency sub-band is a representation of the information at all coarser scales. In this sub band coding procedure the coefficients are grouped into subband oriented groups, i.e. common sub bands with different spatial locations.

In the above multi-resolution representation, coefficients in sub-bands have parent-child relationships. The family tree is constructed as follows (see figure 5.1 (d)). Each coefficient in $LL3$ sub-band has one child in sub-bands $HL3$, $LH3$ and $HH3$. Each

coefficient in $HL3$, $LH3$ and $HH3$ has four children in the corresponding lower sub-band. Note that in a three level decomposition, one coefficient tree corresponds to a 8×8 block of pixels in the original texture image.

5.3.2 Texture Synthesis Process

Generally, the process of texture synthesis can be mathematically represented by equation (5.1), where F is the texture synthesis function, which takes the input texture sample I_{sample} as the input and synthesizes a texture, I_{output} .

$$I_{output} = F(I_{sample}) \quad (5.1)$$

The proposed multiresolution approach to texture synthesis starts by applying a 2D discrete wavelet transform (e.g., Haar Transform) to the sample image, I_{sample} . The application of a single level 2D wavelet transform will result in decomposing I_{sample} into a set of component images (sub-bands).

$$I_{sample} = f(C_{LL1}^s, C_{HL1}^s, C_{LH1}^s, C_{HH1}^s) \quad (5.2)$$

Where $C_{LL1}^s, C_{HL1}^s, C_{LH1}^s, C_{HH1}^s$ are the image sub-bands (coefficient matrices) corresponding respectively to low-resolution approximation, horizontal detail, vertical detail and diagonal detail of the sample input texture. In general we represent an image sub-band as C_{kl}^p where $p \in \{s, o\}$, $k \in \{LL, HL, LH, HH\}$ and $l \in \{1, 2, 3\}$. In the above notation k represents the sub-bands within each decomposition level (LL -low resolution, HL -horizontal, LH -vertical, HH -diagonal), l represents the decomposition level (1-1st level, 2-2nd level, 3-3rd level) and p represents the decomposed image type ($p = s$ for sample image, $p = o$ for output/synthesized image). Note that f represents the inverse and f^{-1} represents the forward, discrete wavelet transform function.

Similarly 2 level and 3 level decompositions are obtained by applying a 2D wavelet transform to the low-resolution sub-band of the previous decomposition level. This could be mathematically presented as follows:

$$C_{LL1}^s = f(C_{LL2}^s, C_{HL2}^s, C_{LH2}^s, C_{HH2}^s) \quad (5.3)$$

$$C_{LL2}^s = f(C_{LL3}^s, C_{HL3}^s, C_{LH3}^s, C_{HH3}^s) \quad (5.4)$$

If a wavelet decomposition strategy similar to the above is performed on the output texture, I_{output} (which is yet to be generated) the following set of equations can be used for its mathematical representation.

$$I_{output} = f(C_{LL1}^o, C_{HL1}^o, C_{LH1}^o, C_{HH1}^o) \quad (5.5)$$

$$C_{LL1}^o = f(C_{LL2}^o, C_{HL2}^o, C_{LH2}^o, C_{HH2}^o) \quad (5.6)$$

$$C_{LL2}^o = f(C_{LL3}^o, C_{HL3}^o, C_{LH3}^o, C_{HH3}^o) \quad (5.7)$$

Therefore the three level wavelet decomposition of the sample and output textures consist of ten sub-bands each (see Fig.5.1(c)). The basic idea of the proposed multiresolution texture synthesis algorithm is to synthesize each sub-band of the output texture by the corresponding sub-band of the input, sample texture. This can be mathematically represented as $C_{kl}^o = F(C_{kl}^s)$ where F represents the synthesis function.

Texture synthesis procedures based on the basic DWT transformations are described in detail below.

5.4 ALGORITHM I & II (LOW FREQUENCY COMPONENT BASED)

5.4.1 Design Details

In section 3, DWT, HVS and the ways of incorporating a HVS model in DWT were discussed in detail. In this section the reader is introduced to the model used in the

first and second algorithms for selecting the wavelet coefficients. Further applicability of this model for different textures is also discussed via examples.

In the algorithm design, the main intention is to obtain the maximum amount of VSI from the minimum number of coefficients. From the extensive experiments carried out by the image compression community it is revealed that use of more than 5 levels of decomposition would not be beneficial in this regard. But in texture synthesis if the number of decomposition levels is increased the amount of candidate tiles available is reduced requiring more and more DWT transforms (see section 5.4.4.1.4). From the experiments carried out it is clear that 3-level decomposition gives the best compromise. This is the reason for the selection of a 3-level DWT transform in the proposed algorithms.

By close observation of the HVS model for DWT (fig. 3.6), it is clear that the visibility thresholds of different frequency sub-bands are different. As a whole it increases with increasing frequency. Further fig. 3.6 suggested that the highest level (3rd level in the proposed algorithm, if only a three level decomposition is obtained) contains the most visually significant data in an image. Based on this idea, in the first algorithm we assume that the approximation sub-band and a combination of either vertical, horizontal or diagonal sub-bands would contain sufficient information for good quality texture synthesis. Some simple experiments were conducted in order to check the veracity of this assumption. Figure 5.2 shows that the approximation details sub-band and the horizontal detail sub-bands provide a good approximation of the texture. This is further improved when 3rd level vertical sub-band details are added. Its appearance is nearly the same as when all three bands are added (fig. 5.2 (f)). Note that both these examples used only 3%-6% of the wavelet coefficients and retained around 80%-90% of the original energy content. Fig. 5.2(f) and fig. 5.3(f) were constructed using 6% of the coefficients.

Figure 5.2: 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

Figure 5.3: Reconstruction of “mat” 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

Figure 5.3 suggests a slightly different view. Adding any one of the horizontal, vertical or diagonal sub-bands to the approximation texture would not provide a good reconstruction. Adding all three components is barely enough for a sufficient approximation (see fig. 5.3(f)). This suggests that when the complexity of the repeating micro pattern increases, the accuracy of the adopted model decreases. This is due to the fact that when the complexity of the texture increases, the high frequency content also goes up making it impossible to give a good approximation when only the low frequency details of the 3rd level of the decomposition are considered.

5.4.2 Algorithm I

Let $B_{kl(x,y)}^s$ represent a general square block of the decomposed sample image (i.e. $p = s$), located at position (x,y) relative to the corresponding sub-band's origin. In the experiments the block size is set to $2^{6-l} \times 2^{6-l}$. In the first phase of the proposed texture synthesis procedure, only the four sub-bands of the 3rd decomposition level of the output image is synthesized. The first considered is the synthesis of the lowest resolution sub-band, i.e. C_{LL3}^o from samples of C_{LL3}^s . A block, $B_{LL3(xr,yr)}^s$ is first picked randomly from C_{LL3}^s and placed in the top left hand corner of the output coefficient image, C_{LL3}^o . Subsequently blocks on other sub-bands, which correspond in location to the random block above (i.e. $B_{kl(xr,yr)}^s$ where $l \in \{1,2,3\}$ and $k \in \{HL, LH, HH\}$) were transferred to the top left hand corners of the associated sub-bands, C_{kl}^o (see figure 5.4). Once the first block has been randomly selected and transferred to the output representation as discussed above, all possible blocks, $B_{LL3(x,y)}^s$ of similar size from the input sample image's C_{LL3}^s component are picked and matched, for a good overlap with the first block. The matching criterion is as follows.

Figure 5.4: Construction of the output texture. (a) First random block ($\dots$) and its best match ($\blacksquare$) (second block) placed on top left hand corner of the output texture together with the corresponding coefficient blocks of detail components in all three levels. (b) Selected best match for the first block with its corresponding coefficient blocks from the input texture (c) The matching criteria

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

$$d(B_{(x1,y1)}^o, B_{(x2,y2)}^s) = \sum_{i \in \partial B} \left[\left\{ \partial B_{LL3(x1,y1)}^o - \partial B_{LL3(x2,y2)}^s \right\}^2 \right] \quad (5.8)$$

Where ∂B_{kl}^p is the edge zone of block $B_{kl(x,y)}^p$ (see figure 5.4(c)) and i is an element (coefficient) within the edge zone.

In the proposed scheme we use the combined matching error of the blocks in its approximation sub-band ($LL3$) and one from $LH3$, $HL3$, $HH3$ sub-bands to compute the total matching error. Figure 5.4(c) illustrates the case where the $HH3$ sub-band together with the $LL3$ sub-band is used. In general, the decision to select a combination between the matching errors (as described above) in the horizontal, C_{LH3}^p vertical, C_{HL3}^p or diagonal, C_{HH3}^p sub-bands, could be taken by comparing the energy level of these components. This is justified as a higher energy detail component of a particular type; say the C_{LH3}^p component, would mean that the horizontal details of the original image would be more significant and visually important than the vertical or diagonal detail. However we avoid high frequency bands, C_{k1}^p in checking the suitability of the candidate sample as high frequency bands usually contain noise, which could be falsely interpreted as high energy.

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 fig 5.4(c)); we use the sum of the squares of the square-errors (eq. 5.8) of the two components, as the matching criteria. For example, if diagonal details are prominent the comparison equation (5.8) can be modified as below:

$$d(B_{(x1,y1)}^o, B_{(x2,y2)}^s) = \sum_{i \in \partial B} \left[\left\{ \begin{aligned} & \left[\partial B_{LL3(x1,y1)}^o(i) - \partial B_{LL3(x2,y2)}^s(i) \right]^2 \\ & + \left[\partial B_{HH3(x1,y1)}^o(i) - \partial B_{HH3(x2,y2)}^s(i) \right]^2 \end{aligned} \right\} \right] \quad (5.9)$$

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 image's C_{k2}^o component, the corresponding coefficient blocks from all detail images of the sample texture image are transferred to the appropriate components of the output texture image. (See figure 5.4 (a) & (b)). This process is continued until the whole output texture image is constructed in a three-level decomposed format.

Note that when constructing blocks other than those belonging to the first row and first column of blocks, the two overlaps, the first corresponding to the overlap with the block in front and the second corresponding to the overlap with the block above, need to be considered. Finally an inverse DWT is performed on the output image decomposition to create the output texture image, I_{output} .

5.4.3 Algorithm II

In order to maintain the global structure of the overall texture it is important to select the block size to be sufficiently large. This also accounts for increased efficiency of the algorithm as the choice of blocks available for filling the output texture becomes less, making the process fast (see section 5.4.4.1.2). Unfortunately the selection of large block sizes makes it increasingly difficult to find overlapping areas providing a good match, which in turn may affect the quality of the resulting texture. On the other hand the use of small block sizes will increase the synthesis time and may have a negative effect on the global structure of the synthesized texture (see section 5.4.4.1.2). It has been shown that the selection of the optimum size of the block is dependent on the repeating pattern contained in the texture to be synthesized [12]. Thus in an effective implementation of the proposed algorithm we need to have a trade off between the image quality and efficiency in selecting the block size. The following procedure is used to achieve optimum trade-off. For blocks in the decomposition level l , we start with a block size of $2^{6-l} \times 2^{6-l}$ coefficients. Subsequently, a maximum allowed overlap error (mean-squared error) threshold δ_1 , beyond which the visual quality is poor, is empirically defined. If the block is to be placed in the middle, the horizontal and vertical overlap errors are considered separately. If the overlap error is greater than δ_1 , the block is split into 2 smaller blocks (see figure 5.5). For example if the horizontal overlap error is higher than the

threshold the block is vertically split and vice versa. The above procedure is repeated for these two blocks. However the process is not repeated iteratively to smaller block sizes due to the necessity of preserving the global structure.

The following section provides detailed experimental results to justify the design and implementation of the proposed algorithm and to critically analyse its performance.

Figure 5.5: Changing the block size adaptively

5.4.4 Experimental Results and Analysis

In order to analyse the performance of the proposed algorithm, experiments were performed on a widely used set of popular test texture images widely used within the texture synthesis community (see figure 5.6), consisting of textures of both regular, fig. 5.6 (a), near regular, fig. 5.6(b) & 5.6(c), irregular, fig. 5.6 (d) & 5.6 (e) and stochastic nature (fig. 5.6 (f)). A typical sample size of 184×184 pixels was used to synthesize textures of size 320×320 pixels and only a quarter of the available candidate samples (see section 5.4.4.1.4) are used in the synthesis. Figure 5.7 provides the results of algorithm II and a performance comparison of the two algorithms is provided in figure 5.8.

Figure 5.6 (l): Multiresolution image quilting results. Sample texture (left)

(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.6 (II): Multiresolution image quilting results. Synthesized textures using the proposed algorithm

Figure 5.7: Texture Synthesis results with adaptive block sizes

Figure 5.8: Performance comparison: The use of fixed (left) vs. adaptive (right) block size

In section 5.4.4.1 the performance of the proposed algorithm is further investigated under three varying design considerations and in section 5.4.5 we compare it directly with that of [56]. A close visual inspection of figure 5.6, illustrates a good synthesized texture quality with little visibility of straight-line edges (i.e. mismatches) between patched blocks. To further support the subjective investigations and to allow a direct comparison with textures synthesized by [56] we propose the use of the objective quality measure, Mean Squared Difference of Slopes (MSDS) [147, 148], which gives a measure of the continuity of the texture across block boundaries. MSDS is defined as the mean of the square of the difference between the slope across two adjacent blocks and the average between the slopes of each of the two blocks close to the boundary. In relation to figure 5.9, where $w(i, j)$ and $f(i, j)$, ($0 \leq i, j \leq N-1$) are two adjacent (patched) $N \times N$ blocks, the MSDS along an edge (boundary) can be mathematically defined as:

$$MSDS_{edge} = \sum_{k=0}^{N-1} \left(\left\{ \frac{f(k, 0) - w(k, N-1)}{2} \right\}^2 \right) / N \quad (5.10)$$

The overall MSDS of the synthesized texture is calculated by totalling the Squared Difference of Slopes over all such edges (boundaries) in the synthesized texture and calculating the mean.

The quality metric MSDS has been used in the literature to evaluate the performance of blocking artefact reduction algorithms in DCT based image coding. It is capable of checking local smoothness across straight edge block boundaries. Thus it is not ideal in quantifying the maintenance of the global structure of synthesized textures. However it provides a reasonable objective metric to compare the performance of texture synthesis algorithms when edge blending is not used. This is particularly true for regular textures.

Figure 5.9: Four adjacent blocks of block 'f' for MSDS

5.4.4.1 Design Considerations

Closer subjective analysis of the synthesized textures in figure 5.6 and further experimental analysis carried out by us have revealed that there are important factors that are crucial in deciding the quality of the final texture synthesized by the proposed algorithm. They are:

- Level and type of detail used in selecting the matching block.
- Size of the unit of construction used.
- Number of candidate tiles available for matching

These factors are discussed in detail in the following sections.

5.4.4.1.1 Level and type of detail:

In contrast to the method proposed by Efros and Freeman, we have adapted a multiresolution matching strategy in selecting the adjacent blocks of the output texture. The use of pixel level detail in [56] would not only make the texture synthesis inefficient, but also unsuitable for real time texture synthesis capabilities expected by modern imaging applications. This is because full-resolution level details (pixel domain) have to be considered within the synthesis algorithm.

In the experiments performed we have assumed a typical three-level decomposition resulting in the lowest resolution component image to be $1/64$ times the size of the original image. The decisions about matching adjacent blocks are taken only considering the above component and a combination of other detail components (see 5.4.2) from the 3rd level of decomposition. Thus the computational complexity is considerably reduced as compared to the method proposed in [56] which performs block matching in the pixel domain, i.e. full resolution level. The exclusive use of the low-resolution component is justified due to the fact that it possesses the most VSI of all four components at a given level of decomposition. Further experimental results indicated that the use of the additional detail component (see section 5.4.1 and figure 5.10) resulted in better matching as compared to using only the lowest resolution component, justifying its use.

Figure 5.10 Effect of addition of sub-bands, left: texture synthesized with only $LL3$ sub band, Right: texture synthesized with all sub-bands of level 3.

Instead of randomly picking a block from a set of blocks, which matches within 10% of the error of the best matching block [56], the proposed texture synthesis method selects the best possible match in terms of the overlap criteria discussed above. The experiments carried out by us revealed that the random block picking technique proposed in [56] is ineffective when used in conjunction with the proposed multiresolution-based approach. This is due to two reasons. The first is the fact that the proposed algorithm performs block quilting (matching) using the $p = LL$ and $p = HH$ (say) sub-bands of the lowest resolution level $l = 3$ and a randomly selected block within 10% of the error of the best matching block could be vastly different from the best matching block. Secondly, in order to keep the computational cost down we do not carry out the minimum error boundary cut proposed in [56] or the feathering technique proposed in [73]. This simplification also means that the proposed technique needs less memory capacity to carry out this task.

5.4.4.1.2 Block size

In section 5.4.3 it was mentioned that the size of the element used in building the overall texture accounts for the overall quality of the texture and the efficiency of the algorithm.

Figure 5.11 comparison with varying block size, Image synthesized using block size (a) 16×16 , (b) 32×32 , (c) 64×64 , (d) 128×128

In figure 5.11 we compare the synthesized texture quality obtained when using block sizes 16×16 , 32×32 , 64×64 and 128×128 (note: these are the equivalent full resolution block sizes). It shows that at larger block sizes the mismatch across the block boundaries are more visible than at intermediate block sizes and more obvious repetition is visible across the synthesized texture. When the block size is small, though the algorithm tries to provide better continuation across the edges it fails to

retain the global structure and therefore leads to large amounts of artefacts in the synthesized texture.

The best size of block is dependent on the repeating pattern contained in the texture to be synthesized. In order to keep the computational complexity of the algorithm low, we have opted to use a fixed block size of 8×8 pixels at the third level of resolution ($l=3$). This is equivalent to using a 64×64 pixel block at full pixel resolution.

In the latter algorithm where the block size is changed adaptively, three different block sizes, 8×8 , 4×4 and 2×2 are used in the resolution level $l=3$. Figure 5.7 shows the results synthesized using the proposed method when the adaptive block size algorithm is used. Figure 5.8 compares the performance of the proposed algorithm using fixed sized blocks with that of using variable size blocks. It shows the improvement of the quilting quality obtainable with the adaptive block size selection algorithm.

5.4.4.1.3 Number of Candidate Tiles Available for Matching

In Efros and Freeman's algorithm, candidate tiles for matching can be selected with single pixel shifts in both directions. For example, if the sample size is 128×128 pixels and the tile size is 64×64 , there are 4096 ($= 64 \times 64$) different tiles to choose from. In the proposed algorithm, initially the sample is subjected to 3 level DWT transform. Due to this, it is not possible to shift by a single coefficient position to select another candidate for matching. When one position is shifted in the lowest resolution sub-band there will be an effective 8 position shift. Therefore from the above sample size we can only have 64 samples to choose from. Though this number of candidate blocks is enough for some textures, this may have a detrimental effect on the quality of the synthesized texture for most texture types. However the speed of finding a match increases as the number of candidate tiles for matching reduces. In order to compromise speed against quality we propose taking multiple numbers of DWT transforms of the samples as the alternative solution.

In order to obtain the maximum possible number of tiles it is necessary to obtain 64 ($= 8 \times 8$) different shifted DWT transforms of the sample image as discussed above (i.e. shifting one pixel at a time), in both directions. However taking 64 different DWT transforms is computationally ineffective and hence a compromise should be made. A careful consideration of the subjective quality of the synthesized textures shows that only one quarter of the total possible samples (i.e. taking 16 shifted DWTs of the sample) is sufficient for synthesizing very good visual quality output texture. The additional computational cost due to the shifted DWT calculations of the blocked sample image recommended above, are largely offset by the above speed-ups and the relatively larger size of the synthesized texture as compared to the sample texture. Further in a practical implementation the use of fast DWT calculations and reusing already calculated coefficients would enable further minimization of computational cost arising from additional DWT requirements.

5.4.5 Comparison with Benchmark

Figure 5.12 compares the performance of the proposed algorithm with that of Efros and Freeman. The results illustrate that the subjective quality of the proposed technique is marginally better than that of [56]. Note that for a fair comparison with [56], we have used the fixed block size version of our algorithm for visual comparison in figure 5.12.

Table.5.1 tabulates the MSDS values of the synthesized textures obtained with the proposed and benchmark algorithms for all test images considered in fig. 5.6 and fig 5.12. These results show that the objective quality performance of the proposed technique is marginally better than that of [56]. In particular for regular textures the proposed algorithm performs relatively better in comparison to the algorithm of [56]. This is because the benchmark algorithm picks up a random block amongst a set of best matching candidate blocks (within 10% of best match) in patching two adjacent blocks. Although when using stochastic textures this process may not create significant visual artefacts along block boundaries, when using regular textures,

Figure 5.12: Comparison of performance of the proposed algorithm with Efros and Freeman's algorithm. Top: Sample input textures. Left: Output textures generated using proposed algorithm. Right: Output texture generated using Efros and Freeman's algorithm.

noticeable artefacts could arise as a randomly selected block would not be the best choice.

Table 5.1: MSDS comparison

	Efros & Freeman	Our algorithm
Fig. 5.6(a)	0.0215	0.0208
Fig. 5.6(b)	0.0152	0.0146
Fig. 5.6(c)	0.0224	0.0216
Fig. 5.6(d)	0.0102	0.0091
Fig. 5.6(e)	0.0243	0.0230
Fig. 5.6(f)	0.0835	0.0776
Text Image in Fig. 5.12	0.0744	0.0661
Granite Image in Fig. 5.12	0.0243	0.0155

Table 5.1 and figure 5.12 show that both the subjective and objective performance of the proposed multiresolution texture synthesis algorithm is slightly better than that of [56]. However due to the use of the multiresolution approach the best contribution of the novel technique is to the algorithmic simplification achieved. Considering the fact that in the proposed technique the output texture is synthesised based only on a search on the lowest resolution level of the decomposed sample texture image, a computational complexity investigation revealed that theoretically the proposed algorithm uses 32 times less (64 times less if only LL3 information is used) number of coefficients in the general case than the benchmark algorithm. Due to the use of only a quarter of the available candidate matching blocks in the synthesis process leads to a further 4 fold theoretical speedup. More detailed computational complexity analysis is done in section 5.8. However when using a PC with a Pentium IV, 1600MHz, non-optimised MATLAB software based wavelet transforms, no hardware acceleration and synthesizing a 512×512 texture from a 184×184 sample the effective speedup finally achieved was approximately 30 fold. In particular when synthesizing textures considered, Efros and Freeman's algorithm took an average of 98.26 seconds per texture where the proposed algorithm took only 3.2 seconds. However when the algorithm is used within a image/video coding systems that require the

multiresolution decomposition of texture detail, the requirement for dedicated wavelet transforms for the purpose of texture quilting becomes non essential (image data is already in the wavelet transformed format). Therefore the effective speedup that could be achieved becomes much higher.

5.5 ALGORITHM III

Observation of the synthesis results of algorithms I & II show that the synthesis speed is satisfactory but further quality improvements are necessary. In the above algorithms only the lowest resolution sub-band, *LL3* together with a combination of *LH3*, *HH3* or *LH3* are used in selecting the block to be synthesized. However there are perceptually important coefficients in sub-bands of higher resolution levels as well as perceptually negligible coefficients in low frequency bands, i.e. in the considered sub-bands. As this was not accounted for in the above texture synthesis process, the quality of the synthesized texture was not very good for certain types of textures. Further computational power was wasted in considering coefficients of insignificant visual impact in the synthesis process.

In this algorithm substantial improvements are proposed in terms of speed and the quality of synthesized texture. Visually significant coefficients, which act as elements of a matching template are used in the texture quilting process. These coefficients which are distributed throughout all frequency bands are determined by an automatically generated threshold. It is shown 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 more realistic texture synthesis. The blending strategy used removes the remaining artefacts across edges further improving the quality. Popular test textures are used to compare our results with the existing state-of-the-art techniques. Many application scenarios for the proposed algorithm are also discussed.

5.5.1 Design Details

The key concept behind the design is the selective use of visually significant coefficients in the quilting process. Important theoretical work by Shapiro [149] in progressive encoding of images, namely Embedded Zerotree Wavelet (EZW) coding, is modified and used in the proposed texture synthesis process in order to pick the visually significant coefficients spread throughout all sub-bands in all the levels. This idea is based on two important observations:

- (i) Natural images in general have a low pass spectrum. 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 sub-bands than in the lower sub-bands (i.e. higher sub-bands only add details).
- (ii) Larger wavelet coefficients are more important than the smaller coefficients.

Further exploitation of this idea with the visibility threshold equation shown in chapter 3 suggests that it is necessary to find a threshold which gives the minimum number of coefficients distributed throughout all frequency bands that will result in the best possible perceptual quality. In order to show the effect of selecting the thresholds, experimental results illustrated in figure 5.13 are used.

The visual quality of fig. 5.13 (b) which uses only 3% of the coefficients is better than fig. 5.2 (f) which uses 6% of the coefficients. Fig. 5.13 (d) that uses 5% of wavelet coefficients outperforms fig. 5.3 (f) that uses 6% of the coefficients in quality. When comparing fig 5.2 & 5.3 with fig. 5.13 it is evident that the threshold base coefficient selection method outperforms the low frequency only selection method explained in section 5.4.1.

Further experiments carried out suggested that the optimum threshold is dependent on the details contained in the texture.

Figure 5.13, Effect of global thresholding. (a) & (c) original images, reconstructed images (b) using only 3% of (d) using only 5% of the frequency information picked from all the bands and all three levels

5.5.2 Proposed Multiresolution Framework

Similar to the above two algorithms the proposed multiresolution approach to texture synthesis starts by applying a three-level, 2D DWT to the sample image, I_{sample} . By combining the equations 5.1 to 5.7, I_{sample} and I_{output} can be represented as,

$$I_{sample} = f(C_{LL3}^s, C_{HL3}^s, C_{LH3}^s, C_{HH3}^s, C_{LL2}^s, C_{HL2}^s, C_{LH2}^s, C_{HH2}^s, C_{HL1}^s, C_{LH1}^s, C_{HH1}^s) \quad (5.11)$$

$$I_{output} = f(C_{LL3}^o, C_{HL3}^o, C_{LH3}^o, C_{HH3}^o, C_{HL2}^o, C_{LH2}^o, C_{HH2}^o, C_{HL1}^o, C_{LH1}^o, C_{HH1}^o) \quad (5.12)$$

The basic idea proposed in this algorithm is to synthesize each sub-band of the output texture by the corresponding sub-band of the sample texture as described in detail below. Unlike in the previous algorithms a tree of blocks is considered in the matching process instead of a few blocks in level 3.

Let $B_{(x,y)}^s$ denote a general, *block tree* (a combination of several adjacent *coefficient trees* (see fig 5.1 (d))) of the decomposed sample (Note: superscript's') image, located at position (x, y) relative to its origin. First a block tree $B_{(x_r, y_r)}^s$, is randomly picked and placed on the top left hand corner of the output coefficient image (see figure 5.14(a)). Let this block tree be denoted by $B_{(0,0)}^o$. Subsequently we create a so-called *matching mask tree* from $B_{(0,0)}^o$ by only selecting the coefficients higher than a pre-defined threshold out of the right hand side edge zone tree of the block $B_{(0,0)}^o$ (see figure 5.14(b)). Let this matching mask tree be denoted by $\partial B_{(0,0)}^o$. This matching mask tree (figure 5.14(c) top) is then moved around the sample image's 3 level DWT representation in search of the best match. When the best match is found, the corresponding block tree (figure 5.14(c) bottom) is picked and placed in the output representation (see figure 5.14(d)).

Note that due to the use of the matching mask tree, the coefficients above the pre-defined threshold in all sub-bands were used in the matching process. All coefficients below the threshold were disregarded in matching, regardless of whether they came from a low resolution sub-band or not. The matching criterion used is described as follows.

Figure 5.14: Construction of the output texture. (a) First random block tree (stippled) placed on the top left hand corner of the output texture (b) Edge zone tree of the first block tree. (c) top: Matching mask tree moved around the sample texture to find the best match (c) bottom: best position is found and the corresponding block tree is picked (d) First random block tree (stippled) and its best match (solid) (second block tree) placed on top left hand corner of the output texture.

In general, if $B_{(x1,y1)}^o$ and $B_{(x2,y2)}^s$ are two block trees to be matched, we say $B_{(x2,y2)}^s$ is the best match for $B_{(x1,y1)}^o$ if $d(B_{(x1,y1)}^o, B_{(x2,y2)}^s)$ is minimum for all possible B^s block trees where,

$$d(B_{(x1,y1)}^o, B_{(x2,y2)}^s) = \sum_{selected \ i} [\{\partial B_{(x1,y1)}^o(i) - \partial B_{(x2,y2)}^s(i)\}^2] \quad (5.13)$$

Where $\partial B_{(x,y)}$ is the *matching mask tree* of the *block tree* $B_{(x,y)}$ (see figure 5.14(c) above) and i is the i^{th} element in the matching mask tree. Note that i does not contain all the elements of the matching mask tree. These elements are selected using a threshold as discussed below.

5.5.3 Determining the Threshold

Threshold selection method is based on the observations and experiments carried out in section 5.5.1. The most desirable condition is to find a threshold, which gives the minimum number of coefficients which results in the best possible perceptual quality. Extensive experiments carried out show that the optimum threshold is dependant on the details contained in the texture and 5% to 20% (depending on the texture image) of the perceptually significant wavelet coefficients provide a sufficient approximation for synthesizing most of the texture types. Therefore the threshold is determined as follows,

Consider a sample texture size of $n \times n$. After the sample image is wavelet transformed, all the coefficients in all three levels are organized in descending order according to their absolute magnitudes. Consequently depending on the energy content in the high frequency bands, the magnitude of the k^{th} coefficient $\{ \text{where } (n^2 \times 0.05) < k < (n^2 \times 0.2) \}$ is selected as the threshold, T . Selection of the k in this way will result in a T , where only 5% to 20% of the high magnitude coefficients are considered.

5.5.4 Mask Creation and Best Match Selection

Once the above threshold is used to filter out the significant coefficients, the matching mask tree is created to be the significant coefficient representation of the *edge zone tree* (see fig. 5.14(b)). This mask is overlapped with the sample coefficient image and moved vertically and horizontally to all possible locations until the best match is found. At the best location, the block tree adjacent to the matching mask tree is picked

and placed in the output representation. This process is repeated until the whole output texture is filled.

5.6 MULTIREOLUTION WEIGHTED EDGE BLENDING TECHNIQUE

In certain cases it becomes impossible for the algorithm to find a good matching block and this leads to artefacts across the block boundaries. In this section, a multiresolution weighted edge blending technique, which utilizes the multi-resolution levels of the wavelet transformation to gain control over the blending of the block edges in different frequency bands is proposed. The proposed technique extends conventional image *feathering* [74] algorithms by offering additional control over the processing of the individual multi-resolution sub-bands that are used in the wavelet representation when moving from the source to the destination image.

In this process, the source image is smoothly faded out and distorted towards the alignment of features in the destination image. The destination image starts distorted to the feature geometry of the source image and while it fades in, the distortion is gradually reduced. The resulting image contains an average of the source and the destination image adjusted to the average feature geometry.

5.6.1 Algorithm IV

Consider two already selected adjacent coefficient blocks B^s and B^o (corresponding to image blocks I_0 and I_1) with corresponding vertical overlapping edge zones ∂B^s (source image) and ∂B^o (destination image) (See fig. 5.15). The edge blending technique is to generate an intermediate image $\partial B^{blended}(d)$ such that $\partial B^{blended}(0) = \partial B^s$ and $\partial B^{blended}(1) = \partial B^o$. We assume that the distance d varies from 0 to 1 when the source image, ∂B^s continuously changes to destination image ∂B^o . Note that source and destination image sub-band coefficients are organized into a tree structured representation (see fig. 5.15) using corresponding coefficients from blocks B^s and B^o .

Figure 5.15. (a) Already selected adjacent blocks showing 3-level wavelet decomposed edge zones (b). The two blocks placed so that they overlap.

It is necessary to define a set of weighting matrices, $W_{(sub)(level)}$, ($level \in \{1, 2, 3\}$ and $sub \in \{LL, LH, HL, HH\}$), where the weights of the matrices are defined such that when sub-bands of the left hand edge zone is multiplied by the corresponding weighting matrices, the left hand edge zones contribution to the overlapping region gradually decreases from left to right.

Similarly, when the right hand edge zone is multiplied by another set of weighting matrices derived from the above weighting matrices, its contribution to the overlapping area increases from left to right. The sizes of these weighting matrices are dependent on the width of the overlap.

To start with, let's consider blending in the pixel domain. If the weighting matrix is w the blending equation can be represented by following equation 5.14. As the overlap used in the proposed algorithm is 8, w is selected empirically as in eqn 5.15.

$$I^{blended} = I^s \cdot w + I^o \cdot (I - w) \quad (5.14)$$

$$w = \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \end{bmatrix} \quad (5.15)$$

Where I is an identity matrix and “dot” represents simple matrix multiplication. By application of a wavelet transform to both sides

$$f(I^{blended}) = f(I^s) \cdot f(w) + f(I^o) \cdot f(I - w) \quad (5.16)$$

This leads to a set of equations (5.17-5.20) where $level = 1$.

$$\begin{aligned} \partial B_{LL(level)}^{blended} &= \partial B_{LL(level)}^s \bullet W_{LL(level)} + \partial B_{HL(level)}^s \bullet W_{LH(level)} + \\ &\quad \partial B_{LL(level)}^o \bullet (I - W_{LL(level)}) + \partial B_{HL(level)}^o \bullet (I - W_{LH(level)}) \end{aligned} \quad (5.17)$$

$$\begin{aligned} \partial B_{HL(level)}^{blended} &= \partial B_{LL(level)}^s \bullet W_{HL(level)} + \partial B_{HL(level)}^s \bullet W_{HH(level)} + \\ &\quad \partial B_{LL(level)}^o \bullet (I - W_{HL(level)}) + \partial B_{HL(level)}^o \bullet (I - W_{HH(level)}) \end{aligned} \quad (5.18)$$

$$\begin{aligned} \partial B_{LH(level)}^{blended} &= \partial B_{LH(level)}^s \bullet W_{LL(level)} + \partial B_{HH(level)}^s \bullet W_{LH(level)} + \\ &\quad \partial B_{LH(level)}^o \bullet (I - W_{LL(level)}) + \partial B_{HH(level)}^o \bullet (I - W_{LH(level)}) \end{aligned} \quad (5.19)$$

$$\begin{aligned} \partial B_{HH(level)}^{blended} &= \partial B_{LH(level)}^s \bullet W_{HL(level)} + \partial B_{HH(level)}^s \bullet W_{HH(level)} + \\ &\quad \partial B_{LH(level)}^o \bullet (I - W_{HL(level)}) + \partial B_{HH(level)}^o \bullet (I - W_{HH(level)}) \end{aligned} \quad (5.20)$$

In general $X_{sub_{level}}$ represents a particular sub-band of an image matrix X in which, $level \in \{1, 2, 3\}$, $X \in \{\partial B^s, \partial B^o, W, \partial B^{blended}\}$ and $sub \in \{LL, LH, HL, HH\}$.

Similarly blended coefficients can be calculated for other levels. Figure 5.16 shows the effect of edge blending using the proposed algorithm. Note that mismatched blocks are selected deliberately to highlight the perceptual effect of blending.

Figure 5.16 above: joining without blending, below: Joined after blending

5.7 EXPERIMENTAL RESULTS AND ANALYSIS

5.7.1 Experimental results

Figure 5.17 comparisons with results of algorithm I; left: results of Algorithm I, Right: results of algorithm III

In order to analyse the performance of the proposed algorithm, experiments were performed on a widely used set of texture images, consisting of textures of both regular and stochastic nature. A typical set of sample images and the output textures obtainable using the proposed texture synthesis algorithm is illustrated in figure 5.17 and 5.18. Figure 5.17 shows the synthesized images without edge blending for a better

comparison with the previous algorithm. However when using a PC with the same configuration as section 5.4.5, non-optimised MATLAB code, software based wavelet transforms, no hardware acceleration, and synthesizing a 512×512 texture, the effective synthesis time was between 3 to 13.2 seconds. The results clearly indicate that the proposed method is capable of providing high quality texture synthesis for a wide variety of textures. We have chosen samples of different categories of textures (regular, near regular, random and stochastic) to illustrate that the proposed algorithm is capable of synthesizing all texture types.

Figure 5.19 compares the performance of the proposed technique with that of Efros and Freeman’s [56, 150], our previous method based on DWT and that of Kwatra [57], for two test images. They clearly illustrate the improved subjective quality performance of the proposed algorithm.

Figure 5.18(i) Synthesized (large) texture samples using the proposed algorithm. The corresponding small textures show the original texture samples.

Figure 5.18(ii) Synthesized (large) texture samples using the proposed algorithm. The corresponding small textures show the original texture samples.

Figure 5.19: Comparison of results; left: Efros and Freeman's algorithm; Middle: Kwatra [57], Right: Proposed algorithm

5.7.2 Detailed Analysis

Closer inspection of results in figures 5.17, 5.18 and further experiments have revealed that apart from the factors discussed in section 5.4.4.1 there are certain important factors that are crucial in deciding the quality of the final texture synthesized by the proposed algorithm. They are:

- (1) Number of coefficients used in the mask in selecting the matching block.
- (2) Width of the mask tree.

5.7.2.1 *Number of coefficients used*

In contrast to the method proposed by Efros and Freeman, we have adapted a multiresolution matching strategy in selecting the adjacent blocks of the output texture. The use of pixel level detail [56] would not only make the texture synthesis

inefficient, but also unsuitable for real time texture synthesis capabilities expected from modern imaging applications. In our previous method, we adopted all coefficients from a fixed number of sub-bands from the lowest level of resolution, for comparisons. Those coefficients, in some instances excluded some significant high frequency coefficients and included certain insignificant low frequency components. Yet it managed to achieve a remarkable speedup in synthesis compared to other state-of-the-art techniques while maintaining reasonably good quality compared to Efros and Freeman's algorithm.

In the proposed algorithm when the threshold is lowered the numbers of coefficients included in the mask tree is increased. This in turn increases the quality of the synthesized texture. In the experiments performed we started with the threshold at a higher value and gradually decreased. Our observations were that the quality of the synthesized texture increased up to a certain maximum level and lowering the threshold further, does not significantly improve the quality of the final output texture. At the same time when the threshold is lowered the number of comparisons increased due to the increased number of coefficients in the mask tree, resulting in an increased computational cost. This increases the synthesis time. Consequently we need to find a trade off between quality and speed. Empirically we found that the best threshold selection method as discussed in section 5.5.3, by using a large number of test images.

5.7.2.2 *Width of the Mask Tree*

In selecting the matching block, the width of the matching mask tree (corresponds to the area of overlap of block trees to be matched) will also account for the quality and the speed of synthesis. Use of a lesser number of overlapping elements (coefficients) results in increased efficiency but more visible artefacts at block boundaries. Increase of overlapping elements results in better quality with less artefacts and increased synthesis time. However, a too extensive increase in overlapping area will result in noticeable artefacts as it makes it more difficult for the algorithm to make the correct decision for the perceptually best matching block. In order to maintain a compromise situation we have adapted an overlap of a single coefficient row (or column) at level

LL3, of decomposition in our experiments. This amounts to an overlap of 8 pixel rows (or columns) in the pixel domain.

5.8 COMPARISON WITH BENCHMARK ALGORITHMS AND COMPUTATIONAL COMPLEXITY ANALYSIS

This section compares the computational complexity of the proposed algorithms with that of Efros & Freeman's [56] and Kwatra's [57].

In contrast to Kwatra's algorithm, in Efros & Freeman's algorithm, the size of the pixel domain patch selected for quilting is independent of the sample size. Thus, if n is the number of pixels in the sample, the selection of a candidate block from the sample for matching with an already quilted block, is of $O(n)$ complexity, whereas, the comparison of two blocks is of constant-time complexity. Hence the overall quilting process for a single quilted block in the synthesized texture is a linear-time operation, having $O(n)$ complexity. Similar to Efros & Freeman's algorithm, the proposed algorithms has linear-time computational complexity as shown below.

Assume that the input texture and sample tiles are square in nature. Let the sample be n pixels large and a tile consist of b pixels. Thus,

$$\text{Sample size} = \sqrt{n} \times \sqrt{n}$$

$$\text{Tile size} = \sqrt{b} \times \sqrt{b}$$

Let the tile overlap considered in matching two adjacent tiles be w pixels wide, along both vertical (with block in front) and horizontal (with the block above) directions. Therefore,

$$\text{Total number of candidate tiles available in sample per synthesized texture tile} = (\sqrt{n} - \sqrt{b})^2$$

Number of candidate tiles we consider per synthesized texture tile = $\frac{(\sqrt{n} - \sqrt{b})^2}{k_1}$

Note that $k_1 = 4$ is considered in general in the experiments. But even taking $k_1 = 16$ gives good results for some textures.

Total number of computations per synthesis of a general output texture tile, without considering the visual significance of certain coefficients = $2w\sqrt{b} - w^2$

Algorithms I & II

Number of computations considered in our experiments, assuming visual significance of certain coefficients = $(2w\sqrt{b} - w^2)/k_2$

Note that $k_2 = 32$ in our experiments.

Thus the total cost of synthesis of a single tile of the output texture =

$$\frac{(\sqrt{n} - \sqrt{b})^2}{k_1} \times \frac{(2w\sqrt{b} - w^2)}{k_2}$$

Here w , k_1 and k_2 are constant and therefore independent of n . However in algorithm II, the block size, b varies in adaptive splitting, but it is not dependent on n .

Therefore in big-O notation this can be expressed as $O(n)$. Thus the proposed algorithms I and II are capable of performing texture synthesis in $O(n)$.

However, due to the selection of a limited number of samples in candidate selection and a limited number of coefficients in block matching, the proposed algorithm's computational cost is reduced by a constant factor, as compared to that of Efros and Freeman's algorithm [56]. Based on the choices made in section 5.4.4.1, the overall number of calculations will be reduced by a factor of:

if only $LL3$ and one other sub-band is selected, $\frac{1}{4} \times \frac{2}{64} = \frac{1}{128}$

But due to non-optimized implementations the practical speedup obtainable is nearly 30 fold, as discussed before.

Algorithm III

In algorithm III the number of coefficients considered in synthesis of each block is not constant because of the threshold used. However this number can be considered as a constant, as it has only a little variation around a constant. Further this value is independent from n . Therefore the number of computations considered in our experiments, assuming visual significance of certain coefficients $= (2w\sqrt{b} - w^2) / k_3$

Note that k_3 varies between 20 and 5 in our experiments.

Thus the total cost of synthesis of a single tile of the output texture =

$$\frac{(\sqrt{n} - \sqrt{b})^2}{k_1} \times \frac{(2w\sqrt{b} - w^2)}{k_3}$$

As w, k_1 and k_3 are constant and therefore independent of n . Therefore in big-O notation this can be expressed as $O(n)$. Thus the proposed algorithms III is also capable of performing texture synthesis in $O(n)$.

Compared to that of Efros and Freeman's algorithm [56] the overall number of calculations will be reduced by a factor of:

if 5% of visually significant coefficients are used $\frac{1}{4} \times \frac{3}{64} = \frac{3}{256}$ (valid only for very few textures)

if 20% of visually significant coefficients are used $\frac{1}{4} \times \frac{12}{64} = \frac{3}{64}$

But due to non-optimized implementations the practical speedup obtainable is nearly 10 fold, as discussed before.

In Kwatra's algorithm, the patch size used for texture synthesis is dependent on the sample size. Though this results in a computational cost of $O(n^2)$ for synthesizing a single texture patch, Kwatra et al. have used a FFT technique to reduce the above cost to $O(n \log n)$. The Graphcut Texture algorithm consists of a pre-processing stage and several iterative passes of the main algorithm, before a texture with good synthesized quality is produced. The complexities of such additional stages have been neglected in comparison to $O(n \log n)$ as they have linear and/or constant complexity. However this assumption is true only for very large values of 'n' and in the presence of small constant factors.

Above, it was stated that similar to [56] the proposed algorithm uses a tile size, which is independent of the sample size. Therefore it reduces the computational cost for synthesizing a single texture tile to $O(n)$. Thus asymptotically the proposed algorithm performs better than that of Kwatra's.

For an evaluation of the practical efficiency of the proposed algorithm, further experiments were carried out by varying the sample size. This was done mainly to compare the proposed algorithm with that of Kwatra [57]. Output size was kept constant at 512×512 (262144) and sample size is varied from 128×128 to 1024×1024 . Figure 5.20 illustrates the plot of synthesis time vs. the sample size (in pixels). It also includes a linear trend line for comparison purposes. The graph shows a linear trend justifying the theoretical claim that the texture synthesis process of the proposed algorithm has computational cost of $O(n)$.

Figure 5.20: Input size (n) Vs Synthesis Time

5.9 APPLICATIONS

The following is a summary of applications that could benefit from the multiresolution design of the proposed fast texture synthesis algorithm.

Progressive 2D texture transmission:

Within a progressive transmission scenario, data are transmitted according to significance. The 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:

The MPEG 4 AFX standard is currently working on progressive transmission of 3D structures. Initially they transmit data sufficient for a coarse representation of a 3D shape. The proposed algorithm can complement this effort by texturing that surface with minimal transmission of texture data. Thus, both the structure as well as the texture can be refined progressively with more data transmission.

5.10 CONCLUSION

In this chapter, four novel approaches are introduced for synthesizing textures under a multi resolution framework. The experimental results and an in-depth analysis suggest that the proposed method works well, producing better output texture quality compared to the method proposed in [56] and equivalent or better quality compared to the method proposed in [57], at much lower computational complexity.

The first algorithm proposed in this chapter performs texture synthesis at remarkably good speeds. Unfortunately the qualities of the synthesized textures are very good only for a limited number of stochastic texture samples. But this algorithm provides acceptable quality for most of the texture types. The second algorithm proposed attempts to provide a solution to the quality problem by adaptively splitting the blocks. The zerotree approach based third algorithm which considers the visually significant information in high frequency bands synthesizes textures with excellent quality in very high speeds compared to the existing algorithms. Since the proposed algorithms perform texture synthesis at high speeds even with non-optimized MATLAB code and without hardware acceleration, the addition of these features will result in a real time texture synthesis algorithm.

The fourth algorithm proposed a wavelet based edge artefact reduction method which further improves the quality of the synthesized texture with very low additional cost. In addition this algorithm can be applied in wide variety of applications such as view morphing; wavelet based video texture synthesis, video editing etc.

The multiresolution nature of the proposed framework also makes it easily applicable to modern imaging applications needing progressive transmission capabilities.

Chapter 6

2D TEXTURE SYNTHESIS IN DISCRETE COSINE TRANSFORM DOMAIN

6.1 OVERVIEW

Almost all the existing images are in their compressed format due to the ease of transmission and storage. DCT is one of the most popular transforms widely used in image compression [151]. To the authors knowledge even though DCT was used in many image processing applications there hasn't been any attempt to use it in texture synthesis. The aim of the proposed algorithm is to explore the possibilities of using DCT in block based texture synthesis. In particular we consider the possibility of using a limited number of visually significant frequency coefficients within the block stitching process enabling high computational speed-ups in texture synthesis, while maintaining the synthesized texture quality at higher or equivalent levels as compared to existing algorithms. Further, a simple weighting function is used in order to avoid occasional visible seams along matched block boundaries. Throughout the algorithm designs, special consideration has been given to minimising the computational cost. We show that the adaptation of the frequency domain approach results in a fast, cost-effective, flexible texture synthesis algorithm that is capable of being used in conjunction with modern, bandwidth-adaptive/compressed domain, graphics applications. A collection of regular and stochastic test textures is used to prove the effectiveness of the proposed algorithm.

For clarity of presentation this chapter is divided into several sections. Section 6.2 starts with justifying the use of DCT in this regard. Section 6.3 introduces the basic technology behind the novel algorithm proposed in this chapter. The design and implementation details are included in section 6.4. Section 6.5 introduces the novel visual significance driven DCT coefficient based weighted edge smoothing technique. Section 6.6 provides the experimental results and an analysis while section 6.7

provides information of more detailed experiments carried out with the proposed algorithm. Section 6.8 compares the performance of the algorithm with the benchmark algorithms and the DWT based algorithm proposed in chapter 5. Sections 6.8 and 6.9 concludes a discussion on possible applications and advantages of the proposed algorithm.

6.2 WHY DCT?

The main reason for using DCT in the proposed algorithm is that the use of visually significant coefficients within texture synthesis enables effective exploitation of the limitations of the HVS, a task which is not possible when using traditional texture synthesis algorithms, that performs texture synthesis in the pixel domain. Another reason is its usefulness in frequency de-correlation, particularly the ability to isolate the visually insignificant coefficients. Further this opens a new paradigm in texture synthesis as it can be carried out in the transform/compressed domain without having to convert the sample texture to the pixel domain.

The ability to incorporate the progressive construction capabilities of DCT into the texture synthesis algorithm is an added advantage which makes this algorithm perform effectively in bandwidth adaptive systems requiring dynamic/speed adjustments. It is noted that texture synthesis in the DCT domain has not been considered before in literature.

6.3 BASIC TECHNOLOGY

The proposed technique adopts a two-stage process towards transform domain texture synthesis, i.e. a DCT transform domain *tile* (note: a tile consists of a collection of neighbouring DCT transformed blocks) quilting algorithm, which is used to form an initial representation of the transform domain synthesized texture and the subsequent edge blending strategy adopted to improve visual quality along the boundaries of quilted blocks. As both these algorithms are based on DCT, the technology is discussed first.

6.3.1 Application of DCT to an image

The proposed algorithm starts with dividing the texture sample into 8×8 pixel blocks. Therefore the analysis here is limited to 8×8 blocks though it could practically be of a general $n \times n$ size. Application of a two-dimensional DCT transform to each block results in the formation of 64 coefficients, organized in a zigzag pattern, starting from the zero frequency DC coefficient at the top left corner and ending with the highest frequency component at the bottom right (see fig. 6.1(b)). This is evident from the basis functions of this block shown in fig. 6.1(a). The HVS model in DCT (shown in fig. 3.8) provides the visibility thresholds corresponding to different frequency components. Close observation of this figure shows that the threshold values rapidly increase with frequency. Thus the resulting representation enables one to effectively group coefficients according to their visual significance. This is the characteristic which make the DCT transform suitable for texture synthesis.

Figure 6.1: DCT basis images of order, $N=8$ (a) Features corresponding to each DCT coefficient (b) Distribution order of frequency components in ascending order

6.3.2 Design Details

The proposed algorithm effectively exploits the limitations of the human visual system (HVS) [152] by considering coefficients that are visually significant in the synthesis process. The coefficient selection process is based on the HVS model in the DCT (fig. 3.8) and the results of the extensive experiments carried out in JPEG image compression standards. Several coefficient selection methods based on the HVS

model are proposed in this regard. The viability of these methods is discussed via examples in the following paragraphs.

One model proposed in this thesis uses only lower frequency coefficients in the synthesis process. Figure 6.2 (ii) illustrates the effect of using DC coefficients and a selected number of AC coefficients in the reconstruction process. According to the results illustrated it is seen that the “brick” texture can be approximated substantially even with only 3 coefficients out of 64 in a block whereas for the “mat” texture at least 10 coefficients are needed. These facts indicate that the use of this method effectively extracts the VSI.

Extensive experiments carried out by the image compression community have revealed the occurrence of visually insignificant information within the low frequency coefficients. In JPEG image compression standards (e.g. JPEG-baseline and MPEG-2 [130, 131]) each DCT block is divided by a scalar quantization matrix to further filter out visually insignificant frequency components. Only the remaining large frequency components are used for image representation. This is the concept used in the second method of coefficient selection proposed. In order to illustrate the effect of the DCT and the scalar quantization stages we consider the numerical example [133] given in figure 6.3.

(a)

Figure 6.2 (i): Effect of selecting only low frequency DCT coefficients (a) original textures representing all coefficients

Figure 6.2 (ii): Effect of selecting only low frequency DCT coefficients, image constructed using DC component and (b) 2, (c) 5, (d) 9 consecutive lowest AC coefficients out of 64 in a block

Figure 6.3: subjecting an image block to JPEG/ Inverse JPEG process

An image block shown in Fig 6.3(a) is first subjected to a forward DCT transform. The resulting coefficient matrix is shown in Fig. 6.3(b). Then it is quantized (see Fig. 6.3(d)) using a quantization matrix illustrated in fig. 6.3(c). This results in having only 6 non-zero (most visually significant coefficients) coefficients out of a possible 64.

In order to observe the damage done due to quantization the matrix is inverse quantized (see fig. 6.3(e)) and reconstructed (see fig. 6.3(f)). By comparing source image block and reconstructed block it is clear that the use of only 6 coefficients out of 64 doesn't have much effect on the overall quality. This is the scenario that encourages its usage in texture synthesis. Note the occurrence of a zero in fig. 6.3(d) in the lowest AC component. This suggests the fact that there is a possibility of occurrence of less visually significant coefficients within the low frequencies and the second model adopted can extract the VSI with a smaller number of coefficients than the first method.

In the proposed algorithm we demonstrate texture synthesis using both quantized and non-quantized coefficients and compare their relative merits.

6.4 TILE QUILTING ALGORITHM

Let us consider a sample texture image I^s of size $m \times n$. The sample texture is first divided into 8×8 pixel blocks, B^s_{ij} , each having 64 pixels, where i and j represent row and column numbers of the block.

$$I^s = \{B^s_{11}, B^s_{12}, \dots, B^s_{ij}, \dots, B^s_{\frac{n}{8} \frac{m}{8}}\} \quad (6.1)$$

Subsequently, a 2D Discrete Cosine Transform is applied to each 8×8 block, B^s_{ij} of I^s . i.e.,

$$DCT(B_{ij}) = b_{ij} \quad (6.2)$$

Where $1 \leq i \leq \frac{n}{8}$ and $1 \leq j \leq \frac{m}{8}$

Therefore we can represent the DCT transformed sample

$$DCT(I^s) = \{b_{11}^s, b_{12}^s, \dots, b_{ij}^s, \dots, b_{\frac{a}{8}\frac{b}{8}}^s\} \quad (6.3)$$

Subsequently a tile size for texture synthesis (a multiple of 8 in both dimensions) is selected. Then from the DCT transformed sample image randomly select a ' p blocks long and q blocks wide' *Tile* (Note: i.e., a tile consists of a block of coefficients of size $(p \times 8) \times (q \times 8)$) and place it (i.e. *quilt* the block) in the top left hand corner of the DCT domain representation of the output texture being created, $DCT(I^o)$, of size $a \times b$, where, using notation similar to that introduced in equations 6.2 and 6.3,

$$DCT(I^o) = \{b_{11}^o, b_{12}^o, \dots, b_{ij}^o, \dots, b_{\frac{a}{8}\frac{b}{8}}^o\} \quad (6.4)$$

Then we consider the rightmost column of blocks (consisting of blocks $\{b_{1q}^o, b_{2q}^o, b_{3q}^o, \dots, b_{pq}^o\}$) of the first quilted block (described above) as a matching mask (see Fig. 6.4). Subsequently this mask is moved throughout the transform domain sample image (i.e. $DCT(I^s)$) until the best possible match is found using the matching criteria described below. The 'tile' of size $(p \times 8) \times (q \times 8)$ immediately to the right of the best matching position is picked from $DCT(I^s)$ and used to quilt the area immediately to the right of the first quilted area of $DCT(I^o)$. This process is continued until the whole output texture, $DCT(I^o)$, is created. Note that in contrast to the vertically oriented overlapping areas assumed in the above discussion, in quilting the first column of tiles of $DCT(I^o)$, horizontally oriented overlapping areas are considered for matching, whereas when quilting all other tiles, L-shaped overlapping areas are considered.

In the experiments the Sum of Squared Error (SSE) is used as a metric to quantify the overlap error. In calculating the overlapping error between two potential neighbouring tiles, only the DC component and the lowest nine AC frequency components are considered from each 8×8 DCT transformed block (see fig 6.4(c)). However within

the detailed experiments to check the robustness of the proposed algorithm (see section 6.7), use of above mentioned coefficients in non-quantized and quantized formats is considered, separately. In addition the use of the DC component and the AC coefficients having the nine largest magnitudes is also considered.

Figure 6.4: (a) Placing tiles in the output representation. Numbers show the order of placing samples (b) Part of an enlarged Tile showing the overlapping area consisting of 8×8 coefficient blocks. (c) Enlarged block showing only visually important coefficients

The proposed transform domain texture synthesis process described above can be summarised using the following pseudo code.

BEGIN

Divide the sample image into 8×8 pixel blocks,

Apply DCT to each block.

Pick a Random $p \text{ Block} \times q \text{ Block}$ *Tile* from transform domain *Sample*,

Place in the top left corner of *OutputTexture*.

LastSynthesizedTile = $p \text{ Blocks} \times q \text{ Blocks}$ *Tile*

WHILE (*OutputTexture* is incomplete)

FOR all $p \text{ Blocks} \times q \text{ Blocks}$ *Tiles* of the *Sample*

Overlap with the *LastSynthesizedTile* &/or *Tile* directly above (depending on position)

Store *OverlapError* and corresponding *TileNumber*.

ENDFOR

```

    Find Minimum OverlapError and corresponding block position
    Patch Block
    Perform Weighted Edge Blending

ENDWHILE

END

```

Note that the DCT based weighted edge blending (see section 6.5) is performed while matching blocks are being quilted. Experiments have shown that this is computationally more efficient as compared to DCT based weighted edge blending considering as a post processing technique to transform domain texture synthesis described in this section.

6.5 WEIGHTED EDGE BLENDING ALGORITHM

The general idea of weighted edge blending is based on gradually reducing the luminance contribution made by a tile to its seams, while moving away to the adjoining block across the seam. In the proposed algorithm weighted edge blending is carried out in the DCT domain, which will be more appropriate for the method of tile quilting used within the proposed methodology.

Let w represent an empirically defined matrix capable of performing weighted edge blending in the pixel domain, via matrix multiplication with the overlap area. The size of w is dependent on the width of the overlap. In our experiments we have assumed an overlap width of eight pixels (hence coefficients, in the DCT domain) and have therefore defined w to be of size 8×8 . We define w as follows (same as in section 5.6.1):

$$w = \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \end{bmatrix} \quad (6.5)$$

$$DCT(w) = \begin{bmatrix} 0.5500 & -0.2278 & 0.0000 & -0.0238 & 0 & 0.0071 & -0.0000 & -0.0018 \\ -0.2278 & 0.5500 & -0.1779 & -0.0000 & -0.0219 & -0.0000 & -0.0063 & 0.0000 \\ 0 & 0.1779 & 0.5500 & -0.1661 & 0 & -0.0181 & 0 & -0.0038 \\ -0.0238 & -0.0000 & -0.1661 & 0.5500 & -0.1623 & 0.0000 & -0.0156 & -0.0000 \\ 0 & -0.0219 & 0.0000 & -0.1623 & 0.5500 & -0.1598 & 0.0000 & -0.0118 \\ -0.0071 & -0.0000 & -0.0181 & 0.0000 & -0.1598 & 0.5500 & -0.1560 & 0.0000 \\ 0.0000 & -0.0063 & 0.0000 & -0.0156 & 0.0000 & -0.1560 & 0.5500 & -0.1442 \\ -0.0018 & 0.0000 & -0.0038 & -0.0000 & -0.0118 & 0.0000 & -0.1442 & 0.5500 \end{bmatrix} \quad (6.6)$$

In general w can be directly used to obtain the pixel domain representation of the overlap area that results in an improved seamlessness between the two tiles. If the pixel domain representation of the overlap area of the left tile is denoted by $T_{leftlap}$ and that of the right tile is represented by $T_{rightlap}$, the pixel domain representation of the resulting blended overlap area, $O_{blendlap}$ can be represented by,

$$O_{blendlap} = T_{leftlap} \circ w + T_{rightlap} \circ (I - w) \quad (6.7)$$

Where $T_{leftlap}$ and $T_{rightlap}$ are of size $N \times 8$ ($N=64$ in our experiments) and I is an identity matrix of size 8×8 .

To obtain a transform domain equivalent, equation 6.7 can be converted to a DCT domain weighted blending algorithm as follows:

$$\begin{aligned} DCT(O_{blendlap})_{6 \text{ Coeff}} &= DCT(T_{leftlap})_{6 \text{ Coeff}} \circ DCT(w)_{6 \text{ Coeff}} \\ &+ DCT(T_{rightlap})_{6 \text{ Coeff}} \circ DCT(I - w)_{6 \text{ Coeff}} \end{aligned} \quad (6.8)$$

Note that equation 6.8 results from 6.7 due to the fact that the DCT transformation matrix we used is bi-orthogonal. In our experiments we have assumed an overlap area of a single column of 8×8 coefficient blocks (i.e. DCT domain), and a tile size of 64×64 . This *DCT coefficient weighting technique* at tile edges is only used to blend the DC coefficient and the first five AC coefficients from each 64×64 block. This leads to a minimisation of computational cost. Blending of sharp edges (high frequency components) can still cause artefacts. Blending only a few low frequency components makes those artefacts invisible.

Figure 6.5(i) Synthesized texture samples using the proposed algorithm. The corresponding small textures show the original texture samples. regular (a) & (b)

Figure 6.5 (ii) Synthesized (large) texture samples using the proposed algorithm. Regular (a) & (b), near regular (c)- (e), irregular (f) & (g), stochastic (h)

Figure 6.6: Comparison of texture samples, synthesized using different algorithms

Performing weighted edge blending in the transform domain benefits the proposed algorithm in two ways. Firstly, it eliminates the use of an inverse DCT stage as that would be required if blending was performed in the pixel domain. Thus the synthesized texture can be formed in the DCT domain for applications requiring pixel domain texture synthesis. Secondly as compared to applying weighted edge blending as a post processing stage, the proposed method would have more accurately blended neighbouring blocks ready in time for the quilting of a given block.

6.6 EXPERIMENTAL RESULTS AND ANALYSIS

In order to evaluate the performance of the proposed algorithm, experiments were performed on a widely used set of texture images, consisting of textures of both regular and stochastic nature. A typical set of sample images and the output textures obtainable using the proposed texture synthesis algorithm is illustrated in figure 6.5. The results clearly indicate that the proposed method is capable of providing high quality texture synthesis for a wide variety of textures.

Figure 6.6 compares the synthesized texture obtained by the proposed algorithm with that of [56, 57] and some other popular algorithms [12, 63, 166]. These results were shown to a group of observers for visual comparison. The group included both men and women as the frequency sensitivity for the two groups are known to be different. The experiments were carried out in general visual conditions for a fair comparison. The fact that the proposed algorithm is capable of producing marginally better quality synthesized texture as compared to the algorithm of [56, 57] and remarkably better quality as compared to all other algorithms considered, was broadly supported by the above subjective test results.

The blending technique used in [56] is computationally costly as it involves dynamic programming. Therefore the proposed blending strategy based on a simple matrix multiplication aids in achieving computational speed-up of the overall texture synthesis algorithm. The use of only six visually significant coefficients in blending as against 64 possible coefficients, further aids the speed-up. The improvement of the

visual quality obtainable by applying the proposed blending strategy is illustrated in figure 6.7. Note that in order to prove the effectiveness of the DCT based WEB process; we have intentionally lowered the subjective quality of the proposed texture synthesis process by only considering the DC values of the transform domain tiles, within the tile matching/quilting process.

Figure 6.7: Comparison of effect of blending (a) without blending (b) with blending (note: compare the encircled areas)

6.7 DETAILED EXPERIMENTS

In this section we further analyse the effects of using shifted DCT's in varying the number of candidate tiles made available for matching and the effects of using different DCT coefficients in tile quilting.

6.7.1 Number of Candidate Tiles Available for Matching

In the proposed algorithm, initially the sample is divided into 8×8 blocks and a DCT transform is applied to these blocks. Due to this, it is not possible to shift by a single coefficient position to select another candidate for matching, similar to the problem in the DWT domain. All DCT transformed block candidates are separated by 8

coefficient positions. Therefore from the above sample size we can only have 64 samples to choose from. This may have a detrimental effect on the quality of the synthesized texture. However the speed of finding a match increases as the number of candidate tiles for matching reduces. In order to compromise speed against quality we propose taking multiple numbers of DCT transforms of the samples as the alternative solution.

In order to obtain the maximum possible number of tiles it is necessary to obtain 64 ($= 8 \times 8$) different shifted DCT transforms of the sample image as discussed above (i.e. shifting one pixel at a time), in both directions. However taking 64 different DCT transforms is computationally ineffective and hence a compromise should be made. A careful consideration of the subjective quality of the synthesized textures shows that only one quarter of the total possible samples (i.e. taking 16 shifted DCTs of the sample) is sufficient for synthesizing very good visual quality output texture. The additional computational cost due to the shifted DCT calculations of the blocked sample image recommended above, are largely offset by the above speed-ups and the relatively large size of the synthesized texture as compared to the sample texture. Further in a practical implementation the use of fast DCT calculations would enable further minimization of computational cost arising from additional DCT requirements.

6.7.2 Use of different coefficients in tile quilting:

As explained in earlier sections the quality of the synthesized texture is dependent on the amount and the visual significance of the coefficients used within the matching process. The use of a higher number of visually significant coefficients improves the synthesized texture quality, but increases computational cost. This feature could effectively be used in achieving scalability in the texture synthesis quality of the proposed algorithm. For this purpose the coefficients must be accessible in order of their visual significance. The zig-zag pattern that is used in the scanning of DCT coefficients (for coding) in the baseline-JPEG standard (see fig 6.1(b)) [30] can be used for this purpose as it organizes the coefficients in ascending order of frequency, starting with the DC component.

Figure 6.8 (I): (a) with only 10 coefficients (b) With highest magnitude coefficients (c) with highest magnitude coefficients after dividing by DCT quantization matrix

Figure 6.8 (II): (a) With only 10 coefficients (b) With highest magnitude coefficients (c) With highest magnitude coefficients after quantization

Note that the experiments carried out within and outside the JPEG standardisation process have shown that the human eye is more sensitive to low frequency information than to high frequency information in images. This is the principle used behind JPEG compression. However considering visual significance of coefficients depending on their frequency alone would not be feasible.

Rather their relative magnitudes should also be considered. This is because in a given textured area, it is likely that certain low frequencies (thus thought to be of high visual significance) would not be present (or would be of very low magnitude) so that its visual presence within the block would be negligible. Under such a circumstance these would be insignificant in the matching process. Instead selecting a much higher frequency coefficient with a relatively higher energy (i.e. magnitude) would be more important. Considering the visual significance of coefficients using the quantized coefficients will be useful in quantized domain texture synthesis. Given the above facts we consider the following three scenarios in selecting the coefficients to be used in the matching process.

1. Ten lowest frequency non-quantized coefficients when arranged in the zig-zag order.
2. All non-quantized coefficients which are above an empirically selected threshold value.
3. All quantized coefficients which are above an empirically selected threshold value.

Figure 6.8 compares the synthesized texture quality of three test images when the above three procedures were followed for coefficient selection. It illustrates that the reduction of the number of coefficients considered within the matching process from 64 to 10 or to a smaller value depending on the empirically selected thresholds, does not have any seriously detrimental effect on the synthesized texture quality. In particular the use of the 10 lowest frequency coefficients in the matching purpose produces visually good quality texture synthesis. For textures consisting of irregular patterns, the synthesis quality of the texture is similar regardless of the scenario used (see above) for coefficient selection. Thus the proposed texture synthesis is generally capable of good quality texture synthesis in the compressed domain. Note that in applications where the speed of synthesis has priority over the quality of the synthesized texture the number of coefficients used in matching could be reduced by increasing the coefficient selection threshold, and vice versa.

6.8 COMPARISON WITH OTHER ALGORITHMS AND COMPUTATIONAL COMPLEXITY ANALYSIS

In this section we compare the computational complexity of proposed algorithm with that of Efros & Freeman's [56] Kwatra's [57] and the DWT based algorithm proposed in Chapter 5.

As explained in section 5.6 Efros & Freeman's paper [56] their algorithm has linear linear-time computational complexity $O(n)$ and Kwatra et al. [57] algorithm has $O(n \log n)$ complexity. The complexity of the proposed algorithm can be calculated as below.

In the proposed texture synthesis algorithm, the size of the sample texture tile, used in the synthesis of the output texture, is fixed, 64×64 pixels in our experiments. I.e., the tile size is not dependent of the sample texture size, n . Based on the above important observation, the total cost for synthesizing a single tile of the synthesized texture can be calculated as follows:

Assume that the input texture and sample tiles are square in nature. Let the sample be n pixels large and a tile consist of b pixels. Thus,

$$\text{Sample size} = \sqrt{n} \times \sqrt{n}$$

$$\text{Tile size} = \sqrt{b} \times \sqrt{b}$$

Let the tile overlap considered in matching two adjacent tiles be w pixels wide, along both vertical (with block in front) and horizontal (with the block above) directions. Therefore, total number of candidate tiles available in sample per synthesized texture

$$\text{tile} = (\sqrt{n} - \sqrt{b})^2$$

$$\text{Number of candidate tiles we consider per synthesized texture tile} = \frac{(\sqrt{n} - \sqrt{b})^2}{k_1}$$

Note that $k_1 = 4$ in our experiments.

Total number of computations per synthesis of a general output texture tile, without considering the visual significance of certain coefficients = $2w\sqrt{b} - w^2$

Number of computations considered in our experiments, assuming visual significance of certain coefficients = $(2w\sqrt{b} - w^2)/k_2$

Note that $k_2 = 10/64$ in our experiments.

Thus the total cost of synthesis of a single tile of the output texture =

$$\frac{(\sqrt{n} - \sqrt{b})^2}{k_1} \times \frac{(2w\sqrt{b} - w^2)}{k_2}$$

As b , w , k_1 and k_2 are constant and therefore independent of n , in big-O notation this can be expressed as $O(n)$. Thus the proposed algorithm is capable of performing texture synthesis in $O(n)$.

For an evaluation of the practical complexity of the proposed algorithm, further experiments were carried out by varying the sample size.

Figure 6.9 illustrates the plot of synthesis time vs. the sample size (in pixels). It also includes a linear trend line for comparison purposes. The graph shows a linear trend justifying the theoretical claim that the texture synthesis process of the proposed algorithm has computational cost of $O(n)$.

Figure 6.9: Sample size Vs synthesis time

However, due to the selection of a limited number of samples in candidate selection (section 6.7.1) and a limited number of coefficients in block matching (section 6.7.2), the proposed algorithm’s computational cost is reduced by a constant factor, as compared to that of [56]. Based on the choices made in sections 6.7.1 and 6.7.2, the overall theoretical speed up factor is $(1/4) \times (10/64) = 5/128$. In the DWT based algorithms the speed up factors are in the range between $1/128$ and $3/64$.

Thus asymptotically the proposed DCT based algorithm also performs better than Kwatra’s.

Table 6.1 tabulates the texture synthesis speed of the proposed algorithm, when a PC with a Pentium IV, 1600 MHz processor, non-optimised MATLAB code, software based DCT transforms were used with no hardware acceleration. The synthesized texture of size 512×512 , sample texture of size 184×184 and a tile size of 64×64 used. The 10 most significant coefficients (out of 64) have been used in tile matching. Note that the 900 sample case of Table 6.1 represents DCTs taken at 4 pixel shifts (i.e. $\frac{1}{4}$ th of total candidates) in both directions and 14400 sample case represents DCTs taken at 1 pixel shifts. It also compares the above performance results with that

of Efros and Freeman’s method and the non-optimised DWT based approach proposed by us in chapter 5, when the same hardware/software configuration is used.

Table 6.1. Texture synthesis times

Synthesis Method	Proposed DCT based algorithm				Efros & Freeman’s	DWT based
candidates	225	900	3600	14400	14400	3600
Synthesis time (s)	2.8	4.2	11.2	35.6	98.26	3 - 13.2

When comparing the performance of the proposed DCT and DWT based approaches, it is seen that the average computational complexities are in the same range. When comparing the practical speeds the wavelet based algorithms takes between 3-13.2 seconds while the DCT based algorithm took an average time of 11.2 seconds for synthesizing textures under similar conditions. Though the performances are nearly the same they have their own functional advantages as below.

DCT based algorithm

- Progressive construction capability.
- Most of the images are still in DCT based compressed (JPEG) format. The synthesis can be performed without having to convert them back to the pixel domain.
- Almost all the videos available today are in compressed format and they are using a DCT based compression format. Proposed DCT based texture synthesis algorithm can be easily adopted for fast video texture synthesis in the compressed domain.
- The flexibility of DCT transforms makes it very efficient in representing image/videos, which are in general non-stationary in nature.

DWT based Algorithm

- Multiresolution construction capability

- Can be easily adopted to 3D subdivision surfaces and compatible with MPEG-4 AFX standards,
- Aliasing distortions can be reduced significantly with proper choice of wavelet filters.
- The basis functions match the characteristics of human visual system
- The synthesized texture can be easily adopted in Region Of Interest (ROI) based applications

6.9 APPLICATIONS

Compressed domain texture synthesis:

Due to the ease of storage and transmission, almost all the images are in their compressed format. In order to synthesize a texture for fast on demand applications currently available texture synthesis algorithms need to uncompress the sample first. Then they perform texture synthesis and again compress the synthesized texture for transmission. But the proposed algorithm can make use of already compressed texture samples for synthesizing a compressed output, eliminating the requirement of unnecessary decompression and compression steps and saving a lot of time.

DCT based video texture synthesis:

The same principles discussed in this chapter can be used for a fast DCT based video texture synthesis algorithm. This was implemented as an MSc project under the supervision of the author. The weighted edge blending technique proposed was also used for smooth visual transition between frames of the video textures. In addition the speed of the video texture synthesis was considerably increased due to the adaptation of the HVS based approach.

6.10 CONCLUSION

We have proposed a fast DCT domain texture synthesis algorithm that is capable of synthesizing textures with quality equivalent to that of the best state-of-the-art texture

synthesis algorithms. We have shown that texture synthesis in the DCT domain enables a limited amount of visually significant texture information to be used within synthesis, thus allowing significant speed ups and additional functionality/flexibility, compared to existing pixel domain state-of-the-art texture synthesis algorithms.

The DCT based weighted edge blending algorithm proposed in this thesis, unlike the DWT based edge blending algorithm proposed in chapter 5, further exploits the limitations of the human visual system by using only a limited number of visually significant coefficients in the blending process. This requires very much less computational power compared to the currently available edge artefact reduction techniques employing dynamic programming or graph cut techniques.

We have provided experimental results and a detailed discussion in support of our claims.

Chapter 7

TEXTURE MAPPING ON 3D SURFACES

7.1 OVERVIEW

As discussed in chapter 2 many investigations have been carried out to provide a solution to the problem of fast texture mapping of arbitrary shaped surfaces without discontinuities and distortion. Despite these efforts fast efficient 3D texture synthesis still remains an open problem. Existing techniques suffer from problems such as (i) occupying a large memory space, (ii) requirement of higher user intervention, (iii) involvement of high computational cost etc.

This chapter proposes a general texture mapping method, where the user provides a sample texture and an arbitrary 3D mesh as input and gets texture coordinates on mesh vertices as the output. The aim of the proposed algorithm is to use the techniques developed in conjunction with a 2D block based texture synthesis algorithm and their modifications in three dimensional surface texture mapping. The implementation of the proposed algorithm is done for the DWT block based texture synthesis idea as a proof of concept. The proposed technique progressively covers the 3D surface using triangular patches. Moreover, the method uses little memory since no global texture map needs to be stored. For ease of understanding, first the algorithm is explained as a pixel domain texture-mapping algorithm. However no implementation of this was done in the pixel domain. Instead the same concept is adopted in the DWT domain. In the pixel domain a similar work is done by Soler et al. [92].

For clarity of presentation the chapter is divided into the following sections. Section 7.2 provides background information on 3D texture synthesis. Section 7.3 describes the pre-processing of 3D information extracted from PLY files in order to make them appropriate for matching when used in the proposed technique. Section 7.4 discusses the basic design features of the algorithm while section 7.5 addresses the

implementation issues. Section 7.6 provides details of the DWT adaptation of the algorithm. Section 7.7 provides experimental results and an analysis. The potential applications of the proposed technique are discussed in section 7.8. Finally, section 7.8 concludes with an insight in to possible future developments.

7.2 BACKGROUND INFORMATION

Mapping a 2D texture onto a surface in 3D requires a parameterization of the surface. This comes naturally for surfaces that are defined parametrically, such as bicubic patches, but less naturally for other surfaces such as polygons and quadrics, which are usually defined implicitly. The parameterization can be by surface coordinates u and v , as in standard texture mapping, by the direction of a normal vector or light ray, as in illumination mapping, or by spatial coordinates x_0, y_0, z_0 for objects that are to appear carved out of a solid material [79].

The main source of problems in the usual mapping techniques is that they generally rely on global surface parameterization. In most cases finding and modelling the correct global parameterization is difficult. Sample based techniques on the other hand keep local parameterization with continuity constraints. Texture mapping is quite popular for surfaces modelled from patches, for two reasons: (a) the parameterization comes for free; (b) the cost of texture mapping is small relative to the cost of patch rendering. The proposed algorithm uses 3D polygonal models where surface modelling is done with patches due to the above advantages and ease of subsequent adaptation to the DWT based approach.

Defining tiling with square patches is convenient in 2D texture synthesis techniques but it creates distortions in 3D. Therefore triangular patches are more suitable in tiling surfaces into regions where a local parameterization is defined. Even though there has been little previous effort [93, 154] in using triangular texture patches to design surface aspects in computer graphics, polygonal tiling adequately captures an object's topology and can be used for mapping textures. Therefore the first step of the

proposed algorithm consists of building triangular tiling of the surface from the set of vertices available.

7.3 PRE-PROCESSING OF POLYGONAL INFORMATION

Prior to performing texture synthesis, the 3D polygonal data needs to be accessed and processed. The proposed algorithm takes polygonal files presented in PLY format as the input for this purpose. A brief description of the PLY file format is presented below.

7.3.1 PLY File Format [164, 165]

The PLY polygon file format was designed as a convenient format for researchers who work with polygonal models for storing graphical objects that are described as a collection of polygons. This provides a format that is not only simple and easy to implement but also general enough to be useful for a wide range of models. The file format has two sub-formats: an ASCII representation for ease of getting started, and a binary version for compact storage and for rapid saving and loading. Early versions of this file format were used at Stanford University and at the University of North Carolina at Chapel Hill.

A typical PLY object definition is simply a list of (x,y,z) triples for vertices and a list of faces that are described by indices into the list of vertices. Most PLY files include this core information. Vertices and faces are two examples of "elements", and the bulk of a PLY file is its list of elements. Each element in a given file has a fixed number of "properties" that are specified for each element. Applications can create new properties that are attached to elements of an object [165].

7.3.1.1 *File Structure*

This is the structure of a typical PLY file:

```
Header
Vertex List
```

Face List

(lists of other elements)

The header is a series of carriage-return terminated lines of text that describe the remainder of the file. The header includes a description of each element type, including the element's name (e.g. 'face'), how many such elements are in the object, and a list of the various properties associated with the element. The header also tells whether the file is binary or ASCII. Following the header is one list of elements for each element type, presented in the order described in the header.

Below is an example giving a complete ASCII description for a cube. The header of a binary version of the same object would differ only in substituting the word "binary_little_endian" or "binary_big_endian" for the word "ascii". The comments in brackets are not part of the file; they are annotations to this example. Comments in files are ordinary keyword-identified lines that begin with the word "comment".

```
ply
format ascii 1.0          { ascii/binary, format version number }
comment made by Greg Turk { comments keyword specified, like all
lines }
comment this file is a cube
element vertex 8          { define "vertex" element, 8 of them in
file }
property float x           { vertex contains float "x" coordinate }
property float y           { y coordinate is also a vertex property }
property float z           { z coordinate, too }
element face 6             { there are 6 "face" elements in the file
}
property list uchar int vertex_index { "vertex_indices" is a list of
ints }
end_header                 {delimits the end of the header}
0 0 0                     {start of vertex list}
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0
4 0 1 2 3                 {start of face list}
```

```
4 7 6 5 4
4 0 4 5 1
4 1 5 6 2
4 2 6 7 3
4 3 7 4 0
```

After reading the polygonal information from the ply file, that information is processed by the proposed algorithm making it suitable for the mapping process.

7.3.2 Triangulation of Faces

Face lists presented in the PLY file can be in different forms such as triangular, quadrangular, etc. As the proposed algorithm uses only triangular texture patches for mapping onto the surface, triangulation of other shapes needs to be done in the pre processing stage.

7.3.3 Adjacency Information Extraction

Neighbourhood information is the key factor in the proposed texture synthesis process. After reading the polygon detail file the adjacency information of the triangular faces need to be extracted. The initial step of this process involves finding the corresponding edge list of each face. Consequently one edge of the face to be considered is picked from its edge list and all other faces are searched in order to find the face sharing the same edge, which is then added to the adjacency list of the first face. This process is repeated for all the edges of all faces.

The algorithm is ready for texture mapping after obtaining this information. For ease of understanding, the proposed algorithm is initially discussed as a pixel domain algorithm.

7.4 TEXTURE-MAPPING WITH TRIANGULAR PATCHES

The faces of the polygonal model are in the 3D space and they need to be mapped onto 2D texture space in order to start the texture-mapping process. For every face on the mesh surface, the parameterization is a linear map defined by the coordinates at the vertices. This linear mapping is obtained by translation and rotation of the mesh

faces such that the surface normal, n of the face is normal to the 2D texture space (see fig. 7.1).

Figure 7.1: Mapping the face in 3D onto the 2D texture space

For the first face the corresponding coordinates of the texture space are recorded as its texture coordinates (see fig. 7.2) and the face is marked as textured. Note that the position of the placement of this face on the texture plane is randomly selected.

The next step involves picking the next face from the queue and checking for its already texture-mapped neighbours. Consequently a mask, the shape of which is determined by the number of already texture-mapped neighbours, is created from a portion of texture adjacent to the common edges of the already textured faces (see section 7.4.1). This mask is moved inside a predefined window (see section 7.4.2) on the texture space for all possible locations. For each potential location, the overlap error is calculated and the position with the minimum overlap error is noted. The face to be textured is placed next to this location and the corresponding coordinates in the 2D texture space are assigned as its texture coordinates. Subsequently this process is repeated until the whole polygonal mesh is textured.

The following section addresses the issues related to the implementation of the proposed algorithm.

Figure 7.2: Assigning texture coordinates to the first triangular face [157]

7.5 IMPLEMENTATION ISSUES

In order to implement the above mentioned algorithm the following issues had to be addressed.

1. Mask creation and best match selection
2. Search window creation
3. Texture storage and rendering

These issues are discussed in detail in the following sections.

7.5.1 Mask creation and best match selection

As explained above, when texture-mapping a face in the mesh, in order to preserve the continuity between texture patches, previously textured neighbours need to be considered. Creating a mask in 3D is not as straightforward as in 2D for the following three reasons:

- (a) Adjacent faces of a candidate may be in different planes.
- (b) In cases where two or three neighbours are already textured, the masks cannot be placed on texture as it is, because the pasting operation is done as a post process. The individual masks need to be rotated for the proper angle between them (see fig. 7.3).

- (c) The size of the texture that needs to be picked is not a fixed square as in the 2D case.

Figure 7.3: Creating a mask for a face having two already textured neighbours. Right: The v-shaped mask is created by rotating the edge regions of the already textured neighbours F_1 and F_2 .

As mentioned before in the synthesis process, the face to be texture-mapped and its neighbours should be mapped onto 2D texture space first. Consequently pixels need to be extracted using a mask from the adjacent neighbours (see fig.7.3). Depending on the number of neighbours already texture-mapped, three algorithms are proposed to this effect.

When only one neighbour is already textured:

If the next face in the queue has only one already textured neighbour, this neighbour is rotated around the common edge into the plane of the face to be textured. Then both faces are mapped onto 2D texture space. Subsequently a mask is created from the texture extracted from the neighbours. The pixel extraction for the mask is performed as follows.

Assume the triangular face F to be textured, has neighbours F_1, F_2 and F_3 , where only F_1 is textured. It is therefore necessary to obtain the trapezoidal area (see fig.7.3) of F_1 next to the common edge, P_1P_2 as the mask.

Figure 7.4: Face to be textured with the neighbourhood

In order to obtain the mask, barycentric coordinates are used in the proposed algorithm. Let P_1, P_2, P_3 be the vertices with coordinates (x_1, y_1) , (x_2, y_2) and (x_3, y_3) of F_1 . Any point P , inside the face, can be expressed in the barycentric coordinates as

$$P = \alpha P_1 + \beta P_2 + \gamma P_3 \quad (7.1)$$

where $\alpha + \beta + \gamma = 1$, $\alpha > 0, \beta > 0, \gamma > 0$ and P lies in the plane of F_1 .

Any point inside the triangle can be expressed in terms of equation (7.1) where $0 < \alpha, \beta, \gamma < 1$.

In order to obtain the texture values inside the trapezoidal region (hatched region in fig. 7.4-right) equation 7.1 is used with the conditions, $0 < \alpha, \beta < 1$ and $0 \leq \gamma \leq k$ where k decides the distance from P_3 . By varying the value of k , it is possible to change the width of the mask.

When two neighbours are textured:

If the face to be texture-mapped F , has two already textured neighbours F_1 and F_2 , initially neighbours are rotated around their common edges until they come to the plane of the face considered. Consequently all three faces F, F_1 and F_2 are mapped

onto the 2D texture space a V-shaped mask is extracted from neighbourhood faces (see fig. 7.5 right).

Figure 7.5, neighbourhood mask of two already textured neighbours

When all three neighbours are textured:

When all three neighbours F_1, F_2 and F_3 of a particular face are already textured the shape of the mask would be triangular and the same searching procedure is followed.

All three procedures are repeated depending on the number of neighbours already texture-mapped, for all the faces of the 3D mesh.

7.5.2 Search Window Creation

The texture mask obtained as above needs to be moved throughout the texture sample in order to find the position with minimum overlap error. The location of the selected position should be such that a texture area similar to the shape and size of the face to be texture-mapped should be available next to the mask. In order to fulfil this requirement the search should be restricted to a window, the size of which is totally dependent on the dimensions and orientation of the face to be texture-mapped and its neighbours.

For example, consider the face F with only one neighbour F_1 already textured. The size of the search window would be as shown in Figure 7.6. The boundaries of the

search window at the left and bottom are different to the texture boundary. This is because after finding the best position a texture portion equal to the size of F needs to be picked as mentioned above. Figure 7.6 illustrates two search locations which are important in locating the boundaries of the search window.

Figure 7.6: Search for the best match. Hatched region shows the search window in the texture sample

A similar criterion is adopted for faces with two or three already textured neighbours.

7.5.3 Texture Storage and Rendering

Once texture coordinates of all the faces are found, the creation and storage of texture can be performed in two ways. The traditional representation is to create a global texture map for the whole mesh and pre-composite all the paste operations in it. Such a global texture map is called a texture atlas.

This thesis introduces an alternative method which consists of storing the computed parameterizations and compositing all the patches at runtime. This method required less memory and can be used in bandwidth adaptive systems as the transmission of a small texture sample and a set of coordinates is sufficient.

7.6 DWT DOMAIN PATCH BASED MAPPING ALGORITHM

If the above mentioned process is implemented in the pixel domain as explained above it would suffer from the same basic problems as in 2D texture synthesis, which were discussed in detail in chapter 4. Considering these facts a DWT based adaptation of the above pixel based 3D texture synthesis algorithm is proposed in this section.

The first step of the proposed DWT based implementation involves the DWT transformation of the sample texture. In n -level DWT decomposition a coefficient in the LLn sub-band corresponds to a $2^n \times 2^n$ block size. When the block size is larger it becomes more difficult to fit into the odd shaped masks, creation of which is necessary in the texture-mapping process. On the other hand, as discussed in chapter 5, 3-level decomposition provides most efficient extraction of VSI. Due to these reasons the number of levels is limited to 3 in the experiments.

Assigning texture coordinates to the first face in the queue is performed in a manner similar to that discussed in section 7.4 but in different scales (see fig. 7.7) at different sub-bands similar to that of chapter 5. When synthesizing textures on other faces in the queue mask creation criterion similar to that of section 7.4 is adopted. But instead of a single mask a trapezoidal mask tree is used in the algorithm. Only a few selected visually significant coefficients spread throughout the mask tree are considered in the matching process. The same threshold selection method used in chapter 5 is used for selecting the visually significant information. After the location with minimum overlap error is found, the texture coordinates corresponding to the shape and size of the face to be texture-mapped are assigned for that particular face. This process is continued until all the faces of the polygonal model are assigned with texture coordinates. Finally these texture coordinates are used for rendering as discussed in section 7.5.3.

Figure 7.7: Wavelet transformed sample and coefficient triangles corresponding to a face.

7.7 EXPERIMENTAL RESULTS AND ANALYSIS

Figure 7.8 shows the polygonal models textured using the proposed algorithm. The polygonal model shown in fig. 7.8(a) has only a few faces textured and lines are drawn across the edges in order to highlight the places where edge matching is done. Figure 7.8(b) shows the same polygonal model mapped with a different texture. It shows some edge artefacts across certain edges. This happens in places where there are three already textured neighbours and finding a good patch to match all becomes very difficult. These artefacts can be minimized by occupying an edge smoothing technique. This thesis concentrates on a basic algorithm design which aids the extension of the novel 2D texture synthesis idea presented in chapter 4-6 to 3D texture synthesis. The improvements and/or variations of this algorithm are not discussed.

Figure 7.8: Texture mapped 3D models

7.8 APPLICATIONS

Figure 7.9: Multi-resolution texture mapping: (a) Input texture. (b)-(e) Synthesized textures onto a 2D surface (f)-(i) Synthesized textures onto a sphere, at progressively higher resolutions. (j)-(m) effect of adding higher level details to the sphere at higher resolution

3D texture mapped objects contain large amounts of information. Due to this in downloading these through the internet, the available bandwidth plays a critical role. Users with a limited bandwidth, such as dial up users probably will not be able to view the object. Progressive transmission can provide a solution for this problem by initially sending a rough texture-mapped shape of the 3D object with few vertices. This is achieved by first subdividing the 3D object into a simpler object with few vertices and quickly mapping texture with lowest resolution band and gradually refining both polygonal model as well as texture. Then the users with limited bandwidth can satisfy with the coarser texture-mapped model while others can gradually receive the rest of the polygonal data and texture data.

The proposed algorithm can be adopted for this purpose as multi-resolution texture mapping is a key advantage provided by it. This is not possible by the adaptation of the pixel based approach to texture mapping. Figure 7.9 illustrates the effect of mapping the texture using a multi-resolution approach. The mapping of the texture in figure 7.9(f) & (j) are done using only the LL_3 sub-band (fig. 7.9(b)) of the sample texture (fig. 7.9(a)). The mapping of texture in figures 7.9(g) & 7.9(k) was done using all sub-bands at resolution level 3. In mapping the texture in figures 7.9(h) & (l) and 7.9(i) & (m), the detailed sub-bands (LH, HL and HH) in the next higher resolution level are respectively considered in addition. The proposed algorithm can be adapted for MESHGRID 3D models [158] having the subdivision capability introduced in MPEG-4 AFX standards.

7.9 CONCLUSION

This chapter explained a 3D texture mapping algorithm that adopts the wavelet based 2D texture synthesis techniques discussed in chapter 5. The design and implementation is carried out in order to show the viability of the proposed algorithm in 3D texture mapping. Due to adaptation of the HVS based approach, the proposed algorithm performs texture-mapping faster than the existing pixel domain texture mapping algorithms. In addition to the other advantages inherited due to the use of

DWT, it has the ability to be used in progressive transmission of 3D texture mapped objects.

The experimental results show that it can be used for a number of textures. Further improvements such as edge blending and clustering the faces can be used for the development of the algorithm.

Chapter 8

CONCLUSIONS AND FUTURE RESEARCH

This chapter summarizes the key results presented in chapters 4, 5, 6 and 7, draws conclusions and emphasizes the important contributions presented in this thesis. It also presents possible future directions of research in order to extend the functionalities and efficiency of the proposed algorithms.

The thesis initially introduced several extensions to the pixel domain texture synthesis algorithm of Efros et al., which has been used as a benchmark. Later it reports the design, development and analysis of four novel algorithms for transform domain 2D texture synthesis, two novel algorithms for removing seam artefacts in block based transform domain texture synthesis and an algorithm for 3D texture mapping.

The research work reported in this thesis commenced with proposing possible improvements to the benchmark algorithm of [56]. Several amendments to the original benchmark algorithm in terms of improving the synthesized texture quality were proposed. They included a modified best patch selection criteria based on minimum path cutting cost, an improved path cutting technique with freedom of horizontal movement, a technique that considers independent vertical and horizontal overlap errors that avoids the selection of matching blocks having a higher error in only one side (vertical or horizontal overlap) and neighbourhood block consideration in selecting the suitable matching block instead of using small overlap areas. It was shown that these amendments resulted in a substantial quality improvement in the synthesized texture. Finally an extension to the benchmark algorithm, which uses the entire sample in the synthesis process instead of sub-blocks from the sample, together with the use of weighted edge blending at block seams was proposed. The synthesis quality improvements possible with this approach were evaluated and conclusions were drawn.

The first three novel texture synthesis algorithms proposed in this thesis are designed under a DWT based multi-resolution framework. The first algorithm effectively exploits the limitations of the human visual system by considering the lowest resolution sub-band, which is more sensitive to the human vision, in the texture synthesis process. It has been shown that this algorithm increases the speed of texture synthesis considerably while maintaining the quality at acceptable levels. The second algorithm, while adopting the same framework, attempts to address the problem of preservation of global structure in the texture synthesis, while maintaining the edge artefacts to a minimum. This is achieved by changing the block size adaptively according to the overlap error throughout the synthesis process. The possible improvements to the quality of the synthesized texture and the speed of synthesis were analysed and conclusions have been drawn.

The third algorithm exploits the limitations of the HVS further by addressing a major draw back in the first algorithm, i.e. ignorance of visually significant information prevailing in high frequency bands and consideration of insignificant information in low frequency bands. The algorithm uses an automatically generated threshold in obtaining visually significant information for the texture synthesis process. It has been shown that this results in improved quality at increased speeds making it possible to synthesize a wide variety of textures, while addressing the problems inherited in SSD in error calculation. It has been shown that the multi-resolution nature of the proposed framework also makes it easily applicable to modern imaging applications needing progressive transmission capabilities. The experimental results and an in-depth analysis have been provided to justify that the proposed method works remarkably fast, producing better output texture quality as compared to the methods proposed in [57] and provides similar quality results but with higher speeds compared to the recent work of Kwatra's et al. [57], which is at present considered as one of the best texture synthesis algorithms amongst the computer graphics community.

The fourth algorithm based on DCT domain texture synthesis opens a new paradigm in texture synthesis. It can be carried out in the transform/compressed domain without having the need to first convert sample texture to the pixel domain, synthesizing the

texture and then having to convert the synthesized texture back to the quantized transform domain for efficient storage/transmission. This enables the potential use of the technique in new application domains. The proposed work is the first cited attempt in performing texture synthesis in the DCT domain. It has been shown that the use of visually significant DCT coefficients within texture synthesis, effectively exploits the limitations of the human visual system (HVS), a task which is not possible when using traditional texture synthesis algorithms that perform texture synthesis in the pixel domain. It has also been shown that this novel approach allows significant speed ups and additional functionality/flexibility, compared to existing pixel domain state-of-the-art texture synthesis algorithms. The extensive experiments carried out within this thesis have proved that the proposed DCT based algorithm is able to perform texture synthesis with $O(n)$ complexity whereas Kwatra's [57] method has a complexity of $O(n \log n)$ even after FFT based acceleration, making the proposed algorithm asymptotically better.

Though minimum error boundary cut techniques are implemented using dynamic programming in most of the pixel domain texture synthesis algorithms, they are computationally costly and do not fully exploit limitations of the HVS. The fifth and sixth novel algorithms were designed to address the shortcomings of these techniques. The proposed transform domain weighted edge blending approaches facilitate the block artefact reduction in above mentioned texture synthesis approaches, without having to convert the sample back to pixel domain. The fifth algorithm performs wavelet based edge smoothing by weighting the frequency components of different frequency bands of different levels, with the use of a weighting matrix in order to provide a smooth visual transition between coefficient blocks to be joined. The DCT domain edge blending proposed in sixth algorithm considers only the visually significant coefficients in the weighting process, saving computational cost and fully exploiting the limitations of the HVS. The computational costs of this edge smoothing techniques are less than dynamic programming techniques used in [56] and the graph cut approach used in [57].

The final algorithm proposed in this thesis performs texture mapping on 3D surfaces. The implementation is a proof of concept to show the possibility of application of one of the wavelet based 2D multi-resolution texture synthesis techniques proposed in the thesis (4th algorithm) to 3D texture synthesis. It has been concluded that the algorithm needs further development in order to be used efficiently in the 3D texture mapping. The possibility of application of this technique in progressive 3D transmission has been discussed.

We are currently working on software optimization techniques and accelerated hardware implementations to further speed up the proposed texture synthesis processes. In addition we are currently looking at extending the idea to transform domain texture synthesis related to 3D objects and arbitrarily shaped surfaces. The proposed transform domain approaches would be particularly suitable in compressed domain texture synthesis applications.

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

The standardization of 3D content in MPEG-4 AFX, and the recent development of MPEG-4 Graphics Framework Extension (GFX) [35] are at present further extending the application areas of the proposed algorithms. One such area is the application of proposed texture mapping ideas in MESHGRID[158], a method used for the representation of 3D objects in MPEG-4 AFX. As MESHGRID based wavelet subdivision of surfaces of arbitrary shaped 3D objects has been already proposed in the above mentioned standards, the proposed DWT based texture mapping algorithm can be adopted to further extend its capabilities.

REFERENCES

- [1] *IEEE standard glossary of image processing and pattern recognition terminology*, standard 610.4-1990: IEEE Press, New York
- [2] R. C. Gonzalez, 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* . Tch. report CMU-RI-TR-04-01, Robotics Institute, Carnegie Mellon University, January, 2004
- [5] G. Winkenbach, D. H. Salesin. *Computer-generated pen-and-ink illustration*. Proceedings of *SIGGRAPH 94*, pages 91–100, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.
- [6] G. Winkenbach, David H. Salesin, *Rendering parametric surfaces in pen and ink*. Proceedings of *SIGGRAPH 96*, pages 469–476, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.
- [7] M. Szummer and R. W. Picard. *Temporal texture modeling*. In *International Conference on Image Processing*, volume 3, pages 823–6, Sep 1996.
- [8] A. Schödl, R. Szeliski, D. H. Salesin, I. Essa, *Video textures*. Proceedings of *SIGGRAPH 2000*, pages 489-498, July 2000
- [9] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, M. Werman, *Texture Mixing and Texture Movie Synthesis Using Statistical Learning*, Volume 7 , Issue 2 Pages: 120 - 135 (April 2001)
- [10] S. Soatto, G. Doretto, and Y. N. Wu. *Dynamic textures*. In *IEEE International Conference on Computer Vision*, pages 439--446, 2001.

- [11] A. C. Beers, M. Agrawala, and N. Chaddha. *Rendering from compressed textures. Proceedings of SIGGRAPH 96*, pages 373–378, August 1996.
- [12] A. Efros and T. Leung. *Texture synthesis by non-parametric sampling. In International Conference on Computer Vision*, volume 2, pages 1033–8, Sep 1999.
- [13] H. Igehy and L. Pereira. *Image replacement through texture synthesis. In International Conference on Image Processing*, volume 3, pages 186–189, Oct 1997.
- [14] H. Yamauchi, J. Haber and H.P. Seidel, "Image Restoration using Multiresolution Texture Synthesis and Image Inpainting," *Proc. Computer Graphics International (CGI) 2003*, 2003, pp.120-125, 9-11 July, Tokyo, Japan.
- [15] H. Chong, *GPU Image Inpainting via Texture Synthesis*, www.people.fas.harvard.edu/~hchong/goodies/inpaint.pdf
- [16] J. Min, M. Powell, and K. Bowyer, "Automated performance evaluation of range image segmentation algorithms," in H. I. Christensen and J. Phillips, eds., *Empirical Evaluation Methods in Computer Vision*, World Scientific Press, Singapore, 2001.
- [17] P. Felzenszwalb, D. Huttenlocher. *Image segmentation using local variation. Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 98–104, 1998.
- [18] G. Paschos, *Fast color texture recognition using chromacity moments. Pattern Recognition Letters* **21** (2000) 837-841
- [19] P. Wu, Y.M. Ro, C.S. Won, Y. Choi. *Texture descriptors in MPEG-7. In W. Sharbek, editor, CAIP 2001, Lecture Notes in Computer Science 2124*, pages 21–28, Warsaw, Poland, September 2001. Springer Verlag.
- [20] R. Smith, *Getting Started: Image Classification* Lincoln, Nebraska: Micro Images. 23p. (1997)

- [21] T. Caelli, D. Reye, *On the classification of image regions by color, texture and shape*. Pattern Recognition 26 (1993) 461-470
- [22] J. R. Bergen, B. Julesz, "*Rapid discrimination of visual patterns*," IEEE Transactions on Systems, Man, and Cybernetics, vol. 13, no. 5, pp. 857-863, 1983.
- [23] B. Julesz, "*Visual pattern discrimination*," IRE Transactions on Information Theory, vol. 8, pp. 84-92, 1962.
- [24] W. K. Pratt, O. D. Faugeras, A. Gagalowicz, "*Visual discrimination of stochastic texture fields*," IEEE Transactions on Systems, Man and Cybernetics, vol. 8, pp. 796-804, Nov. 1978.
- [25] T. Caelli, B. Julesz, E. Gilbert, "*On perceptual analyzers underlying visual texture discrimination: Part II*," Biological Cybernetics, vol. 29, no. 4, pp. 201-214, 1978.
- [26] B. Julesz, "*Textons, the elements of texture perception, and their interactions*," Nature, vol. 290, pp. 91-97, Mar. 1981.
- [27] 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.
- [28] "*Coding of moving pictures and associated audio for digital storage at up to about 1.5 Mbits/s*," ISO/IEC 11172-2: Video, November 1999.
- [29] G. K. Wallace, "*The JPEG still picture compression standard*," IEEE Trans. on consumer electronics, December 1999
- [30] *Digital compression and coding of continuous-tone still images, Part 1, Requirements and guidelines*. ISO/IEC JTC1 Draft international standards 10918-1, November 1991.
- [31] *Digital compression and coding of continuous-tone still images, Part 2, Compliance testing*. ISO/IEC JTC1 Committee draft 10918-2, December 1991.
- [32] *Lossless and near-lossless coding of continuous tone still images (JPEG-LS)*, ISO/IEC JTC 1/SC 29/WG1 FCD 14495 – public draft, 1997

- [33] MPEG-4 Standards -MPEG97/N1796 - ISI/IEC JTC 1/SC29/WG11, July 1997.
- [34] M. Sevenier, E.S Jang, *An introduction to the MPEG-4 animation framework eXtension* Bourges- Circuits and Systems for Video Technology, IEEE Transactions on Volume 14, Issue 7, July 2004 Page(s):928 - 936
- [35] ISO/IEC 14496-21, Coding of audio-visual objects, Part 21: MPEG-J Graphics Framework extension (GFX).
- [36] R.M. Haralick, *Statistical image texture analysis*. In *Handbook of Pattern Recognition and Image Processing*, volume 86, pages 247–279. Academic Press, 1986.
- [37] L. VanGool, P. Dewaele, A. Oosterlinck. *Texture analysis anno 1983*. *Computer Vision, Graphics, and Image Processing*, 29(3):336–357, March 1985.
- [38] G. Smith. *Image Texture Analysis using Zero Crossings Information*. PhD thesis, Department of Computer Science and Electrical Engineering, University of Queensland, St Lucia 4072, Australia, 1998.
- [39] A.R. Rao. *A Taxonomy for Texture Description and Identification*. Springer-Verlag, 1990.
- [40] A. Turing. *The chemical basis of morphogenesis*. *Philosophical Transactions of the Royal Society (B)*, 237:37–72, 1952.
- [41] 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).
- [42] 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.
- [43] H. Meinhardt, *Models of Biological Pattern Formation*, Academic Press, London, 1982.

- [44] A. Witkin and M. Kass. *Reaction-diffusion textures*. In T.W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 299–308, July 1991.
- [45] G. Turk, *Generating textures for arbitrary surfaces using reaction-diffusion*. In *Computer Graphics (SIGGRAPH '91 Proceedings)* (July 1991), T. W. Sederberg, Ed., vol. 25, pp. 289–298.
- [46] K. Perlink, *An image synthesizer*. In *Computer Graphics (SIGGRAPH '85 Proceedings)* (July 1985), B. A. Barsky, Ed., vol. 19, pp. 287–296.
- [47] S. P. Worley, *A cellular texture basis function*. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 291–294. ACM SIGGRAPH, Addison Wesley, August 1996.
- [48] K.W. Fleischer, D. Laidlaw, B. Currin, A. Barr. *Cellular texture generation*. *Proceedings of SIGGRAPH 95*, pages 239–248, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California
- [49] K. W. Fleischer, Alan H. Barr, *A simulation test bed for the study of multicellular development: The multiple mechanisms of morphogenesis*. In *Artificial Life III*. Addison-Wesley, 1994.
- [50] W. Becket, N. Badler, *Imperfection for realistic image synthesis*. *Journal of Visualization and Computer Animation* 1, 1 (Aug. 1990), 26–32.
- [51] S. Hsu, T. Wong, *Simulating dust accumulation*. *IEEE Computer Graphics and Applications*, 15, 1 (Jan. 1995), 18–22.
- [52] G. Miller, *Efficient algorithms for local and global accessibility shading*. In *Computer Graphics Proceedings (1994)*, Annual Conference Series, ACM SIGGRAPH, pp. 319–326.
- [53] P. Hanrahan, W. Krueger, *Reflection from layered surfaces due to subsurface scattering*. In *Computer Graphics Proceedings (1993)*, Annual Conference Series, ACM SIGGRAPH, pp. 165–174.

- [54] J. Dorsey, P. Hanrahan. *Modeling and rendering of metallic patinas*. In *Computer Graphics Proceedings, Annual Conference Series*, pages 387–396. ACM SIGGRAPH, August 1996.
- [55] J. Dorsey, E. Edelman, J. Legakis, H.W. Jensen, H.K. Pedersen, “*Modelling and rendering of weathered stone*”, *Proceeding of SIGGRAPH '99*, Pages 225-234, and August 1999.
- [56] A. Efros and W. T. Freeman, “*Image Quilting for Texture Synthesis and Transfer*”, *Proceedings of SIGGRAPH '01*, pages 341-346, Los Angeles, California, August, 2001
- [57] V. Kwatra, A. Schödl, I. Essa, Greg Turk, A. Bobick, *Graphcut Textures: Image and Video Synthesis Using Graph Cuts*, *ACM Transactions on Graphics*, vol. 22, 3 (2003), pp 277 286. (*SIGGRAPH 2003 proceedings*).
- [58] R. L. Cook, *Shade trees*. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (1984), ACM Press, pp. 223-231.
- [59] J. Dorsey, A. Edelman, J. Legakis, H. Wann Jensen, H.K. Pedersen. *Modeling and Rendering of Weathered Stone*. *Proceedings of SIGGRAPH 99*, 225—234
- [60] K. Popat, R.W. Picard. *Novel cluster-based probability model for texture synthesis, classification, and compression*. In *Visual Communications and Image Processing*, pages 756–68, 1993.
- [61] D.J. Heeger, J.R. Bergen. *Pyramid-Based texture analysis/synthesis*. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 229–238. ACM SIGGRAPH, AddisonWesley, August 1995.
- [62] R. Paget, I.D. Longstaff. *Texture synthesis via a noncausal nonparametric multiscale Markov random field*. *IEEE Transactions on Image Processing*, 7(6):925–931, June 1998.
- [63] J. S. De Bonet, *Multiresolution sampling procedure for analysis and synthesis of texture images*. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 361–368. ACM SIGGRAPH, AddisonWesley, August 1997.

- [64] S. Zhu, Y. Wu, D. Mumford, *Filters, random fields and maximum entropy (FRAME) - towards a unified theory for texture modeling*. *International Journal of Computer Vision*, 27(2):107–126, 1998.
- [65] E. Simoncelli, J. Portilla. *Texture characterization via joint statistics of wavelet coefficient magnitudes*. In *Fifth International Conference on Image Processing*, volume 1, pages 62–66, October 1998.
- [66] A. Efros, T. K. Leung, *Fast Texture synthesis by tree structured vector quantization sampling*. In *International Conference on Computer Vision*, pages 1033–1038, Corfu, Greece, September 1999.
- [67] S.C. Zhu, X.W. Liu and Y.N. Wu, "*Exploring Julesz Ensembles by efficient Markov Chain Monte Carlo---towards a trichromacy theory of texture*", *IEEE Trans. PAMI*, vol 22, No. 6, pp554-569, 2000.
- [68] M. Ashikhmin, "*Synthesizing Natural Textures*" *The proceedings of 2001 ACM Symposium on Interactive 3D Graphics*, Research Triangle Park, North Carolina March 19-21, pp. 217-226
- [69] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, D. Salesin. *Image Analogies*, SIGGRAPH 2001 Conference Proceedings
- [70] S. Zelinka, M. Garland, *Towards Real-Time Texture Synthesis with the Jump Map*, In *Proceedings of the Thirteenth Eurographics Workshop on Rendering Techniques (2002)*, Eurographics Association, pp. 99-104.
- [71] X. Tong, J. Zhang, L. Liu, X. Wang, B.N. Guo, H.Y. Shum, *Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces*, Siggraph 2002, Pages: 665 - 672
- [72] Y. Q. Xu, B. N. Guo, H.Y. Shum, "*Chaos Mosaic: Fast and Memory Efficient Texture Synthesis*", MSR TR-2000-32, April, 2000.
- [73] L. Ling, C. E. Liu, Y. Xing, B. Guo, H. Shum. "*Real Time Texture synthesis by patch based Sampling*", *ACM Transaction on Graphics*, Vol 20, No 3, July 2001, pp.127-150.

- [74] R. Szeliski and H.Y. Shum. *Creating full view panoramic mosaics and environment maps*. Proceedings of SIGGRAPH 97, pages 251–258, August 1997.
- [75] A. Nealen, M. Alexa, *Hybrid Texture Synthesis*. proceedings of the Eurographics Symposium on Rendering 2003 (14th Eurographics Workshop on Rendering), pages 97-105 ,2003
- [76] Y. Boykov, O. Veksler, R. Zabih, *Fast approximate energy minimization via graph cuts*. In *International Conference on Computer Vision*, 377–384. 1999.
- [77] M. Cohen, J. Shade, S.Hiller, O. Deussan, *Wang Tile for image and texture generation*. SIGGRAPH 2003 conference proceeding, Volume 22 , Issue 3 (July 2003), Pages: 287 - 294 .
- [78] B. Gr"unbaum, G.C. Shepard, *Tiling and patterns*, W.H. Freeman and company, New York, 1986.
- [79] P. Heckbert, *Survey of Texture Mapping*, IEEE Computer Graphics and Applications, Nov. 1986
- [80] E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces*, PhD thesis, Dept. of CS, U. of Utah, Dec. 1974.
- [81] J.F. Blinn, "*Simulation of wrinkled surfaces*", Computer Graphics 11, ACM SIGGRAPH annual conference series, 1978, pp 286-292.
- [82] N. Max, "*Horizon mapping: shadows for bump-mapped surfaces*", The Visual Computer 4, 1988, pp. 109-117.
- [83] K. Perlin, "*Hypertextures*", Computer Graphics 23(3), ACM Siggraph annual Conference Series, July 1989, pp. 253-262.
- [84] J. Kajiya, T. Kay, "*Rendering Fur with three dimensional textures*", Computer Graphics 23(3), ACM Siggraph annual Conference Series, July 1989, pp. 271-298.
- [85] K.W. Fleischer, D.H. Laidlaw, B.L. Currin, A.H. Barr, "*Cellular Texture Generation*", Computer Graphics 29, ACM Siggraph annual Conference Series, 1995, pp. 239-248.

- [86] J.M. Dischler, D. Ghazanfarpour, "*Interactive Image-Based Modeling of Macrostructured Textures*", IEEE Computer Graphics and Applications 19(1), 1999, pp. 66-74.
- [87] G. Turk, *Texture Synthesis on Surfaces*, SIGGRAPH 2001, pp. 347-354.
- [88] L.Y. Wei and M. Levoy, *Texture synthesis over arbitrary manifold surfaces*, SIGGRAPH 2001, Computer Graphics Proceeding Pages 355-360, 2001.
- [89] G. Turk, *Re-tiling polygonal surfaces*, SIGGRAPH 1992, pp. 55-64, 1992.
- [90] L. Ying, A. Hertzmann, H. Biermann, D. Zorin, *Texture and shape synthesis on surfaces*. In 12th Eurographics Workshop on Rendering, pages 301--312, June 25-27 2001. 4
- [91] E. Praun, A. Finkelstein, H. Hoppe, *Lapped Textures*. Siggraph 2000, Computer Graphics Proceedings, 465-470, 2000.
- [92] C. Soler, M.P. Cani, A. Angelidis, *Hierarchical Pattern Mapping*. SIGGRAPH 2002, Conference Proceedings, 673-680, 2002.
- [93] F. Neyret, M.P. Cani, *Pattern-based texturing revisited*. Proceedings of SIGGRAPH 99 (August), 235-242, 1999
- [94] T. Chang, C.C. J. Kuo, "*Texture analysis and classification using tree-structured wavelet transform*," IEEE Trans. Image Processing, vol. 2, no. 4, pp. 429-441, Oct. 1993.
- [95] M. Unser, "*Texture classification and segmentation using wavelet frames*," IEEE Trans. Image Processing, vol. 4, no. 11, pp. 1549-1560, Nov. 1995.
- [96] G. V. Wouwer, P. Scheunders, D.V. Dyck, "*Statistical texture characterization from wavelet representations*," IEEE Trans. Image Processing, vol. 8, no. 5, pp. 592-598, May 1999.
- [97] A. Mojsilovic, M.V. Popovic, D. M. Rackov, "*On the selection of an optimal wavelet basis for texture characterization*", IEEE Trans. Image Processing, vol. 9, no. 12, pp. 2043-2050, Dec.2000.

- [98] H. Choi, R. Baraniuk, "*Multiscale image segmentation using wavelet-domain hidden Markov models*," IEEE Trans. Image Processing, vol. 10, no. 9, pp. 1309-1321, September 2001.
- [99] T.I. Hsu, R. Wilson, "*A two-component model of texture for analysis and synthesis*," IEEE Trans. Image Processing, vol. 7, no. 10, pp. 1466-1476, Oct. 1998.
- [100] J. Zhang, D. Wang, Q. N. Tran, "*A wavelet-based multiresolution statistical model for texture*," IEEE Trans. Image Processing, vol. 7, no. 11, pp. 1621-1627, Nov. 1998.
- [101] M. J. Wainwright, E. P. Simoncelli, A. S. Willsky, "*Random cascades on wavelet trees and their use in analyzing and modelling natural images*," Applied and Computational Harmonic Analysis, vol. 11, no. 1, pp. 89-123, July 2001.
- [102] G. Fan, X.G. Xia, "*Wavelet-based Texture Analysis and Synthesis Using Hidden Markov Models*", IEEE Trans. Circuits and Systems, Part I, Vol. 50, No. 1, pp. 106-120, Jan. 2003
- [103] S. Daly, *A visual model for optimizing the design of image processing algorithms*. In Proc. of IEEE ICIP'94, volume II, pages 16-20, Austin, Texas, USA, November 1994.
- [104] M.P. Eckert, A.P. Bradley, *Perceptual quality metrics applied to still image compression*. Signal Processing, 70(3):177-200, November 1998.
- [105] B.A. Wandell, *Foundations of vision*. Sinauer Associates, Inc., Sunderland, Massachusetts, Stanford University edition, 1995.
- [106] S. Zeki, *A vision of the brain*. Blackwell scientific publications, Oxford, 1993.
- [107] G. W. Larson, H. Rushmeier, C. Piatko, "*A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes*", Technical Report LBNL-39882, Lawrence Berkeley National Laboratory, March 1997

- [108] J. A. Ferwerda, S. N. Pattaniak, P. Shirley, D. P. Greenberg, "*A Model of Visual Adaptation for Realistic Image Synthesis*", *Computer Graphics*, pp. 249-258, 1996.
- [109] J. L. Mannos, D. J. Sakrison, "*The Effects of a Visual Fidelity Criterion on the Encoding of Images*", *IEEE Transactions on Information Theory*, pp. 525-535, Vol. 20, No 4, (1974)
- [110] S. Gortler, P. Schröder, M. Cohen, P. Hanrahan, "*Wavelet Radiosity*", *Computer Graphics, Annual Conference Series (Siggraph'93 Proceedings)*, pp. 221-230, 1993.
- [111] D. Berman, J. Bartell, D. Salesin, "*Multiresolution Painting and Compositing*", *Computer Graphics, Annual Conference Series (Siggraph'94 Proceedings)*, pp. 85-90, 1994.
- [112] A. Finkelstein, D. Salesin, "*Multiresolution Curves*", *Computer Graphics, Annual Conference Series (Siggraph'94 Proceedings)*, pp. 261-268, 1994.
- [113] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsberry, W. Stuetzle, "*Multiresolution Analysis of Arbitrary Meshes*", *Computer Graphics, Annual Conference Series (Siggraph'95 Proceedings)*, pp. 173-182, 1995
- [114] L. Lippert, M. Gross, "*Fast Wavelet Based Volume Rendering by Accumulation of Transparent Texture Maps*", *Eurographics'95 Conference Proceedings*, *Computer Graphics Forum* Vol. 14, Nr. 3, pp. 431-443, 1995
- [115] C. Jacobs, A. Finkelstein, D. Salesin, "*Fast Multiresolution Image Querying*", *Computer Graphics, Annual Conference Series (Siggraph'95 Proceedings)*, pp. 277-286, 1995
- [116] Z. Liu, S. Gortler, M. Cohen, "*Hierarchical Spacetime Control*", *Computer Graphics, Annual Conference Series (Siggraph'94 Proceedings)*, pp. 35-42, 1994
- [117] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, W. Stuetzle, "*Interactive Multiresolution Surface Viewing*", *Computer Graphics, Annual Conference Series (Siggraph'96 Proceedings)*, pp. 91-98, 1996
- [118] E.J. Stollnitz, T. D. DeRose, D. H. Salesin, *Wavelets for Computer Graphics - Theory and Applications*, Morgan Kaufmann Publishers Inc., San Francisco, California, 1996

- [119] P. Schröder, W. Sweldens, M. Cohen, T. DeRose, D. Salesin, *Wavelets in Computer Graphics*, SIGGRAPH 96 Course Notes
- [120] C. K. Chui, *"An Introduction to Wavelets"*, Academic Press Inc., Boston, MA 1992
- [121] I. Daubechies, *"Ten Lectures on Wavelets"*, Vol. 61 of CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, PA 1992
- [122] Robi Polikar, Wavelet Tutorial,
<http://www.engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html> ;
- [123] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA: SIAM, 1992.
- [124] R. De Vore, B. Jawerth, B. Lucier, *"Images Compression through Wavelet Transform Coding."* *IEEE Trans. Information Th.* 38, 719-746, 1992.
- [125] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *"Wavelet Transforms."* §13.10 in *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd ed. Cambridge, England: Cambridge University Press, pp. 584-599, 1992.
- [126] G. Strang, *"Wavelet Transforms Versus Fourier Transforms."* *Bull. Amer. Math. Soc.* 28, 288-305, 1993.
- [127] S.G. Mallat, *A theory of multiresolution signal decomposition: the wavelet representation.* *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7), 674-693. (1989)
- [128] A.B. Watson, G.Y. Yang, J.A. Solomon, J. Villasenor, *Visibility of Wavelet Quantization Noise.* *IEEE Trans. of Image Processing*, 1997, vol. 6, p. 1164 – 1175.
- [129] D. Levicky, P. Foris, *Human Visual System Models in Digital Image Watermarking*, *Radio Engineering*, pp 38-43 Vol. 13, No. 4, December 2004.
- [130] W.B. Pennebaker, *"JPEG Technical Specification Revision 8,"* Working Document No. JTC1/SC2/WG10/JPEG-8-R8, Aug. 1990.
- [131] W.B. Pennebaker, J.L. Mitchell, *JPEG Still Image Compression Standard*, Van Nostrand Reinhold, New York, 1993.

- [132] H.A. Peterson, A.J. Ahumada, A.B. Watson, *An Improved Detection Model for DCT Coefficient Quantization*. In *Proc. SPIE Conf. Human Vision*, 1993, vol. 1913, p.191 - 201.
- [133] E.A.Edirisinghe, "Data Compression in Stereo Images and Video", PhD thesis, Loughborough University, UK, October 1999.
- [134] MATLAB –Image Processing Toolbox, Mathworks Inc. 1997
- [135] A. McAndrew, *Introduction to Digital Image Processing with MATLAB*, Thomson Course Technology, 2004
- [136] R.C. Gonzalez, R.E. Woods, S.L. Eddins, *Digital Image Processing Using MATLAB*, Prentice Hall, 2004
- [137] P. Baurke, Texture Library, <http://astronomy.swin.edu.au/~pbourke/texture/>
- [138] "Vis Tex's" texture library Vision Textures,
<http://vismod.media.mit.edu/vismod /imagery/VisionTexture/ vistex.html>
- [139] Brodatz Textures, gray scale texture library,
<http://www.ux.his.no/~tranden/brodatz.html>
- [140] J.S. De Bonet's texture collection
<http://www.debonet.com/Research/TextureSynthesis/>
- [141] J. Davis, *Mosaics of scenes with moving objects*. In *Proc. IEEE Conf. on Comp.Vision and Patt. Recog.*, 1998.
- [142] L. Ford, D. Fulkerson, *Flows in Networks*, Princeton University Press, 1962. 3.1
- [143] S.L. Kilthau, M. Drew, T. Moller, *Full search content independent block matching based on the fast fourier transform*. In *ICIP02*, I: 669–672. 2002
- [144] E.V. DeNardo, *Dynamic Programming: Models and Applications*, Courier Dover Publications, April 2003.
- [145] J.Jing, E.A.Edirisinghe, *Algorithm for compression of Stereo Image Pairs*. *Electronic Letters*, 1997. 33(12): p. 1034-1035.

- [146] ISO/IEC, Int. Standard DIS 10918, "*Digital compression and coding of continuous-tone still images.*"
- [147] S.Minami, A.Zakhor, "*An Optimization Approach for Removing Blocking Effects in Transform Coding*", IEEE Trans. Circuits & Systems for Video Technology, vol. 5, pp.64-82, April 1995.
- [148] G. Lakhani, N. Zong, "*Derivation of Prediction Equations for Blocking Artifact Reduction*", IEEE Trans. Circuits & Systems for Video Technology, vol. 9, No 3, pp.415-418, April 1999.
- [149] J.M. Shapiro, "*Embaddded image coding using zero trees of wavelet coefficients*", IEEE Trans. On Signal processing, Vol. 41, No. 12, December 1993, pp 3445-3462.
- [150] A. Efros, T. Leung, "*Texture synthesis by non-parametric sampling*". In International Conference for Computer Vision, Volume 2, pages 1033-1038, September 1999.
- [151] K.R. Rao, P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press Inc.(London) Ltd, September 1990.
- [152] P. Jacob, M. Jeannerod, *Ways of Seeing: The Scope and Limits of Visual Cognition* (Oxford Cognitive Science S.), Oxford University Press, October 2003.
- [153] A. Efros and T. Leung. *Texture synthesis by non-parametric sampling*. In *International Conference on Computer Vision*, volume 2, pages 1033–8, Sep 1999.
- [154] F.C. Crow, "*Summed-Area Tables for Texture Mapping*", *Computer Graphics*, (SIGGRAPH'84 Proceedings), Vol. 18, No. 3, July 1984, pp. 207-212.
- [155] L. A. Ferrari, Jack Sklansky, "*A Fast Recursive Algorithm for Binary-Valued Two-Dimensional Filters*", *Computer Vision, Graphics, and Image Processing*, Vol. 26, No. 3, June 1984, pp. 292-302.
- [156] K. Perlin, "Course Notes", *SIGGRAPH '85 State of the Art in Image Synthesis seminar notes*, July 1985.

- [157] J. Maillot, H. Yahia, A. Verroust, *Interactive texture mapping*. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH 93 Proceedings)*, volume 27, pages 27–34, August 1993.
- [158] H. K. Pedersen, *Decorating implicit surfaces*. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, pages 291–300. ACM SIGGRAPH, Addison Wesley, August 1995.
- [159] H. K. Pedersen, *A framework for interactive texturing operations on curved surfaces*. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, pages 295–302. ACM SIGGRAPH, Addison Wesley, August 1996.
- [160] E.A. Feibush, M. Levoy, R.L. Cook, “*Synthetic Texturing Using Digital Filters*”, *Computer Graphics*, (SIGGRAPH '80 Proceedings), Vol. 14, No. 3, July 1980, pp. 294-301.
- [161] M. Gangnet, D. Perny, P. Coueignoux, “*Perspective Mapping of Planar Textures*”, *Eurographics '82*, 1982, pp. 57-71, (slightly superior to the version that appeared in *Computer Graphics*, Vol. 16, No. 1, May 1982).
- [162] F. Neyret, M.P. Cani, *Pattern-based texturing revisited*. in *SIGGRAPH 99*. 1999.
- [163] Texture mapping: a new tool for molecular graphics, <http://www.pc.chemie.tu-darmstadt.de/research/molcad/texture.shtml>
- [164] A. Salomie, R. Deklerck, A. Munteanu, J. Cornelis. *The MeshGrid Surface Representation*. Vrije Universiteit Brussel, Department. ETRO-IRIS, Technical Report IRIS-TR-0082. [Online]. Available: http://www.etro.vub.ac.be/Publications/technical_reports.asp (2002)
- [165] The Ply file format, http://www.cc.gatech.edu/projects/large_models/ply.html
- [166] PLY - Polygon File Format (Also known as the Stanford Triangle Format), <http://astronomy.swin.edu.au/~pbourke/geomformats/ply/>
- [167] L.Y. Wei, M. Levoy, Fast texture synthesis using tree-structured vector quantization, *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp.479-488, July 2000.

- [168] E.W. Weisstein, "Voronoi Polygon", From *MathWorld*—A wolfram web resource, <http://mathworld.wolfram.com/voronoipolygon.html>.
- [169] Allen B. Poirson & Brian A. Wandell, Pattern-color separable pathways predict sensitivity to simplified coloured patterns, Vision Research, 1995