# Image Enhancement Techniques by Texture Synthesis

By

**Krupa Shah**

**08MCE016**


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**AHMEDABAD-382481**

**May 2010**

# Image Enhancement Techniques by Texture Synthesis

**Major Project**

Submitted in partial fulfillment of the requirements

For the degree of

**Master of Technology in Computer Science and Engineering**

By

**Krupa Shah**

**08MCE016**


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**AHMEDABAD-382481**

**May 2010**

# Declaration

This is to certify that

i) The thesis comprises my original work towards the degree of Master of Technology in Computer Science & Engineering at Nirma University and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

**Krupa Shah**

# Certificate

This is to certify that the Major Project entitled **"Image Enhancement Techniques by Texture Synthesis"** submitted by **Krupa Shah** (08MCE016), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Major Project-I, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. S.N. Pradhan
Guide,P.G.Coordinator,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D. J. Patel
Professor and Head,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. K Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

# Abstract

In many image processing applications,it is often important to accurately expand images without loss of clarity. Edge and Textures are among the most important features of an image. They have however very different characteristics, suggestions that they should be enhanced using different techniques. For edges, we propose a sobel techniques and finding the edge of all objects .For textures we have used Pixel base Texture Synthesis and Patch based texture Synthesis approach.

Texture synthesis is the process of algorithmically constructing a large digital image from a small digital sample image.Texture synthesis can be used to fill in holes in images, create large non-repetitive background images and expand small pictures and also removing noise.

For cleaning gray scale image two methods Pixel Based Texture Synthesis and Patch Based Texture Synthesis are implemented and their result have been compared.Also taking different size and shapes of patch and then using both methods noise is removed. PSNR is used as measure of quality. For Color image(RGB image) cleaning the image by selecting area(select image area ) and segmentation(object select by boundary).

This dissertation addresses two parts. The first part of this dissertation considers Texture Synthesis for Gray scale images by Pixel based and Patch based Texture Synthesis. The second part deals Cleaning the color images by Texture Synthesis.

# Acknowledgements

I am deeply indebted to my thesis guide **Dr. S.N.Pradhan** for his constant guidance and motivation. He has devoted significant amount of his valuable time to plan and discuss the thesis work. Without his experience and insights, it would have been very difficult to do quality work.I would like to extend my gratitude to Dr. S.N.Pradhan for his continuous encouragement.

I am thankful to Asst. Prof. Swati Jain and all professors at Nirma Institute of technology who taught the fundamental essentials to undertake such a project. Without their valuable guidance it would have been extremely difficult to grasp and visualize the project theoretically.
I am also thankful to members of my class for their delightful company which kept me in good humor throughout the year.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

**- Krupa Shah**

**08MCE016**

# Contents

**Declaration** iii

**Certificate** iv

**Abstract** v

**Acknowledgements** vi

**List of Tables** x

**List of Figures** xi

**Abbreviation** xii

**1 Introduction** 1
- 1.1 Objectives . . . 1
- 1.2 statement of Problem . . . 1
- 1.3 Texture . . . 2
- 1.4 Texture Synthesis . . . 3
  - 1.4.1 Definition . . . 3
  - 1.4.2 Methods . . . 3
- 1.5 Application . . . 5
  - 1.5.1 Distorted synthesis . . . 5
  - 1.5.2 User control . . . 6
  - 1.5.3 Rendering . . . 6
  - 1.5.4 Animation . . . 6
  - 1.5.5 Compression . . . 7
  - 1.5.6 Restoration and Editing . . . 7
  - 1.5.7 Computer Vision . . . 8
- 1.6 Goal . . . 8
- 1.7 Tools And Techniques . . . 8
- 1.8 Thesis Organization . . . 8

**2 Literature Survey** **10**
2.1 General . . . 10
2.2 Literature Review . . . 10
2.2.1 Image Replacement through Texture Synthesis . . . 10
2.2.2 Texture Analysis and Synthesis using Tree Structure Vector Quantization . . . 11
2.2.3 State of the Art in Example-based Texture Synthesis . . . 14
2.2.4 Real-Time Texture Synthesis By Patch-Based Sampling . . . 16
2.2.5 Feature Matching . . . 19
2.2.6 Markov Random Field and Gibbs Sampling . . . 19
2.2.7 Edge Handling . . . 20
2.2.8 Image Enhancement By Texture Synthesis . . . 21
2.3 Conclusion of Research Papers . . . 24

**3 Comparision** **25**
3.1 Explicit v.s. Implicit Texture Synthesis . . . 25
3.2 Comparison of Texture Synthesis Methods . . . 26
3.2.1 Tiling Based Texture Synthesis . . . 26
3.2.2 Pixel Based Texture Synthesis . . . 27
3.2.3 Patch Based Texture Synthesis . . . 27

**4 Segmentation** **29**
4.1 Texture Segmentation . . . 30
4.1.1 Edge Detection . . . 30
4.2 Example of Segmentation . . . 31

**5 Methodologies** **32**
5.1 Gray Scale Image . . . 32
5.1.1 Pixel Based Texture Synthesis . . . 32
5.1.2 Patch Based Texture Synthesis . . . 33
5.1.3 Summary . . . 34
5.2 Color Image . . . 35
5.2.1 Selecting Area . . . 35
5.2.2 Edge Detection Using Segmentation . . . 36

**6 Implementation** **37**
6.1 Fixed Patch Size . . . 37
6.1.1 Pixel based Texture Synthesis . . . 37
6.1.2 Patch Based Texture Synthesis . . . 38
6.1.3 Noise Comparison . . . 39
6.2 Different size of Patch . . . 39
6.2.1 Noise Comparision . . . 40
6.3 PSNR for Texture Synthesis . . . 40
6.3.1 Result . . . 41

6.4 Program Complexity . . . . . 41
6.5 Color Image . . . . . 42
6.5.1 Cleaning image by Selecting Area . . . . . 42
6.5.2 Cleaning image by Edge Detection . . . . . 44

**7 Conclusion and Future Scope** **46**
7.1 Conclusion . . . . . 46
7.2 Future Scope . . . . . 47

**Website References** **48**

**References** **49**

**Index** **51**

# List of Tables

2.1 Symbols . . . . . 14
2.2 Pseudocode of the algorithm . . . . . 15
2.3 Conclusion of Research Papers . . . . . 24

6.1 Noise Ratio . . . . . 41

# List of Figures


1.1 Distorted Synthesis . . . . . 5
1.2 User Control . . . . . 6

2.1 Example of Image Replacement . . . . . 11
2.2 Synthesis Process . . . . . 12
2.3 Process of Texture Synthesis . . . . . 16
2.4 Multi resolution Neighbor . . . . . 20
2.5 Image Enhancement By Texture Synthesis . . . . . 23

4.1 Example-1 of Segmentation . . . . . 31
4.2 Example-2 of Segmentation . . . . . 31

5.1 Match neighborhood Pixel . . . . . 32
5.2 Flow of Pixel Based Texture Synthesis . . . . . 33
5.3 Pixel in Patch . . . . . 34
5.4 Flow of Patch Based texture Synthesis . . . . . 34
5.5 Program Flow . . . . . 35

6.1 Remove noise using Pixel based Texture Synthesis . . . . . 38
6.2 Remove noise using Pixel based Texture Synthesis . . . . . 39
6.3 Remove noise . . . . . 40
6.4 Select Area . . . . . 42
6.5 Cleaning Portion . . . . . 42
6.6 Example-2 of Cleaning Image . . . . . 43
6.7 Example-3 of Cleaning Image . . . . . 43
6.8 Input Image . . . . . 44
6.9 Edge Detection . . . . . 44
6.10 Output Image . . . . . 45

# Abbreviation

**MRF** Markov Random Field

**MSE** Mean Square Error

**PSNR** Peak Signal to Noise Ratio

**RETS** Real time image enhancement using texture synthesis

**TSVQ** Tree Structured Vector Quantization

# Chapter 1

# Introduction

## 1.1 Objectives

The primary challenge was that simply stretching the digital image created a blocky result, because a digital image contains only small, finite sampling of the continuous function represent.

Using an interpolation algorithm reduces the blocky appearance but does not add details. There is a need for a image enhancement algorithm that synthesized a higher-resolution image and automatically inserts visually appropriate details.

## 1.2 statement of Problem

During large few years, researchers have developed proven techniques for enlarging images. The primary challenges was that simply stretching the digital image created a blocky result, because a digital image contains only a small , finite sampling of a continous function is represents.

## 1.3 Texture

Reproducing detailed surface appearance is important to achieve visual realism in computer rendered images. One way to model surface details is to use polygons or other geometric primitives. However, as details becomes finer and more complicated, explicit modeling with geometric primitives becomes less practical. An alternative is to map an image, either synthetic or digitized, onto the object surface, a technique called texture mapping [1] [2].

The mapped image, usually rectangular, is called a texture map or texture. A texture can be used to modulate various surface properties, including color, reflection, transparency, or displacements. In computer graphics the content of a texture can be very general; in mapping a color texture, for example, the texture can be an image containing arbitrary drawings or patterns. Unfortunately, the meaning of texture in graphics is somehow abused from its usual meaning. The Webster's dictionary defines texture as follows:

Texture,

a. Something composed of closely interwoven elements; specifically a woven cloth

b. The structure formed by the threads of a fabric ...

In other words, textures are usually refereed to as visual or tactile surfaces composed of repeating patterns, such as a fabric. This definition of texture is more restricted than the notion of texture in graphics. However, since a majority of natural surfaces consist of repeating elements, this narrower definition of texture is still powerful enough to describe many surface properties. This definition of texture is also widely adopted in computer vision and image processing communities.

## 1.4 Texture Synthesis

### 1.4.1 Definition

**Texture synthesis** is the process of algorithmically constructing a large digital image from a small digital sample image by taking advantage of its structural content. Texture synthesis can be used to fill in holes in images, create large non-repetitive background images and expand small pictures[1]. The goal of texture synthesis can be stated as follows: Given a texture sample, synthesize a new texture that, when perceived by a human observer, appears to be generated by the same underlying process.

The major challenges are:

**Modeling** How to estimate the texture generation process from a given finite texture sample. The estimated process should be able to model both the structural and stochastic parts of the input texture. The success of modeling is determined by the visual fidelity of the synthesized textures with respect to the given samples.

**Sampling** How to develop an efficient sampling procedure to produce new textures from a given model. The efficiency of the sampling procedure will directly determine the computational cost of texture generation.

### 1.4.2 Methods

- **Tiling**

  The simplest way to generate a large image from a sample image is to tile it. This means multiple copies of the sample are simply copied and pasted side by side. The result is rarely satisfactory. Except in rare cases, there will be the seams in between the tiles and the image will be highly repetitive.

- **Stochastic Texture Synthesis**

  Stochastic texture synthesis methods produce an image by randomly choosing color values for each pixel, only influenced by basic parameters like minimum brightness, average color or maximum contrast. These algorithms perform well with stochastic textures only, otherwise they produce completely unsatisfactory results as they ignore any kind of structure within the sample image.

- **Single purpose structured Texture Synthesis**

  In this use a fix procedure to create an output image, i. e. they are limited to a single kind of structured texture. Thus, these algorithms can both only be applied to structured textures and only to textures with a very similar structure. For example, a single purpose algorithm could produce high quality texture images of stonewalls; yet, it is very unlikely that the algorithm will produce any viable output if given a sample image that shows pebbles.

- **Chaos Mosaic**

  This method, proposed by the Microsoft group for internet graphics, is a refined version of tiling and performs the following three steps:

  a. The output image is filled completely by tiling. The result is a repetitive image with visible seams.

  b. Randomly selected parts of random size of the sample are copied and pasted randomly onto the output image. The result is a rather non-repetitive image with visible seams.

  c. The output image is filtered to smooth edges.

  The result is an acceptable texture image, which is not too repetitive and does not contain too many artifacts. Still, this method is unsatisfactory because the smoothing in step 3 makes the output image look blurred.

- **Pixel-Based Texture Synthesis**

  They typically synthesize a texture in scan-line order by finding and copying

pixels with the most similar local neighborhood as the synthetic texture. These methods are very useful for image completion. They can be constrained, as in "Image Analogies", to perform many interesting tasks. They are typically accelerated with some form of Approximate Nearest Neighbor method since the exhaustive search for the best pixel is somewhat slow.

- **Patch-Based Texture Synthesis**

  Patch-based texture synthesis creates a new texture by copying and stitching together textures at various offsets. These algorithms tend to be more effective and faster than pixel-based texture synthesis methods.

- **Chemistry Based Texture Synthesis**

  Realistic textures can be generated by simulations of complex chemical reactions within fluids, namely Reaction-diffusion systems. It is believed that these systems show behaviors which are qualitatively equivalent to real processes (Morphogenesis) found in the nature, such as animal markings (shells, fishes, wild cats.).

## 1.5 Application

### 1.5.1 Distorted synthesis

Using a random order of visit of the output pixels and modifying the shape of the current neighborhood according to the local distortion of the output image, the synthesis produces images with both the texture information and the distortion one.

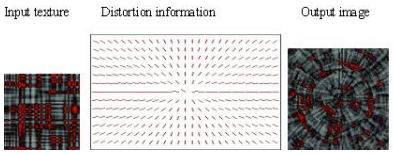



Figure 1.1: Distorted Synthesis

### 1.5.2 User control

Using a image painted by the user instead of the random noised image, with few modifications of the algorithm of Ashikhmin or other method , it is possible to force the synthesis to collocate the different features of the texture in different places of the output image.

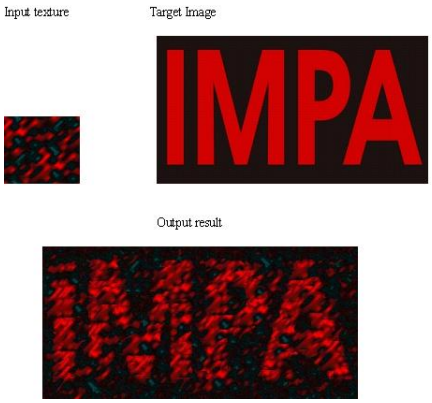



Figure 1.2: User Control

### 1.5.3 Rendering

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

### 1.5.4 Animation

Computer generated animations often contain scripted events and random motions. Scripted events are non-repetitive actions such as opening a door or picking up an object, and are usually rendered under direct control. On the contrary, random motions are repetitive background movements such as ocean waves, rising smoke, or a burning fire. These kind of motions have indeterminate extent both in space and time, and

are often refered as temporal textures[3].

These temporal textures are often difficult to render using traditional techniques based on physical modeling, since different textures are often generated by very different underlying physical processes. By treating them as textures, we can model and synthesize them using a single texture synthesis algorithm. In addition to temporal textures, certain motions such as joint angles of articulated motions, could also be modeled as one dimensional textures. These textures can be synthesized on the fly to simulate delicate motions such as eye blinking or human walking.

### 1.5.5 Compression

Images depicting natural scenes often contain large textures regions, such as a grass land, a forest, or a sand beach. Because textures often contain significant high frequency information, they are not well compressed by transform-based techniques such as JPEG. By segmenting out these textured regions in a preprocessing step, they might be compressible by a texture synthesis technique. In addition to image compression, texture synthesis can also be employed for synthetic scenes containing large amounts of textures [4].

### 1.5.6 Restoration and Editing

Photographs, films and images often contain regions that are in some sense flawed. A flaw can be a scrambled region on a scanned photograph, scratches on an old film, wires or props in a movie film frame, or simply an undesirable object in an image. Since the processes causing these flaws are often irreversible, an algorithm that can fix these flaws is desirable. Often, the flawed portion is contained within a region of texture, and can be replaced by texture synthesis[5][6].

### 1.5.7 Computer Vision

Several computer vision tasks use textures, such as segmentation, recognition, and classification. These tasks can benefit from a texture model, which could be derived from a successful texture synthesis algorithm[7].

## 1.6 Goal

We have to synthesize a high resolution version of a low resolution source image. We should be able to insert appropriate detail from sample texture and also generate a new texture and also remove the noise. The system should support multiple sample textures.

## 1.7 Tools And Techniques

- To implement Texture Synthesis, we will need the following installed.
  - Mat Lab
- In Texture synthesis we will use Pixel Based Texture Synthesis and Patch Based Texture Synthesis for Gray scale image. For cleaning the Color images used Selection area and segmentation.

## 1.8 Thesis Organization

The rest of the thesis is organized as follows.

**Chapter 2**, *The Texture Synthesis and Related Terminologies*, describes the methods and how its work and what is the result of it.

**Chapter 3**,*Comparision*, describes the comparison of Methods Tiling based ,pixel and Patch Based Texture Synthesis.

**Chapter 4**,*Segmentation*,describes finding the boundary of objects.

**Chapter 5**, *Methodologies*, describes the methodologies of Texture Synthesis.

In **chapter 6**, *Implementation*, a new algorithm for performing the Texture Synthesis. It remove noise using Pixel Based and Patch Based Texture Synthesis.Also Cleaning the Image by segmentation.

Finally, in **chapter 7**, Concluding remarks and scope for future work is presented.

# Chapter 2

# Literature Survey

## 2.1 General

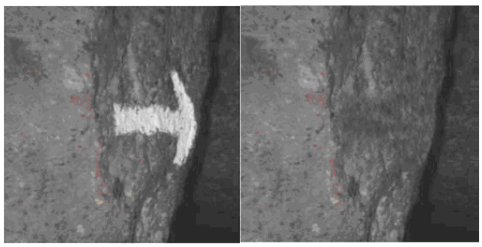
Literature in form of research papers, books regarding various aspects of performance of Texture Synthesis are referred and review is presented in this chapter. The objective of literature review is to develop basic understanding about different type of Texture Synthesis and their result.

## 2.2 Literature Review

Various literatures have referred for basic understanding and analysis of structure with Texture Synthesis. Some of that are summarized below in the form of books and research papers.

### 2.2.1 Image Replacement through Texture Synthesis

Texture synthesis can be useful in a variety of images which need the replacement of large areas with stochastic textures. This technique, however, is useless for images that need the replacement of areas with structured texture.



(a) Input Image (b) Output Image

Figure 2.1: Example of Image Replacement

Image replacement through texture synthesis provides a new solution to the image replacement problem for a certain class of images. Direct composition of a pure synthetic texture into an image often works well for stochastic textures, but may occasionally result in undesirable artifacts due to differences in the grain of the real texture and the imperfect synthetic texture. By integrating composition into the texture synthesis algorithm, a smooth transition between real and synthetic texture grain may be realized[8].

### 2.2.2 Texture Analysis and Synthesis using Tree Structure Vector Quantization

In the synthesis process, the output texture will be transformed from a random noise to a new image based on the estimated F.

The synthesis process can be described by the following pseudo code:

a. Loop through all pixels (x,y) in the output texture in raster scan order.

b. Collect the neighborhood vector, $N(_{x,y})$, of pixel (x,y).

c. Assign $F(N(_{x,y}))$ to be the synthesized color of pixel (x,y).

**Neighborhood:** The output texture is synthesized in a raster scan ordering, we restrict our neighborhood system, $N(_{x,y})$, to a causal neighborhood, which means that

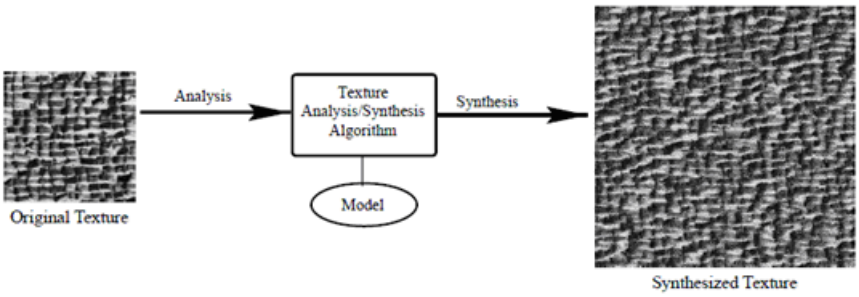



Figure 2.2: Synthesis Process

$N(_{x,y})$ depends only on the previous pixels in the raster scan ordering. A noncausal neighborhood $N(_{x,y})$ will lead to an iterative synthesis algorithm, which will take longer computation time[9].

Because the set of local neighborhoods N(P)is used as the primary model for textures, the quality of the synthesized results will depend on its size and shape. Intuitively, the size of the neighborhoods should be on the scale of the largest regular texture structure; otherwise this structure may be lost and the result image will look too random[10].

**Advantage:**The key advantage of this approach is that it can efficiently synthesize a wide variety of textures. At the same time, it is simple to implement since the most complex component is tree structure VQ. This is also easy to use: image along with a few parameters are required to generate a new texture of any size and shape[9]. Texture synthesis is an alternative way to create textures. Because synthetic textures can be made any size, visual repetition is avoided. Texture synthesis can also produce tileable images by properly handling the boundary conditions.

Given a texture sample, synthesize a new texture that, when perceived by a human observer, appears to be generated by the same underlying stochastic process.

The major challenges are

- modeling- how to estimate the stochastic process from a given finite texture sample

- sampling- how to develop an efficient sampling procedure to produce new textures from a given model.

**Algorithm:**MRF

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. 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 and is independent of the rest of the image.
Using Markov Random Fields as the texture model, the goal of the synthesis algorithm is to generate a new texture so that each local region of it is similar to another region from the input texture. First describe how the algorithm works in a single resolution, and then extend it using a multi resolution pyramid to obtain improvements in efficiency. List the symbols used in Table1 and Summarize the algorithm in Table2.

The algorithm starts with an input texture sample $I_a$ and a white random noise $I_s$. We force the random noise $I_s$ to look like a $I_a$ by transforming $I_s$ pixel by pixel in a raster scan ordering, i.e. from top to bottom and left to right.
To determine the pixel value p at $I_s$ , its spatial neighborhood $N(_p)$ is compared against all possible neighborhoods N(pi) from Ia. The input pixel pi with the most similar $N(_{pi})$ is assigned to p. We use a simple L2 norm (sum of squared difference) to measure the similarity between the neighborhoods. The goal of this synthesis process is to ensure that the newly assigned pixel p will maintain as much local similarity between $I_a$ and $I_s$ as possible. The same process is repeated for each output pixel until all the pixels are determined. This is akin to putting together a jigsaw puzzle: the pieces are the individual pixels and the fitness between these pieces is determined

Table 2.1: Symbols

| *Symbol* | *Meaning* |
|---|---|
| $I_a$ | Input Texture Sample |
| $I_s$ | Output Texture Sample |
| $G_a$ | Gaussian pyramid built from $I_a$ |
| $G_s$ | Gaussian pyramid built from $I_s$ |
| $p_i$ | An input pixel in $I_a$ or $G_a$ |
| p | An output pixel in $I_s$ or $G_s$ |
| N(p) | Neighborhood around the pixel p |
| G(L) | $L^{th}$ level of pyramid G |
| G(L,x,y) | Pixel at level L and position (x,y) of G |
| {RxC,k} | (2D)neighborhood containing k levels, with size RxC at the top level |
| {RxCxD,k} | 3D neighborhood containing k levels, with size RxCxD at the top level |

by the colors of the surrounding neighborhood pixels.

**Application:**One of the chief advantages of our texture synthesis method is its low computational cost[10].

a. Constrained Texture Synthesis

b. Temporal Texture Synthesis

### 2.2.3 State of the Art in Example-based Texture Synthesis

The process of texture synthesis could be decomposed into two main components, analysis and synthesis:

**Analysis:** How to estimate the underlying generation process from a given finite texture sample. The estimated process should be able to model both the structural

Table 2.2: Pseudocode of the algorithm

function $I_s \longleftarrow$ Texture Synthesis($I_a$,outputSize)

$I_s \longleftarrow$ Initialize(outputSize);
$G_s \longleftarrow$ BuildPyramid($I_a$);
$G_a \longleftarrow$ BuildPyramid($I_s$);
for each level L from lower to higher resolution of $G_s$
loop through all pixels ($x_s$,$y_s$) of $G_s$(L)
C $\longleftarrow$ FindBestMatch($G_a$,$G_s$,L,$x_s$,$y_s$);
$G_s$(L,$x_s$,$y_s$) $\longleftarrow$ C;
$I_s \longleftarrow$ ReconPyramid($G_s$);
return $I_s$

function C $\longleftarrow$ FindBestMatch($G_a$,$G_s$,L,$x_s$,$y_s$);

$N_s \longleftarrow$ BuildNeighborhood($G_s$,L,$x_s$,$y_s$)
$N_a^{best} \longleftarrow$ null; C $\longleftarrow$ null;
loop through all pixels ($x_s$,$y_s$) of $G_s$(L)
$N_a \longleftarrow$ BuildNeighborhood($G_a$,L,$x_a$,$y_a$);
if Match ($N_a$,$N_s$) $\rangle$ Match($N_a^{best}$,$N_s$)
$N_a^{best} \longleftarrow N_a$ ; C $\longleftarrow G_a$(L,$x_a$,$y_a$);
return C;

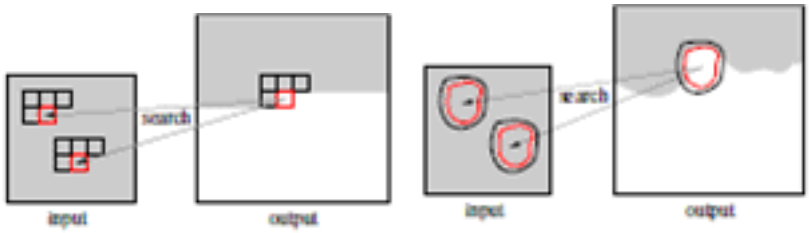
and stochastic parts of the input texture. The success of the model is determined by the visual fidelity of the synthesized textures with respect to the given samples.

**Synthesis:** How to develop an efficient generation procedure to produce new textures from a given analysis model. The efficiency of the sampling procedure will directly determine the computational cost of texture generation.

**Basic algorithm:**

- Pixel based Texture synthesis
- Patch Based Texture Synthesis

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




(a) Pixel Based (b) Patch Based

Figure 2.3: Process of Texture Synthesis

within the same copied patch ought to look good with respect to each other[11].

### 2.2.4 Real-Time Texture Synthesis By Patch-Based Sampling

High-quality texture can be synthesized in realtime. A key ingredient of the algorithm we propose is a patch-based sampling scheme that uses texture patches of the sample texture as building blocks for texture synthesis.

The advantages of patch-based sampling include

- Speed:For synthesizing textures of the same size and comparable (or better) quality, our algorithm is orders of magnitude faster than existing texture synthesis algorithms, including TSVQ-accelerated non-parametric sampling. As a result, high-quality texture synthesis is now a real-time process on a midlevel PC.

- Quality:The patch-based sampling algorithm synthesizes high-quality textures for a wide variety of textures ranging from regular to stochastic. Like , that is also a greedy algorithm for non-parametric sampling. However, the patches in sampling scheme implicitly provide constraints for avoiding garbage. For this reason, algorithm continues to synthesize high-quality textures even when and cease to be effective. For natural textures, the results of patch-based sampling look subjectively better.

### Patch Based Sampling

The patch-based sampling algorithm uses texture patches of the input sample texture $I_{in}$ as the building blocks for constructing the synthesized texture $I_{out}$ In each step, paste a patch $B_k$ of the input sample texture $I_{in}$ into the synthesized texture $I_{out}$. To avoid mismatching features across patch boundaries, select $B_k$ based on the patches already pasted in $I_{out}$,$\{B_0$ .... $B_{k-1}\}$. The texture patches are pasted in the order. For simplicity,use square patches of a prescribed size $w_B$ x $w_B$.

### Sampling Strategy

$I_{R1}$ and $I_{R2}$ be two texture patches of the same shape and size. Say that $I_{R1}$ and $I_{R2}$match if $d(R1, R2) < \delta$ , where the $d()$ represents the distance between two texture patches and $\delta$ is a prescribed constant.

Assuming the Markov property, the patch-based sampling algorithm estimates the local conditional MRF (FRAME or Gibbs) density $p(I_R|I_{\partial R})$

in a non-parametric form by an empirical histogram. Define the boundary zone $\partial R$ of a texture patch $I_R$ as a band of width $w_E$ along the boundary of R . When the texture on the boundary zone $I_{\partial R}$ is known, we would like to estimate the conditional probability distribution of the unknown texture patch $I_R$ . Instead of constructing a model, we directly search the input sample texture $I_{in}$ for all patches having the known $I_{\partial R}$ as their boundary zones. The results of the search form an empirical histogram for the texture patch $I_R$ . To synthesize $I_R$ , we just pick an element from at random. Mathematically, the estimated conditional MRF density is

$$p(I_R/I_{\partial R}) = \sum_i \alpha_i \delta(I_R - I_{R^i}), \sum_i \alpha_i = 1 \tag{2.1}$$

where $I_{R^i}$ is a patch of the input sample texture $I_{in}$ whose boundary zone $I_{\partial R^i}$ matches the boundary zone $I_{\partial R}$. The weight $\alpha_i$ is a normalized similarity scale factor. With

patch-based sampling, the statistical constraint is implicit in the boundary zone $\partial R$. A large boundary zone implies a strong statistical constraint. Generally speaking, a non-parametric local conditional PDF is faster to estimate than the analytical FRAME model. On the down side, the nonparametric density estimation is subject to greater statistical fluctuations, because in a small sample texture $I_{in}$ there may be only a few sites that satisfy the local statistical constraints.

A more serious problem with existing non-parametric sampling techniques is that they tend to wonder into the wrong part of the search space and grow garbage in the synthesized texture. The patches in our sampling scheme implicitly provide constraints for avoiding garbage.

**Patch Based Sampling Algorithm**

a. Randomly choose a $w_B$ x $w_B$texture patch B0 from the input sample texture $I_{in}$. Paste $B_0$ in the lower left corner of $I_{out}$. Set k = 1.

b. Form the set $\Psi_B$ of all texture patches from $I_{in}$ such that for each texture patch of $\Psi_B$, its boundary zone matches $E_k^o ut$.

c. If $\Psi_B$ is empty, set $\Psi_B = \{B_{min}\}$ where $\{B_{min}\}$is chosen such that its boundary zone is the closest to $E_k^o ut$.

d. Randomly select an element from $\Psi_B$ as the $k^{th}$ texture patch $B_k$. Paste $B_k$ onto the output texture $I_{out}$. Set k = k + 1.

e. Repeat steps (b), (c), and (d) until $I_{out}$ is fully covered.

f. Perform blending in the boundary zones.

The patch-based sampling algorithm is easy to use and flexible. It can generate tileable textures if so desired. It can be used for constrained synthesis as well. The algorithm has an intuitive randomness parameter. The user can use this parameter

to interactively control the randomness of the synthesized texture.

Algorithm combines the strengths of nonparametric sampling and patch-pasting . In fact, both patch-pasting and the pixel-based non-parametric sampling are special cases of the patch based sampling algorithm. The patches in our sampling scheme implicitly provide constraints for avoiding garbage. For this reason, algorithm continues to synthesize high-quality textures even when cease to be effective. For natural textures, the results of patch-based sampling look subjectively better[12].

### 2.2.5 Feature Matching

Some algorithms model textures as a set of features, and generate new images by matching the features in an example texture These algorithms are usually more efficient than Markov Random Field algorithms. Heeger and Bergen [13]model textures by matching marginal histograms of image pyramids. Their technique succeeds on highly stochastic textures but fails on more structured ones. De Bonet [14]synthesizes new images by randomizing an input texture sample while preserving the cross-scale dependencies.

This method works better than on structured textures, but it can produce boundary artifacts if the input texture is not tileable. Simoncelli and Portilla [15]generate textures by matching the joint statistics of the image pyramids. Their method can successfully capture global textural structures but fails to preserve local patterns.

### 2.2.6 Markov Random Field and Gibbs Sampling

Many algorithms model textures by Markov Random Fields (or in a different mathematical form, Gibbs Sampling), and generate textures by probability sampling[5].Since Markov Random Fields have been proven to be a good approximation for a broad range of textures, these algorithms are general and some of them produce good results.

A drawback of Markov Random Field sampling, though, is that it is computationally expensive: even small texture patches can take hours or days to generate.

### 2.2.7 Edge Handling

Proper edge handling for $N_(p)$ near the image boundaries is very important. For the synthesis pyramid the edge is treated toroidally. In other words, if $G_s(L; x; y)$ denotes the pixel at level L and position (x; y) of pyramid $G_s$, then $G_s(L; x; y)$=$G_s(L; xmodM; ymodN)$, where M and N are the number of rows and columns, respectively, of $G_s(L)$. Handling edges toroidally is essential to guarantee that the resulting synthetic texture will tile seamlessly.

A causal multi resolution neighborhood with size f5x5,2g. The current level of the

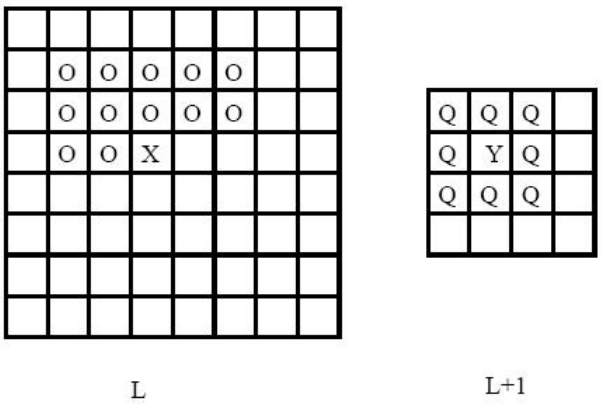



Figure 2.4: Multi resolution Neighbor

pyramid is shown at left and the next lower resolution level is shown at right. The current output pixel p, marked as X, is located at (L; x; y), where L is the current level number and (x; y) is its coordinate. At this level L of the pyramid the image is only partially complete. Thus, we must use the preceding pixels in the raster scan ordering (marked as O). The position of the parent of the current pixel, located at (L + 1; x $/2; y/$ 2 ), is marked as Y. Since the parents level is complete, the neighborhood can

contain pixels around Y, marked by Q. When searching for a match for pixel X, the neighborhood vector is constructed that includes the Os, Qs, and Y, in scanline order.

For the input pyramid Ga, toroidal neighborhoods typically contain discontinuities unless Ia is tileable. A reasonable edge handler for Ga is to pad it with a reflected copy of itself. Another solution is to use only those $N(_{pi})$ completely inside Ga, and discard those crossing the boundaries. Because a reflective edge handler may introduce discontinuities in the derivative, we adopt the second solution which uses only interior blocks.

### 2.2.8 Image Enhancement By Texture Synthesis

Real time Enhancement using texture synthesis combines interpolation ,classification and patch based texture synthesis to enhance low resolution imagery. RETS uses as input a low resolution source image and several high resolution sample textures. The output of RETS is a high resolution image with the structure of the source image, but with detail consistent with the high resolution sample textures.

Image Interpolation: Interpolation is the primary technique used for image scaling. Image scaling is the process of taking a source image and extending it to create a large image. The primary problem with enlarging images using interpolation is that the large result contains the same amount of discrete data as smaller source image.

Two types of interpolation are bilinear and bicubic. Bilinear interpolation uses 2x2 neighborhood of data points to calculate pixel color between data points. Bicubic interpolation uses 4x4 neighborhood of data points to calculate pixel color between data points[16].

Using texture Synthesis solve two problems

- It allows to user to specify the detail to be inserted into the output image by providing a representative sample for the system to replicate.
- Applying texture synthesis appropriately will allow us to avoid the unnatural repetition of texture tiles that can occur with standard texture mapping.

Image enhancement using texture synthesis combines interpolation, classification and texture synthesis to enhance low resolution imagery, particular aerial imagery. In that input as low resolution source image and several high resolution sample textures. The output is high resolution image with the structure of the source image but with detail consistent with the high resolution sample textures[16].

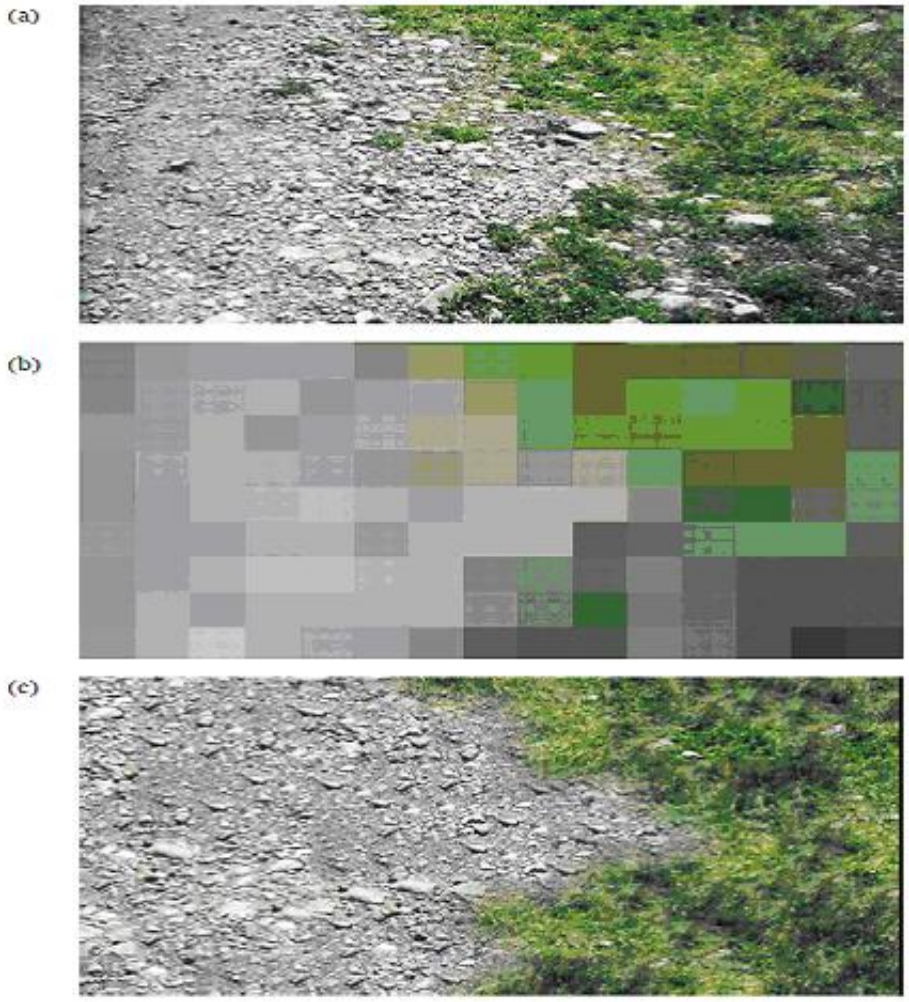



Figure 2.5: Image Enhancement By Texture Synthesis
(a)Original Photograph (b)Sample version of Image (c)Enhanced version of Image

## 2.3 Conclusion of Research Papers

Table 2.3: Conclusion of Research Papers

| No. | Title Of Paper | Summary | Method |
|---|---|---|---|
| 1 | Image Replacement through Texture Synthesis | Image replacement texture synthesis provides a new solution to the image replacement problem for a certain class of images. | Stochastic |
| 2 | Deterministic Texture Analysis and Synthesis using Tree Structure Vector Quantization | Advantage of approach is that it can efficiently synthesize a wide variety of textures.At the same time,it is simple to implement since the most complex component is tree structure VQ. | Pixel |
| 3 | Composite Texture Synthesis | Segmentation uses fixed filters, which are texture- and mutually-independent, while the synthesis uses an optimal texture- and mutually-dependent pixel pair type selection obtained during the analysis-by-synthesis procedure[17]. | Pixel |
| 4 | Fast Texture Synthesis using Tree-structured Vector Quantization | Texture synthesis method is its low computational cost.This permits us to explore a variety of applications, in addition to the usual texture mapping for graphics, that were previously impractical. | Patch |
| 5 | Synthesis Algorithms | Patch based approach can produce better synthesis results while requiring less computation[18]. | Patch |
| 6 | Real time image enhancement by texture synthesis | Combination of Interpolation and patch based texture synthesis to enhance low resolution images while they are being large. | Patch |
| 7 | An Efficient Texture Synthesis Algorithm Based On WT | An algorithm to speed up texture synthesizing using WT technique using patch based texture synthesis[19]. | Patch |

# Chapter 3

# Comparision

## 3.1 Explicit v.s. Implicit Texture Synthesis

Texture synthesis techniques can be classified as either explicit or implicit[20].An Explicit algorithms generates a whole texture directly while an Implicit algorithm answers a query about a particular point (much like scan-converting polygons versus raytracing implicit surfaces). Most existing statistical texture synthesis algorithms are explicit; because the value of each texture pixel is related to other pixels (such as spatial neighboring ones in Markov Random Field approaches) it is impossible to determine their values separately. On the other hand, most procedural texture synthesis techniques are implicit since they allow texels to be evaluated independently (such as Perlin noise).

Implicit texture synthesis offers several advantages over explicit texture synthesis. Because only those excels that are actually used need to be evaluated, implicit methods are usually computationally cheaper than the explicit ones. Implicit methods often consume less memory since they don't need to store the whole texture (especially for high dimensional textures).

Implicit methods are also more flexible since they allow texture samples to be evaluated independently and in any order. Unfortunately, implicit methods are usually less general than explicit ones. Because of the requirement of independent texel evaluation, implicit methods cannot use general statistical texture modeling based on inter-pixel dependencies.

## 3.2 Comparison of Texture Synthesis Methods

Here there are comparison of three method of Texture synthesis.

a. Tiling Based Texture Synthesis

b. Pixel Based Texture Synthesis

c. Patch Based Texture Synthesis

### 3.2.1 Tiling Based Texture Synthesis

The simplest way to generate a large image from a sample image is to tile it. This means multiple copies of the sample are simply copied and pasted side by side. The result is rarely satisfactory. Except in rare cases, there will be the seams in between the tiles and the image will be highly repetitive.
**Synthesization:** Synthesization is fast[21].
**Time:** Neighbor pixel comparison is not done so not more time consuming.
**Execution:** Execution is faster.
**Seams:** In between two texture seams are present.

### 3.2.2 Pixel Based Texture Synthesis

They typically synthesize a texture in scan-line order by finding and copying pixels with the most similar local neighborhood as the synthetic texture. These methods are very useful for image completion. They can be constrained, as in "Image Analogies", to perform many interesting tasks. They are typically accelerated with some form of Approximate Nearest Neighbor method since the exhaustive search for the best pixel is somewhat slow.

**Synthesization:** Too slow when the synthesized image is large.

**Time:** Finding the neighbor so more time taken and also control over individual pixel value[22].

**Execution:** Execution is slower[23].

**Seams:** Here seams are not present.

In this approach employs a pixel-based multi-resolution texture synthesis algorithm, which is based on a non-parametric sampling method. In it assumes a Markov random field texture model, which means a pixel value at a certain location only depends on its immediate neighborhood.

### 3.2.3 Patch Based Texture Synthesis

Patch-based texture synthesis creates a new texture by copying and stitching together textures at various offsets. These algorithms tend to be more effective and faster than pixel-based texture synthesis methods.

**Synthesization:** Synthesizing high quality textures as they can maintain global structure of the texture. But slow when the synthesized image is large.

**Time:** Compare to pixel based texture synthesis this is faster [24].

**Execution:** Execution is faster than pixel based texture synthesis.

**Seams:** Here seams are present but we overlapping the patch so remove the seams.

The method synthesis's a new image by stitching together small patches from the sample image. In this method synthesis's a result image block by block in raster order. Square blocks are used to capture the primary pattern in the sample texture. First, a block is randomly selected from the sample image and pasted into the new image beginning at the first row and the first column. Then another block is selected as a candidate neighbor. It is placed next to the first block so that they overlap one another.

# Chapter 4

# Segmentation

Image segmentation is to cluster pixels into salient image regions, i.e., regions corresponding to individual surfaces, objects, or natural parts of objects. A segmentation could be used for object recognition, occlusion boundary estimation within motion or stereo systems, image compression, image editing, or image database look-up.

Image Segmentation is a subset of an expansive field of Computer Vision which deals with the analysis of the spatial content of an image. In particular, it is used to separate regions from the rest of the image, in order to recognize them as objects. It is a method used in the vast field of Artificial Intelligence.

Region Growing is an approach to image segmentation in which neighboring pixels are examined and added to a region class if no edges are detected. This process is iterated for each boundary pixel in the region. If adjacent regions are found, a region-merging algorithm is used in which weak edges are dissolved and strong edges are left in tact.

Region Growing offers several advantages over conventional segmentation techniques. Unlike gradient and Laplacian methods, the borders of regions found by region growing are perfectly thin (since we only add pixels to the exterior of our

region) and connected. The algorithm is also very stable with respect to noise. Our region will never contain too much of the background, so long as the parameters are defined correctly. Other techniques that produce connected edges, like boundary tracking, are very unstable. Most importantly, membership in a region can be based on multiple criteria. We can take advantage of several image properties, such as low gradient or gray level intensity value, at once.

There are, however, several disadvantages to region growing. First and foremost, it is very expensive computationally. It takes both serious computing power (processing power and memory usage) and a decent amount of time to implement the algorithms efficiently.

## 4.1 Texture Segmentation

The purpose of texture segmentation is to differentiate textured regions from the rest of the image, which include smooth regions and well-defined edges such as object boundaries. Noticing the fact that textured regions are usually covered with dense edges when performing the edge detection, we use the local edge pixel number as a feature for texture segmentation.

### 4.1.1 Edge Detection

By applying the horizontal and vertical Sobel operators to the luminance channel Y of the image, we can obtain the gradient magnitude and direction for each pixel. Pixels with gradient magnitude larger than a certain threshold $T_g$ are initialized as edge pixels. Since a lot of textures in video images are blurred, in order to obtain a sufficient number of edge pixels in textured regions for the purpose of segmentation, we set a low threshold $T_g = 10$ based on experiments. Raw edge maps usually contain thick

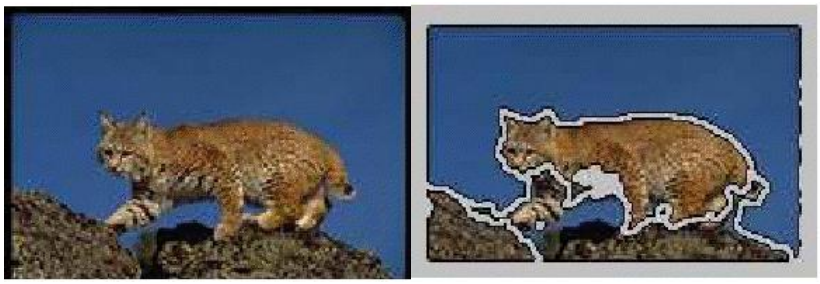
edges and are not suitable to be directly used for feature extraction and edge-based interpolation. Therefore we carry out non-maximal suppression to thin the edges. This operation re-classifies an edge pixel to be non-edge if its gradient magnitude is less than that of either of its two neighbors along the gradient direction[25].

## 4.2 Example of Segmentation

Simple Scenes Segmentations of simple gray-level images and color images (RGB images )can provide useful information about the surfaces in the scene.

(a) Input Image (b) Segmented Image

Figure 4.1: Example-1 of Segmentation



(a) Input Image (b) Segmented Image

Figure 4.2: Example-2 of Segmentation

# Chapter 5

# Methodologies

## 5.1 Gray Scale Image

### 5.1.1 Pixel Based Texture Synthesis

pixel-based multi-resolution texture synthesis algorithm, which is based on a non-parametric sampling method. It also assumes a Markov random field texture model, which means a pixel value at a certain location only depends on its immediate neighborhood[18]. When choosing the value of the next pixel in the output image

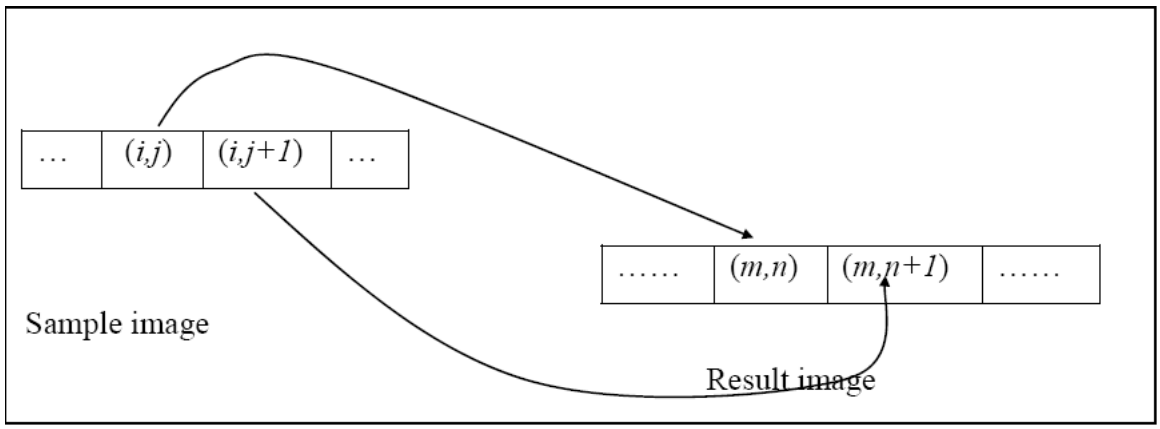



Figure 5.1: Match neighborhood Pixel

the algorithm uses the populated portion of the pixels neighborhood to exhaustively search for the best matched region in the sample image.

synthesizing texture in raster order. If pixel $(i, j)$ has the most similar neighbor-

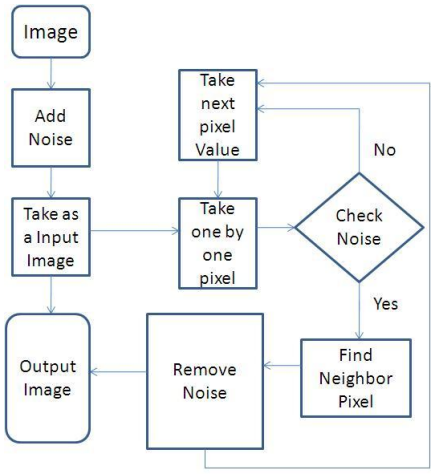



Figure 5.2: Flow of Pixel Based Texture Synthesis

hood, the value of pixel $(i, j)$ in the sample pyramid is assigned to pixel $(x, y)$ in the result pyramid.

### 5.1.2 Patch Based Texture Synthesis

The method synthesists a new image by stitching together small patches from the sample image. The method in synthesists a result image block by block in raster order. Square blocks are used to capture the primary pattern in the sample texture.

A block is randomly selected from the sample image and pasted into the new image beginning at the first row and the first column. Then another block is selected as a candidate neighbour. It is placed next to the first block so that they overlap one another. newpage

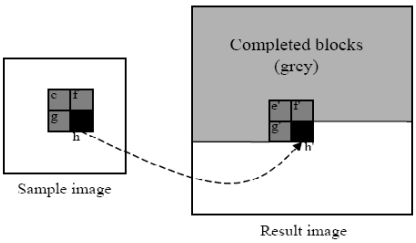



Figure 5.3: Pixel in Patch

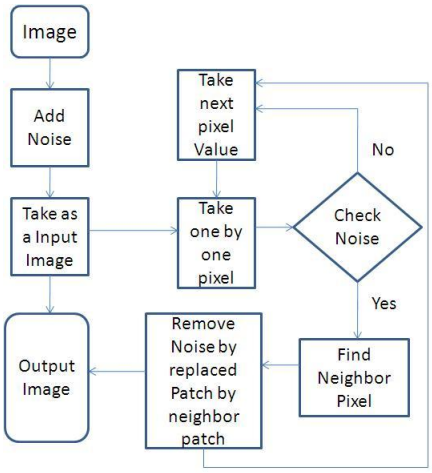



Figure 5.4: Flow of Patch Based texture Synthesis

### 5.1.3 Summary

In Pixel Based Texture Synthesis assumes a Markov Random Field texture model, which means a pixel value at a certain location only depends on its immediate neighborhood. A multi-resolution scheme is applied to construct the neighborhood around a given pixel.

In Patch Based Texture Synthesis can produce high-quality synthesis results while requiring little computation.

## 5.2 Color Image

In Image select four points using data cursor and take this four points as input and store its position value in variable. Using that position value of four points to select area which we want to remove or replace with some other texture which are surrounding this selecting area.

### 5.2.1 Selecting Area

**Flow Chart**

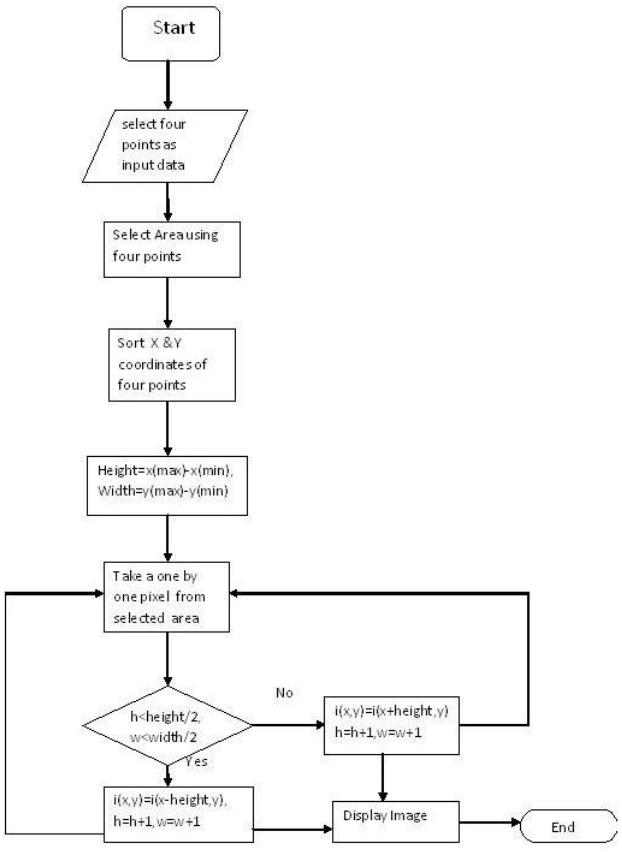



Figure 5.5: Program Flow

Four points are rectangles points then we are sorting x and y coordinates for this selecting four points. Find height and width.

$$Height = x(max) - x(min), \tag{5.1}$$

$$Width = y(max) - y(min) \tag{5.2}$$

Select one by one pixel from selecting area and check the value or height and width. Selecting pixel h<height/2 and w<width/2 then replace that pixel value by

$$I(x, y) = I(x - height, y) \tag{5.3}$$

Or $h > height/2$ and $w > width/2$ then replace that pixel value by

$$I(x, y) = I(x + height, y) \tag{5.4}$$

Now,using get area which we want to replaced. After, cleaning that area we can that output. And highlight that area which can be clean using Texture Synthesis.

### 5.2.2 Edge Detection Using Segmentation

Using Sobel Operation of segmentation finding the Edges of all Objects and then selecting the pixel of Object which we want to replaced for cleaning the image.
For this first convert color image into Gray scale image by function “RGB2gray” and then apply sobel operation and finding the edge and select the pixel of object. And which pixel of objects are replaced that changes are made in color image.

# Chapter 6

# Implementation

For implementation select one image and then add some noise in that image and taken that image as an input image. Then after removing the noise from that input image and get the output image. Below implementation of the methods for removing the noise from that input image.

## 6.1 Fixed Patch Size

Here taken noise patch is fixed size.

### 6.1.1 Pixel based Texture Synthesis

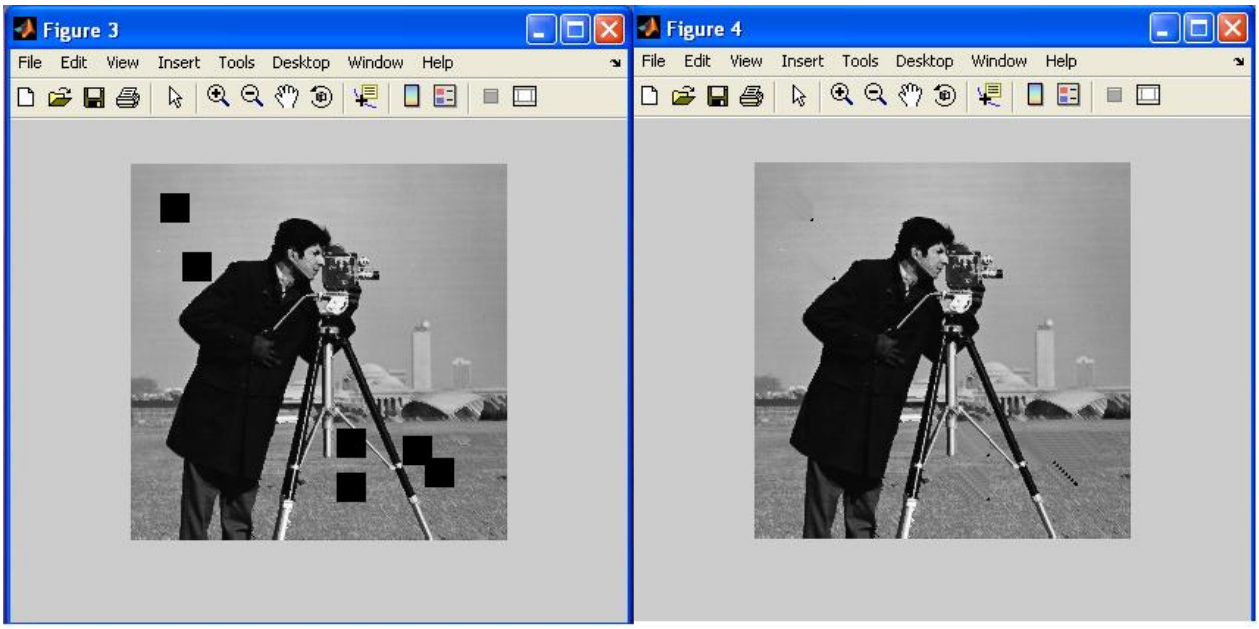
A pixel value at a certain location depends only on its immediate neighborhood. In this approach, for a certain percentage of the selections,use the next column neighbor pixel'.

There are two common steps:

a. Searching for the best match for the current output neighborhood within the sample texture

b. Merging a patch or a pixel with the synthesized output texture.

After this two steps we get an output image which is noiseless. The result is shown below.




(a) Input Image (b) Output Image

Figure 6.1: Remove noise using Pixel based Texture Synthesis

### 6.1.2 Patch Based Texture Synthesis

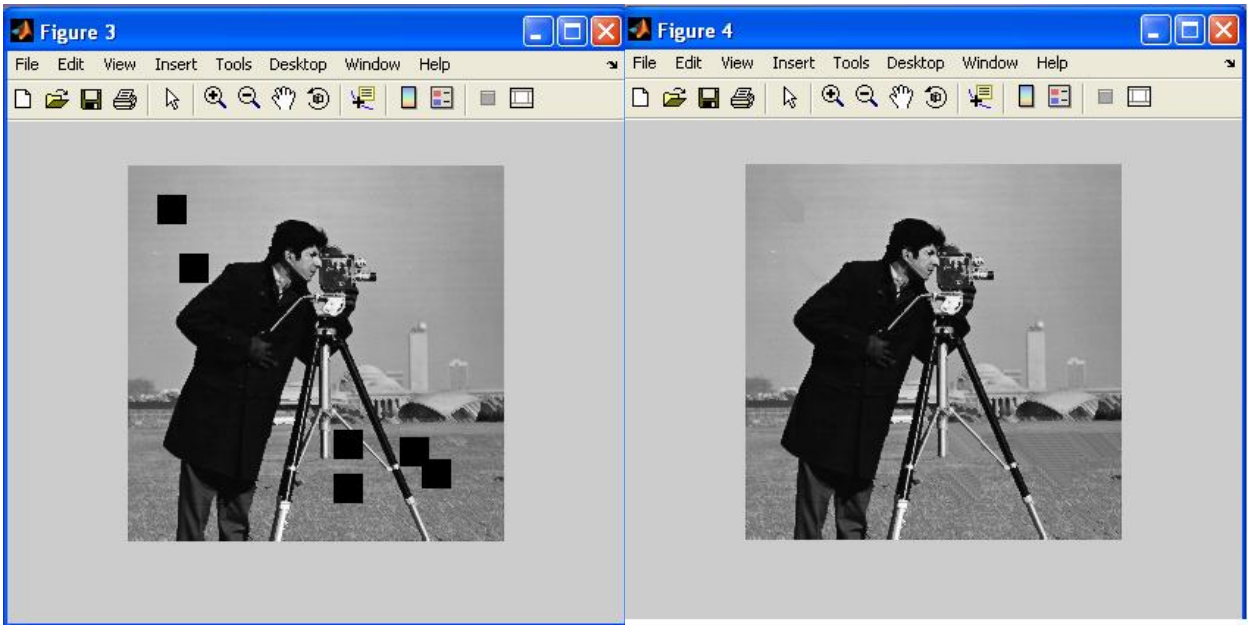
In patch based approach synthesiss the result image by stitching together small patches selected from the sample image. In this method synthesiss a result image block by block in raster order. Square blocks are used to capture the primary pattern in the sample texture.

There are two common steps:

a. Searching for the best match for the current output neighborhood within the sample texture

b. Merging a patch or a pixel with the synthesized output texture.

After this two steps we get an output image which is noiseless. The result is shown below.




(a) Input Image (b) Output Image

Figure 6.2: Remove noise using Pixel based Texture Synthesis

### 6.1.3 Noise Comparison

After analyzing both the output images(Pixel based and Patch based) it can be concluded that Patch Based Texture Synthesis is better than Pixel Based Texture Synthesis because in Pixel Based Texture Synthesis the portion of noise is not completely removed so it can be possible that some information will be lost. Patch Based Texture Synthesis gets good effect after removing the noise in image compare to Pixel Based Texture Synthesis.

## 6.2 Different size of Patch

Taking different size of matrix $n \times n$ as noise ($10 \times 10$ ,$20 \times 20$) in image and remove that noise and compare that in which method removal of noise is better.


(a) Input Image (b) pixel Output Image (c) Patch Output Image

Figure 6.3: Remove noise

### 6.2.1 Noise Comparision

In Pixel Based Texture Synthesis some Information is lost because of different size of patch is taken. Using Patch Based Texture Synthesis information is not lost and give better output than the Pixel Based Texture Synthesis.

## 6.3 PSNR for Texture Synthesis

The PSNR block computes the peak signal-to-noise ratio, in decibels between two images. The MSE and PSNR are the two error metrics used to compare image compression quality. The MSE represents the cumulative squared error between the compressed and the original image, whereas PSNR represents a measure of the peak error. The lower the value of MSE, the lower the error.

To compute the PSNR, the block first calculates the mean-squared error using the following equation:

$$MSE = (\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2)/(M \times N) \tag{6.3.1}$$

In the above equation, M and N are the number of rows and columns in the input images, respectively.

Then the block computes the PSNR using the following equation:

$$PSNR = 10 \times (\log_{10}(R^2/MSE)) \tag{6.3.2}$$

### 6.3.1 Result

Table 6.1: Noise Ratio

| **Patch Size** | **PSNR for Pixel Based Texture Synthesis** | **PSNR for Patch Based Texture Synthesis** |
|---|---|---|
| 5x5 | 27.5197 | 27.5197 |
| 10x10 | 27.4716 | 27.4946 |
| 15x15 | 27.4548 | 27.4532 |
| 20x20 | 27.3606 | 27.3555 |
| 25x25 | 24.0651 | 24.3302 |
| 30x30 | 21.0408 | 21.3612 |

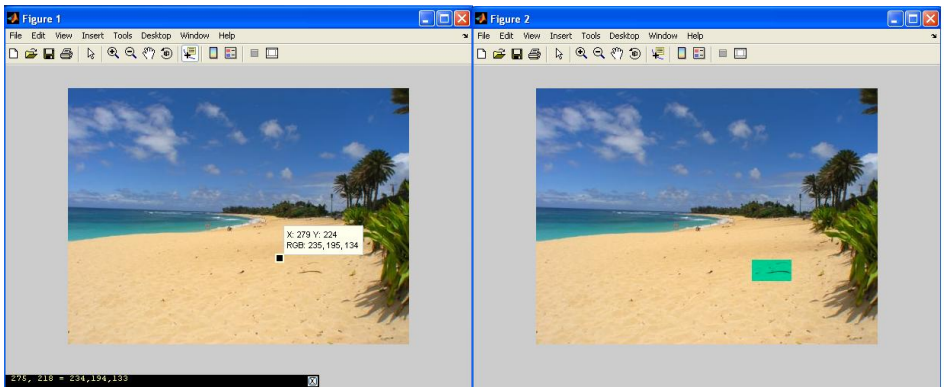
The higher the PSNR, the better the quality of the reconstructed image.

## 6.4 Program Complexity

In Pixel Based Texture Synthesis taking One by one pixel and then find its near neighbor pixel and replaced it. So, program complexity is $\Theta(n^2)$. In Patch Based Texture Synthesis taking a group of pixels. so not taking one by one pixel so complexity is $\Theta(n^2) - \sum(x \times y)$ where x is a patch size and y is number of patches. So, Patch Based Texture Synthesis complexity is less than Pixel Based Texture Synthesis.

## 6.5 Color Image

### 6.5.1 Cleaning image by Selecting Area

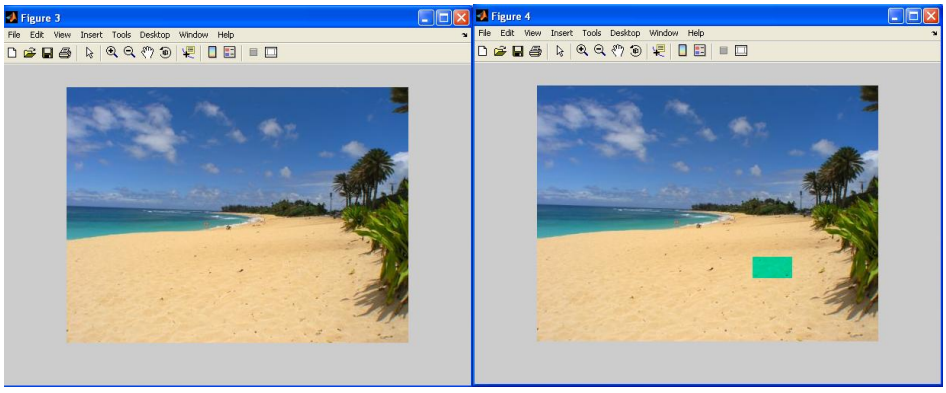



(a) Input Image (b) selected area

Figure 6.4: Select Area

We can get area which is to be replaced by selecting four points. By applying Texture on that area we can get the output in which that area is replaced by surrounding pixels to clean that area.

In Fig. 6.5a shown that area is cleaned and in Fig. 6.5bit is highlighted.




(a) Cleaning Area (b) Output Image

Figure 6.5: Cleaning Portion

In Fig. 6.6 first select an area enclosed by arrow and in the output image that area is removed.

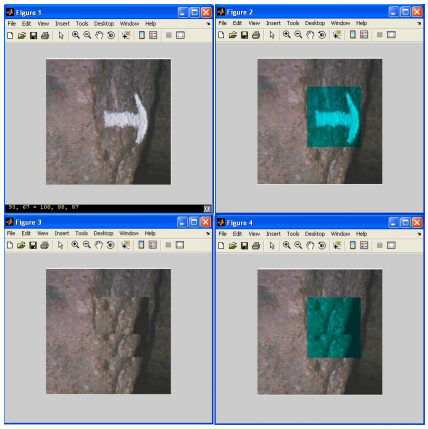


Figure 6.6: Example-2 of Cleaning Image

In Fig. 6.7 first select an area enclosed by donkey and in the output image that area is removed.

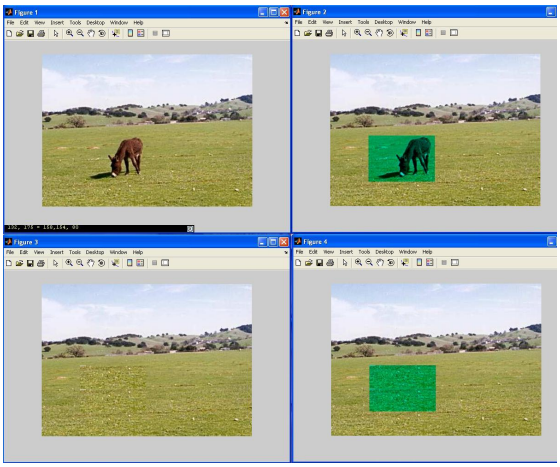


Figure 6.7: Example-3 of Cleaning Image

### 6.5.2 Cleaning image by Edge Detection

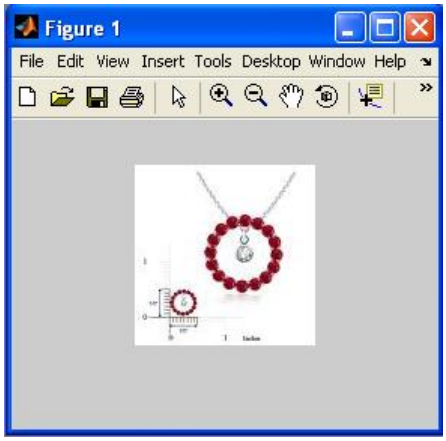



Figure 6.8: Input Image

By applying the horizontal and vertical Sobel operators to the luminance channel Y of the image, we can obtain the gradient magnitude and direction for each pixel. Pixels with gradient magnitude larger than a certain threshold are initialized as edge pixels. Since a lot of textures in video images are blurred, in order to obtain a sufficient number of edge pixels in textured regions for the purpose of segmentation.

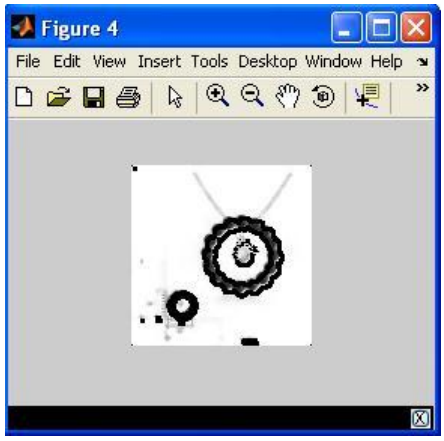



Figure 6.9: Edge Detection

After Finding the Edge of all objects choose one pixel of that object which we want to replaced by other texture. So,after choosing the pixel check the surrounding pixel values. when we get change in that surrounding pixel value we have to stop this process and replace selected pixel by surrounding values.

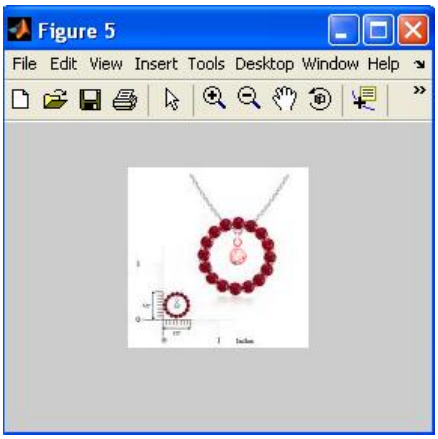



Figure 6.10: Output Image

# Chapter 7

# Conclusion and Future Scope

## 7.1 Conclusion

Patch Based Texture Synthesis is better than Pixel Based Texture Synthesis. Using Patch Based Texture Synthesis removal of noise is better than Pixel Based Texture Synthesis. Sometimes some information will be lost in Pixel Based Texture Synthesis. Compare to Pixel Based Texture Synthesis, loss of information is less in Patch Based Texture Synthesis. In PSNR there is less difference between Patch Based Texture Synthesis and Pixel Based Texture Synthesis but Program complexity is less in Patch Based Texture Synthesis than Pixel Based Texture Synthesis.

In color image applying Pixel based Texture Synthesis on selected area some information is lost even though it is not an object. In Segmentation we have to find edges of objects only so selecting and cleaning that object give better result compare to selecting an area of the image.

## 7.2 Future Scope

Although the goals we set when this research began have been met, there is much room for future research.

- One area of particular interest is real time enhancement using Oriented texture synthesis.
- A second area of interest is that of oriented motion synthesis. Whereas image enhancement using texture synthesis synthesized static texture, there is a need for motion synthesis. Example of motion synthesis are: waves moving towards shore, a stream flowing, grass blowing in the wind, etc.
- A third area of interest is that more than one object will be removing. And for that more than one pixel will be selecting so know that which object would removed by user.

# Website References


[1] http://en.wikipedia.org/wiki/Texture_synthesis

[2] http://www.mathworks.com/access/helpdesk/help/toolbox/vipblks/ref/psnr.html

[3] http://www.math.ucla.edu/~getreuer/matlabimaging.html

[4] http://www-iplab.ece.ucsb.edu/courses/ece178/W00/matlabip.htm

[5] http://matlab.izmiran.ru/help/techdoc/ref/image.html

[6] http://cnx.org/content/m15696/latest/

[7] http://www.aquaphoenix.com/lecture/matlab10/page3.html

# References


[1] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, (19):542546, 1976.

[2] E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Phd thesis, Computer Science Department, University of Utal,, Salt Lake City, Utah, 1974.

[3] Martin Szummer and Rosalind W. Picard. Temporal texture modeling. *International Conference on Image Processing*, 3:823–826, sep 1996.

[4] Maneesh Agrawala Andrew C. Beers and Navin Chaddha. Rendering from compressed textures. *Proceedings of SIGGRAPH 96*, pages 373–378, August 1996.

[5] Alexei Efros and Thomas Leung. Texture synthesis by non-parametric sampling. *International Conference on Computer Vision*, 2:1033–1038, sep 1999.

[6] Homan Igehy and Lucas Pereira. Image replacement through texture synthesis. *International Conference on Image Processing*, 3:186–189, oct 1997.

[7] Li-Yi Wei. *Texture Synthesis By Fixed Neighborhood Searching*. PhD thesis, STANFORD UNIVERSITY, November 2001.

[8] Lucas Pereira Homan Igehy. Image replacement through texture synthesis. *Computer Science Department,Stanford University*.

[9] LI-YI WEI. Deterministic texture analysis and synthesis using tree structure vector quantization. *Gates Computer Science Building,Stanford University,CA 94309, U.S.A.*, (386).

[10] Li-Yi Wei Marc Levoy. Fast texture synthesis using tree-structured vector quantization. *Stanford University*.

[11] Vivek Kwatra Greg Turk4 Li-Yi Wei, Sylvain Lefebvre. State of the art in example-based texture synthesis. *The Eurographics Association*, 2009.

[12] Yingqing Xu Baining Guo Lin Liang, Ce Liu and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *Technical Report*, (MSR-TR-2001-40), March 2001.

[13] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. *SIGGRAPH 95 Conference Proceedings*, pages 229–238, Aug 1995.

[14] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. *SIGGRAPH 97 Conference Proceedings*, pages 361–368, August 1997.

[15] E. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. *Fifth International Conference on Image Processing*, 1:62–66, oct 1998.

[16] Matthew Sorenson. Real time image enhancement using texture synthesis. November 2004.

[17] G. Caenen L. Van Gool A. Zalesny, V. Ferrari. Composite texture synthesis. *International Journal of Computer Vision*, 2004.

[18] D. Scharstein. *Synthesis Algorithm*, volume 1583/1999 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1999.

[19] Hai-Feng Cui Xin Zheng Tong Ruan. An efficient texture synthesis algorithm based on wt. 6:3472–3477, 2008.

[20] Darwyn Peachey Ken Perlin David S. Ebert, F. Kenton Musgrave and Steven Worley. Texturing and modeling. In *A Procedural Approach*. Morgan Kaufmann Publishers, 1998.

[21] Ning Zhou Weiming Dong, Ning Zhou. Optimized tile-based texture synthesis. *Graphics Interface, Montreal, Canada*, 2007.

[22] W. Guo Y. Meng, W.H. Li and Y.L. Liu. Particle swarm optimization method used in pixel-based texture synthesis.

[23] Aaron Bobick Nipun Kwatra Vivek Kwatra, Irfan Essa. Texture optimization for example-based synthesis.

[24] Pizzanu Kanongchaiyosy Jakrapong Narkdej. An efficient parameters estimation method for automatic patch-based texture synthesis.

[25] Xiaojun Feng and Jan P. Allebach. Segmented image interpolation using edge direction and texture synthesis. *IEEE Xplore*, 2008.

# Index

Advantage, 12
Algorithm, 15
Analysis, 15
Application, 5

Challenges, 12
Comparision, 25
Complexity, 41

Edge detection, 30

Flow chart, 33, 35

MRF, 13, 32
MSE, 40

Neighbor, 33
Noise, 39

Patch Based Texture Synthesis, 5, 38
Pixel Based Texture Synthesis, 4, 37
PSNR, 40

RETS, 21
RGB, 31, 36

Segmentation, 29
Synthesis, 15

Texture, 2
Texture Synthesis , 3