Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

EXAMPLE BASED TEXTURE SYNTHESIS AND QUANTIFICATION OF TEXTURE QUALITY

by
Chandralekha De

Textures have been used effectively to create realistic environments for virtual worlds by reproducing the surface appearances. One of the widely-used methods for creating textures is the example based texture synthesis method. In this method of generating a texture of arbitrary size, an input image from the real world is provided. This input image is used for the basis of generating large textures. Various methods based on the underlying pattern of the image have been used to create these textures; however, the problem of finding an algorithm which provides a good output is still an open research issue. Moreover, the process of determining the best of the outputs produced by the existing methods is a subjective one and requires human intervention. No quantification measure exists to do a relative comparison between the outputs. This dissertation addresses both problems using a novel approach. The dissertation also proposes an improved algorithm for image inpainting which yields better results than existing methods.

Firstly, this dissertation presents a methodology which uses a HSI (hue, saturation, intensity) color model in conjunction with the hybrid approach to improve the quality of the synthesized texture. Unlike the RGB (red, green, blue) color model, the HSI color model is more intuitive and closer to human perception. The hue, saturation and intensity are better indicators than the three color channels used in the RGB model. They represent the exact way, in which the eye sees color in the real world.

Secondly, this dissertation addresses the issue of quantifying the quality of the output textures generated using the various texture synthesis methods. Quantifying the quality of the output generated is an important issue and a novel method using statistical measures and a color autocorrelogram has been proposed. It is a two step method; in the first

step a measure of the energy, entropy and similar statistical measures helps determine the consistency of the output texture. In the second step an autocorelogram is used to analyze color images as well and quantify them effectively.

Finally, this disseratation prsesents a method for improving image inpainting. In the case of inpainting, small sections of the image missing due to noise or other similar reasons can be reproduced using example based texture synthesis. The region of the image immediately surrounding the missing section is treated as sample input. Inpainting can also be used to alter images by removing large sections of the image and filling the removed section with the image data from the rest of the image. For this, a maximum edge detector method is proposed to determine the correct order of section filling and produces significantly better results.

EXAMPLE BASED TEXTURE SYNTHESIS AND QUANTIFICATION OF TEXTURE QUALITY

by
Chandralekha De

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

May 2012

APPROVAL PAGE

EXAMPLE BASED TEXTURE SYNTHESIS AND QUANTIFICATION OF TEXTURE QUALITY

Chandralekha De

Dr. Frank Y. Shih, Dissertation Advisor — Date
Professor of Computer Science, NJIT

Dr. Vincent Oria, Committee Member — Date
Associate Professor of Computer Science, NJIT

Dr. Dimitrios Theodoratos, Committee Member — Date
Associate Professor of Computer Science, NJIT

Dr. Zhi Wei, Committee Member — Date
Assistant Professor of Computer Science, NJIT

Dr. Yun Q. Shi, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Chandralekha De

Degree: Doctor of Philosophy

Date: May 2012

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2012
- Bachelor of Technology in Computer Science Engineering and Applications,
  National Institute of Technology, Rourkela, Orissa, India, 2004

Major: Computer Science

Presentations and Publications:

C. De and F. Y. Shih. Autocorrelogram Based Quantification of Enhanced Hybrid Texture Synthesis. (under preparation).

C. De and F. Y. Shih. Improved Image Inpainting using Maximum Value Edge Detector. *Southwest Symposium on Image Analysis and Interpretation*, Santa Fe, New Mexico, USA, April 2012.

C. De and F. Y. Shih. Quantification and Relative Comparison of Synthesized Texture. *International Conference on Image Processing, Computer Vision, Pattern Recognition*, Las Vegas, Nevada, USA, July 2011.

For those who had faith in me when mine faltered.

ACKNOWLEDGMENT

I would like to thank everyone who helped me these last five years and made my dream of attaining a PhD a reality. First I would like to thank my advisor Dr. Frank Y. Shih for his constant support and encouragement. He gave me enough leeway and time to start my research and had faith in the fact that I could accomplish significant results in my work.

I am extremely grateful to my committee members Dr. Vincent Oria, Dr. Dimitrios Theodoratos, Dr. Zhi Wei and Dr. Yun Q. Shi for agreeing to serve on my commmittee and for their valuable input and support.

I would like to thank Dr. David Nassimi for providing me with the departmental teaching assistantship these five years. I would also like to extend my thanks to Dr. George Olsen and Ms. Angel Bell for all the help they provided during my stay here.

I am hugely indebted to my parents and sister for supporting all my life choices. They stood by me every step of the way and let me make all my choices, some good, some not so good. They convinced me that I could do anything I set my heart on and motivated me to move forward no matter what.

I am grateful to all my friends both at NJIT and across the globe for their help and encouraging words which helped me get through the entire period as a graduate student. I couldn't have come so far without them.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Introduction to Textures

Texture refers to properties that represent the surface or structure of an object [1]. It can be obtained from a variety of sources, such as hand-drawn pictures or scanned photographs. Hand-drawn pictures can be aesthetically pleasing, but it is hard to make them photo-realistic. In addition to that, it requires artists and their level of expertise, and could be difficult for ordinary people to come up with good texture images. In industry, there is a specific profession in game and film studios named texture artist, whose job is to produce high quality textures. To recreate the surface appearance of objects textures can be applied to objects by a technique called texture mapping [2, 3]. Scanned images, however, may be of inadequate size or quality, e.g., containing non-uniform lighting, shadows, or geometry; and could lead to visible seams or repetition if directly used for texture mapping.

Example based texture synthesis is a highly prevalent technique for creating textures of arbitrary sizes. This method of texture synthesis takes an image from the real world and uses that to generate a larger image such that the new generated image appears to have the same underlying structure as the input image. This is a two-step method comprising of analysis and synthesis. Analysis involves examination and determination of the underlying structure of the texture and establishing a mathematical pattern for it. Synthesis on the other hand involves using the data generated by the analysis phase to recreate a texture with the same underlying pattern as the sample image.

Example based texture synthesis algorithms can be broadly classified as pixel based [4, 5, 6] and patch based [7]. Pixel based texture synthesis algorithms introduce blurs in the texture, but helps maintain a consistent texture appearance. Patch based texture synthesis

algorithms on the other hand, maintain the global structure of the texture, but sometimes introduce visual artifacts along the edges in the texture.

Textures generated by all the existing algorithms work well on some categories of textures and not so well on others, determining the best output from the algorithms still requires human intervention for the most part and is still an open field for research. The work of Chieh [8] introduced a method; however it is applicable only to a single class of textures, the near regular textures.

Figure 1.1 Image without texture: A green patch representing grass.

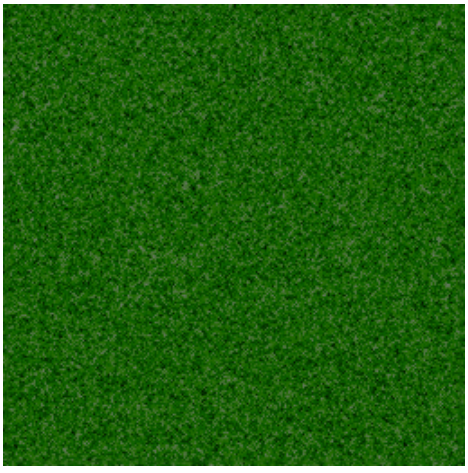


Figure 1.2 Image with texture: A grass texture used. The quality of output is significantly improved using the texture allowing for a more realistic look and feel.

Figure 1.1 and Figure 1.2 shows two images representing greenery. Figure 1.1 shows a flat image with green color whereas Figure 1.2 shows grass rendered using a texture image. The second image is a significant improvement from the first in terms of appearing realistic. It uses an image of grass captured with a traditional camera and mapped onto a surface. In a computer generated image of a field depicting grass the second image is more convincing and visually appealing as compared to the first image. It is closer to how grass is perceived in the actual world and not merely a green patch representing it.

The role texture plays in creating computer generated images cannot be over emphasized. It helps give all surfaces and models in a scene more depth and realistic feel. Textures have been used widely in multimedia [9], gaming [10, 11] and animation [10, 11] to create a better virtual world. However different types of textures have been used in each of these applications to create a realistic feel.

The image in Figure 1.3 shows four images from the MIT VisTex Image database [12] representing flowers, fabric, food and grass. These images have been widely used in the study and analysis of textures and texture synthesis algorithms.

1.1.1 Types of Textures

Textures used in computer graphics comprise of different structures and patterns and layouts. Textures can be used for rendering surfaces of wood, brick, stone, water, fire, sky and various other objects and surfaces. The images used for synthesis vary in many ways both structurally and in color.

Texture can be broadly classified into the following five subtypes based on their structure and appearance

a) Regular textures are textures with periodic patterns where the color/shape of all texels (texture elements) are repeated in equal intervals along two linearly independent directions. Regular textures are defined as wallpaper-like, congruent 2D tiling whose structural regularity can be completely characterized by the 17 wallpaper groups [13, 14]. The

Figure 1.3 A subset of texture images from the VisTex image database. The images from the MIT VisTex database has been widely used in texture synthesis algorithms and texture analysis.

underlying lattice structure of each regular texture can thus be represented and generated by a pair of linearly independent translations.

b) Near regular textures are statistical departures of regular textures along different directions. They can be defined as textures which are regular globally but are random locally. A near-regular texture deviates geometrically and photometrically from a regular congruent tiling. Although near-regular textures are ubiquitous in the man-made and natural world, they present computational challenges for state of the art texture analysis and synthesis algorithms [15].

c) Irregular textures are textures which have no periodicity in them. They comprise of texels which cannot be isolated and no single texel can help recreate the entire texture.

These textures are however not as random as stochastic textures, discussed next, but are much harder to synthesize on larger scale on account of the fact that the underlying structure varies a lot.

d) Stochastic textures are textures which appear like noise. This type of texture is generally used to represent features like sand, smoke, fire and similar features. In most cases synthesizing stochastic texture involves randomly choosing color values based on parameters like minimum brightness, average colour or maximum contrast. They provide low quality output but suffice when used for such textures.

e) Near stochastic textures are textures which appear similar to a stochastic texture but comprises less of noise.

1.1.2 Homogeneous and Non Homogeneous Textures

Textures can be classified as homogeneous or non homogeneous based on their details. Typically homogeneous textures consist of exclusively of single texels. Specifically, homogeneous texture contains ideal repetitive structures, and such uniformity produces idealised patterns. Weak homogeneity involves local spatial variation in texture elements or their spatial arrangement, which leads to more or less violates the precise repetitiveness. A non homogeneous texture mostly refers to an image where repetition and spatial self-similarity are absent. Since spatial homogeneity is considered an essential property of a texture, a non homogeneous image is not treated as a "texture" in this dissertation.

1.1.3 Parametric and Non Parametric Textures

Textures can also be classified in a different manner based on the way they can be synthesized. Parametric and non parametric textures are two broad classifications of textures based on the manner of synthesis. The parametric method is the use of oriented linear kernels at multiple spatial scales for image analysis and representation. The widespread use of such kernels as descriptions of early visual processing in mammals inspired a large

number of models for texture classification and segmentation. The development of wavelet representations, which are based on such kernels, have revolutionized signal and image processing. A number of inspirational results in texture synthesis are based on multi-scale decompositions [16, 17].

1.2 Texture Synthesis

Texture Synthesis is the mechanism of building large textures of arbitrary shapes using a small input image so that the new texture appears to be a part of the existing texture.

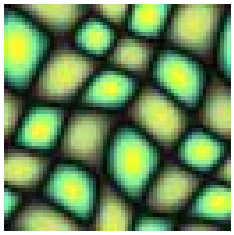


Figure 1.4 An example image showing a sample input texture to be used for synthesis.

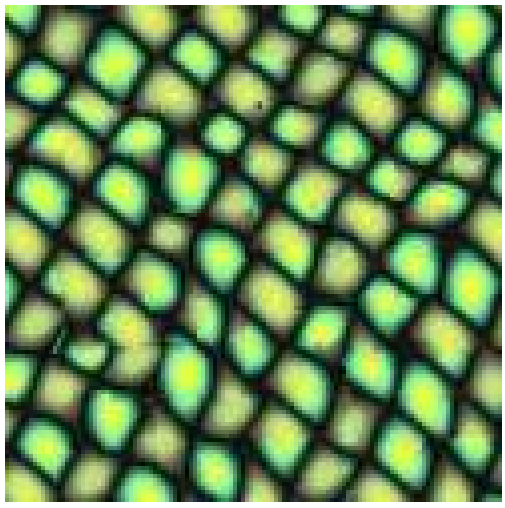


Figure 1.5 A synthesized texture generated using provided sample input texture.

1.2.1 Motivation

The problem is to generate an output texture which is as close to the input image structurally and tonally without showing visible seams or artifacts. The new image should look like it has been created by a continuation of the original image. It should have the same underlying structure as the input image and the same characteristics. The problem is significant because it helps create large textures of arbitrary size with a small sampler image. Scaling the input image will not have the same effect. It will compromise the quality of the output and also scale the texels which forms the basis of the image as opposed to using the texels to create a large output. Figure 1.4 shows an original texture image which is provided as an input to generate the larger image in Figure 1.5. The Efros and Freeman [18] method has been used to generate the texture in Figure 1.5.

1.2.2 Markov Random Field

Although a variety of texture models have been proposed throughout the history, so far the most successful model for graphics applications is based on Markov Random Field (MRF).

The MRF model is a 2 dimensional extension of the single dimensional Markov Chain, inspired by the Shannon's work on modelling the English language using n-grams [19]. The Markov Chain is a sequence of random variables with the Markov Property; given the present state, the past and the future states are independent of each other. If the system is at state X_i and moves to a new state X_j the subsequent state after X_j depends only on X_j and not on X_i or any earlier state [20].

Markov Random Field methods model a texture as a realization of a local and stationary random process. That is, each pixel of a texture image is characterized by a small set of spatially neighboring pixels, and this characterization is the same for all pixels. Imagine that a viewer is given an image, but only allowed to observe it through a small movable window. As the window is moved the viewer can observe different parts of the image. The image is stationary if, under a proper window size, the observable portion always appears

similar. The image is local if each pixel is predictable from a small set of neighboring pixels.

Based on this Markov-Random-Field model, the goal of texture synthesis can be formulated as follows: given an input texture, synthesize an output texture so that for each output pixel, its spatial neighborhood is similar to at least one neighborhood at the input. The size of the neighborhood is a user-specifiable parameter and is proportional to the feature size of the texture patterns. Due to the MRF assumption, similarity of local neighborhoods between input and output will guarantee their perceptual quality similarity as well. In addition to this quality concern, the texture synthesis algorithm should also be efficient and controllable.

1.2.3 Texture Synthesis over Surface

Turk's [21] and Wei and Levoy's [5] (WLS) surface synthesis methods both densely tessellate the input mesh using Turk's re-tiler [22] and then perform a per-vertex color synthesis. These two approaches are very similar (both use a multiresolution mesh hierarchy), yet have three quite significant differences. (1) WLS uses both random and symmetric vector fields, whereas Turk's always uses a user defined, smooth vector field. (2) Turk's uses a sweeping order derived from the smooth vector field for vertex traversal, WLS visits the mesh vertices in random order. (3) Turk's uses surface marching to construct the mesh neighborhood, while WLS performs flattening and resampling of the mesh. Results of the two methods are comparable in quality.

Ying et al. [23] wish to overcome the drawbacks of surface marching methods (such as [21]): (1) the sampling pattern is not guaranteed to be even in the presence of irregular geometry, (2) the 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-texel using a texture atlas of the polygonal mesh (a collection of rectangular domains U_i on

which the surface is smoothly parameterized) and a common planar domain, the chart V , from which neighborhood sample positions in the domains U_i are gathered. These positions in the U_i then correspond to the neighborhood on the original surface along a previously defined, orthogonal unit tangent vector field. They apply both Wei/Levoy [5] and Ashikhmin [4] per-pixel synthesis strategies with convincing results [23].

Praun et al.s Lapped Textures [24] extend the chaos mosaic [25] 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 parametrization 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.

1.3 Texture Quantification

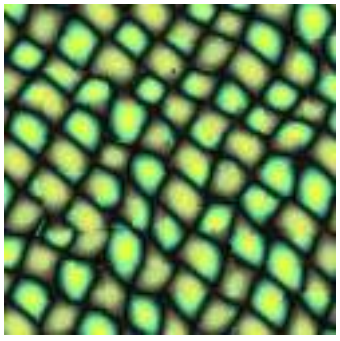


Figure 1.6 Synthesized texture: Image showing a texture synthesized using the Efros Freeman method.

Figure 1.6 and Figure 1.7 show two textures synthesized using two different algorithms [18, 6]. The results of the two algorithms are different and the choice between the

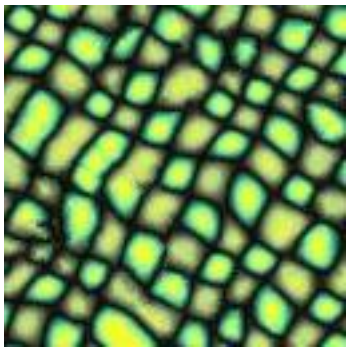


Figure 1.7 Synthesized texture: Image showing a texture synthesized using the Efros Leung method.

two algorithms becomes a subjective one. This problem is interesting and more challenging than just studying images as they involve comparing the input texture with all output textures generated.

It involves studying which texture has consistently retained the global and local features of the input image, which image consistently recreates the structure of the input image and does not cause seams or noise to appear. As is visible in the above figures some algorithms produce significantly better output than the other and at such points mere inspection of the output is good enough to determine the better of the two. However in many cases this decision is not easy when the margin of error is small.

1.4 Image Inpainting

Digital images are subject to the introduction of noise and corruption. Removing the noise and restoring images to their original format is a widely researched topic and many different approaches have been adopted to achieve this goal. Image inpainting is an extension of this issue which deals with filling corrupt section in images or even filling a large section of the image which has been removed. It allows an end user to pick an image which has speckles [26, 27, 28] or large sections obfuscated and restore it to its original format [29].

As explained by Bertalimio and Sapiro [30] the modification of images in a way that is non-detectable for an observer who does not know the original image is a practice as old as artistic creation itself. Medieval artwork started to be restored as early as the Renaissance, the motives being often as much to fill in any gaps [31, 32]. This practice is called retouching or inpainting. The object of inpainting is to reconstitute the missing or damaged portions of the work, in order to make it more legible and to restore its unity [32].

Figure 1.8 Corrupt image with speckles. Courtesy: G. Sapiro, Geometric Partial Differential Equations and Image Processing, Cambridge University Press, January 2001.

Image inpainting focuses on automatically repairing the missing regions in a visually plausible way. In cases where small sections have been removed image inpainting can help restore the image to its original state, however in cases where an entire object or person has been removed from the original image inpainting allows restoring the image in such a manner that the restored image appears to have never had the removed object. This is useful when undesired objects are present in an image and need to be removed.

Figure 1.9 and Figure 1.10 shows an example where a large section of the image has been removed and then restored using the surrounding neighborhood.

Figure 1.9 Image with section removed. People in the image have been manually removed from the image leaving a void in the image.

1.5 Overview

This dissertation deals with these three problems and tries to address them. The following chapters dwells deeper into each of the topics. Chapter 2 talks about using a HSI color model for texture synthesis using the hybrid texture synthesis method, Chapter 3 deals with quantification of the textures generated using the various algorithms. Chapter 4 deals with the last topic of image inpainting and our solution using a maximum edge detector to achieve better results. Chapter 5 provides the conclusions and the future work.

Figure 1.10 Image with section restored. Removed section of the image is refilled using the texture of the neighboring region.

CHAPTER 2

TEXTURE SYNTHESIS

2.1 Synthesis

The goal of texture synthesis can be stated as follows: given a texture sample, generate a new texture that, when perceived by a human observer, appears to be generated by the same underlying process.

Example based texture synthesis algorithm has been one of the most widely used and explored method for texture synthesis. This method accepts a real world image and tries to generate a texture of arbitrary size from the input texture. The goal is to make the new texture perceptually similar to the input structure. The global features of the texture are retained and it appears to have the same underlying pattern as the input texture. This method of texture synthesis is an alternative way to create textures. As mentioned in Chapter 1 the process of texture synthesis comprises of the two sub-processes of analysis and synthesis.

Example based texture synthesis algorithms for the most part uses a pixel based or patch based approach. The pixel based approach generates the new texture one pixel at a time in scan-line order whereas patch based approach generates textures using a collection of pixels called patches. Hybrid texture synthesis algorithm [33] combines these two methods to improve the quality of the output texture.

2.1.1 Pixel Based Algorithm

Pixel-based algorithms synthesize a new texture pixel by pixel [5], with the value of each new pixel determined by its local neighborhood. A very simple and yet elegant algorithm works as follows. First, the output texture is seeded from a portion of the input, e.g. a 3 $\times$ 3 region. Starting from this seed, it generates new pixels outward in a spiral fashion.

For each output pixel under investigation, a fixed-size window (with size picked by the user) centering on the pixel is intersected with already synthesized pixels. This collection of intersection pixels is then searched throughout the input image to find the most similar N candidates. Finally, a random candidate is drawn uniformly from this candidate set, and the input pixel centered on this neighborhood is copied to the target pixel. This process is repeated for each not-yet-synthesized output pixel in a spiral fashion until the entire output is synthesized.

2.1.2 Patch Based Algorithm

The quality and speed of pixel-based approaches can be improved by synthesizing patches rather than pixels. Intuitively, when the output is synthesized by assembling patches rather than pixels from the input, the quality ought to improve as pixels within the same copied patch ought to look good with respect to each other. In pixel-based synthesis, the output is synthesized by copying pixels one by one from the input. The value of each output pixel is determined by neighborhood search to ensure that it is consistent with already synthesized pixels. Patch-based synthesis is very similar to pixel-based synthesis, except that instead of copying pixels, we copy patches. To ensure output quality, patches are selected according to its neighborhood, which, just like in pixel-based synthesis, is a thin band of pixels around the unit being copied (being pixel in pixel-based synthesis or patch in patch-based synthesis).

Cohen et al. [34] presents a tile-based texturing algorithm that shares similar philosophy to previous patch-based algorithms. Similar to patch-based synthesis, Cohen et al. [34] used tiles as the basic units for synthesis. However, instead of using arbitrarily shaped patches and handle patch overlaps via blending or cutting, Cohen et al. [34] used a special kind of tile called Wang Tiles which have no overlap with each other and whose contents are constructed carefully so that the texture patterns are continuous cross tile edges that

have compatible colors. Once this tile set is constructed, an arbitrarily large texture can be constructed by assembling these tiles so that adjacent tiles share identical edge colors.

2.1.3 Other Techniques

In addition to the the pixel and patch based approach other techniques to generate texture synthesis exist. The work of Kwatra et. al [35] suggests an alternative method for synthesizing texture. Like the pixel based gorithms, texture optimization synthesizes an output texture in the units of pixels. However, unlike previous pixel-based methods which synthesize pixels one by one in a greedy fashion, this technique considers them all together, and determine their values by optimizing a quadratic error energy function. The error function is determined by mismatches of input and output neighborhoods, so minimizing this function leads to better output quality [36].

2.2 Previous Work

Non parametric texture synthesis specifically example based texture synthesis has long been a widely used method for generating textures in computer graphics.

One of the earliest and significant work was that done by Popat [37]. This texture synthesis algorithm can be best classified as a nonparametric Markov chain synthesis algorithm. The basis of the algorithm is to order the pixels and then synthesize a new pixel from a nonparametric representation of the conditional probability function derived from samples of the input texture. This algorithm proposed to use stochastic sampling of the conditional probability function and also to compress the conditional probability function via a set of Gaussian kernels. This compression allowed for fast look ups, but limited the neighbourhood order that could be successfully modelled. The one flaw in this algorithm's approach was that it was causal. That is, the synthesis was performed in a sequential sequence starting from a "seed" and gradually moving 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 get lost in a space where only garbage is produced. This severely limited the algorithm from synthesising large areas of texture.

The problem with the sequential approach of Popat's [37] algorithm is solved by the work of Paget and Longstaff [38]. This algorithm 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".

The noncausal nature of Paget and Longstaff [38] algorithm not only could synthesise a texture to any size, but also better supported the idea that arbitrary textures could be modelled by a Markov random field. The number and range of textures that could be synthesised by their scheme also showed that nonparametric Markov random field models were the models of choice for natural textures that included both stochastic and deterministic properties. However, the one draw back to their scheme was speed.

Another important algorithm was that proposed by Heeger and Bergen [39]. This paper was one of the first papers to show the possibility of synthesising colour textures to the graphics community. They did it using a combination of Laplacian and Steerable pyramids to deconstruct an input texture. The histograms from each of the pyramid levels was used to reconstruct a similar pyramid. Unfortunately the deconstruction was not orthogonal, which meant that the algorithm had to use an iterative approach of matching the histograms and expanding and reducing the pyramid.

A novel algorithm was suggested by Zhu [40] which amalgamated the filter technology and MRF technology to produce the so-called FRAME model. It did this by comparing the histograms of both the filter responses from the original texture and that of the synthetic texture. The synthetic texture was then continually updated with respect to an evolving MRF probability function, that was defined with respect to modulated filter responses. The modulation was defined with respect to differences between the expected filter response

using the current MRF probability function and the filter response from the original texture. 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, but the synthesis/modelling process was very slow.

A similar technique to that of Heeger and Bergen [39] was suggested by Simoncelli and Portilla [41], but where Heeger and Bergen [39] updated the complete filter response using histogram equalisation, Simoncelli and Portilla [41] updated each point in the pyramid of filter responses with respect to the correlations using a method similar to projection onto convex sets (POCS). They did this by finding an orthogonal projection from the filter response of the synthetic texture to that of the original. After the projection of all filter responses, the wavelet pyramid was collapsed, further projection was performed, and then the pyramid was reconstructed. This iteration continued until a convergence was reached. Simoncelli and Portilla [41]found that only a few minutes of processing time was required to produce reasonable results. However they still had some failures, and had difficulty maintaining fidelity with textures containing structure.

The work of Efros and Leung [6] gave a rendition of Popat's [37] work. However in their case they explicity used the exhaustive nearest neighbour searching. They also initialised 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 [37], which had inherent stability problems.

In subsequent work both Wei [42] and Wei and Levoy [5] used Popat's [37] approach. However they did not change the synthesis order for a very good reason, speed. Keeping the synthesis order as proposed by Popat [37], meant that the Markov neighbourhood was fairly consistent over the whole synthesis process. This allowed for data compression. Popat [37] had used Gaussian kernels to define a pdf, Wei and Levoy [5] used tree-structured vector

quantisation to quickly search for the nearest neighbour. As highlighted by Efros and Leung [6], synthesis via nearest neighbour gave superior results.

In subsequent work Turk [21] shows how the nonparametric MRF techniques of Paget and Longstaff [38] and Wei and Levoy [5] can be extended to texture synthesis on arbitrary meshes. He uses a similar coarse to fine synthesis approach as presented in both papers. He also supplements the synthesis process with a directional field over the mesh to help maintain a consistent orientation to the texture.

The work of Ashikhmin [4] was the first real solution to the time consuming procedure of exhaustive nearest neighbour searching, without loss of quality as in Wei and Levoy's approach. In fact the method of Ashikhmin [4] 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 Manhatten or Euclidean distance, as he suggests that these may not be the best measure 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 correspond to a pixel within the input image. Speed can be gained if, instead of doing a exhaustive search, we only sample from those pixels with a corresponding neighbour. Unfortunately he applied his new technique to Wei and Levoy's [5] algorithm, which meant that his results were not as good as they should have been. Applying his technique to Paget and Longstaff's [38] algorithm, we get an algorithm that is arguable one of the best for synthesising natural textures.

Developed concurrently with Liang [7] approach, Efros and Freeman [18] take patch-based texture synthesis a step further. Instead of blending overlapping edges with a filter, they propose cutting and joining the respective patches along a boundary for which the difference in pixel values is minimal.

What if nearest neighbour comparisons could be avoided during synthesis? This is what Zelinka [43] tries to accomplish by creating a k nearest neighbour lookup table as part

Figure 2.1 (a) Input texture (b) Output generated by hybrid texture synthesis algorithm (c) Output generated by enhanced hybrid texture synthesis algorithm.

of an input texture analysis stage. They then use this table to make random jumps during their sequential texture synthesis stage. No neighbourhood comparisons are done during synthesis, which makes the algorithm very fast.

The hybrid texture synthesis algorithm [33] combines the pixel and the patch based algorithm to yield more effective results. Figure 2.1 shows an input texture and the output generated using the hybrid texture synthesis algorithm and our enhanced hybrid texture synthesis algorithm. As is evident, the outputs generated by both are close to the input texture. The blurring caused in right bottom corner figure figure 2.1 (b) using the hybrid texture synthesis algorithm has been eliminated in the figure 2.1 (c) using the enhanced hybrid texture synthesis algorithm.

2.3 Hybrid Texture Synthesis

The name, hybrid texture synthesis algorithm [33] was derived from the nature of the algorithm which combines the strengths of pixel-based algorithms, patch-based algorithms and their variants. The algorithm starts with an initially empty output image R, which is partitioned by a set of non-overlapping, connected pixel patches $p_i \in P$. All patches p_i combined resemble a tiling of R, formally

$$R = \bigcup_i p_i \tag{2.1}$$

and

$$\forall i, j.i \neq j : p_i \cap p_j = \emptyset \tag{2.2}$$

In algorithm step i, a patch of connected pixels P_i is selected from an example texture T, which best fits the target region p_i in the partially synthesized result R_{i-1}. This selection/search procedure is constrained by overlap of ov pixels with R_{i-1}. The algorithm constructs this overlap for arbitrarily shaped pixel patches p_i by dilating the binary support of p_i once with a square shaped structuring element of corner length $2ov + 1$. A candidate patch $P_i \subset T$ is used either if the overall error in the overlap region is below the maximum overlap error $\Delta_{\max}$ or the patch cannot be further split - otherwise the search process is repeated using smaller patches (a partition of p_i). If a patch satisfying the error bound has been found, every pixel in the overlap region exceeding the pixel error threshold $\partial_{\max}$ is marked as invalid and the binary support of the valid region is dilated to form a pixel traversal-map M_i, thereby defining the order in which erroneous overlap pixels are then re-synthesized on a per-pixel basis. This ensures sufficient valid neighborhoods for each re-synthesized pixel. The algorithm repeats this process until each patch p_i in the initial patch list P has been assigned pixel values.

The Hybrid texture synthesis algorithm [33] uses every patch in the output texture to find the best possible patch in the input texture. It uses the error image defined by

$$E_i = ERRORIMAGE(T, I_i, J_i) \tag{2.3}$$

where E_i is the 2D error image calculated using I_i , the 2D image mask and J_i the 2D binary support function for I_i

Given I_i , input texture T and the weighted error $\Delta_i \in [0, 1]$ can be calculated between the mask I_i and a circular shift x_0 of T.

$$E_i(x_0) = \frac{1}{\mathrm{K}_i} \sum_x \sum_c [J_i(x) W_c (I_{i,c}(x) - T_c(x + x_o))^2] \tag{2.4}$$

with $\mathrm{K}_i = \sum_x J_i(x)$, c=R,G,B and $\sum_c W_c = 1$. W_c is a three element color channel weighting vector.

Table 2.1 shows the variables used in the algorithm and their meanings. Algorithm 1 shows the pseudocode for the original Hybrid Texture synthesis algorithm described above. It is a recursive algorithm as explained, which repeats until the error margin is reduced or the algorithm is left with a single pixel.

Table 2.1 List of variables

Variable	Meaning
T	2D Input Texture
R,R_i	resulting 2D Texture(final/in step i)
ov,ov_s	patch overlap in pixels (initial/split)
Δ_{max}	user defined patch overlap error tolerance in [0,1]
∂_{max}	user defined pixel overlap error tolerance in [0,1]
P, P_s	list of non-overlapping pixel-patches p_i, which, when combined, resemble a tiling of the resulting texture area $R = \bigcup_i p_i$ (initial/split)
i	i integer algorithm step, starting at 1
p_i	a patch of connected pixels with index i in list P. p_i defines a region in R with Size(p_i) > 0
$R_{i,composite}$	intermediate 2D result after Compositing
I_i	2D image mask extracted from R_{i-1}, using p_i and o_v
J_i	2D binary support function for I_i
E_i, $E_{i,trim}$	2D error image computed from T, I_i, J_i
P_i	2D texture patch picked from T in step i
Δ_i	patch overlap error in the interval [0,1] for P_i
S_i	2D error surface between R_{i-1} and P_i
M_i	2D pixel traversal-map, defining the order in which erroneous overlap pixels are re-synthesized
W_c	a three-element, color channel weighting vector (c = R, G,B) with $\sum_c W_c = 1$
x, x_0	a 2D vector/coordinate (x, y)

Algorithm 1 HybridSynthesize:$(T, P, ov, \Delta_{\max}, \partial_{\max}, R : R)$

for all patches $p_i \in P$ **do**

[P$_i$, Δ_i] $\leftarrow$ FINDBESTPATCH(T, R_{i-1}, p_i, ov)

if ($\Delta_i < \Delta_{\max}$ or ISSINGLEPIXEL(p_i)) **then**

$S_i \leftarrow$ ERRORSURFACE(P_i,R_{i-1})

$P_i \leftarrow$ MARKINVALID(P_i,R_{i-1})

$M_i \leftarrow$ BUILDTRAVERSALMAP(P_i,R_{i-1})

$R_{i,composite} \leftarrow$ COMPOSE(P_i,R_{i-1})

$R_i \leftarrow$ OVERLAPRESYNTHESIZE(P_i,R_{i-1})

else

$P_s \leftarrow$ SPLITPATCH(p_i)

$ov_s \leftarrow \max(3, ceil(ov/2))$

$R_i \leftarrow$ HybridSynthesize(T,P_s,ov_s,$\Delta_{\max}$,$\partial_{\max}$,R_{i-1})

end if

end for

2.4 Color Models

Color is the brain's reaction to visual stimulus. Although we can precisely describe color by measuring its spectral power distribution this leads to a large degree of redundancy. The reason for this redundancy is that the eye's retina samples color using only three broad bands, roughly corresponding to red, green and blue light. The signals from these cones, together with those from the rods, are combined in the brain to give several different "sensations" of the color. These sensations have been defined as follows:

o Brightness: The human sensation by which an area exhibits more or less light.

o Hue: The human sensation according to which an area appears to be similar to one, or to proportions of two, of the perceived colors red, yellow, green and blue.

o Colorfulness: The human sensation according to which an area appears to exhibit more or less of its hue.

o Lightness: The sensation of an area's brightness relative to a reference white in the scene.

o Chroma: The colorfulness of an area relative to the brightness of a reference white.

o Saturation: The colorfulness of an area relative to its brightness

The tri-chromatic theory describes how three colors can form the basis of all colors. This theory is the basis for photography and print media.

A color model is a means for specifying, creating and visualizing colors. Various color models have been proposed over the years based on the needs. Color is mostly specified using three parameters. The various color models specify three of the "sensation" described above as parameters. The value of these parameters defines the position of the color in the color space.

Due to the availability of a large number of color models choosing the right model becomes a daunting task. In most cases color models are chosen as per the need of the application and the keeping in mind the limiting factors that dictate the size and type of color space that can be used. Some color spaces are perceptually linear, i.e. a 10 unit change in stimulus will produce the same change in perception wherever it is applied. Many color spaces, particularly in computer graphics, are not linear in this way. Some color spaces are intuitive to us thus making it easy for the user to navigate within them and creating desired colors relatively easily. Other spaces are confusing for the user with parameters with abstract relationships to the perceived color. Finally, some color spaces are tied to a specific piece of equipment (i.e. are device dependent) while others are equally valid on whatever device they are used.

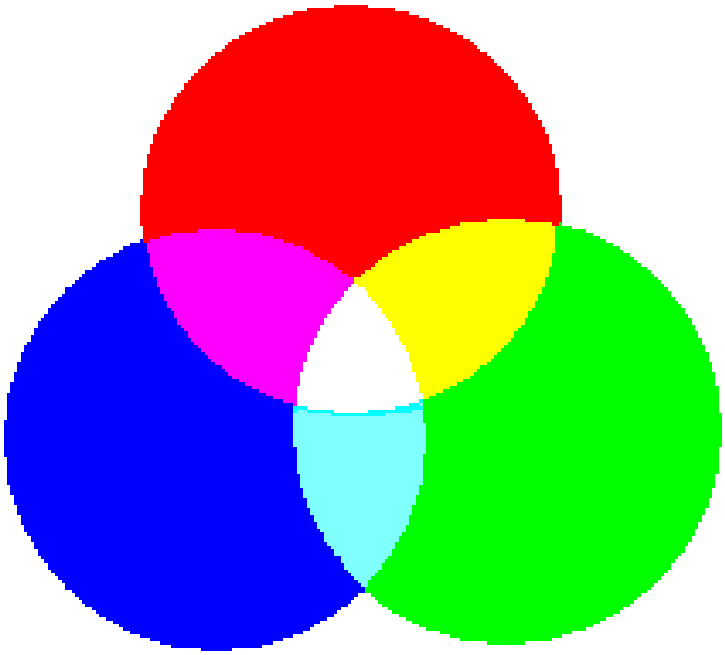


Figure 2.2 An image showing the RGB color triad. The three colors form the primary colors.

2.4.1 RGB Color Space

The RGB color space is the most widely used color models in image processing and computer graphics. In the RGB model, each color appears as a combination of red, green, and blue. This model is called additive, and the three colors, red, green and blue are called primary colors. The primary colors can be added to produce the secondary colors of light - magenta (red plus blue), cyan (green plus blue), and yellow (red plus green). The combination of red, green, and blue at full intensities makes white. Figure 2.2 shows the three primary colors and the secondary colors formed by their combination. Thus, images in the RGB color model consist of three independent image planes, one for each primary color. RGB is a basic color model for computer graphics because color displays use red, green, and blue to create the desired color. Therefore, the choice of the RGB color space simplifies the architecture and design of the system. Besides, a system that is designed

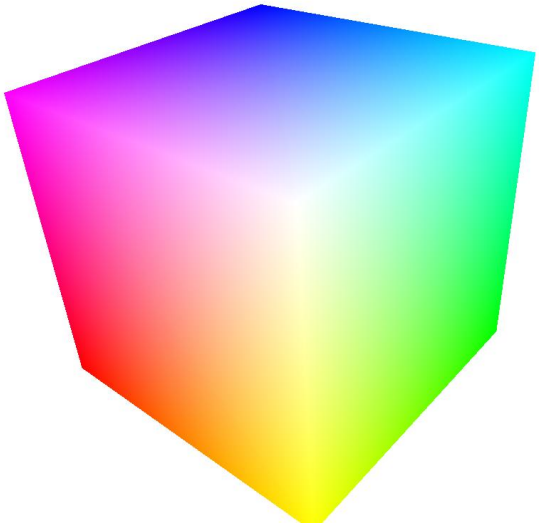


Figure 2.3 An image showing the color cube. The cube helps represent the various colors and their gradients as they change from one color to the other.

using the RGB color space can take advantage of a large number of existing software routines, because this color space has been around for a number of years. However, RGB is not very efficient when dealing with real-world images. To generate any color within the RGB color cube, all three RGB components need to be of equal pixel depth and display resolution. Also, any modification of the image requires modification of all three planes.

2.4.2 HSI Color Space

The HSI color space as described by Gonzalez and Woods [44] is an extension of the HSL (Hue Saturation Light) model for colors. The HSL type color models are a deformation of the RGB color cube. Imagine the RGB color cube tipped onto the black corner then the line through the cube from black to white defines the lightness axis. The color is then defined as a position on a circular plane around the lightness axis. Hue is the angle from a nominal point around the circle to the color while saturation is the radius from the central lightness axis to the color.

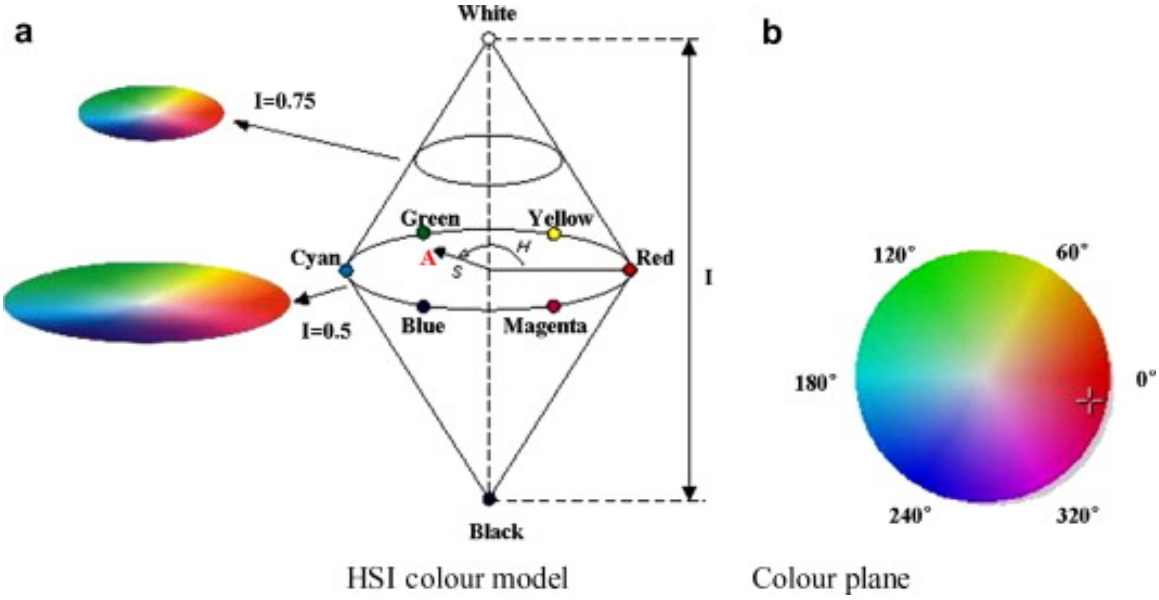


Figure 2.4 Illustration of the HSI color model. The HSI color model is based an the hue, saturation and intensity component.

Figure 2.3 shows a RGB color cube with the white corner oriented towards the viewer. The diagonally opposite corner of this cube corresponds to the black color.

The HSL color space rotates this cube about the diagonal running from the black to the white corner.

The HSI color model uses the hue, saturation and the intensity instead of the value to create and visualize color. The hue component describes the color itself in the form of an angle between [0,360] degrees. 0 degree mean red, 120 means green, 240 means blue, 60 degrees is yellow, 300 degrees is magenta. The saturation component signals how much the color is polluted with white component. The range of S is [0 1]. The intensity ranges between [0 1] where 0 means black and 1 means white

Figure 2.4 shows an illustration of the HSI color model. In this model the point "A" (shown in red), is an arbitrary color point, the angle form the red axis gives the hue. The length of the vector represents the saturation normally ranging from [0 1].

The conversion formulae between RGB and HSI is given as follows

Intensity is defined as

$$I = \frac{(R+G+B)}{3} \tag{2.5}$$

Saturation is defined as

$$S = 1 - \frac{3}{(R+G+B)} * a \tag{2.6}$$

where a is min(R,G,B).

and hue is defined as

$$H = \cos^{-1} \frac{(0.5*(R-G)+(R-B))}{(((R-G)^2+(R-B)(G-B))^{0.5})} \tag{2.7}$$

2.5 Enhanced Hybrid Texture Synthesis

The Enhanced Hybrid Texture synthesis algorithm builds on the idea of the hybrid texture synthesis algorithm and extends it to improve the quality of the images synthesized.

2.5.1 Approach

This algorithm alters the value of the weighting vector to reflect the intensity parameter of the HSI model to thus changing the color model used in the algorithm. Algorithm 2 shows the pseudocode for the enhanced hybrid texture synthesis algorithm. It shows how the new color model has been effectively incorporated into the existing algorithm. The HSI color model is used instead of the traditional RGB (red, green and blue) space because the HSI model has a better color description for human interpretation [44].

Algorithm 2 EnhancedHybridSynthesize:$(T, P, ov, \Delta_{\max}, \partial_{\max}, R : R)$

$W \leftarrow$ HSIWEIGHT(R,G,B)

for all patches $p_i \in P$ **do**

[P$_i, \Delta_i$] $\leftarrow$ FINDBESTPATCH(T, R_{i-1}, p_i, ov)

if ($\Delta_i < \Delta_{\max}$ or ISSINGLEPIXEL(p_i)) **then**

$S_i \leftarrow$ ERRORSURFACE(P_i,R_{i-1},W)

$P_i \leftarrow$ MARKINVALID(P_i,R_{i-1},W)

$M_i \leftarrow$ BUILDTRAVERSALMAP(P_i,R_{i-1},W)

$R_{i,composite} \leftarrow$ COMPOSE(P_i,R_{i-1},W)

$R_i \leftarrow$ OVERLAPRESYNTHESIZE(P_i,R_{i-1},W)

else

$P_s \leftarrow$ SPLITPATCH(p_i)

$ov_s \leftarrow \max(3, ceil(ov/2))$

$R_i \leftarrow$ EnhancedHybridSynthesize(T,P_s,ov_s,$\Delta_{\max}$,$\partial_{\max}$,R_{i-1})

end if

end for

return R

2.5.2 Steps Involved

The proposed algorithm involves modifying the color channel weighing vector while calculating the error image.

1. Build a dataset of textures to test and compare the new algorithm with the original one.

2. Execute the hybrid texture algorithm to generate textures of size 128×128 using the original color model

3. Modify the weight vector W_c in the algorithm to reflect the value of the parameters from the HSI color model

4. Execute the enhanced hybrid texture algorithm to generate the textures of size 128×128 for the same set of input textures as in Step 1.

2.6 Experimental Results

We tested our algorithm on twenty four test texture images and on a subset of the MIT VisTex database. The first set of textures varied from regular to stochastic and were obtained from the PSU texture database. Figure 2.5 shows the steps involved in comparing the output of the two algorithms.

Figure 2.6 shows the input dataset we used for our experiment from the PSU database. Of the texture set the three images on the first column are regular, six images are near-regular, six are irregular, six are near-stochastic and three are stochastic. This set of test data has a mix of different types of texture and serves well in testing the validity of our algorithm. Figure 2.7 - Figure 2.10 show a subset of our results. The first image in the set is the initial input image. The second, the result generated using the hybrid texture synthesis algorithm and the third is the result generated using our enhanced hybrid texture synthesis algorithm.

The second dataset was a subset of textures from the MIT VisTex texture database. Figure 2.13 - Figure 2.15 show the comparison result for the output generated using two images from the MIT VisTex database. A total of forty-three images have been used for testing from this database.

2.7 Conclusions

The textures generated using our enhanced hybrid texture synthesis shows that the result is appreciably better than that generated using the hybrid texture synthesis algorithm. The results however differ with each texture and show noticeable improvement in some and and very little improvement in a few others. The task of determining the better output from the generated texture requires manual inspection and with slight improvement it becomes

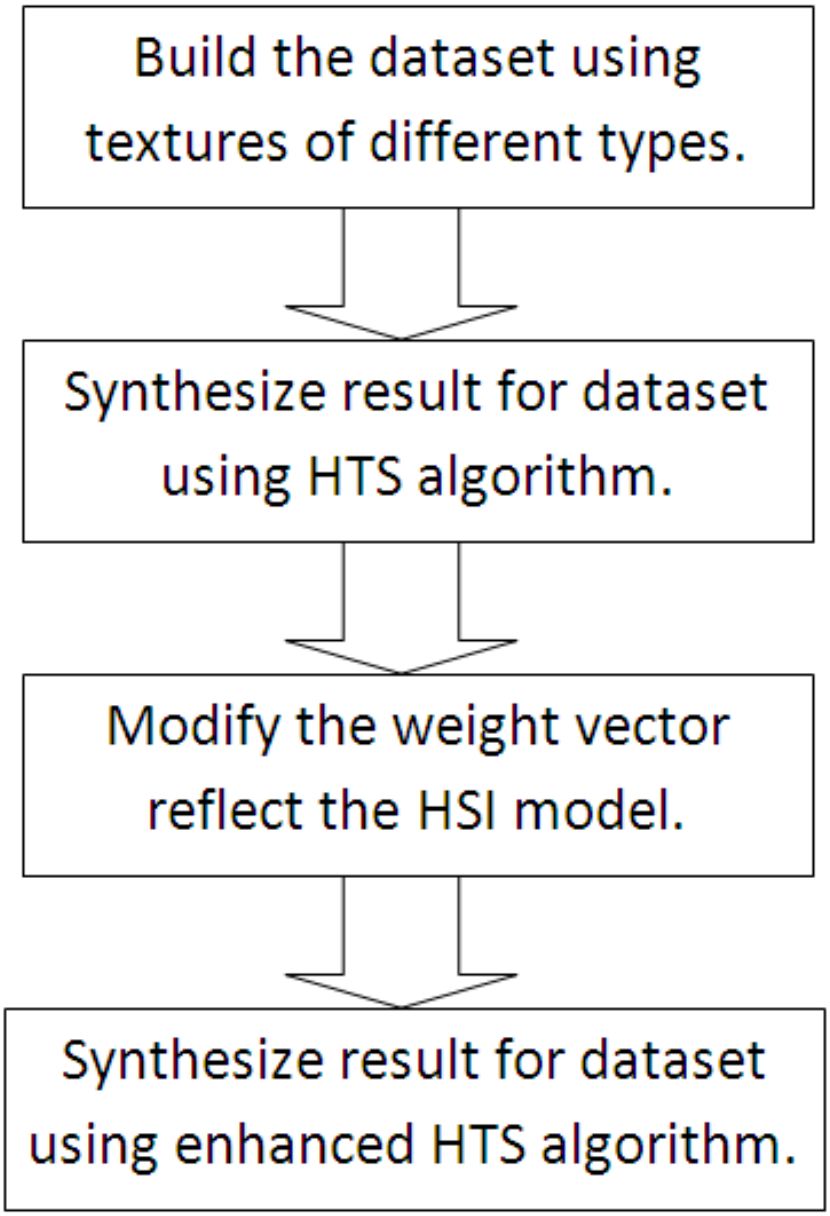


Figure 2.5 Image showing the steps involved in the enhanced hybrid texture synthesis.

Figure 2.6 Input data set consisting of 24 images across five different texture types. These images form the data set for the work.

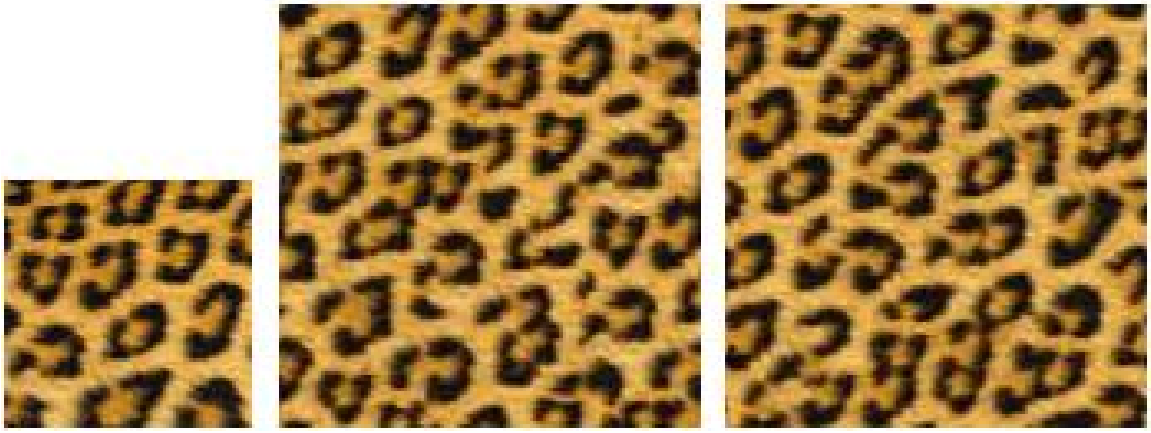


Figure 2.7 Image showing the results obtained by the texture synthesis methods for the tiger print texture image. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

a difficult task. The next chapter deals with this issue of quantifying textures based on the output generated using texture synthesis algorithms.

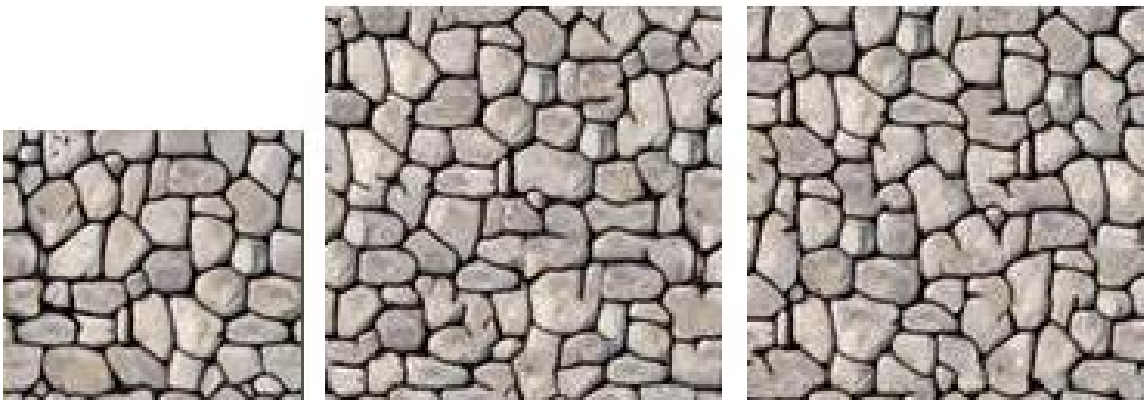


Figure 2.8 Image showing the results obtained by the texture synthesis methods for the stone wall texture image. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

Figure 2.9 Image showing the results obtained by the texture synthesis methods for the pebbles texture image. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

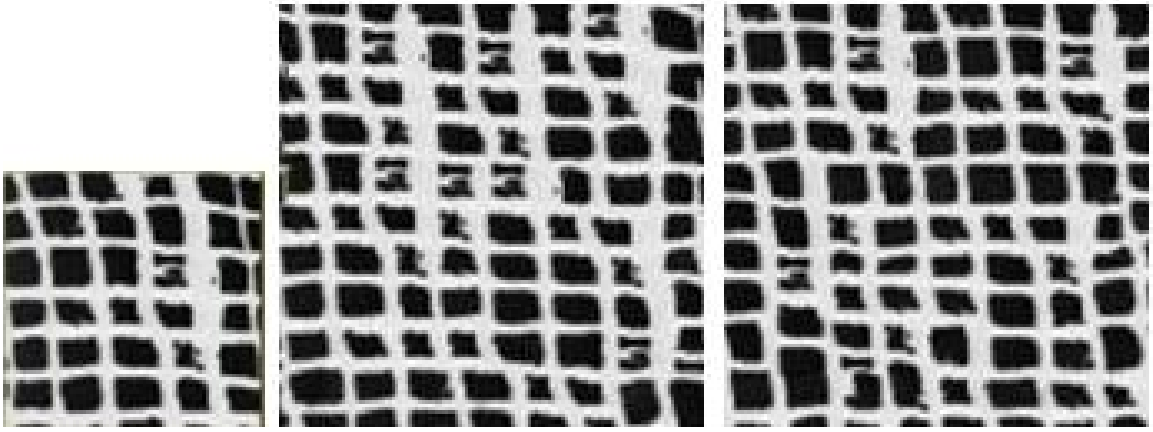


Figure 2.10 Image showing the results obtained by the texture synthesis methods for the mesh texture image. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

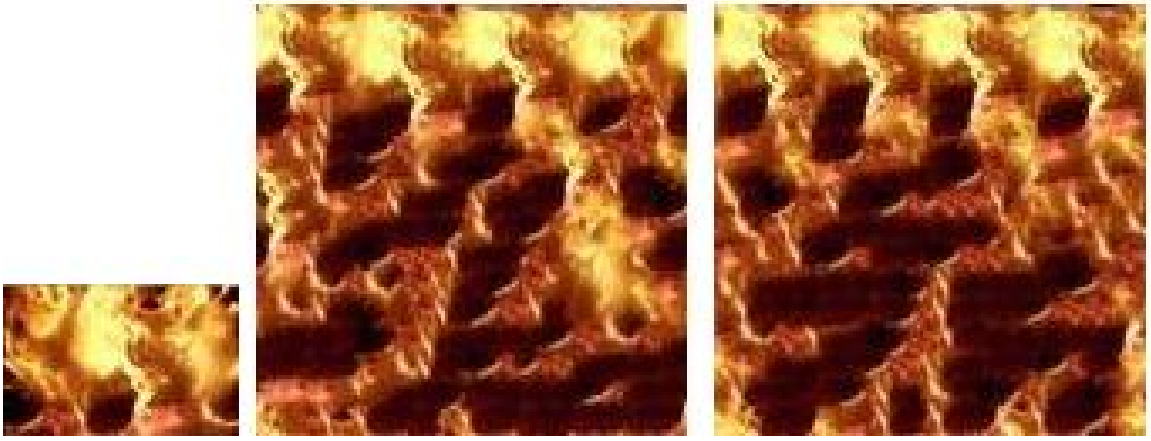


Figure 2.11 Image showing the results obtained by the texture synthesis methods for the fire texture image. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

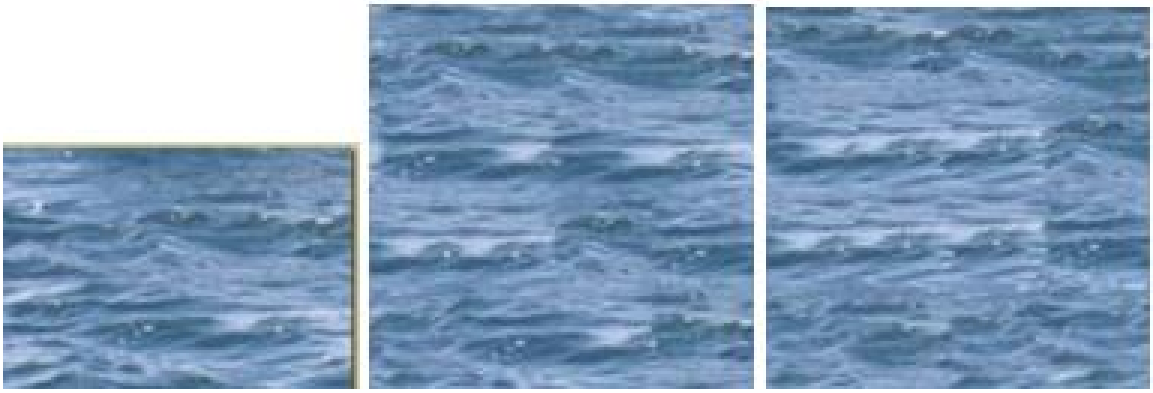


Figure 2.12 Image showing the results obtained by the texture synthesis methods for the water texture image. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

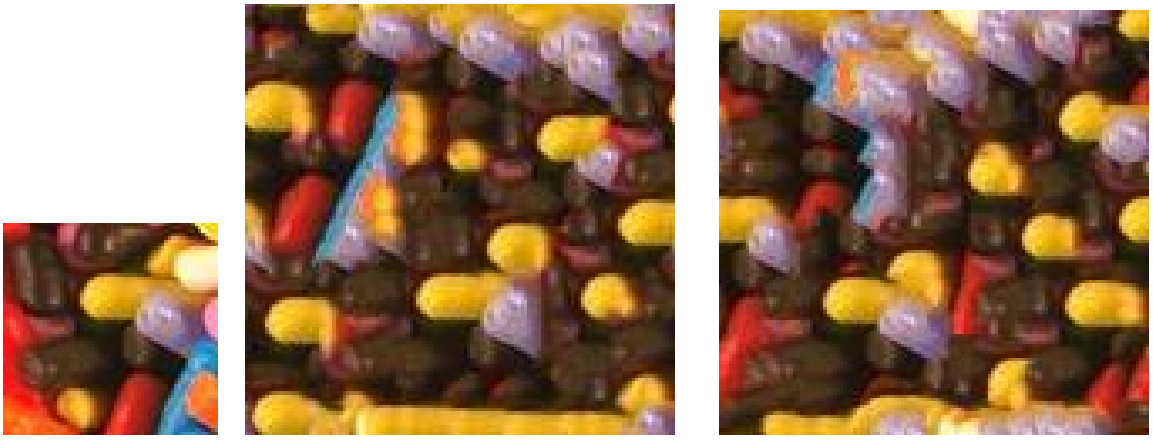


Figure 2.13 Image showing the results obtained by the texture synthesis methods for the food texture image from the MIT VisTex texture database. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

Figure 2.14 Image showing the results obtained by the texture synthesis methods for the food texture image from the MIT VisTex texture database. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

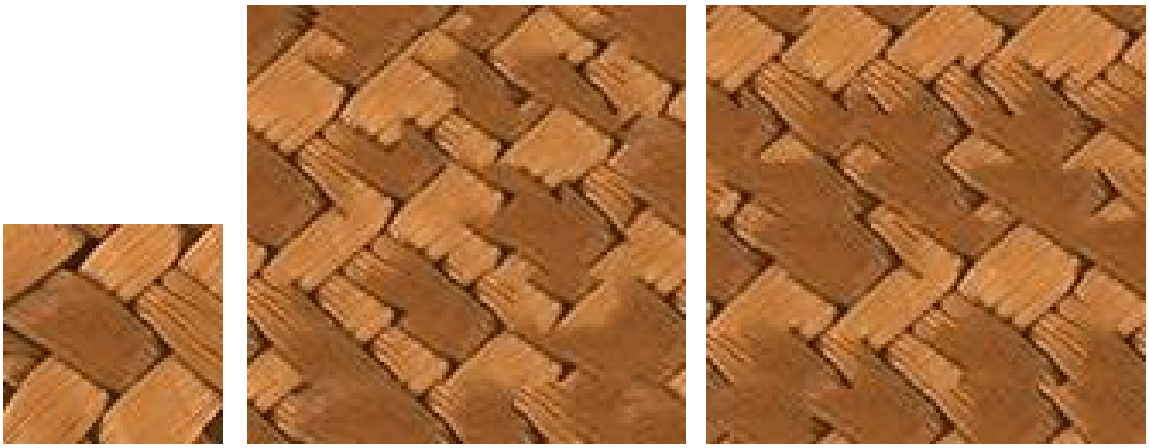


Figure 2.15 Image showing the results obtained by the texture synthesis methods for the fabric texture image from the MIT VisTex texture database. Original image (left), output of hybrid texture synthesis (center), output of enhanced hybrid texture synthesis (right).

CHAPTER 3

TEXTURE QUANTIFICATION

3.1 Quantification

As discussed in Chapter 2 there are many algorithms for texture synthesis using both the pixel and the patch based approach. Each of these algorithm provides an output which though a close match to the original image is not an absolute match. Comparing the different output textures in relation to the input texture provided would be a good approach to provide a basis for quantification and as such determining which output is the closest match.

The best matching output will have to consistently recreate the underlying pattern of the input texture, including all the global and the local features.

Figure 3.1 shows an input texture and the output generated using the Efros and Leung [6] and the Efros and Freeman algorithms [18]. As is evident, the outputs generated by both are close to the input texture, but not an exact match. Determining which of the two outputs is better is a highly subjective matter. This could be done using manual inspection and human interaction. However, to automatically determine which of the two is better and reduce the amount of human involvement would be an ideal option for us as computer scientists. This dissertation presents an algorithm to eliminate human involvement in quantifying the output using classical statistical measures and a color correlogram.

3.2 Existing Techniques

3.2.1 Quantitative Evaluation on Near Regular Texture Synthesis

The method suggested by Chieh et. al. [8] is applicable for the near-regular class of textures and not other classes. The algorithm compares the synthesis results of four algorithms on regular textures and near-regular textures by comparing their underlying lattices. As a

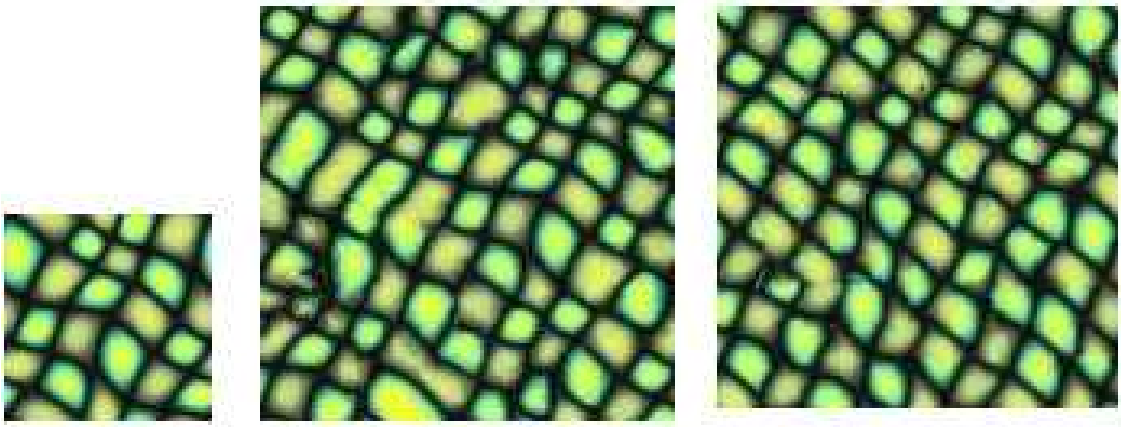


Figure 3.1 (a) Input texture (b) Output generated by Efros and Leung algorithm (c) Output generated by Efros and Freeman algorithm

basis for comparison, the lattices of regular textures are automatically extracted, and those of near-regular textures are specified interactively. The regularity preservation test and the user evaluation test are used as a complementary pair to evaluate the synthesis results.

The first evaluation is objective since it checks if the global regularity is preserved based on a pair of well-defined, quantitative geometric and appearance regularity measures. The basic idea is to view a near-regular texture as a statistical distortion of a regular, wallpaper-like congruent tiling and the degree of its regularity is determined by how much the geometry and appearance of individual tiles differ from their regular state. Since a near-regular texture can be transformed to its regular state by deforming its underlying lattice, the geometric regularity of the near-regular texture is obtained by computing the amount of deformation between the underlying lattice of the near-regular texture and that of its regular counterpart. This method uses two measures called the Geometric Regularity (G-score) and the Appearance Regularity (A-score) for the objective study.

The second evaluation is subjective, human subjects are used to examine and score the overall quality of the synthesized textures in comparison to the input texture in terms of color/intensity, statistical variations, and structures.

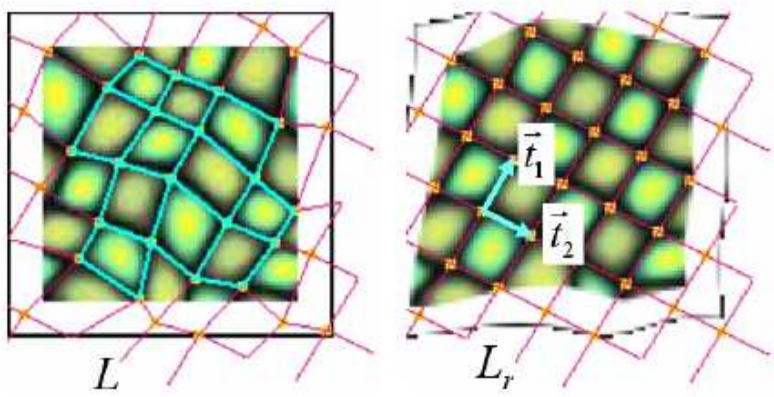


Figure 3.2 A near regular texture overlaid with its lattice(left) and its geometric regular counterpart(right).

Figure 3.2 shows an example of a near regular texture overlaid with its lattice and its geometric counterpart.

3.2.2 Disadvantages

This basis of the work described in [8] is the fact that the near regular textures are departures from their regular equivalent. This poses a problem for using this technique for evaluating different classes of textures. The irregular, near-stochastic and the stochastic textures have no regular conterpart to which they can be compared and which would be the basis for underlying grid.

3.3 Color Correlogram

A color correlogram expresses how the spatial correlation of color changes with distance. It is a color feature which can be used in various image processing and video processing tasks as described by the work in [45, 46, 47, 48]. A color histogram captures only the color distribution in an image and does not include any spatial information. Thus, the correlogram is one kind of spatial extension of the histogram [49]. The highlights of this feature are: (i) it describes the global distribution of local spatial correlations of colors,

(ii) it is easy to compute, and (iii) the size of the feature is fairly small, (iv) it is stable to tolerate large appearance changes, and (v) it is also scalable to large image databases.

Informally, a correlogram is a table indexed by color pairs and distance, where the k-th entry for $< i, j >$ specifies the probability of finding a pixel of color j at a distance k from a pixel of color i. Such an image feature turns out to be stable in tolerating large changes in appearance of the same scene caused by changes in viewing positions, changes in the background scene, partial occlusions, camera zoom that causes radical changes in shape, etc.

Let I be an n $\times$ n image. (For simplicity of exposition, we assume that the image is square.) The colors in I are quantized into m colors $c_1; :::; c_m$. (In practice, m is deemed to be a constant)

For a pixel $p = (x, y) \in I$, let $C(p)$ denote its color. Let

$$I_c \triangleq \{p|C(p) = c\} \tag{3.1}$$

Thus, the notation $p \in I_c$ is synonymous with $p \in I; C(p) = c$. For convenience the L_∞ norm is used to measure the distance between pixels, i.e, for pixels $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$ we define $|p_1 - p_2| \triangleq \max\{(x_1 - x_2), (y_1 - y_2)\}$.

The histogram h of I is defined as follows; for a color $c_i, i \in |m|$.

$$H_{c_i}(I) \triangleq ||I_c|| \tag{3.2}$$

i.e., the number of pixels of color c_i in I. Randomly taking any pixel p from the image I, the probability that the color of p is c_i is

$$h_{c_i}(I) = \Pr[p \in I_{c_i}] = \frac{H_{c_i}(I)}{n^2} \tag{3.3}$$

Figure 3.3 Two sample images with same image histogram but different image correlogram.

The histogram is easily computed in $O(n^2)$ time, which is linear in the size of I. Note that the above definition of the histogram is as a probability distribution h.

In the light of viewing the histogram as a probability distribution of colors, consider this question: pick any pixel p_1 of color c_i in the image I, at distance k away from p_1 pick another pixel p_2, what is the probability that p_2 is of color c_j ? This gives us the conditional probability distribution that depicts the spatial correlation between pixels. We define the correlogram formally below. Let a distance $d \in [n]$ be fixed a priori. Then, the correlogram of I for $i; j \in [m]$, and $k \in [d]$, is defined to be

$$\gamma^k_{c_i c_j}(I) \triangleq \Pr[p_2 \in I_{c_j}, |p_1 - p_2| = k, p_1 \in I_{c_i}] \tag{3.4}$$

Therefore the correlogram of an image I is a table indexed by color pairs and distance, where the k-th entry for $< i, j >$ specifies the probability of finding a pixel of color j at a distance k from a pixel of color i. Note that the size of the correlogram is $O(m^2 d)$.

If only the correlation between the identical colors is considered, the result is the autocorrelogram of I, which is defined by

$$\alpha_c^k(I) \triangleq \gamma_{c,c}^k(I) \tag{3.5}$$

While choosing d to define the correlogram, the following trade-off has to be addressed. A large d would result in expensive computation and large storage requirements. A small d might compromise the quality of the feature.

To compute the correlogram, it suffices to compute the following count, which is similar to the co-occurrence matrix [50] for texture analysis of gray images.

If

$$\Gamma_{c_i,c_j}^k(I) \triangleq |\{p_1 \in I_{c_i}, p_2 \in I_{c_j}, |p_1 - p_2| = k\}| \tag{3.6}$$

then,

$$\gamma_{c_i,c_j}^k(I) = \frac{\Gamma_{c_i c_j}^k}{h_{c_i}(I).8k} \tag{3.7}$$

The denominator is the total number of pixels at distance k from any pixel of color c_i. The factor 8k is due to the eight neighbors which are k distance (in L_∞-norm) away from the center pixel. As the histogram is the color distribution in I, the correlogram is the color correlation distribution in I. Therefore both qualify as global features.

3.4 Algorithms Compared using Proposed Method

The texture synthesis algorithms analysed in this dissertation are the following: a) Efros and Leung Algorithm [6] b) Efros and Freeman Algorithm [18], c) Hybrid Texture synthesis Algorithm [33] and d)Proposed Algorithm.

3.4.1 Efros and Leung

Efros and Leung Algorithm [6] is modeled as a Markov Random Field (MRF) []. That is, the algorithm assumes that the probability distribution of brightness values for a pixel given the brightness values of its spatial neighborhood is independent of the rest of the image. The neighborhood of a pixel is modeled as a square window around that pixel. The size of the window is a free parameter that specifies how stochastic the user believes this texture to be. More specifically, if the texture is presumed to be mainly regular at high spatial frequencies and mainly stochastic at low spatial frequencies, the size of the window should be on the scale of the biggest regular feature.

They synthesize a texture I_o by repeatedly matching the neighborhood around the target pixel in the synthesis result with the neighborhood around all pixels in the input texture I_{in}, starting from a seed pixel and growing outwards. For each to-besynthesized pixel p_{out} and its neighborhood of already synthesized pixels Δ_i, an approximation to the conditional probability $P(p|\omega(p))$ is constructed for each $p_{in} \in I_{in}$. This is achieved by computing a gaussian weighted, normalized sum of square differences (SSD) between $\omega(p_{out})$ and the pixel neighborhoods of each candidate in the input texture, $\omega(p_{in})$. A target pixel is then selected from a set of pixels p_{in} with high conditional probability. The algorithm performs an exhaustive search in I_{in} for each synthesized pixel and is therefore quite slow. Also, the algorithm has a tendency to slip into the wrong part of the search space and start growing garbage or to perform verbatim copying of the input.

3.4.2 Efros and Freeman

The Efros and Freeman Quilting algorithm [18] is a patch based texture synthesis algorithm. It is based on the notion that a lot of searching work is wasted on pixels that can be assigned values using the existing information. It seems then, that the unit of synthesis should be something more than a single pixel, a “patch” perhaps. Then the process of texture synthesis would be akin to putting together a jigsaw puzzle, quilting together the

patches, making sure they all fit together. The Efros and Freeman quilting algorithm presents a very naive version of stitching together patches of texture to form the output image. This method is called “image quilting”.

Efros and Freeman Image Quilting algorithm [18] iterates through a uniform quadrilateral grid of patches, which, combined, resemble a tiling of the output texture. In scanline order, the algorithm selects, for each output patch, a congruent patch of pixels from the input texture, constrained by overlap with the already synthesized result. It then performs a minimum-error-boundary-cut (MEBC) within the overlap region of adjacent texture patches to reduce artifacts.

The MEBC is implemented by employing dynamic programming. The algorithm is also well suited for texture transfer, which is demonstrated in the paper. Synthesis results presented in are equal to or better than Efros/Leung-like, pixel-based algorithms. Still, as also pointed out by [7], hard color changes along patch boundaries, termed boundary mismatch, tend to occur.

3.4.3 Hybrid Texture Synthesis

The Hybrid Texture Synthesis algorithm [33] is an adaptive and hybrid algorithm. This algorithm adaptively splits patches so as to use as large as possible patches while staying within a user-defined error tolerance for the mismatch in the overlap region. Using large patches improves the reproduction of global structure. The remaining errors in the overlap regions are eliminated using pixel-based re-synthesis. It introduces an optimized ordering for the re-synthesis of these erroneous pixels using morphological operators, which ensures that every pixel has enough valid (i.e., error-free) neighboring pixels. A more detailed explanation of this algorithm has been provided in Chapter 2.

3.5 Proposed Method

Our approach deals with quantifying and identifying the best of the synthesized texture in a quantifiable manner. To better understand our method, the following section gives a brief overview of the classical statistical measures. Figure 3.7 shows the steps involved in our quantification process.

3.5.1 Statistical Measures

Classically, a large number of statistical measures have been used to evaluate texture examples. It has been widely used in pattern recognition and computer vision fields. Statistical texture method describes texture as a feature vector of properties which represents a point in multidimensional feature space. For the purpose of quantification the co-occurrence matrix of texture description and the autocorrelation method, which is based on spatial frequencies [50, 51, 52] has been used. The co-occurrence matrix method of texture description is based on the gray level configuration in the texture. For fine textures this configuration varies rapidly with increased distance, for coarse textures it varies gradually. The following five properties of the co-occurrence matrix, a) Correlation b) Contrast c) Energy d) Homogeneity and e) Entropy have been used for the proposed method.

To understand the properties used for analysis, a brief understanding of the co-occurrence matrix is useful. The co-occurrence matrix consists of values $p(i, j)$ which are created by calculating how many times a pixel with gray level value i occurs horizontally from a pixel with a gray level value j. The size of the matrix depends on the intensity levels in the input image.

a) Correlation of an image is a measure of how much a pixel is correlated to its neighbor over the entire image. The range of correlation lies between -1 and 1.

Formally, correlation is defined as follows

$$\sum_{i,j} \frac{(i-\mu_i)(j-\mu_j)p(i,j)}{\sigma_i \sigma_j} \tag{3.8}$$

Where μ_i and μ_j are the means and σ_i and σ_j are the standard deviations.

b) Contrast of an image is a measure of the intensity contrast between a pixel and its neighbor over the whole image. The contrast of an image is defined as follows

$$\sum_{i,j} |i-j|^2 p(i,j) \tag{3.9}$$

For a constant image the contrast is 0.

c) Energy of an image is the sum of the square of the probabilities of the intensity of the image. If the cooccurrence matrix exists the energy calculation merely involves calculating the sum of the square of that matrix.

The energy of an image is defined as follows

$$\sum_{i,j} p(i,j)^2 \tag{3.10}$$

d) Homogeneity of an image returns a value that measures the closeness of the distribution of elements in the gray scale co-occurrence matrix with the elements which lies on its diagonal.

The homogeneity of an image is defined as follows

$$\sum_{i,j} \frac{p(i,j)}{1+|i-j|} \tag{3.11}$$

e) Entropy of an image is a useful statistical measure for determining the randomness of an image. Entropy is defined by

where p is the probability of the intensity in the image

$$-\sum p * \log(p) \tag{3.12}$$

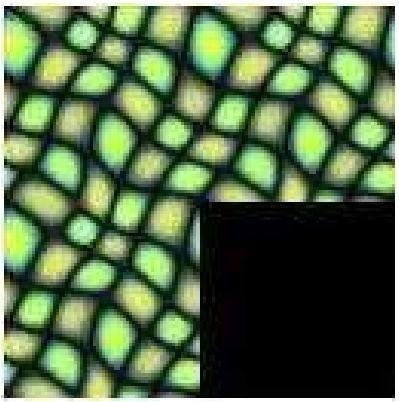


Figure 3.4 A toroidal image set next to copies of itself merges into a single image without any seams.

3.5.2 Toroidal Image

To set a benchmark for this approach a toroidal image is used to create an ideal texture image from an input texture. A toroidal image is one which wraps around itself without creating any overlaps or inconsistency in the appearance of the image. This image successfully preserves all global features of the input texture and does not generate any artifacts in the presence of high frequency features.

To set a benchmark the "green scale" tordoial image has been used.

Figure 3.4 shows the green scale image (64×64) being used to generate a (128×128) image. Three copies of the image have been placed in the larger image and no seams are visible.

3.5.3 Steps Involved in Proposed Method

The approach proposed is a two step process. In the first step the gray scale statictical measures are used to determine the best output. In the second step an autocorrelogram based approach is used to determine the best output match. This result is then used in conjunction with the first set of outputs to determine the overall best output.

Figure 3.5 Data set of images used in the experiments. The images are numbered and results are shown using these numbers.

The first step using statictical measures involves a number of sub steps which initially starts with generating output textures for different examples. To ensure a consistency in the approach texture of types, regular, near-regular, irregular, near stochastic and stochastic types have been used. The Figure 3.7 shows a schematic representation of the approach.

1. Create a dataset using twenty-four different images of different types of texture. The initial sizes of all our images are roughly 64×64 pixels. Figure 3.5 shows a subset of the initial dataset.

2. Generate synthesis results using the three different algorithms. Figure 3.6 shows a subset of the result of the three algorithms along with the original image. The first column shows the original image, the second the result generated by Efros and Leung algorithm, the third the result generated by Efros and Freeman algorithm and the fourth the result generated by the Hybrid algorithm.

Figure 3.6 Subset of result showing original image and result with three algorithms.

3. Calculate the energy, entropy, correlation, contrast and homogeneity for each of the image generated using the various texture synthesis algorithms.

4. Plot the difference in the statistical values of each of the synthesized image with the original images. Figure 3.20 - Figure 3.24 show the respective plotted errors.

5. Pick the synthesized texture image with minimum error. This gives the best match based on the statistical measures.

The comparison of the values in the synthesized case and the values obtained with the original texture will give an estimate of the quality of the in a more specific way. A finite number can be associated which will determine the quality of the output.

The second step uses an autocorrelogram to determine measures for all images. The distance of $k = 1, 2, 3, 4, 5, 6$ have been used to generate the values for each of the images.

1. The synthesized texture images to be compared are first quantized into 32 colors.

2. The autocorrelogram of each image is generated using the distance all the different values of k.

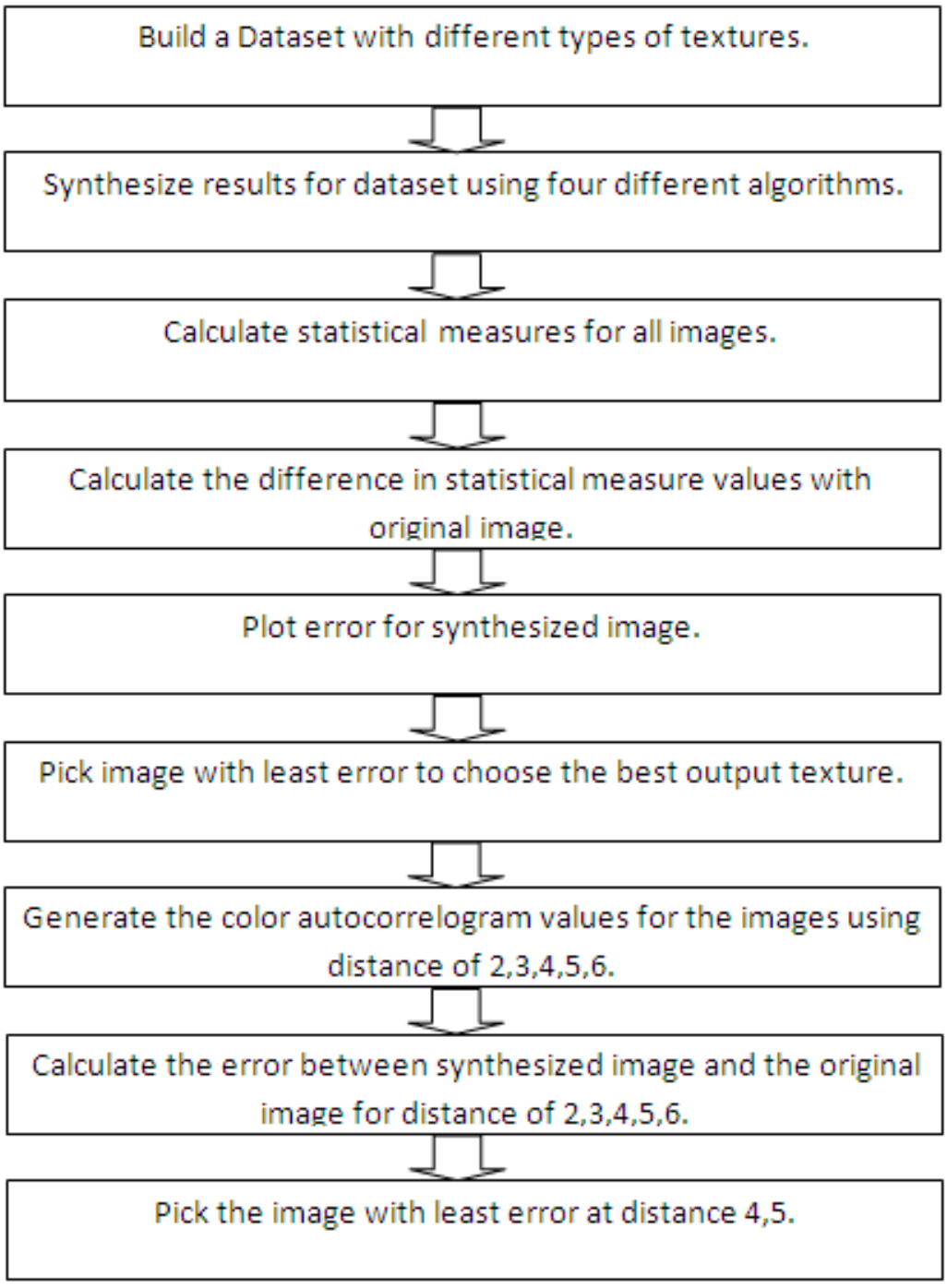


Figure 3.7 Quantification of image quality.

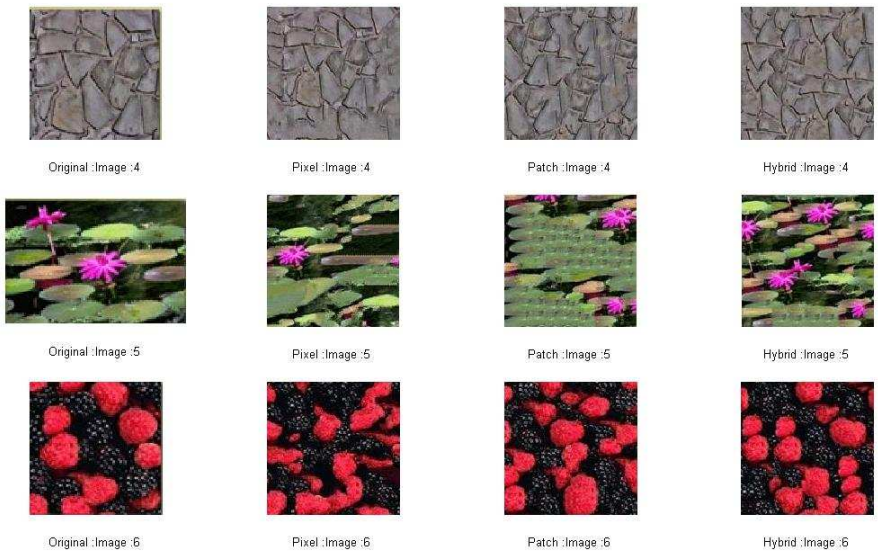


Figure 3.8 Output image set. This shows the results generated using three algorithms on images numbered 4, 5 and 6. The left most image is the original input texture.

3. The autocorrelogram error is then generated for all images using the input image as a standard.

3.6 Experimental Results

In this section the proposed method has been tested on a variety of texture images set comprising of various types of textures. Figure 3.15 - Figure 3.19 shows the statistical measures data for twenty four images. This data has been used to generate the error data for each synthesized texture shown in Figure 3.20 - Figure 3.24. The first set of results shown in Table 3.3 is generated using this data. This table shows the actual output of the quantification process, where "Ef-Le" refers to Efros and Leung's method [6], "Ef-Fre" refers to Efros and Freeman's method [18], "HTS" refers to Hybrid Texture Synthesis method [33] and "Tie" indicates a tie between the output generated.

In the second step the k value associated with the autocorrelogram is set to the following, k=1, 2, 3, 4, 5, 6. The results to determine the best output has been obtained

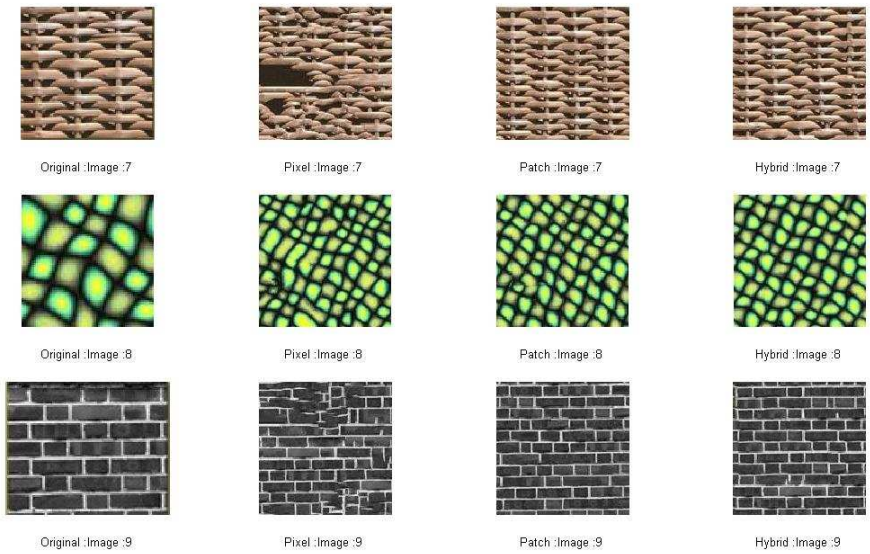


Figure 3.9 Output image set. This shows the results generated using three algorithms on images numbered 7, 8 and 9. The left most image is the original input texture.

using the value k=4. Figure 3.25 - Figure 3.28 shows the plot for the autocorrelogram error using the different values of k for a subset of the images. The Figure 3.25 shows the autocorrelogram error value for image number 2 using four different algorithms. The fourth algorithm considered in this scenario is the proposed method described in Chapter 2. For the purpose of determining the best match we have selected k=4 and compared the error with the input image. Table 3.2 shows the accuracy of the proposed method. Using only the statistical measures we have a 62.5% accuracy whereas using the correlogram we get a accuracy of 79.16%.

All the experiments are performed on a 1.5GHz Core 2 Duo machine with a 2GB RAM.

3.7 Conclusions

The tabulated result clearly shows that the ideal synthesized texture has a significantly smaller error and ideally as in the case of the entropy no error at all. The co-occurrence

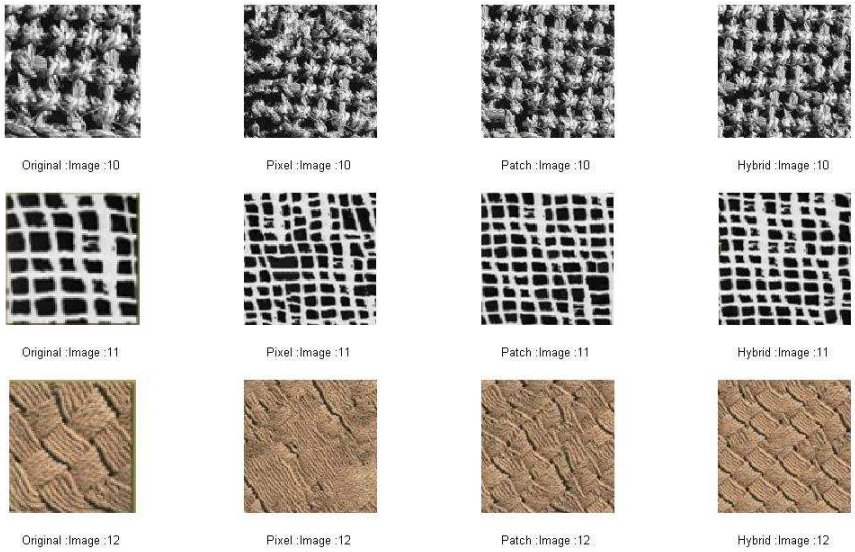


Figure 3.10 Output image set. This shows the results generated using three algorithms on images numbered 10, 11 and 12. The left most image is the original input texture.

matrix method describes second-order image statistics and works well for a large variety of textures. However it does not consider primitive shapes and colors and thus cannot be used for a reasonable analysis of textures with large primitives. This method is also computationally expensive and might be limited with respect to time. This approach can still be used in such scenarios without the ideal texture to compare with. In such situation, the input image can be used as a benchmark. The color correlogram based approach shows a good set of output based on the autocorrelogram using variable distance k. This direction of research can be taken forward using larger image

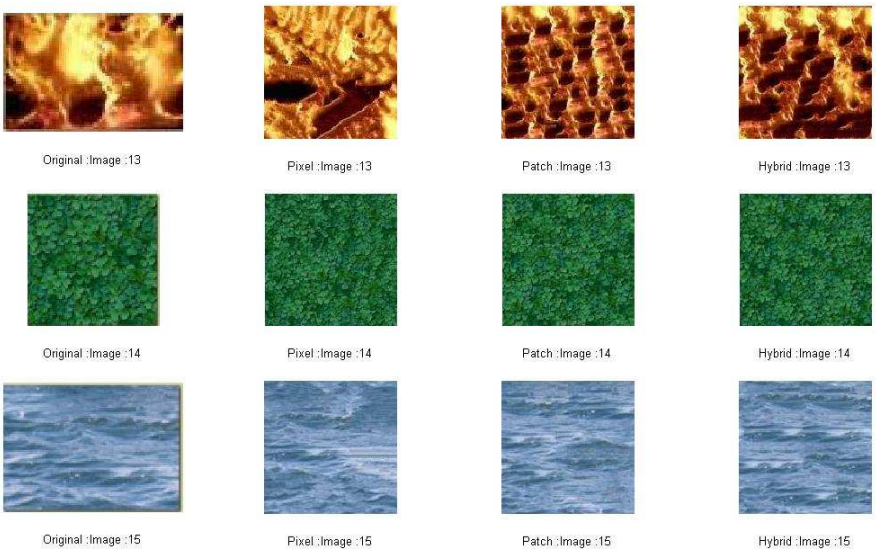


Figure 3.11 Output image set. This shows the results generated using three algorithms on images numbered 13, 14 and 15. The left most image is the original input texture.

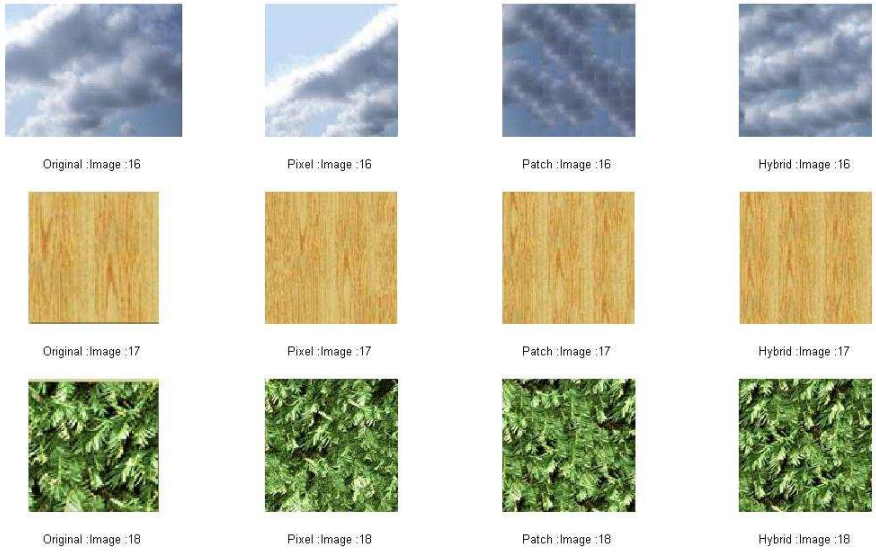


Figure 3.12 Output image set. This shows the results generated using three algorithms on images numbered 16, 17 and 18. The left most image is the original input texture.

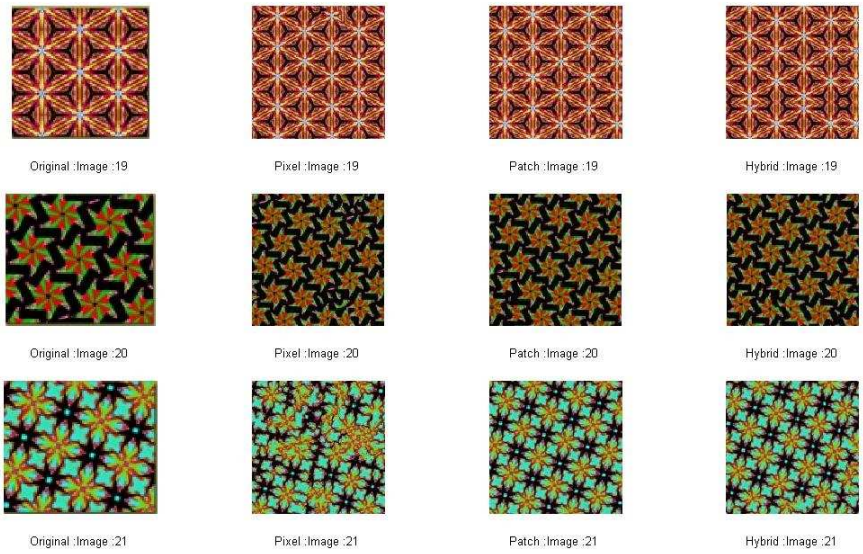


Figure 3.13 Output image set. This shows the results generated using three algorithms on images numbered 19, 20 and 21. The left most image is the original input texture.

Figure 3.14 Output image set. This shows the results generated using three algorithms on images numbered 22, 23 and 24. The left most image is the original input texture.

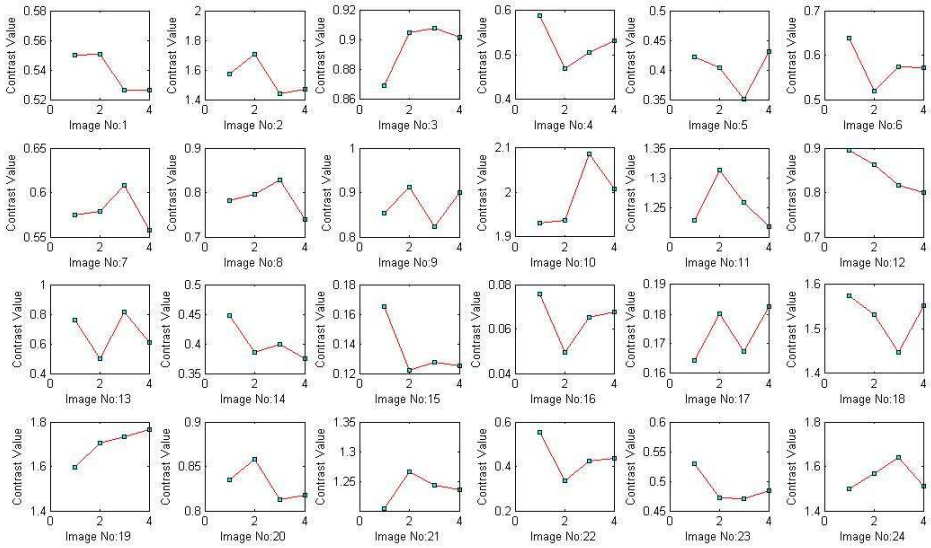


Figure 3.15 Contrast data. This shows the contrast data for the 24 original and synthesized images using the three algorithms. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

Table 3.1 Accuracy of the Proposed Method (Using 24 Images)

Method	Accuracy
Statistical Measure Based	62.5%
Correlogram Based	79.16%

Table 3.2 Accuracy of the Proposed Method (Using subset of MIT VisTex)

Method	Accuracy
Correlogram Based	69.04%

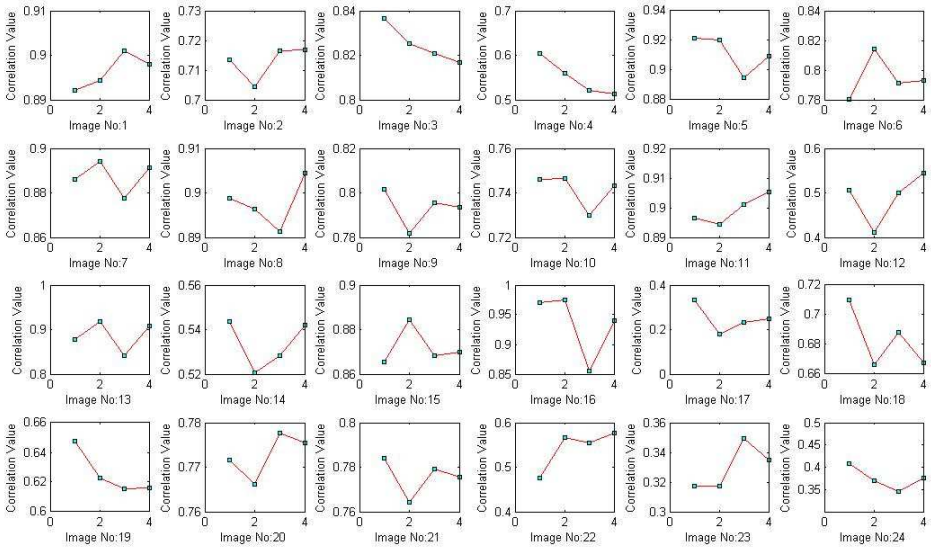


Figure 3.16 Correlation data. This shows the correlation data for the 24 original and synthesized images using the three algorithms. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

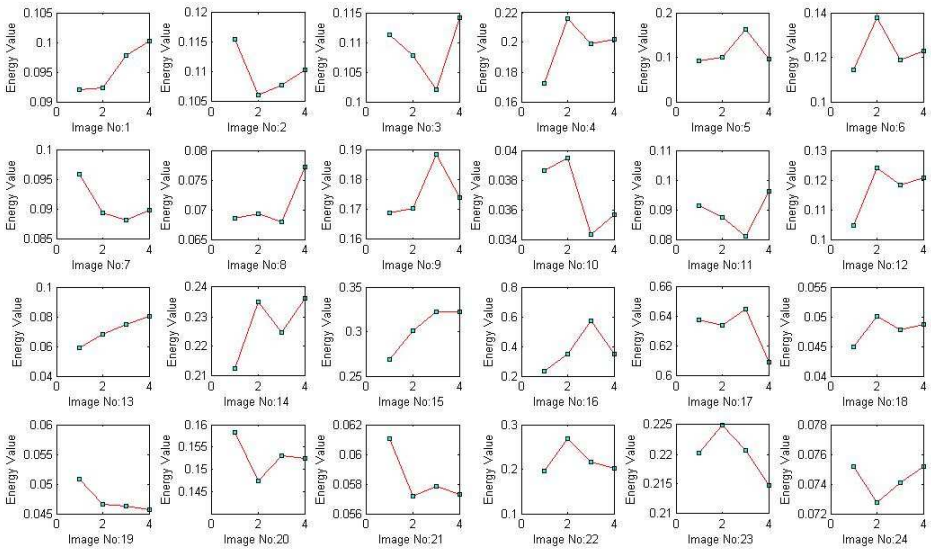


Figure 3.17 Energy data. This shows the energy data for the 24 original and synthesized images using the three algorithms. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

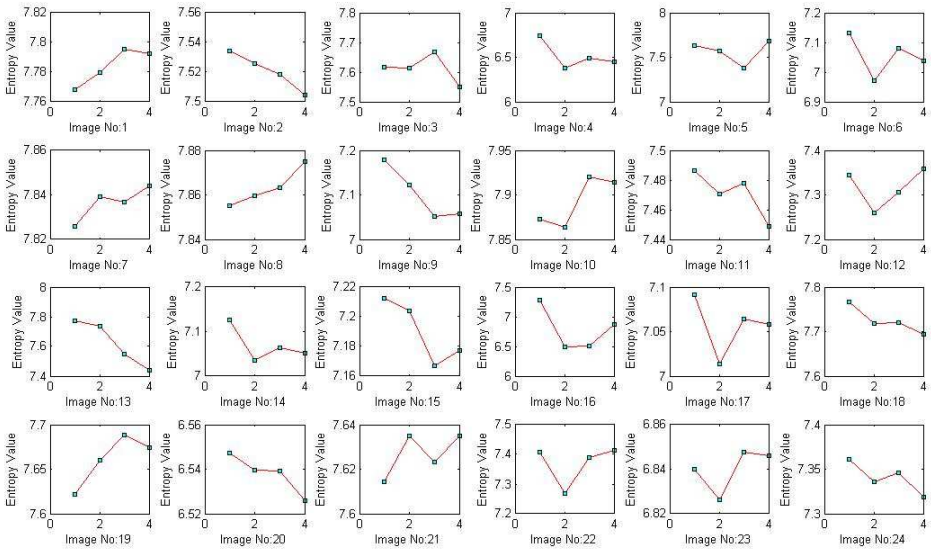


Figure 3.18 Entropy data. This shows the entropy data for the 24 original and synthesized images using the three algorithms. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

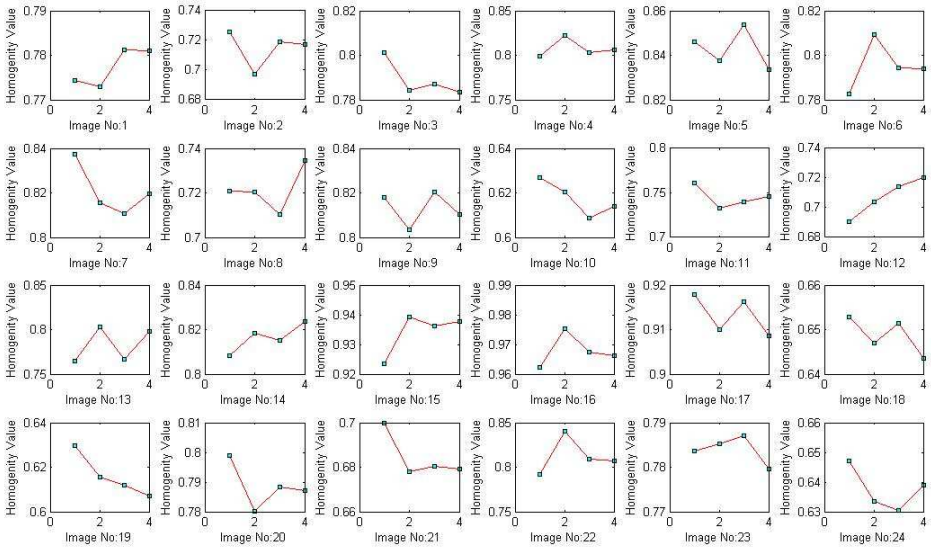


Figure 3.19 Homogenity data. This shows the homogenity data for the 24 original and synthesized images using the three algorithms. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

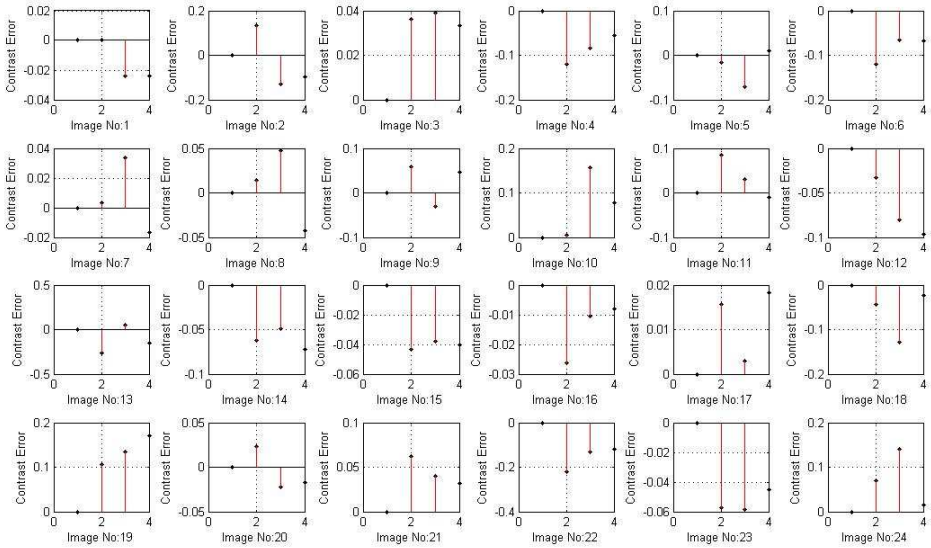


Figure 3.20 Contrast error data. This shows the contrast error data for the 24 original images. The input image is used as standard to calculate the error. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

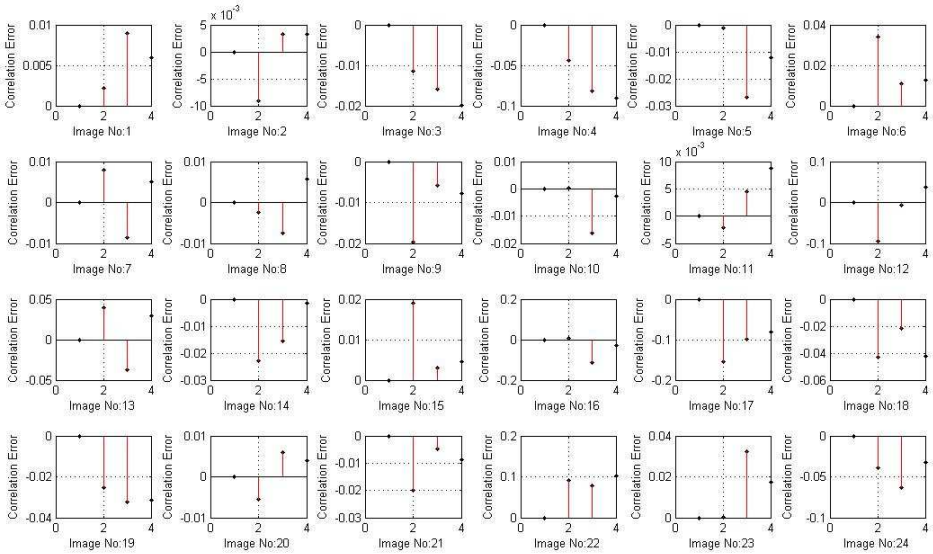


Figure 3.21 Correlation error data. This shows the correlation error data for the 24 original images. The input image is used as standard to calculate the error. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

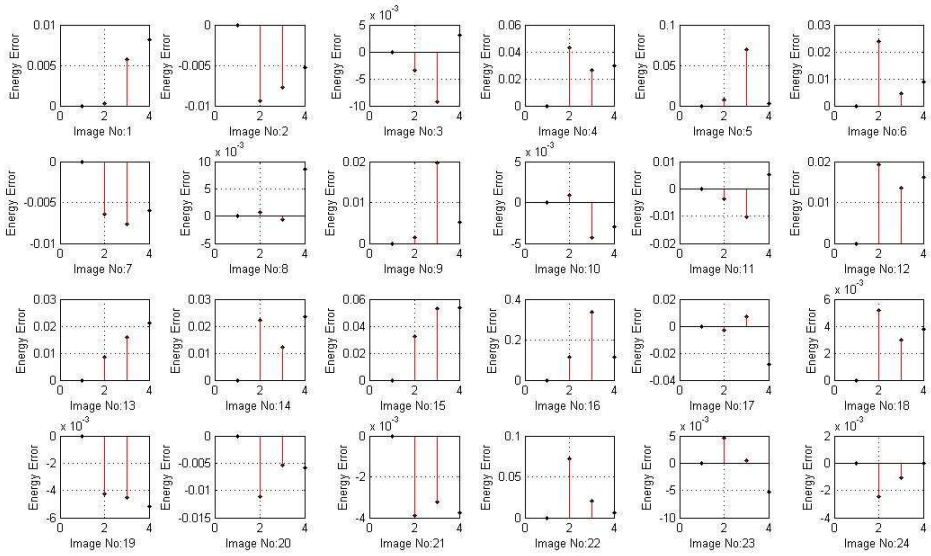


Figure 3.22 Energy error data. This shows the energy error data for the 24 original images. The input image is used as standard to calculate the error. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

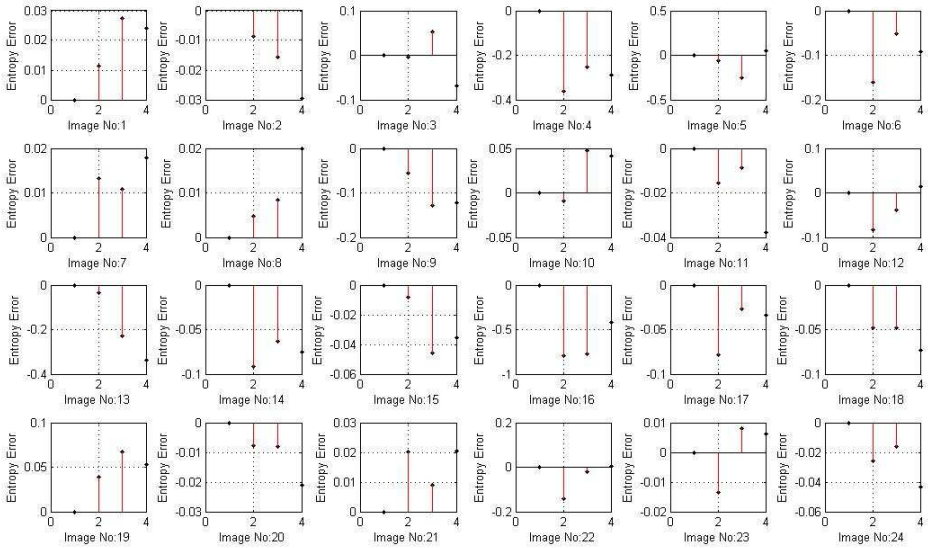


Figure 3.23 Entropy error data. This shows the entropy error data for the 24 original images. The input image is used as standard to calculate the error. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

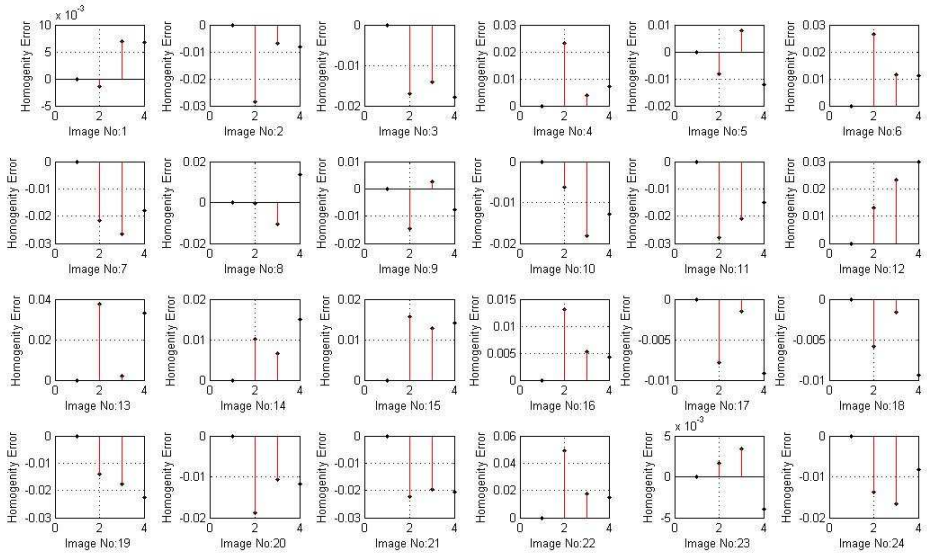


Figure 3.24 Homogenity error data. This shows the homogenity error data for the 24 original images. The input image is used as standard to calculate the error. The x axis refers to an image, image 1 is the original image, 2 the image synthesized using pixel based approach, 3 the image synthesized using patch based approach and 4 is the image synthesized using hybrid approach.

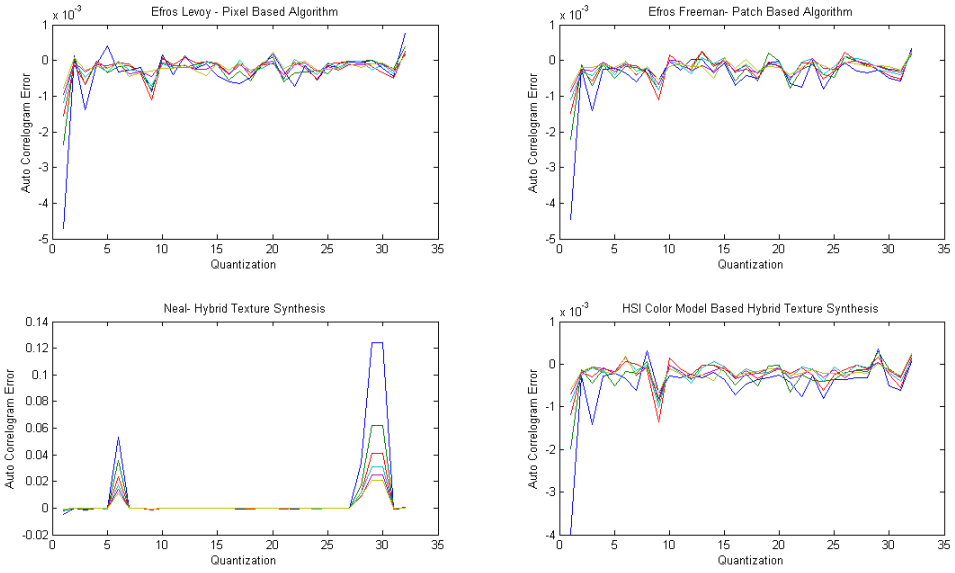


Figure 3.25 Sample output showing the autocorrelogram values for image number 2. The autocorrelogram values are show for distance k, where k=1,2,3,4,5,6.

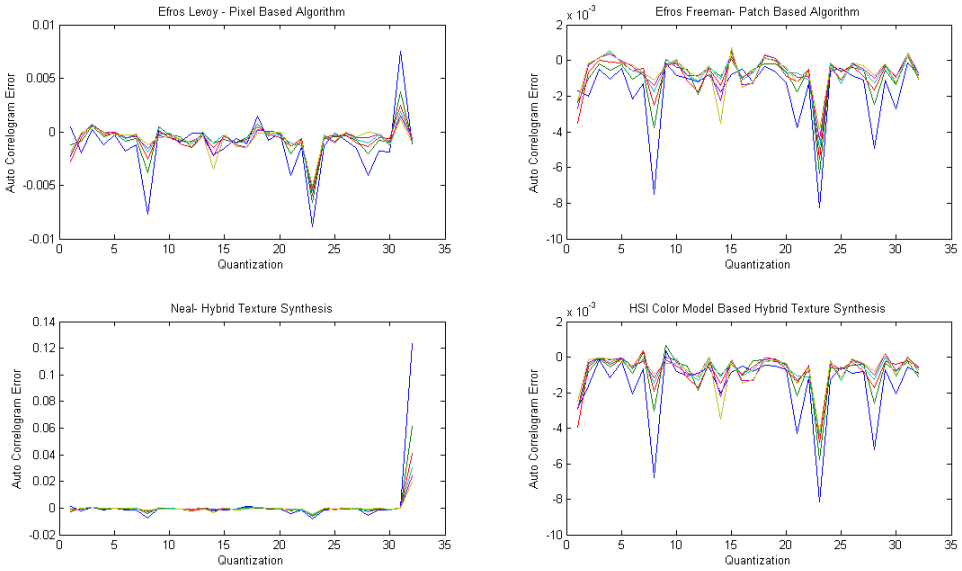


Figure 3.26 Sample output showing the autocorrelogram values for image number 15. The autocorrelogram values are show for distance k, where k=1,2,3,4,5,6.

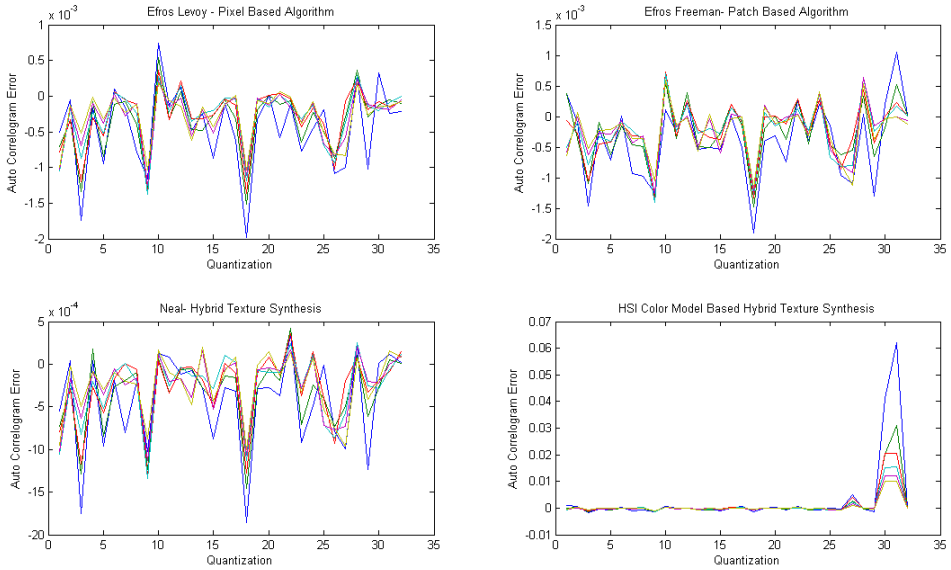


Figure 3.27 Sample output showing the autocorrelogram values for image number 17. The autocorrelogram values are show for distance k, where k=1,2,3,4,5,6.

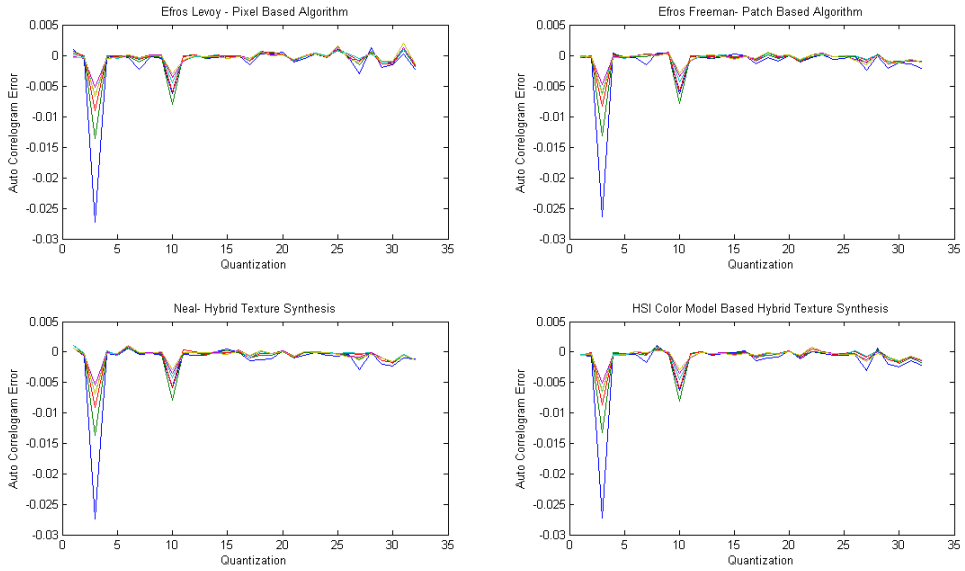


Figure 3.28 Sample output showing the autocorrelogram values for image number 6. The autocorrelogram values are show for distance k, where k=1,2,3,4,5,6.

Table 3.3 Output Comparison for 24 Input Images

Image Number	Data Based	Actual	Result
1	Ef-Le	Ef-Le	True
2	Tie	HTS	False
3	Tie	HTS	False
4	Ef-Fre	Ef-Fre	True
5	HTS	HTS	True
6	Ef-Fre	Ef-Fre	True
7	HTS	HTS	True
8	Ef-Le	Ef-Fre	False
9	Ef-Fre	Ef-Fre	True
10	Ef-Le	Ef-Fre	False
11	Tie	Ef-Fre	False
12	Tie	HTS	False
13	Tie	HTS	False
14	Ef-Le	Ef-Le	True
15	Ef-Fre	Ef-Fre	True
16	Ef-Le	Ef-Le	True
17	Ef-Fre	Ef-Fre	True
18	Ef-Fre	Ef-Fre	True
19	Ef-Le	Ef-Le	True
20	Tie	HTS	False
21	Ef-Fre	Ef-Fre	True
22	HTS	HTS	True
23	Tie	HTS	False
24	HTS	HTS	True

Table 3.4 Autocorrelogram Data Based Output Comparison

Image Number	Correlogram Based	Actual	Result
2	Tie	HTS	False
3	Ef-Fre	HTS	False
8	HTS	Ef-Fre	False
10	HTS	Ef-Fre	False
11	Ef-Fre	Ef-Fre	True
12	HTS	HTS	True
13	HTS	HTS	True
20	HTS	HTS	True
23	Ef-Fre	HTS	False

CHAPTER 4

IMPROVED INPAINTING USING MAXIMUM EDGE POINT DETECTOR

This chapter talks about the image inpainting method proposed and developed using maximum edge point detector method.

4.1 Introduction

Texture synthesis is an alternative way to create textures. In most cases exemplar texture synthesis algorithms have been used to generate a large image based on a smaller input image. The new image generated using this method has a similar structure and property as that of the input image, thus perceptively it appears to be a continuation of the same input image. The idea of texture synthesis has been extended and used in a novel method for fixing damaged images called inpainting. This is based on a large body of texture synthesis research, which seeks to replicate texture ad infinitum, given a small source sample of pure texture [4, 53, 54, 55, 56].

Image inpainting allows repair of damaged images. One of the critical step in inpainting is determining the order in which the damaged patch is fixed. Various algorithms have dealt with this problem and suggested different solutions. Initial image inpainting algorithms have been presented to repair small missed or damaged sections in an image [57, 58, 30], such as speckles, scratches and text in image. The texture synpaper method [6, 59] can effectively repair the damaged image, but is more suitable for repairing the image with single or simple texture. Since texture synpaper [6, 59] is based on a known texture sample, it is not appropriate for repairing images involving complex and unknown textures.

The most important issue for image in-painting of large missing area is to determine the structure information of these areas, and repair them seamlessly from existing surround-

ing information. Previous studies address this issue by solving two problems: analyzing the structural information from the remainder image (e.g., the contour and shape), and keeping the texture information in order to reduce the visual artifact of repaired image. The work of Bertalmio [58] and [30] adopted a method that utilizes partial differential equations to propagate the structure of the missing area, but they produces a vague repaired result if the missed area is large. The work of Jia [60] addressed color and texture segmentation, applying the tensor voting algorithm to segment the image for propagating the structure of the missed area. The resulting structure is used to identify the searching range and the filling order.

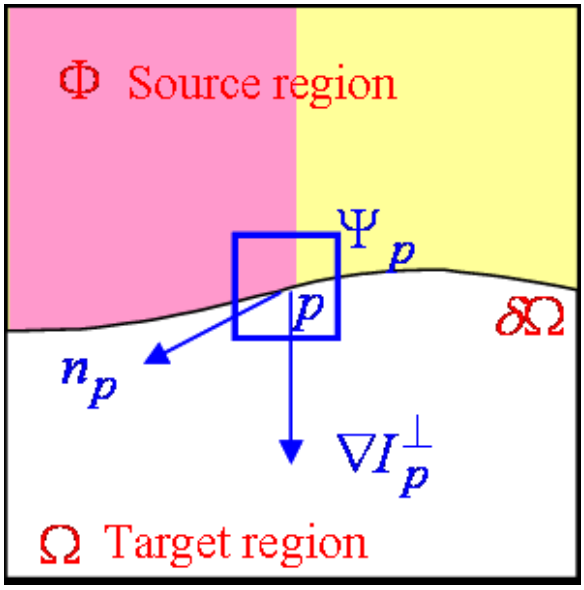


Figure 4.1 Schematic representation of an image showing the source region Φ, target region Ω and the contour $\delta\Omega$. The point p is a point on the contour, Ψ_p is a square patch, centered at p, $_p$ is the normal vector of the contour at p.∇ is the gradient operator and $\perp$ is an isophote orthogonal to the gradient.

Since two parameters control the smoothness of propagate curve, the algorithm can repair not only linear structures, but also non-linear structures, in a missed area. However, it takes a long time for color segmentation and structure propagation. To reduce the time taken, the missed area can be filled manually according the structure drawn. The algorithm

of Wang [61] and Rares [62] presented similar algorithms to Jia [60]. The method of Rares [62] adopted the constrained texture synthesis suggested by Wang [61] to analyze the texture distribution, and then defined the searching range for the filling image. Due to the visual artifacts in the repaired image, an algorithm was presented to detect the artifacts and re-fill these regions again. Rares [62] proposed an algorithm that first extracts the edges or corners of an image and then defines an order for edge permutation. However, the algorithm is only suitable for some structure distribution and produces a blur effect in the repaired image.

To repair an image effectively without apparent artifacts, Criminisi [63] adopted the magnitude of the gradient and undamaged area of an image to define the filling order and searching direction for finding the best patches. The advantage of this algorithm is that it is very suitable for repairing linear structures in the missed area, and it smoothly connects the repaired region with the nearby undamaged region.

The method of Criminisi [63] was extended by Chang [64] to modify the order determination method. They use a value, structure priority value different from the priority value used by Criminisi [63]. This method modifies the priority value to calculate the edge pixels in a patch called E(R). This value along with the confidence term helps determine the exact order of filling to repair a damaged image.

4.2 Image Inpainting

Image inpainting has long been one of the most interesting ways of repairing damaged images. The method suggested by Bertalmio [65] suggests an algorithm for the simultaneous filling-in of texture and structure in regions of missing image information is presented here. The basic idea is to first decompose the image into the sum of two functions with different basic characteristics, and then reconstruct each one of these functions separately with structure and texture filling-in algorithms. The first function used in the decomposition is of bounded variation, representing the underlying image structure, while the second

function captures the texture and possible noise. The region of missing information in the bounded variation image is reconstructed using image inpainting algorithms, while the same region in the texture image is filled-in with texture synthesis techniques. The original image is then reconstructed adding back these two sub-images. Another method suggested by Cheng [66] focuses on the development of a generic priority function to provide robust performance. However this algorithm requires user intervention to be applied to any image contents with different characteristics. The algorithm proposed by Li [67] uses a two pass matching algorithm for efficiency. Their method also reduces growing errorenous regions within the image.

Of the various inpainting algorithms Ciminisi's [63] and [68] makes a significant contribution in this field. Their work presents a novel and efficient algorithm that combines the advantages of two different approaches for object removal and region filling in images. The first approach is based on texture synthesis which generates large regions and the other is the inpainting approach which generates small regions. It is noted that exemplar-based texture synthesis contains the essential process required to replicate both texture and structure; the success of structure propagation, however, is highly dependent on the order in which the filling proceeds. This algorithm proposes a best-first algorithm in which the confidence in the synthesized pixel values is propagated in a manner similar to the propagation of information in inpainting. Figure 4.1 shows a schematic representation of an image I with the source region, target region and the contour of the target region.

Many algorithms have used the approach suggested by Criminisi [63] and further improved on it. The method of Nie [69] uses a similarity based approach. In this method, a more robust method is adopted to compute the filling order, which overcomes the problems arising when the image gradients of some pixels on the source region contour are just zeros. Another method suggested by Wang [61] uses the inpainting method for smart cameras.

4.3 Existing Methods

4.3.1 Previous Work

As explained in the work of Bertalmio [30] classical image denoising algorithms do not apply to image inpainting. In common image enhancement applications, the pixels contain both information about the real data and the noise (e.g., image plus noise for additive noise), while in image inpainting, there is no significant information in the region to be inpainted. The information is mainly in the regions surrounding the areas to be inpainted. There is then a need to develop specific techniques to address these problems.

Mainly three groups of works can be found in the early literature related to digital inpainting. The first one deals with the restoration of films, the second one is related to texture synthesis, and the third one, a significantly less studied class is related to disocclusion.

Kokaram et al. [70] use motion estimation and autoregressive models to interpolate losses in films from adjacent frames. The basic idea is to copy into the gap the right pixels from neighboring frames. The technique can not be applied to still images or to films where the regions to be inpainted span many frames.

Hirani and Totsuka [71] combined frequency and spatial domain information in order to fill a given region with a selected texture. This is a very simple technique that produces incredible good results. On the other hand, the algorithm mainly deals with texture synthesis (and not with structured background), and requires the user to select the texture to be copied into the region to be inpainted. For images where the region to be replaced covers several different structures, the user would need to go through the tremendous work of segmenting them and searching corresponding replacements throughout the picture. Although part of this search can be done automatically, this is extremely time consuming and requires the non-trivial selection of many critical parameters, e.g., [6]. Other texture synthesis algorithms, e.g., [6, 39, 41], can be used as well to re-create a pre-selected texture to fill-in a (square) region to be inpainted.

In the group of disocclusion algorithms, a pioneering work is described in [72]. The authors presented a technique for removing occlusions with the goal of image segmentation. The basic idea is to connect T-junctions at the same gray-level with elastica minimizing curves. The technique was mainly developed for simple images, with only a few objects with constant gray-levels, and will not be applicable for the examples with natural images presented later in this paper. Masnou and Morel [26] recently extended these ideas, presenting a very inspiring general variational formulation for disocclusion and a particular practical algorithm (not entirely based on PDEs) implementing some of the ideas in this formulation. The algorithm performs inpainting by joining with geodesic curves the points of the isophotes (lines of equal gray values) arriving at the boundary of the region to be inpainted. As reported by the authors, the regions to be inpainted are limited to having simple topology, e.g., holes are not allowed. In addition, the angle with which the level lines arrive at the boundary of the inpainted region is not (well) preserved: the algorithm uses straight lines to join equal gray value pixels.

In more recent work, several researchers have considered texture synthesis as a way to fill large image regions with pure textures repetitive two-dimensional textural patterns with moderate stochasticity. This is based on a large body of texture synthesis research, which seeks to replicate texture ad infinitum, given a small source sample of pure texture. The ones of particular interest are exemplar-based techniques which cheaply and effectively generate new texture by sampling and copying colour values from the source. One of the earliest work done with replacing image sections with texture synthesis was done by Harrison [73].

As effective as these techniques are in replicating consistent texture, they have difficulty filling holes in photographs of realworld scenes, which often consist of linear structures multiple textures interacting spatially. The main problem is that boundaries between image regions are a complex product of mutual influences between different textures. In

constrast to the two-dimensional nature of pure textures, these boundaries form what might be considered more one-dimensional, or linear, image structures.

A number of algorithms specifically address the image filling issue for the task of image restoration, where speckles, scratches, and overlaid text are removed. These image inpainting techniques fill holes in images by propagating linear structures (called isophotes in the inpainting literature) into the target region via diffusion. They are inspired by the partial differential equations of physical heat flow, and work convincingly as restoration algorithms. Their drawback is that the diffusion process introduces some blur, which becomes noticeable when filling larger regions.

4.3.2 Criminisi Algorithm

The core of Criminisi's algorithm [63] is an isophote-driven image sampling process. It is well-understood that exemplar-based approaches perform well for two-dimensional textures. But, it is to be noted that in addition that exemplar-based texture synthesis is sufficient for propagating extended linear image structures, as well i.e., a separate synthesis mechanism is not required for handling isophotes. The filling order in the repairing process is determined by the gradient information of an exemplar. Figure 4.1 illustrates a single iteration in filling the target region centered at point p. An image I is divided into disjoint regions, the target region Ω and the source region Φ. The target region is the selection area that needs to be filled in, and the corresponding contour is denoted as $\delta\Omega$. An image excluding the target region and the contour of is defined as the source region, i.e.$\Phi = I - \Omega$. Ψ_p denotes a square patch centered at a point p, where p is located at $\delta\Omega$.

A key part of the inpainting process is to determine the filling order for the target region. The priority value of each square patch $P(p)$ is adopted to find the filling order for all unfilled pixels. The priority value is defined as the product of two terms

$$P(p) = C(p) \cdot D(p) \tag{4.1}$$

where $C(p)$ and $D(p)$ are called confidence term and data term, respectively. These terms are defined as follows

$$C(p) = \frac{\sum_{q \in \Psi_P \cap (I-\Omega)} C(q)}{|\Psi_p|} \tag{4.2}$$

$$D(p) = \frac{\left|\nabla I_p^{\perp} \overrightarrow{n}_p\right|}{\alpha} \tag{4.3}$$

where $|\Psi_p|$ denotes the area of Ψ_p and α is a normalized parameter.

4.3.3 Enhanced Examplar Based Algorithm

The work described by Chang [64] builds on the method described by Criminisis algorithm [63] to improve the image inpainting algorithm. This method changes the method for determining the filling order. Instead of using the priority value they use what is called the structure priority value which is defined as follows

$$SP(p) = C(p) \times ER(p) \tag{4.4}$$

where $C(p)$ is the confidence value that determines the reliability of the corresponding patch and $ER(p)$ denotes the edge value. The term $ER(p)$ is defined as the ratio of edge points relative to the source region inside the patch, and the equation is as follows

$$ER(p) = \frac{\sum_{q \in \Psi_p \cap \Phi} e(q)}{\sum_{q \in \Psi_p \cap \Phi} C(q)} \tag{4.5}$$

$$e(q) = \begin{cases} 1 \; if \; q \in edge \\ 0 \; otherwise \end{cases} \tag{4.6}$$

where $e(q)$ denotes whether pixel p is an edge point. For the detection of the edge value this algorithm uses the canny edge detector.

4.4 Proposed Method

The method builds on the idea introduced by the enhanced examplar based method. As with the existing algorithms the proposed algorithm is a two step process which involves finding the filling order of the target region and then determining the best match for that patch from the existing source region. As with most algorithms it also focuses on the determination of the filling order to effectively remove objects and fill gaps in images. Using the structure priority value from [64] the algorithm has

$$SP(p) = C(p) \times ERI(p) \tag{4.7}$$

where $C(p)$ is the confidence value that determines the reliability of the corresponding patch and $ERI(p)$ defines the modified edge value which is defined as follows

$$ERI(p) = \frac{\max(S_r)}{\sum_{q \in \Psi_p \cap \Phi} C(q)} \tag{4.8}$$

where S_r is defined as follows

$$S_r = \sum_{q \in \Psi_p \cap \Phi} e(q) \tag{4.9}$$

Here the value r varies based on the number of edge detection algorithm used. Different edge detection algorithms provide results with varying accuracy. This is a key issue in detecting the contours accurately. The proposed method uses more than one edge detection algorithm where r represents the number of algorithm used. When all structure priority values of the patches along $\delta\Omega$ have been computed, the patch with the maximum

priority value is selected as the first one to be filled. If $\Psi_{\hat{p}}$ is with the maximum priority value after the filling order determination, the next step is to search for a patch around the source region that is most similar to $\Psi_{\hat{p}}$ according to the following equation

$$\Psi_{\widehat{q}} = \arg\min_{\Psi_q \in \Phi} d(\Psi_{\widehat{p}}, \Psi_q) \tag{4.10}$$

where the distance $d(\Psi_{\hat{p}}, \Psi_q)$ between two patches $\Psi_{\hat{p}}$ and Ψ_q is defined as sum of square different (SSD) of the already repaired pixels in this two patches. After a patch has been filled, the confidence is updated as follows:

$$C(p) = \gamma C(\hat{p}) \forall p \in \Psi_{\hat{p}} \cap \Omega \tag{4.11}$$

4.5 Experimental Results

In this section the proposed method has been tested on a wide variety of image set comprising both natural and computer generated images. The natural images are complex images with a multitude of textures. The computer generated images incorporate a broad band of colors from the color spectrum and strong edges and contours. The results have been compared with the results obtained by the algorithm in [64] and have shown a comparative result for our images. In addition for the purpose of testing the value of r =3 is set thereby using three different edge detection algorithms. The three algorithms selected by are the canny, the prewitt and the sobel edge operators. The results have been discussed with a subset of four images tested using this algorithm, two from each class. All the experiments are performed on a 1.5GHz Core 2 Duo machine with a 2GB RAM. Figure 4.2 and Figure 4.3 show the result of testing two computer generated images using the proposed algorithm and then comparing them to the output produced by [64]. Figure 4.2(a) and Figure 4.3(a) shows the original image, Figure 4.2(b) and Figure 4.3(b) shows the damaged

image, Figure 4.2(c) and Figure 4.3(c) the result produced by [64] and finally Figure 4.2(d) and Figure 4.3(d) shows the result produced by the proposed method.

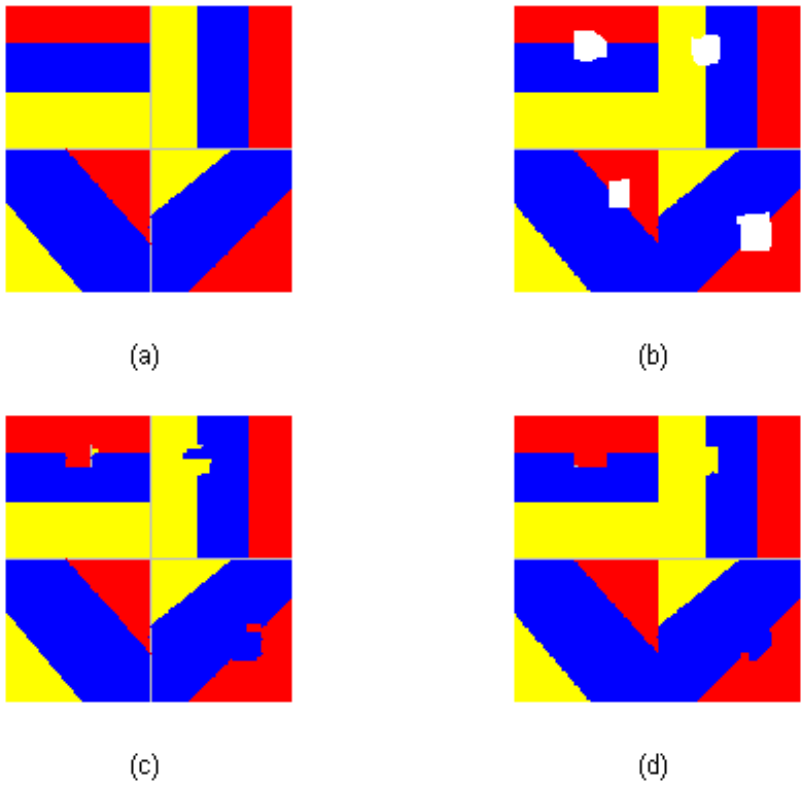


Figure 4.2 Test image. A synthetic image generated using four regions with three colors. The subfigure (a) shows the original image. The subfigure (b) shows the corrupted image which needs to be repaired. The subfigure (c) shows the results using Chang's algorithm. The subfigure (d) shows the results using proposed algorithm.

Figure 4.2(d) shows the result generated by the proposed method. The section in lower left corner is restored accurately using both algorithms. The size of the section being small it poses no challenge to either algorithm. However with larger patches the problem becomes significant and the accuracy of the algorithms come into play. As is evident from Figure 4.3(d) the results produced by the proposed method shows significantly better output than that produced by Chang [64]. On closer examination of the output produced by the proposed method it is observed that though the quality is significantly better there still

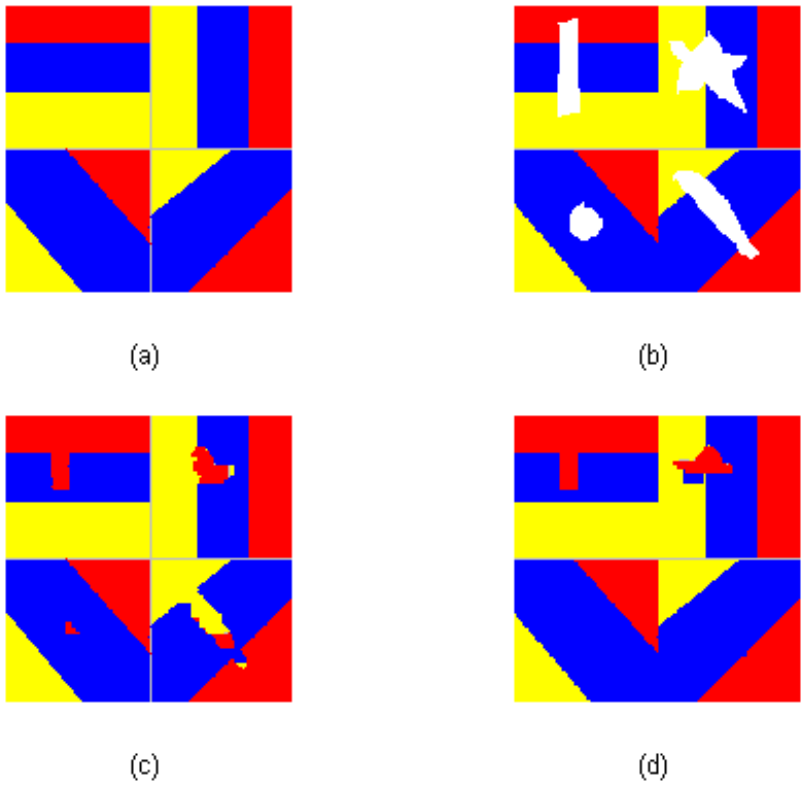


Figure 4.3 Test image. A synthetic image generated using four regions with three colors. The subfigure (a) shows the original image. The subfigure (b) shows the corrupted image which needs to be repaired. The subfigure (c) shows the results using Chang's algorithm. The subfigure (d) shows the results using proposed algorithm.

exists errors and due to the nature of the image and the colors used the errors look more obvious. However in Figure 4.3(d) it is seen that a large section of the image removed from the lower right corner is restored accurately as opposed to the output generated in Figure 4.3(c). However the sections in the top right and top left corners have not been restored accurately and their output is almost comparable to the output produced by [64]. Figure 4.5 and Figure 4.6 show the result of testing two real images using the proposed algorithm and then comparing them to the output produced by [64]. As in the earlier examples Figure 4.5(a) and Figure 4.6(a) shows the original image, Figure 4.5(b) and Figure 4.6(b) shows a foreground section of the image which is to be removed, Figure 4.5(c)

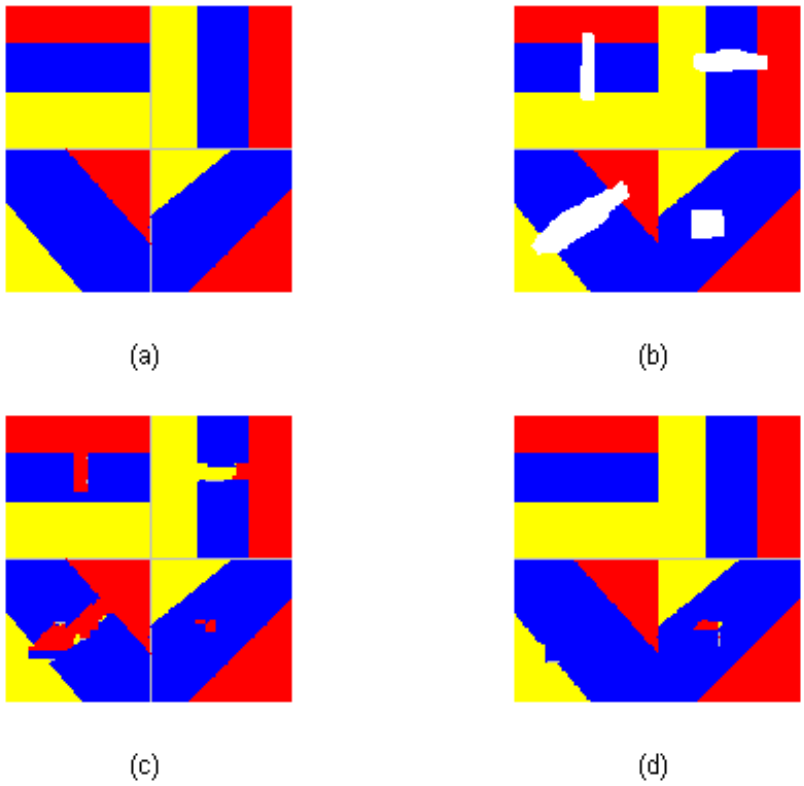


Figure 4.4 Test image. A synthetic image generated using four regions with three colors. The subfigure (a) shows the original image. The subfigure (b) shows the corrupted image which needs to be repaired. The subfigure (c) shows the results using Chang's algorithm. The subfigure (d) shows the results using proposed algorithm.

and Figure 4.6(c) the result produced by [64] and finally Figure 4.5(d) and Figure 4.6(d) shows the result produced by the proposed method. Figure 4.5 shows an image where the people in the foreground need to be removed from the image leaving the image of the sea behind. Figure 4.5(c) shows the result where a section of the image has been altered using [64] and we see a section of the sky has been wrongly copied; this error has been corrected using the proposed method

Figure 4.6 shows an image with a person standing against a rock-face. Figure 4.6(c) shows the person removed using [64] but instead of a continuous rocky background the result shows a section of the sand being copied in. This error is corrected using the proposed

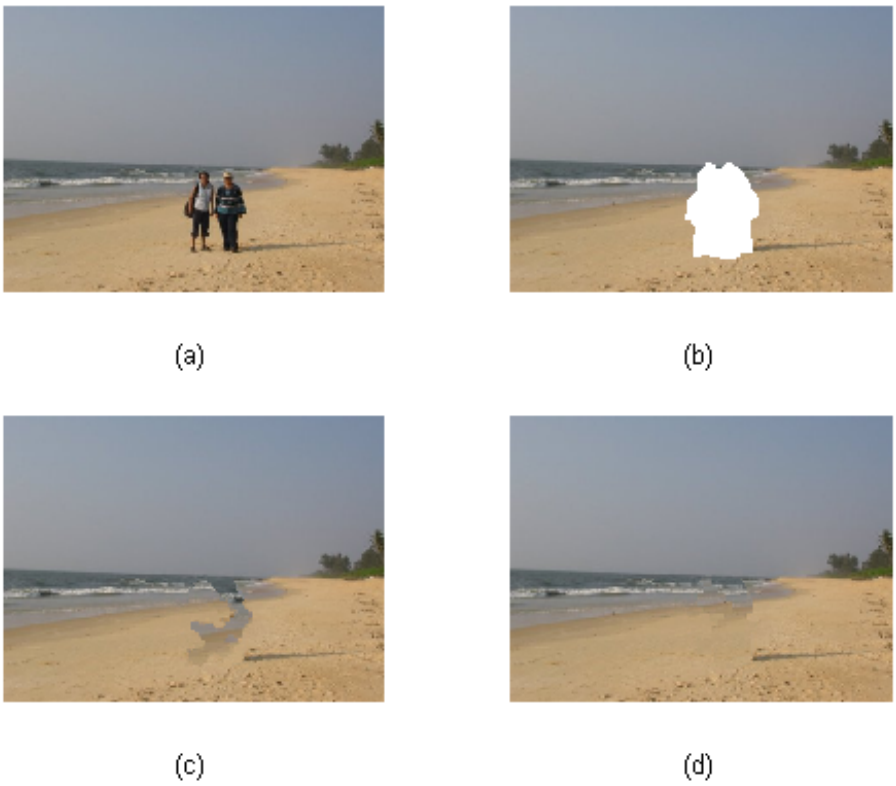


Figure 4.5 Test image. A natural image showing a scene in the background and people in the foreground. The subfigure (a) shows the original image. The subfigure (b) shows the image with a section removed which needs to be filled with the background. The subfigure (c) shows the results using Chang's algorithm. The subfigure (d) shows the results using proposed algorithm.

method in Figure 4.6(d) resulting in a more consistent background. The algorithm used in [64] uses the canny edge detector to detect the edges. As is evident when the value max(S_r) is the value calculated by the canny edge operator, equation (8) is same as equation (5). This will lead to identical results for certain images. However the particular set of images set for testing did not produce any identical results

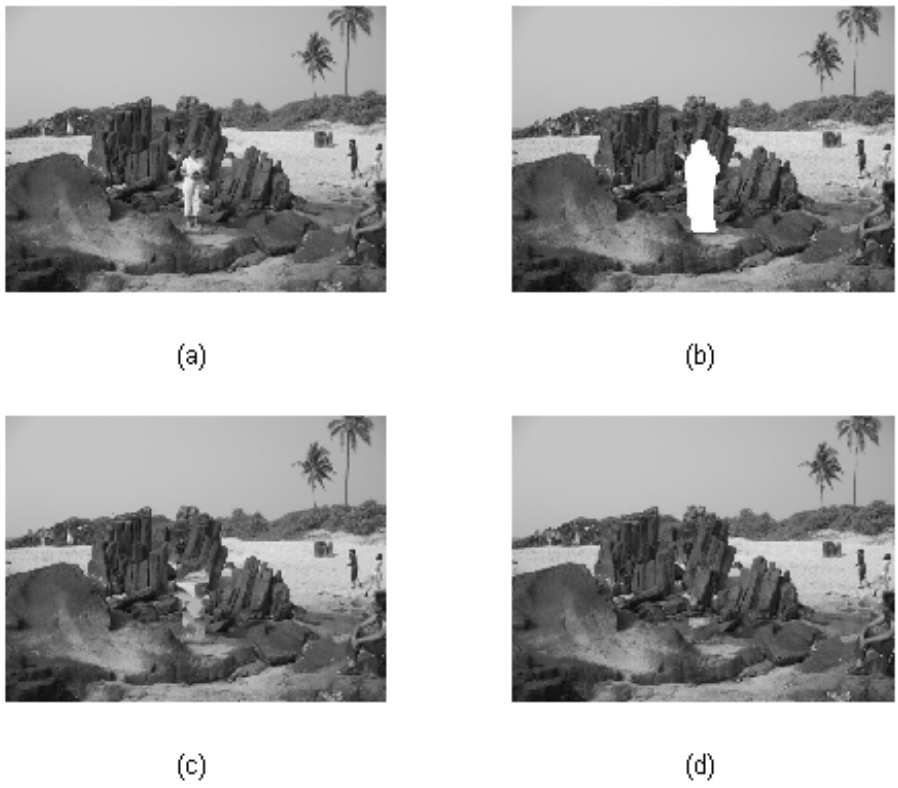


Figure 4.6 Test image. A natural image showing a scene in the background and a human being in the foreground. The subfigure (a) shows the original image. The subfigure (b) shows the image with a section removed which needs to be filled with the background. The subfigure (c) shows the results using Chang's algorithm. The subfigure (d) shows the results using proposed algorithm.

4.6 Conclusions

The proposed method is based on using an edge detection algorithm which will provide a clear contour of the edges. By using the max(S_r) value a significant improvement is made in determining the number of edge pixels thus providing a higher accuracy in detecting the edges. Since determining the filling order is critical in accurately inpainting an image the proposed method gives significantly improved results as compared to the existing algorithms. The next phase of the work presented in this chapter will involve

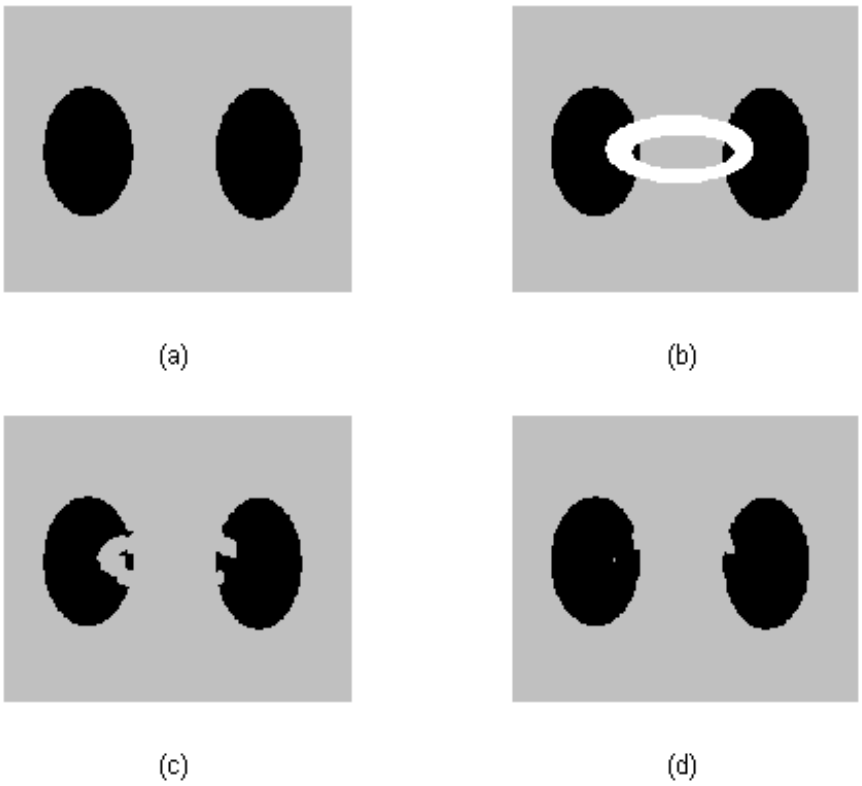
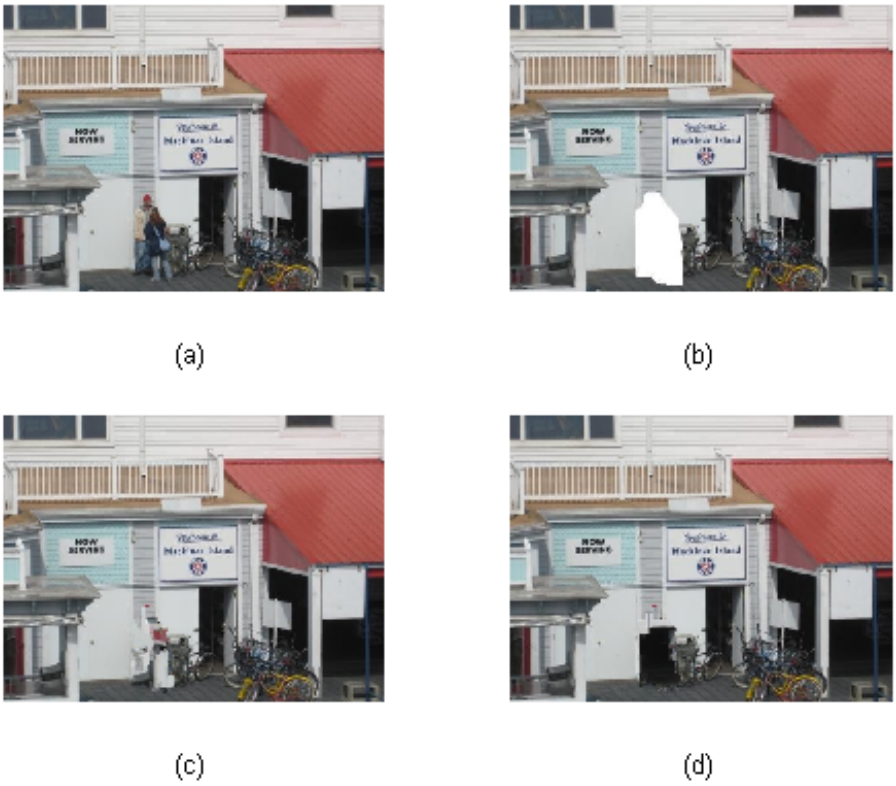


Figure 4.7 Test image. A synthetic gray scale image showing two ellipses. The subfigure (a) shows the original image. The subfigure (b) shows the image with a section removed due to corruption. The subfigure (c) shows the results using Chang's algorithm. The subfigure (d) shows the results using proposed algorithm.

further improvement of the structural priority value to refine the filling order. In addition to focusing on the edge ratio the work also intends improve the confidence value.



(a) (b)

(c) (d)

Figure 4.8 Test image. A natural image showing a scene in the background and a human being in the foreground. The subfigure (a) shows the original image. The subfigure (b) shows the image with a section removed which needs to be filled with the background. The subfigure (c) shows the results using Chang's algorithm. The subfigure (d) shows the results using proposed algorithm.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this dissertation, advanced algorithms are designed and implemented for high quality texture generation using sample input texture, quantification of texture quality based on an autocorrelogram and effective inpainting using a maximum value edge detector. These algorithms can be used for generating textures used in a large number of commercial tools, automatic quantification of image quality and restoration and effective manipulation of digital images.

In the first case, an enhanced hybrid texture synthesis method is proposed and implemented, which uses a HSI color model instead of the standard RGB color model to create large textures from a provided input sample. The HSI color model is closer to human perception as it represents colors similarly to how the human eye senses colors. Used in conjunction with the hybrid texture synthesis approach, it proves more effective in accurately reproducing local and global features of the texture in such a manner that the larger texture created appears to be a continuation of the provided input texture. The new color model is better than the one used in the earlier algorithms to preserve features and reproduce the texture.

In future work a hybrid color model for enhancing hybrid texture synthesis will be used. It will help determine the features in underlying textures irrespective of the texture type. Hybrid color models have been effectively used in various fields of image processing and computer vision very effectively to achieve what traditional color models have been unable to. It is also possible to extend the work to allow texture synthesis for 3D object surfaces and video synthesis. By using GPU driven hardware, the texture synthesis method can be significantly improved and made more efficient.

In the second case, an efficient quantification method is proposed and implemented. This combines gray-scale statistical measures with a color autocorrelogram to efficiently choose the best texture output generated. This method generates a large number of intermediate data values. The entire bunch of data is used to generate error data and use that for determination of the best match. This is a two-step method which involves using first the gray-scale statistical measures and then the color correlogram. To determine the accuracy of the method, user input is used to check validity of the output. However, as always the user input on the best texture is inconsistent and leaves the accuarcy test a little subjective. Each user provides a different input while choosing the best generated texture. The color correlogram, however, proves to be a good measure for color textures, and when the distance measure is chosen intelligently, it results in good output. Reducing the error to a minimum is an effective approach though not the most elegant one.

In future work a special dataset will be used, which consists of entirely tordoial images. Since real life textures are almost never tordoial, the data set would consist of synthetic textures, allowing to accurately measure the statistical values and comparing against the texture generated using the tordoial image. This would completely eliminate the need of human intervention and would give a better measure of the accuracy of the proposed method. With the accuracy proved, this methodology can be used to quantify real life textures. Moreover, the autocorrelogram approach can be further improved using dynamic programming and also by using the feature to select the best match using a SVM.

In the third case, an improved inpainting algorithm is proposed and implemented, which uses a maximum edge value detector to perform inpainting. Determining edges accurately plays a crucial role in the inpainting process, and the proposed method uses a modified edge value. A better choice of the edges helps determine which region of the surrounding area needs to be copied onto the removed section. The method, however, improves only on the edge value but not the confidence value associated.

In the future work, the confidence value will be improved ensuring that the structured priority values show a better improvement. In addition, future work will also involve improving the quality of the output for synthetic images, which so far have not shown as significant an improvement as real life images.

REFERENCES

[1] M. Sonka., V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision, 3 edition*. CL Engineering, USA, 2007.

[2] E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surface*. Phd dissertation, University of Utah, Salt Lake City, Utah, 1974.

[3] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, October 1976.

[4] M. Ashikhmin. Synthesizing Natural Textures. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[5] L. Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. ACM SIGGRAPH, ACM Press*, pages 479–488, 2000.

[6] A. A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Proc. IEEE International Conference on Computer Vision*, pages 1033–1038, 1999.

[7] L. Liang, C. Liu, Y. Xu, B. Guo, and H. Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20:127–150, 2001.

[8] Wen-Chieh Lin, J. H. Hays, C. Wu, V. Kwatra, and Y. Liu. Quantitative evaluation on near regular texture synthesis. In *Computer Vision and Pattern Recognition Conference (CVPR '06)*, volume 1, pages 427–434, Jun 2006.

[9] K. B. Eom. Synthesis of color textures for multimedia applications. *Multimedia Tools and Applications*, 12:81–98, 2000.

[10] E. Angel. *Interactive Computer Graphics: A Top-Down Approach with OpenGL* . Addison Wesley, New York, NY, USA, 3rd edition, July 2002.

[11] E. Angel. *Interactive Computer Graphics: A Top-Down Approach with OpenGL.* Addison Wesley, New York, NY, USA, 5th edition, April 2008.

[12] MIT Media Lab. Vision texture. http://vismod.media.mit.edu/vismod/imagery/visiontexture/, 1995.

[13] B. Grunbaum and G. C. Shephard. *Tilings and Patterns*. W. H. Freeman and Company, New York, 1987.

[14] D. Schattschneider. The plane symmetry groups: Their recognition and notation. *The American Mathematical Monthly*, 85:439–450, 1978.

[15] Y. Liu, Wen-Chieh Lin, and J. Hays. Near-regular texture analysis and manipulation, 2004.

[16] D. Cano and T. H. Minh. Texture synthesis using hierarchical linear transforms. *Signal Processing*, 15(2):131–148, Sep 1988.

[17] M. Porat and Y. Y. Zeevi. Localized texture processing in vision: Analysis and synthesis in gaborian space. In *IEEE Trans. on Biomedical Engineering*, volume 36, pages 115–129, 1989.

[18] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH 2001 Conference*, pages 341–346, 2001.

[19] C. E. Shannon. A mathematical theory of communication. *The Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[20] C. A. Bouman. Markov random fields and stochastic image models. In *Tutorial at Proc. of IEEE International Conference on Image Processing*, Oct 1995.

[21] G. Turk. Texture synthesis on surfaces. In *Proc. ACM SIGGRAPH*, pages 347–354, Aug 2001.

[22] Greg Turk. Re-tiling polygonal surfaces. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 55–64, New York, NY, USA, 1992. ACM.

[23] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin. Texture and shape synthesis on surfaces. In *Proc. 12th Eurographics Workshop on Rendering Techniques*, pages 301–312, 2001.

[24] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proc. SIGGRAPH , Computer Graphics, Annual Conference Series*, pages 465–470, 2000.

[25] B. Guo, H. Shum, and Y. Q. Xu. Chaos mosaic: Fast and memory efficient texture synthesis. Technical Report MSR-TR-2000-32, Microsoft Research, 2000.

[26] S. Masnou and J. M. Morel. Level lines based disocclusion. In *Proc. International Conference on Image Processing*, pages 259–263, 1998.

[27] T. F. Chan and J. Shen. Non-texture inpainting by curvature-driven diffusions (cdd). *Journal of Visual Comm. Image Rep*, 12:436–449, 2001.

[28] C. Ballester, V. Caselles, J. Verdera, M. Bertalmio, and G. Sapiro. A variational model for filling-in gray level and color images. In *Proc. International Conference on Computer Vision*, pages 10–16, 2001.

[29] A. Zalesny, V. Ferrari, G. Caenen, and L. van Gool. Parallel composite texture synthesis. In *Texture 2002 workshop - ECCV*, June 2002.

[30] M. Bertalmio, G. Sapiro, and V. Caselles. Image inpainting. In *Proc. ACM Conference on Computer Graphics (SIGGRAPH)*, pages 417–424, 2000.

[31] S. Walden. *The Ravished Image*. St. Martins Press, New York, 1985.

[32] G. Emile-Male. *The Restorers Handbook of Easel Painting*. Van Nostrand Reinhold, New York, 1976.

[33] A. Nealen and M. Alexa. Hybrid texture synthesis. In *Proc. Eurographics Symposium on Rendering*, 2003.

[34] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 287–294, New York, NY, USA, 2003. ACM.

[35] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 795–802, New York, NY, USA, 2005. ACM.

[36] V. Kwatra, S. Lefebvre, G. Turk, and L. Y. Wei. Example-based texture synthesis, 2007.

[37] K. Popat and R. W. Picard. Novel cluster-based probability model for texture synthesis, classification, and compression. In *Visual Communications and Image Processing*, pages 756–768, 1993.

[38] R. Paget and I. D. Longstaff. Texture synthesis via a noncausal nonparametric multiscale markov random field, 1998.

[39] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *Proc. ACM SIGGRAPH*, pages 229–238, 1995.

[40] S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (frame) - towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):1–20, 1998.

[41] E. P. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *IEEE Computer Society*, pages 4–7, 1998.

[42] L. Y. Wei. Deterministic texture analysis and synthesis using tree structure vector quantization. In *Proc. of the XII Brazilian Symposium on Computer Graphics and Image Processing*, SIBGRAPI '99, pages 207–214, Washington, DC, USA, 1999. IEEE Computer Society.

[43] S. Zelinka and M. Garland. Towards real-time texture synthesis with the jump map. In *Proc. Eurographics Workshop on Rendering, Eurographics Association*, pages 99–104, 2002.

[44] R. Gonzalez and R. Woods. *Digital image processing*. Prentice-Hall Inc., Upper Saddle River, NJ, 2002.

[45] J. Huang, S. R. Kumar, Mandar M. Mitra, W. J. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proc. of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 762–, Washington, DC, USA, 1997.

[46] J. Huang, S. R. Kumar, and M. Mitra. Combining supervised learning with color correlograms for content-based image retrieval. In *Proc. of the fifth ACM international conference on Multimedia*, MULTIMEDIA '97, pages 325–334, New York, NY, USA, 1997.

[47] J. Huang, S. R. Kumar, Mandar M. Mitra, W. J. Zhu, and R. Zabih. Spatial color indexing and applications. *International Journal of Computer Vision*, 35(3):245–268, 1999.

[48] J. Huang, S. R. Kumar, and R. Zabih. An automatic hierarchical image classification scheme. In *Proc. of the sixth ACM international conference on Multimedia*, pages 219–228, New York, NY, USA, 1998.

[49] J. Huang. *Color-Spatial Image Indexing and Applications*. Phd dissertation, Cornell University, Itahca, New York, 1998.

[50] R. M. Haralick. Statistical and structural approaches to texture. In *Proc. of the IEEE*, pages 786–804, May 1979.

[51] R.M. Haralick, Dinstein, and K. Shanmugam. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(3):610–621, Nov 1973.

[52] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision: Vol 1*. Addison-Wesley, Boston, Massachusetts, USA, 1992.

[53] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proc. SIGGRAPH, Annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001.

[54] H. Igehy and L. Pereira. Image replacement through texture synthesis. In *Proc. International Conference on Image Processing*, pages 186–189, 1997.

[55] W. T. Freeman and E. C. Pasztor. Learning low-level vision. *International Journal of Computer Vision*, 40:25–47, 2000.

[56] D. Garber. *Computational Models for Texture Analysis and Texture Synthesis*. Phd dissertation, University of Southern California, Los Angeles, California, 1981.

[57] A. Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 1:23–34, 2004.

[58] M. Bertalmio, A.L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 355–362, Dec 2001.

[59] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. In *Proc. ACM Transactions on Graphics*, volume 22(3), Jul 2003.

[60] J. Jia and C. K. Tang. Image repairing: robust image synpaper by adaptive nd tensor voting. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 643–650, Jun 2003.

[61] J. F. Wang, H. J. Hsu, and S. C. Liao. A novel framework for object removal from digital photograph. In *Proc. IEEE International Conference on Image Processing*, volume 2, pages 73–76, Sep 2005.

[62] A. Rares, M.J.T. Reinders, and J. Biemond. Edge-based image restoration. In *Proc. IEEE Transactions on Image Processing*, volume 14, pages 1454–1468, Oct 2005.

[63] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 721–728, Jun 2003.

[64] I. C. Chang and C. W. Hsu. Image inpainting using an enhanced exemplar-based algorithm. *Multimedia Security: Watermarking, Steganography, and Forensics edited by F.Y. Shih, CRC Press*, 2012.

[65] M. Bertalmio, L. Vese, G. Sapiro, , and S. Osher. Simultaneous structure and texture image inpainting. In *Proc. IEEE Transactions on Image Processing*, volume 12, pages 882–889, Aug 2003.

[66] W. H. Cheng, C. W. Hsieh, S.K. Lin, C. W. Wang, and J. L. Wu. Robust algorithm for examplar based image inpainting. In *Proc. IEEE International Conference on Computer Graphics, Imaging and Vision*, pages 26–29, Jul 2005.

[67] B. R. Li and X. K. Shen. An image inpainting method. In *Proc. IEEE Ninth International Conference on Computer Aided Design and Computer Graphics*, pages 531–536, 2005.

[68] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. In *Proc. IEEE Transactions on Image Processing*, volume 13, Issue 9, pages 1200–1212, Sep 2004.

[69] D. Nie, L. Ma, and S. Xiao. Similarity based image inpainting method. In *Proc. IEEE International Conference on Multi-Media Modelling*, Jan 2006.

[70] A. C. Kokaram, R. D. Morris, W. J. Fitzgerald, and P. J. W. Rayner. Detection of missing data in image sequences. *IEEE Transactions on Image Processing*, 4(11):1496–1508, 1995.

[71] A. N. Hirani and T. Totsuka. Combining frequency and spatial domain information for fast interactive image noise removal. In *Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 269–276, 1996.

[72] M. Nitzberg, D. Mumford, and T. Shiota. *Filtering, Segmentation, and Depth*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1993.

[73] P. Harrison. A non-hierarchical procedure for re-synthesis of complex textures. In *WSCG 2001 Conference proceedings*, pages 190–197, 2001.