Example Based Image Processing

A dissertation submitted to the University of Dublin
for the degree of Doctor of Philosophy

Claire Catherine Gallagher
Trinity College Dublin, October 2006

Signal Processing and Media Applications
Department of Electronic and Electrical Engineering
Trinity College Dublin

To Mum, Dad, Deirdre and Declan

Abstract

This thesis is concerned with example based image processing. Example based image processing is a general term for any class of image processing operation where the manipulation and analysis of the image in question is guided by some set of example images. This thesis focuses on two applications, texture synthesis and image segmentation, in which example based image processing is proposed.

Given an example texture, the goal of a successful texture synthesis algorithm is to generate new texture which is perceptually similar to the sample texture. One of the main challenges in this process is the modelling of the example texture. Previous work has shown that those algorithms based on implicit modelling are more successful than those based on the more rigid explicit models. Based on this observation a new texture synthesis algorithm is developed which combines the strength of the implicit modelling technique with wavelet based image analysis. The Dual-Tree Complex Wavelet Transform used in this work has associated with it good directional selectivity and shift invariance. Both of these properties make it well suited for texture analysis. The new synthesis algorithm exploits this by performing synthesis in the wavelet domain. The result is a scale independent efficient process which is robust enough to work for a wide range of textures.

Building on the strength of this texture synthesis algorithm, the focus then turns to image segmentation. The goal of any image segmentation process is to assign to each pixel in an observed image a label indicating to which region or class that pixel belongs. Fully automated or unsupervised segmentation is an ill-posed problem and so in order to constrain the solution an example image set whose content is similar to that to be segmented is given as an input. This example image set has been segmented *a priori* and so can be used to guide the segmentation process. This type of semi-automated segmentation can be viewed as the interleaving of segmentation and object recognition. As part of this work on example based processing, a new segmentation algorithm has been developed. This example based segmentation algorithm is based on the same implicit modelling technique as the synthesis process. However, in order to regularise the solution, implicit modelling of the observed image is combined with an explicit modelling of the label field. The Bayesian framework provides a natural expression for such parallel modelling techniques. The new algorithm is presented under this framework and some sample segmentation results are given.

Declaration

I hereby declare that this thesis has not been submitted as an exercise for a degree at this or any other University and that it is entirely my own work.

I agree that the Library may lend or copy this thesis upon request.

Signed,

Claire Gallagher

October 29, 2006.

Acknowledgments

All the work has been done, the results are in, the thesis is written and all that is left to do is to thank those people who helped me throughout this work. There really are too many to give them all a mention here but I would like to take this opportunity to acknowledge a few people.

Firstly, I wish to express my deep gratitude to my supervisor Dr. Anil Kokaram for his guidance, advice and help. To all the staff of the Electronic and Electrical Engineering Department in Trinity College who have helped me throughout the years. In particular I would like to thank the past and present members of the SIGMEDIA group. Most especially Dr. Naomi Harte, Dr. Hugh Denman, Dr. Niall Rea, Louise Moriarty and Dr. Angela Quinlan.

This work was part financed by the Embark Initiative, operated by the Irish Research Council for Science, Engineering and Technology (IRCSET) and funded by the State under the National Development Plan. My thanks to all involved with the Embark Initiative.

To my wonderful family who have always supported me in the pursuit of further education. To my father for his advice and support and for allowing me the luxury to sometimes forget my student status! For my mother whose constant encouragement gave me the impetus to complete this degree. To my sister whose friendship and sense of humour I could not do without. Finally, a special word of thanks to Francis for all his help, kindness and patience.

Go raibh míle maith agaibh go léir.

Contents

List of Acronyms

1D	**1**-**D**imensional
2D	**2**-**D**imensional
AGP	**A**ccelerated **G**raphics **P**ort
ANN	**A**pproximate **N**earest **N**eighbor
AR	**A**uto **R**egressive
BP	**B**elief **P**ropagation
CDF	**C**ummulative **D**ensity **F**unction
CHF	**C**ircular **H**armonic **F**unction
CPU	**C**entral **P**rocessing **U**nit
CWT	**C**omplex **W**avelet **T**ransform
CBIR	**C**ontent **B**ased **I**mage **R**etrieval
DI	**D**iscrimination **I**nformation
DT-CWT	**D**ual **T**ree-**C**iscrete **W**avelet **T**ransform
DWT	**D**iscrete **W**avelet **T**ransform
EM	**E**xpectation **M**aximisation
FCM	**F**uzzy **C** **M**eans
GFLOPS	**G**iga **F**loating **P**oint **O**perations per **S**econd
GMRF	**G**aussian **M**arkov **R**andom **F**ield
GPGPU	**G**eneral Purpose Computations on Graphics Processing Units
GPU	**G**raphics **P**rocessing **U**nit
GRF	**G**ibbs **R**andom **F**ield
HMM	**H**idden **M**arkov **M**odel
HMT	**H**idden **M**arkov **T**ree
HSV	**H**ue **S**aturation **V**alue
HVS	**H**uman **V**isual **S**ystem
ICM	**I**terated **C**onditional **M**odes
IID	**I**ndependent **I**dentically **D**istributed

LBP	**L**oopy **B**elief **P**ropagation
LCPDF	**L**ocal **C**onditional **P**robability **D**ensity **F**unction
MAE	**M**ean **A**bsolute **E**rror
MAP	**M**aximum **A** **P**osteriori
MCMC	**M**arkov **C**hain **M**onte **C**arlo
ML	**M**aximum **L**ikelihood
MPM	**M**aximiser of **P**osterior **M**arginals
MR	**M**agnetic **R**esonance
MRF	**M**arkov **R**andom **F**ield
PCIe	**P**eripheral **C**omponent **I**nterconnect **e**xpress
PDF	**P**robability **D**ensity **F**unction
PMF	**P**robability **M**ass **F**unction
RGB	**R**ed **G**reen **B**lue
ROYGBIV	**R**ed **O**range **Y**ellow **G**reen **B**lue **I**ndigo **V**iolet
SA	**S**imulated **A**nnealing
SM	**S**ignal **M**odel
TSVQ	**T**ree **S**tructured **V**ector **Q**uantisation
WT	**W**avelet **T**ransform

1 Introduction

Once the task of the skilled professional, the popularity of the digital camera and the multitude of photo editing suites that exist, has brought even the most complicated image processing operation within a mouse click of the creative consumer. Fuelled by the commercial success of image processing software and coupled with the ever increasing demands of the professional image editor, the research area of image processing is one of the most vibrant sectors of information technology.

Image processing is a blanket term that can be used to describe any operation that acts to improve, correct, analyse, manipulate or render an image. In example based image processing the mechanism by which an image is manipulated or analysed is influenced directly by a set of example images. The driving force behind example based image processing is that many complicated image processing tasks can be simplified considerably if some information on the desired effect or outcome is given as an input. This thesis demonstrates the strength of example based image processing by focusing on two traditional image processing operations: texture synthesis and image segmentation.

1.1 Texture Synthesis

Texture synthesis by definition falls under the category of example based processing. Given an example texture sample, the goal of a successful texture synthesis algorithm is to synthesise or generate new texture which is perceptually similar to the inputted texture example. Texture synthesis is a large research area in the computer graphics industry and algorithms of this type

are widely used in image post-production [1], restoration [123] and compression [41,179]. One of the main challenges in texture synthesis involves extracting and describing the characteristics or behaviour of the example texture so that it can be replicated in the new texture. Extracting and characterising behaviour is the task of image modelling and so it can be said that the accuracy of the synthesised result will be dependent on the suitability of the image model to the example texture.

There have been many types of image models proposed and a review of some of the existing texture synthesis algorithms is given in chapter 2. In order to illustrate the strengths and weaknesses of the more pertinent approaches, some of the algorithms are implemented and synthesised results are presented. Based on the results of these algorithms, it was concluded that methods which were based on an implicit rather than an explicit model are more robust and successful. On the downside, these implicit modelling techniques have associated with them some inherent limitations which compromise their efficiency and flexibility. One of the contributions of this work is the development of a new texture synthesis algorithm which overcomes these limitations. This new algorithm is presented in chapter 4 and the strength of this new approach is illustrated through a comprehensive comparison between it and other popular/successful approaches.

1.2 Image Segmentation

Moving on from the work on texture synthesis, the focus of this thesis then turns toward the problem of image segmentation. As a formal description, the aim of a segmentation process is to assign to each pixel in an image a label indicating to which region or class that pixel belongs. In automated segmentation no information regarding the image to be segmented is given to the segmentation process and so by nature fully automated segmentation is an ill posed problem[1] which ultimately offers no means of judgment on the outcome. As a compromise much of the recent segmentation research has been focused on semi-automated segmentation where some clue as to the image content is given as an input [89, 130, 146, 174, 178, 190]. Semi-automated segmentation can be considered as the interleaving of object recognition and segmentation and the task is now that of: given an example object, does this object exist in the image and if so isolate and label it.

The adroitness of the human visual system (HVS) means that humans can ascertain the similarity between two objects within fractions of a second. However, as is often the case in image processing, devising an algorithm which can replicate artificially on a computer what the HVS does subconsciously is by no means a trivial task. In order to make the problem more manageable, it is broken down into smaller components, the first of which is how to characterise or describe each object. There have been many different object descriptors proposed. For example, in Video Google [190] and OBJ CUT [130], objects are described in terms of their shape and intensity, the Magic Wand tool from Adobe Photoshop 7 [205] uses colour intensity.

[1] A problem is well-posed if there exists a unique solution which is continuously dependent on the data [92].

Some of the most commonly used feature descriptors are reviewed in chapter 5 and of these it was found that characterising and identifying objects using their texture component is the most robust since by definition texture is composed of both intensity and spatial information. This is the approach taken by Mignotte [146] and the approach taken here will follow a similar vein.

Identifying and modelling objects in terms of their texture component allows the problem of object recognition to be re-formulated as that of texture discrimination and distinction. Under this formalisation there exists a symbiosis between texture synthesis and image segmentation whereby, the modelling process evoked in the synthesis process can be used as a means to identify texture (or equivalently objects) in the segmentation domain. A further contribution of this thesis is the development of a new segmentation algorithm which can be classified as example based segmentation. This new algorithm builds on the strength of the new texture synthesis algorithm and incorporates the implicit modelling process into a traditional segmentation framework. Segmentation results obtained using this algorithm are presented and the strengths and weaknesses of this approach are discussed.

1.3 Thesis Outline

This thesis is concerned with example based image processing and in particular focuses on two types of image processing operations: texture synthesis and image segmentation. The thesis is arranged in 7 chapters of which chapters 4 and 6 present the novel contributions of this research.

Chapter 2: The State of Texture Synthesis Today

The concept and significance of the visual feature that is texture is discussed in this chapter. The intrinsic features that can be used to describe texture are illustrated and the problem of texture synthesis is presented. A review of some of the various texture synthesis algorithms that have been developed is also given. These algorithms are categorised based on the type of underlying modelling process they use. Based on this categorisation, the strengths and weakness of each type of modelling process can be ascertained.

Chapter 3: Aspects of Wavelets

This chapter provides a brief introduction to the wavelet transform. The manner in which the wavelet transform is suited to image analysis is discussed and some examples of a wavelet decompositions are given. The chapter finishes by describing the wavelet transform which will be used to represent images in this work.

Chapter 4: Wavelet Based Texture Synthesis

A new texture synthesis algorithm is presented in this chapter. This new algorithm combines the strengths of the existing texture synthesis approaches investigated in chapter 2 with the power of wavelet based analysis. Results of this algorithm are given and a comparison is performed against results obtained using some of the more relevant previous approaches. Results from this chapter have been published in [76–78].

Chapter 5: An Introduction to Image Segmentation

This chapter introduces the problem of segmentation and discusses some of the issues that are involved in the segmentation process. A review of some of the features used to describe image content is given and a taxonomy of some of the existing segmentation algorithms is given. A Bayesian framework is presented and used to unify a large class of techniques.

Chapter 6: Example Based Image Segmentation

Under the Bayesian framework described in chapter 5, a new segmentation algorithm is presented in this chapter. This new algorithm called Example Based Image Segmentation is semi-automatic and builds on the success of the texture synthesis algorithm presented in chapter 4. A description of this algorithm is given as well as some of the segmentation results obtained.

Chapter 7: Final Comments and Future Work

The final chapter of the thesis assesses the contribution of the research presented here and outlines some possible ideas for future work.

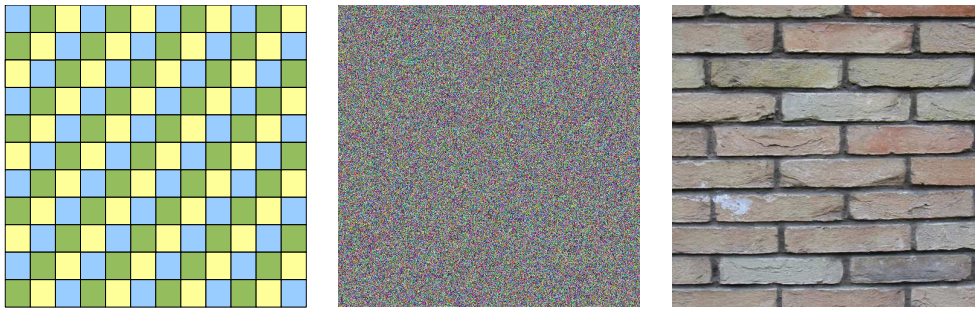
2

The State of Texture Synthesis Today

The problem of texture synthesis is a large research area in the computer graphics industry and has received much interest in recent years. Given an example of texture as a small sub image, the idea behind a successful texture synthesis algorithm is to create a new (larger) image by generating or *synthesising* more texture. This new synthesised texture should be perceptually similar and thus give the impression of being generated from the same underlying statistical process as the example texture. One main assumption in all texture synthesis algorithms is that the example texture is large enough to capture the underlying statistics of the overall infinite texture. The result of a successful texture synthesis process is shown in Figure 2.1. In this case, the example texture images (i) and (iii) are of size 128×128 pixels and the new synthesised

(i) Example (ii) Synthesised (iii) Example (iv) Synthesised

Figure 2.1: *Given an example texture image ((i) and (iii)) as a "seed", a (typically larger) texture image ((ii) and (iv)) is synthesised. This new image should be perceptually similar to the example texture from which it was generated.*



(i) Deterministic (ii) Stochastic (iii) Observable

Figure 2.2: *Three classes of textures: (i) Deterministic, (ii) Stochastic and (iii) Observable.*

images (ii) and (iv) measure 256×256 pixels.

Texture synthesis algorithms are commonly used in many image processing and computer vision applications. For example, in digital image post-production a large area can be covered with some sample texture in order to make the scene more realistic, e.g. in picture editing and restoration it is often required to interpolate missing information or remove certain objects from the original pictures. Texture synthesis processes are often used in such cases as they allow the missing pixels to be filled in with reasonable material [23, 53, 70, 123, 202]. Another large application of texture synthesis algorithms is in image compression, where the goal is to render object surfaces as visually similar to the real ones or as realistic as possible [41, 179].

There are two main challenges to be overcome when designing a stable, accurate, texture synthesis algorithm. These are:

1. *Modelling:* How to estimate and accurately reproduce the texture generation process that generated the original finite example texture. This modelling process may be implicit or explicit. The visual quality of the synthesised texture will be dependent on the accuracy with which the example texture is modelled.

2. *Sampling:* How to define a suitable procedure in order to generate samples from the proposed model. The computational efficiency of the texture synthesis algorithm will be directly related to the efficiency of the sampling procedure.

Before discussing some of the previous approaches that have been proposed to address the texture synthesis problem, the concept of texture and the features used to define it will be discussed in greater detail.

2.1 Texture Properties

Despite the many attempts that have been made to formally describe the essence of the feature that is image texture, no single definition has been agreed upon. However, a useful definition of image texture is that it should be spatially homogeneous, typically containing repeated structures

that often occur with some random variation [159]. In general, and as a means of classification and simplification, textures can be categorised into one of three types [33]. These are,

- **Deterministic:**
  A texture is classified as deterministic if its global structure is regular and well defined. Generally, deterministic texture is composed of a number of features that always appear in the same logical order. As a result, texture of this type can be described by a list of these features together with a set of rules that govern the manner in which they appear. An example of deterministic texture is shown in Figure 2.2 (i).

- **Stochastic:**
  Stochastic texture gives the impression of being generated from a totally random process. However, close inspection will normally reveal that it generally obeys certain statistical laws. An example of a stochastic texture is the white noise sample shown in Figure 2.2 (ii).

- **Observable:**
  Some call this type of texture *visual texture* [208] or *real-world texture* [45], but the slightly ambiguous name of *observable texture* [33] will be adopted here. Observable texture contains characteristics from both deterministic and stochastic texture classes but cannot be classified as purely deterministic or purely stochastic. Figure 2.2 (iii) shows an example of observable texture. Initial impressions indicate that this texture is repetitive in a manner similar to deterministic texture. However, this repetitiveness is not strict and although many of the features are repeated, they are not exactly identical. Therefore, there exists a random stochastic element which makes it difficult to model and hence synthesise.

As stated earlier the success of a texture synthesis algorithm will depend on its ability to accurately model the texture behaviour. Thus it is paramount to understand and appreciate some of the underlying features used to define the texture. Coarseness, directionality, contrast and randomness are just some of the adjectives that have been used to describe and define various textures. Other descriptors of texture are illumination, structure and stochastic measure (randomness) [45]. Two of the most basic characteristics that can be used to describe a texture are *regionality* and *resolution*.

Figure 2.3 illustrates the concept behind texture *regionality*, beginning with a 2×2 pixel area on the left and moving to a more complex 24×24 texture pattern on the right. As more pixels are introduced, the true pattern of the texture becomes evident. Depending on the size of the area or number of pixels given, information on the structure of the texture can change considerably. Thus, the regionality property of texture refers to how much texture, or more precisely how many pixels, must be represented in order to visually ascertain the true texture pattern. Clearly then in texture synthesis, the example texture must be large enough to capture the regionality, or the underlying statistics of the true texture pattern.

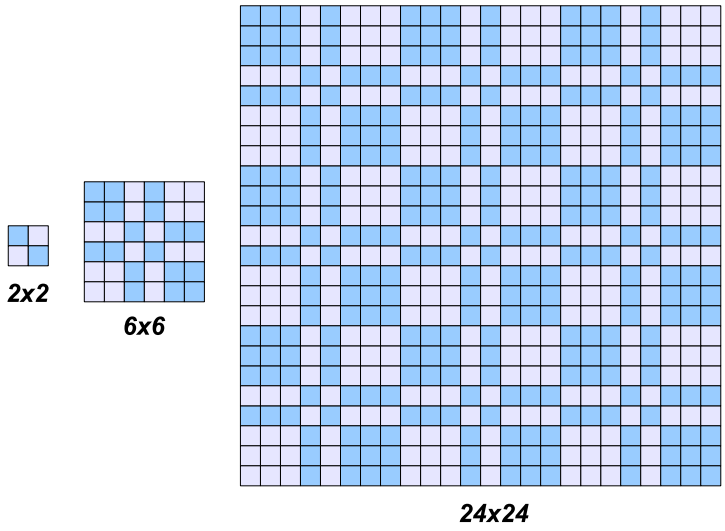
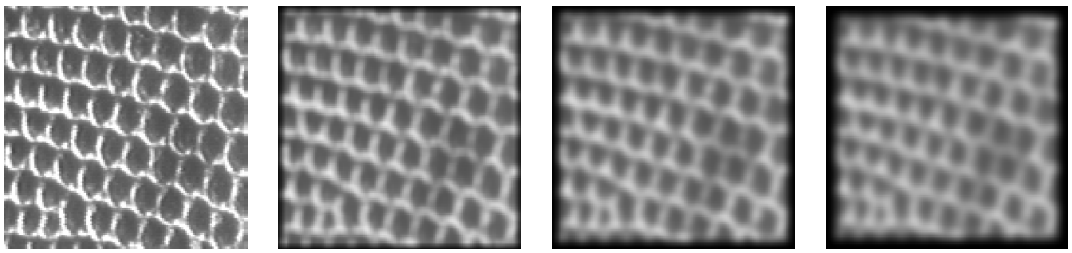


Figure 2.3: *Rregionality: increasing the texture area allows the true texture pattern to be realised.*

The other characteristic property of image texture is *resolution*. The resolution of a signal is linked to its frequency content. Textures by definition are composed of both high and low frequency components. High frequency components represent the edges or ridges present in a texture pattern. Here the change in intensity between neighbouring pixels is large. In contrast, low frequency information represents the building blocks of the texture, where pixel intensity remains almost uniform across neighbouring pixels. Figure 2.4 shows a multi-resolution image decomposition for a Brodatz [34] texture image. The original image (i) is filtered with a 10×10 averaging filter producing a lower resolution image[1] (ii). This removes some of the high frequency information by averaging it out. Similarly, image (iii) is obtained by low pass filtering (ii) and image (iv) is obtained by low pass filtering (iii) with the same spatial averaging filter. Since textures are characterised by their frequency components, it is a natural progression to model textures in the multi-resolution domain. This allows the different frequency components of the texture to be analysed separately.

Because texture is an area feature the regionality and resolution properties of a given texture are linked. That is, the regionality of a texture refers to the area of texture that needs to be present in order to ascertain the true texture pattern, while resolution is concerned with what frequency components characterise the texture. The texture area specified must be large enough to capture all the resolution of the texture. If the regionality of the texture is misinterpreted then some of the resolution present in the texture will be lost. Therefore, in order to accurately reproduce a given texture, the modelling process must capture both the regionality and resolution properties of the underlying infinite texture.

[1]Note the boundary around the edges of the low resolution images is introduced as a result of the zero padding used in the filtering process.



(i) (ii) (iii) (iv)

Figure 2.4: *Multi-resolution image decomposition. Image (i) is the original, image (ii)-(iv) are lower resolution versions of (i) obtained by removing the high frequency content by low pass filtering.*

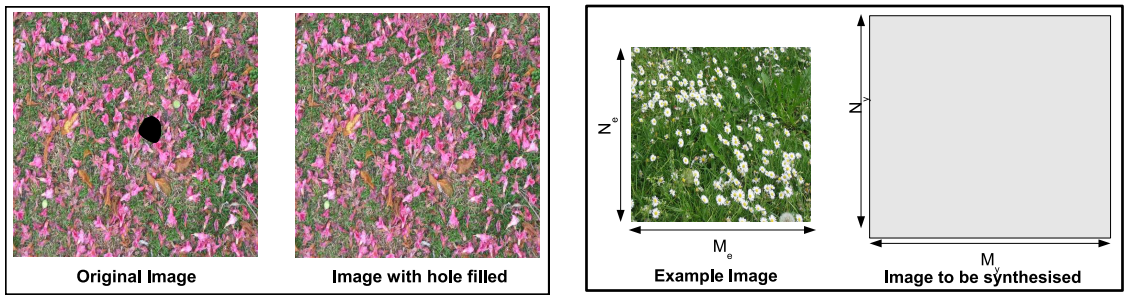
Moving on from this discussion on texture properties, the next section will describe some of the more popular texture synthesis algorithms.

2.2 A Taxonomy of Texture Synthesis Algorithms

There have been many approaches proposed to solve the texture synthesis problem. As stated earlier there are two main challenges in texture synthesis algorithm design. The first challenge is how to model the sample texture, so that the new synthesised texture will give the impression of being generated from the same underlying process as the sample texture. Under this modelling framework, the second challenge is concerned with how to sample efficiently from the model in order to generate the new texture. The first challenge of determining or defining a suitable model for the texture, is similar in goal to the problem of texture analysis.

Texture analysis is concerned with the implicit or explicit modelling of texture regions, so that different texture regions can be identified and defined. Texture analysis methods form an integral part of many image processing applications and in particular image segmentation. The development of an example based segmentation algorithm is one of the main goals of this work. The development of a successful texture model for application to texture synthesis will prove useful within an image segmentation framework. Thus, texture synthesis and image segmentation are closely related. Note that while texture synthesis and texture analysis methods are intrinsically linked, the review given here will focus on algorithms that have been specifically designed to solve the texture synthesis problem. Reviews of texture analysis methods can be found in [33, 95, 166, 168].

So far texture synthesis has been described as the problem of generating a (typically) larger output texture image that is perceptually similar to some example input texture. However, there exists another instance in which the texture synthesis problem can present itself. Figure 2.5 illustrates the two types of texture synthesis problem. In Figure 2.5 (i) texture *in-painting* or *filling-in* is shown. The original image (left) contains missing pixels and the goal of the in-painting algorithm is to generate new pixel values to fill in these missing regions. There should



(i) In-painting or Filling-in (ii) Texture Synthesis

Figure 2.5: *Two types of texture synthesis: (i) Texture in-painting or filling in and (ii) Texture synthesis in which an new image is generated from an example texture.*

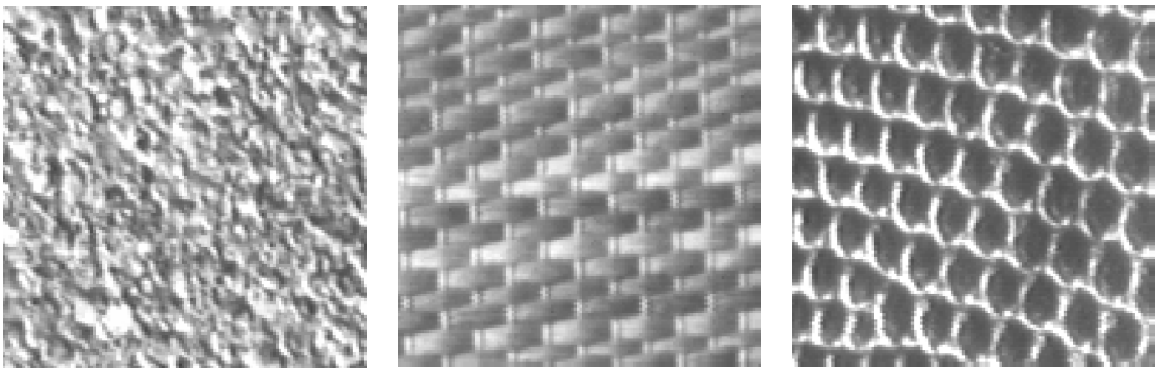
be no discontinuity between the original pixel values and the new generated or synthesised pixels. Texture in-painting algorithms can be interpreted as texture synthesis with boundary conditions. Normally areas to be filled in with new texture are small and the greatest concern is to interpolate smoothly between the boundaries and new synthesised region. Descriptions of these algorithms [19, 88, 123, 155, 210] will not be discussed here as they are generally not suitable for synthesising large regions.

Figure 2.5 (ii) shows the other type of texture synthesis problem which is the focus of this work. Let $\mathbf{I_e}$ denote the input texture sample texture defined on an $M_e \times N_e$ lattice $\mathbf{X_e}$. Each site in $\mathbf{I_e}$ can be indexed using the spatial vector $\mathbf{p} = [x, y]^T$, such that $I_e(\mathbf{p})$ denotes the pixel at location $\mathbf{p}$ in $\mathbf{I_e}$, where $\mathbf{p} \in \mathbf{X_e}$. The goal of the texture synthesis process is to generate a new perceptually similar texture image $\mathbf{I_s}$ that will be defined on a new (typically larger) $M_s \times N_s$ lattice $\mathbf{X}$. Each site in $\mathbf{I_s}$ can be indexed by the spatial vector $\mathbf{x} = [x, y]^T$ such that $I_s(\mathbf{x})$ represents a pixel at location $\mathbf{x}$ in $\mathbf{I_s}$, where $\mathbf{x} \in \mathbf{X}$.

Existing texture synthesis algorithms that generate an output texture image from a given sample texture can be roughly categorised into three classes [131]: parametric, non-parametric and patch based. Some of the approaches that are especially pertinent to this work have been implemented and synthesised results are shown. In order to appreciate the subtle differences between the results obtained using the various approaches, the same set of 128×128 pixel texture images will be used as the example texture. The example texture images were taken from the Brodatz collection [34] and are given in Figure 2.6. The images synthesised using this example set were sized 128×128.

2.3 Parametric Approaches

Parametric approaches explicitly model the texture using some definable process and characterise each texture using a finite set of parameters. There have been many different processes proposed to model texture behaviour and parametric texture models are widely used in texture



(i) (ii) (iii)

Figure 2.6: *Texture images taken from the Brodatz [34] collection. These images measure* 128×128 *pixels. These texture are all observable and so contain a mixture of stochastic and deterministic components. Texture image (i) is more random and so is mainly stochastic, while images (ii) and (iii) are more structured and thus deterministic.*

analysis and recognition [182]. This section will outline some of the more popular parametric approaches.

Parametric Pyramid Based Synthesis [100], (1995)

Heeger and Bergen [100] use histograms at different frequency bands as a texture description. Their approach is based on the assumption that all of the spatial information characterising a texture image can be captured in the first order statistics of a defined set of linear filter outputs. They model the texture in a multi-resolution domain using either a *Laplacian* [38] or *Steerable* pyramid structure [189]. An image pyramid is created by convolving and sub-sampling an image with a bank of linear filters. The defining characteristic of an image pyramid is that the basis/projection functions are translated and dilated copies of one another (translated and dilated by a factor of 2^i for some integer i). Hence, each of the sub-band images are of different size and correspond to predefined frequency bands. Figure 2.7 illustrates the structure of a three level pyramid.

Using either a Laplacian or a Steerable pyramid, the Heeger and Bergen texture synthesis algorithm begins with the example texture $\mathbf{I_e}$ and a uniform sample of white noise of the same dimension as the image to be synthesised $\mathbf{I_s}$. To modify the noise sample to resemble the example texture, the algorithm performs histogram matching across the different levels of the pyramid structure. The Laplacian pyramid transform represents a given image as a sum of shifted, scaled and dilated (approximately) Gaussian functions. The name Laplacian pyramid name comes from the fact that the projection functions of the transform are (approximately) Laplacian-of-Gaussian. While this transform is easy to compute, its basis functions are (approximately)

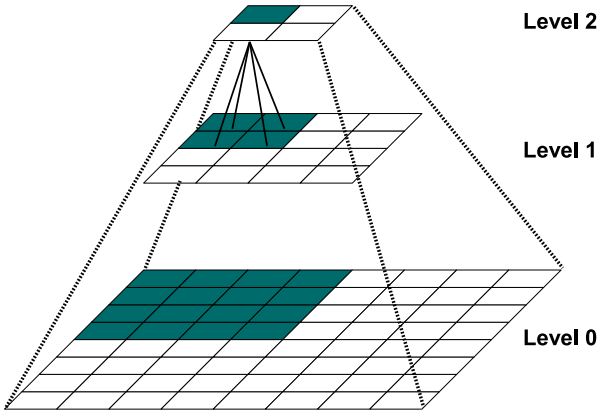


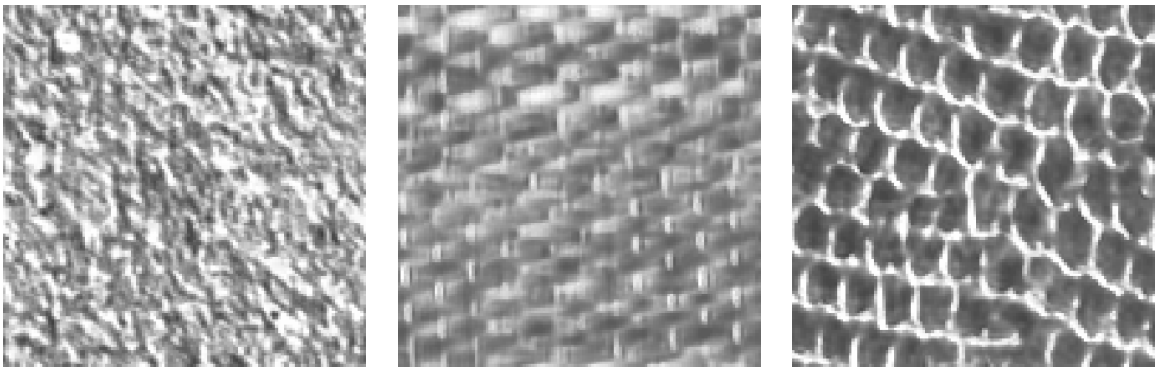
Figure 2.7: *Image pyramid representation showing three levels.*

radially symmetric and as a result it fails to capture the structural information of the texture. To obtain a more realistic representation for more complicated textures, the Steerable pyramid is preferred by Heeger and Bergen. Similar to the Laplacian pyramid, it decomposes an image into different spatial frequency bands. However, it also further divides each frequency band into a set of orientation bands.

Within the pyramid structure and treating each sub-band independently, the algorithm modifies the white noise sample to resemble the example texture by performing histogram matching across each frequency band. Histogram matching is a generalisation of histogram equalisation [100]. The idea is to take a particular image and coerce it via a pair of lookup tables to have a particular histogram. The two lookup tables are the cumulative distribution function of one image and the inverse cumulative distribution function of the other image. Their algorithm is iterative and includes the following steps.

1. Match the histograms of the noise sample $\mathbf{I_s}$ to that of the input texture $\mathbf{I_e}$.
2. Compute the pyramids for both the (modified) noise and texture images.
3. Loop through the two pyramid structures and match histograms of each of the corresponding pyramid sub-bands.
4. Collapse the (histogram-matched) noise pyramid to generate a preliminary version of the $\mathbf{I_s}$.
5. Repeat steps 1-4 for several iterations.

The Heeger and Bergen synthesis algorithm is efficient and works well for simple stochastic textures. However, by treating each sub-band independently, the algorithm fails to reproduce the extended structural elements which occur over many frequency bands. Thus it does not work well for deterministic textures.



(i) (ii) (iii)

Figure 2.8: *Synthesised textures using the Simoncelli and Portilla [159, 188] algorithm.*

Parametric Wavelet Based Synthesis [159, 188], (1998, 2000)

Simoncelli and Portilla [159,188] extend the approach taken by Heeger and Bergen by replacing the pyramid transform with a wavelet transform. This resulted in a larger more complete parameter set to characterise the texture. To capture both the structural and random aspects of the texture, their texture descriptions are based on: (i) the local spatial correlation of wavelet coefficients within each sub-band, (ii) the local spatial correlation of wavelet coefficient magnitudes, (iii) the cross-correlation between coefficient magnitudes at adjacent scales and orientations and (iv) the first few moments of the pixel histogram. As with the Heeger and Bergen algorithm, the image to be synthesised $\mathbf{I_s}$ is initialised with white Gaussian noise. Taking the non-deciminated wavelet transform of both the example texture $\mathbf{I_e}$ and the image to be synthesised $\mathbf{I_s}$, they modify the wavelet tree of the image to be synthesised by forcing it to satisfy the (i)-(iv) texture descriptions of the sample texture. They do this by finding an orthogonal projection from the filter responses of the synthetic texture to that of the sample texture. After the projection of all filter responses, the wavelet tree is collapsed, further projection is performed at the single resolution level and then the wavelet tree is reconstructed. This iteration is repeated until convergence is reached. Their method works well but still has some difficulty in replicating the structural component of deterministic textures. Evidence of this is shown in Figure 2.8 which shows some synthesised textures obtained using the Simoncelli and Portilla algorithm. The example textures are those given in Figure 2.6. Each synthesised image measures 128×128 pixels. Four levels of the wavelet tree were used with four different orientations. The local spatial correlation around wavelet coefficients and coefficient magnitudes was measured using a 9×9 pixel neighbourhood.

Synthesis By Analysis [107], (1998)

Jacovitti et al. [107] proposed a texture synthesis-by-analysis method based on a Hard limited Gaussian process. Their two stage algorithm aims to approximate the first- and second-order dis-

tributions of the sample texture, according to the Julesz conjecture [113]. The Julesz conjecture states that humans cannot distinguish between textures with identical second-order statistics. Thus, by this rationale, if the synthesised image has the same second order statistics as the sample texture, then both textures give the impression of being generated from the same underlying statistical process. The algorithm described in [107] can be divided into two stages: analysis and synthesis. During the analysis stage the binary textural behaviour of the sample texture $\mathbf{I_e}$ is represented by means of a hard-limited Gaussian process. During the synthesising stage, the binary hard limited Gaussian is passed through a linear filter and a zero memory histogram equaliser. However, this method was designed to work with artificial texture used in the computer graphics industry and does not work well on observable textures.

Synthesis Using Circular Harmonic Functions [40], (2002)

Improving on the texture synthesis-by-analysis algorithm, Campisi and Scarano [40] combined the methods proposed in Jacovitti et al. [107] with that of Heeger and Bergen [100] and Simoncelli and Portilla [159,188] to propose a multi-resolution approach for texture synthesis using circular harmonic functions (CHF). Similar to [107], the algorithm proposed in [40] seeks to satisfy the Julesz conjecture. However, this time the task is accomplished using a hybrid approach which operates partially in the spatial domain and partially in the circular harmonic function domain. The multi-resolution circular harmonic function domain was chosen as it has been proven to be well suited for mimicking the behaviour of the human visual system (HVS). Similar to the Jacovitti et al. approach, in the algorithm proposed by Campisi et al. the binary excitation with the desired spatial correlation (taken from the sample texture) is generated by hard limiting a filtered white Gaussian random field. In addition, the synthetic binary excitation is passed through a filter bank which performs a multi-frequency channel decomposition. The coincidence between the first-order distribution of the filter bank outputs and the corresponding components obtained in the parameters identification stage is imposed. Finally, the inverse transform filter bank is applied and the output feeds a synthesis filter whose role is to add details to the image. This method has been tested on some of the texture images taken from the Brodatz collection. It works well for the random stochastic textures but fails to replicate the structural components associated with deterministic textures.

Modelling Texture with Markov Random Fields

In order to capture the local characteristics of the texture it is often modeled as a realisation of a *Markov Random Field* (MRF). The random field $\mathbf{I}$ is called a MRF with respect to the neighbourhood system $\mathbf{N} = \{N(\mathbf{x}), \forall \mathbf{x} \in \mathbf{X}\}$ if,

$$p(I(\mathbf{x})|I(\mathbf{r})), \mathbf{x} \neq \mathbf{r}) = p(I(\mathbf{x})|I(\mathbf{r}), \mathbf{r} \in N(\mathbf{x})) \tag{2.1}$$

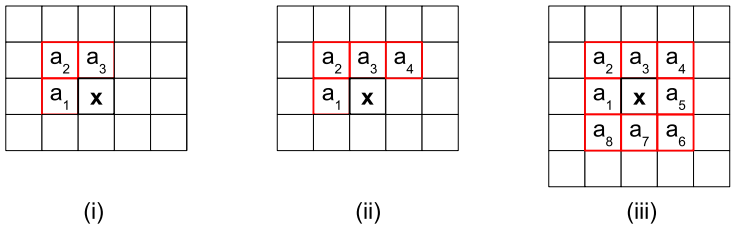


Figure 2.9: *The geometry of typical 2D AR neighbourhood. Neighbourhood (i) is a 3 tap causal model, neighbourhood (ii) is semi-causal and there are four pixels used in the predication of the pixel at* $\mathbf{x}$*. In this case the offset vectors* $\mathbf{q_k}$ *are* $[-1,0],[-1,-1],[0,-1],[1,-1]$*. Neighbourhood (iii) is non-causal neighbourhood and there are eight pixels used in the prediction of the pixel at* $\mathbf{x}$*.*

for all $\mathbf{x} \in \mathbf{X}$, $\mathbf{r} = [x, y]^T$ is a spatial vector where $\mathbf{r} \in \mathbf{X}$ and $N(\mathbf{x})$ represents the neighbourhood of pixel values centred at location $\mathbf{x}$. The Markovian property given in (2.1) states that the probability distribution for one site given the value of its spatial neighbours is independent of the rest of the field. It expresses the fact that all the information about one variable is carried by the value of its neighbours. Note that this does not mean that two variables on sites that are not neighbours are independent of each other, as all variables are in general mutually dependent, but only through successive *local interactions*. These local interactions are specified by the local conditional probabilities $p(I(\mathbf{x})|I(\mathbf{r}), \mathbf{r} \in N(\mathbf{x}))$.

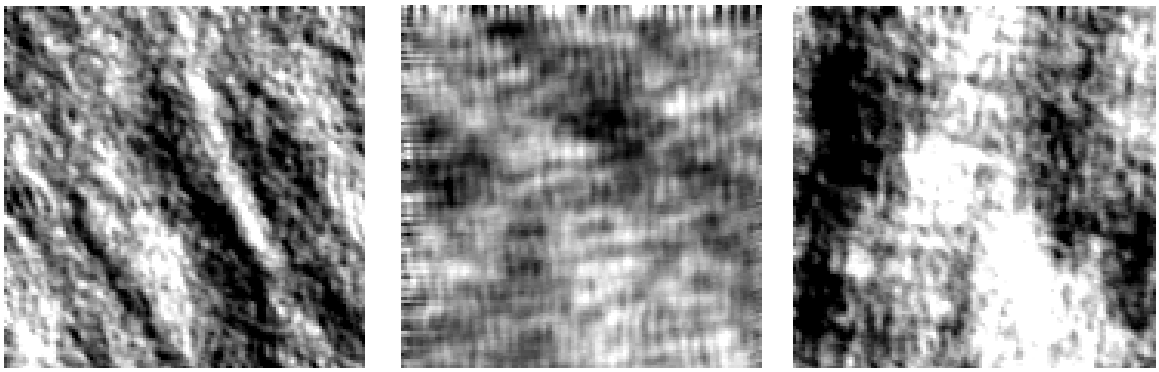
Parametric Markov Random Field Model [47, 124], (1985, 2004)

Using the Markovian definition, Chellappa and Kashyap [47] model texture using the $2D$ non-causal autoregressive (AR) model. The 2D AR model is a subset of the Gaussian Random Field model (GMRF). Similarly, Kokaram [124] uses a 2D AR process to model texture. Under the 2D AR model, each pixel $I(\mathbf{x})$ at site $\mathbf{x} \in \mathbf{X}$ is given as a linear combination of pixels in the neigbourhood around $\mathbf{x}$ plus an excitation or residual error $e(\mathbf{x}) \sim \mathcal{N}(0, \sigma_e(\mathbf{x})^{\mathbf{2}})$. That is,

$$I(\mathbf{x}) = \sum_{k=1}^{P} a_k(\mathbf{x}) I(\mathbf{x} + \mathbf{q_k}) + e(\mathbf{x}) \tag{2.2}$$

The P coefficients of the model are denoted $\mathbf{a} = [\mathbf{a_1}, \cdots, \mathbf{a_P}]$. The pixels used in the predication are called the support or neighbourhood of the model and are mapped by the P spatial offset vectors $\mathbf{q_k}$. Figure 2.9 illustrates some typical model neighbourhoods.

In all parametric texture synthesis algorithms, the overall problem can be divided into firstly estimating the model parameters that characterise the texture, and then using these parameters to estimate the new pixel values. In the case of the 2D AR model, the model parameters are made up of $[\mathbf{a}, \sigma_e^2]$. The model parameters are calculated using the least squares estimate over the example image. Once these parameters have been estimated, new pixels are generated using the relationship given in (2.2). Either raster or checkerboard [123] scans can be used to govern



(i) (ii) (iii)

Figure 2.10: *Synthesised texture using the 2D AR process proposed in [124].*

the order in which pixels are synthesised in the new image.

Figure 2.10 shows some results obtained using the AR algorithm described in [124]. A 9×9 causal neighbourhood was used and the residual error was given as $e(\mathbf{x}) \sim \mathcal{N}(\mu_e, \sigma_e^2)$, where μ_e and σ_e were calculated from the sum of the residual errors in the example image. Pixels were synthesised in a raster scan order and the synthesis process was repeated for 5 iterations. The synthesised images in Figure 2.10 are not an accurate reproduction of the original example images in Figure 2.6. These poor results can be attributed to the fact that the AR model used is quite basic and the order of the model remains invariant to the type of input texture. Secondly, it is found that in general images do not obey Gaussian behaviour and so assuming a Gaussian distribution over the residual error is audacious. Improved results could be obtained if the residual error in the synthesised image was more in line with that in the model of the example image. Chellappa and Kashyap recognise this and use the histogram of residual values in the example image as a basis for determining residual errors in the synthesised image. As a result, more accurate synthesised results can be found in their paper [47].

An alternative idea is to implicitly model the residual error in the synthesised image using the residual error in the example image. That is, given the map of residual errors in the example image, the idea would be to grow residual errors in the synthesised region using a non-parametric modelling process derived from the Efros and Leung [68] texture synthesis process. This idea will not be further investigated at this point but is a possible direction for future work. A description of the Efros and Leung algorithm will be given later in the review of non-parameter texture synthesis algorithms.

For single resolution MRF models, one of the limitations is the size of the neighbourhood that can be used. Large neighbourhood sizes imply large areas of pixel dependency and so are computationally expensive. One way to overcome this is to introduce the MRF model over a multi-resolution domain. At a coarser resolution large features are represented by fewer pixels and by using a MRF model at this level longer range pixel dependencies can be introduced.

The power of multi-resolution analysis was demonstrated in the discussion on texture charac-

teristics where resolution or frequency composition was identified as one of the defining features of a given texture. Because texture is composed of multiple frequencies, it makes sense to represent the texture in a domain where each of these frequencies can be analysed separately. There are a number of multi-resolution transforms available and some of the algorithms discussed previously [40, 100, 159, 188] exploited the power of multi-resolution texture analysis by performing synthesis over a number of frequency bands. Performing synthesis in a multi-resolution domain allows the dominant frequencies present in the texture to be exploited. In addition, multi-resolution synthesis is computationally attractive. Due to sub-sampling inherent with multi-resolution transforms, at a coarse resolution the features or *textons*[2] that characterise the textures are represented by fewer pixels. Therefore, analysis and synthesis of the texture can be performed on a much reduced data set. Finer level features can then be added in accordance with the sub-sampling law that is associated with the transform.

Synthesis using Hidden Markov Models [69], (2003)

Fan and Xia [69] proposed a wavelet based texture synthesis model based on the Hidden Markov Model (HMM). The wavelet transform (WT) is a type of multi-resolution transform which has good directional selectivity and as a result is commonly used in texture analysis algorithms. The texture synthesis algorithm developed as part of this work will be based in the wavelet domain and so some aspects of the wavelet transform will be presented in chapter 3. The HMM is a variant of the basic MRF model and can essentially be classified as a doubly stochastic random field in which there exists a hidden parameter whose value in not directly observable to the user. Crouse et al. [55] proposed the use of the HMM in order to characterise the wavelet dependencies among wavelet coefficients at different levels of the wavelet tree. In [69] the HMT-3S model is based on that proposed in [55] and was developed by integrating three sub-bands into one tree structure using a graphical grouping technique [73]. Using a similar framework to Simoncelli and Portilla [160,188] the HMT-3S is applied in order to impose cross-correlations and marginal statistics. The HMT-3S texture synthesis algorithm was tested on some of the Brodatz [34] textures and found that it worked well for stochastic textures but failed for deterministic textures. Results obtained using this approach are shown in [69].

The above discussion outlines some of the more popular parametric methods that have been developed. In general it is found that while they are efficient and work well for stochastic textures, they have difficulty in modelling the more complicated deterministic textures. Given the finite set of parameters associated with parametric models, it is unrealistic to assume that these models will be able to handle the wide range of possible real-world textures. Motivated by these limitations, researchers turned their attention to an alternative type of modelling process. Non-parametric based approaches will be discussed next.

[2]Textons are the repeated elements that make up a texture pattern [45]

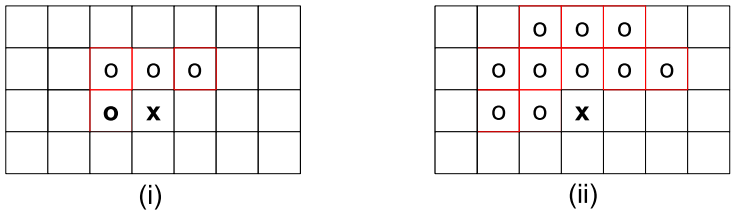


Figure 2.11: *Two causal conditional neighbourhoods (i) and (ii) as used by Popat and Picard [158] in their cluster-based probability model for texture synthesis.*

2.4 Non-Parametric Approaches

Parametric approaches aim to explicitly model the texture using some definable process, thus characterising it using some set of finite parameters. In contrast non-parametric methods offer no such model and rather implicitly model the texture using statistics heuristically obtained from the texture image itself or some other image of similar content. In order to obtain heuristic measurement of image statistics, non-parametric models generally adopt a Markovian assumption over the image field. This Markovian assumption allows the value of the pixel $I(\mathbf{x})$ at any site $\mathbf{x}$ to be dependent only on the pixels values in the neighbourhood $N(\mathbf{x})$ surrounding $\mathbf{x}$. While this assumption has been proved to be valid [68], one major problem associated with it is determining what sort of neighbourhood should be used. Since the heuristic measurements and hence the implicit modelling are very much dependent on the neighbourhood used, discerning a suitable neighbourhood structure for the given texture is fundamental to the modelling process. For the texture model to be robust and flexible, this neighbourhood structure should remain constant for all texture types. If such a condition is met, then the algorithm will be scale invariant. That is, the model will be robust enough to model all types of texture regardless of the underlying texture scale. The notion of texture scale relates to the regionality and resolution properties discussed earlier. That is, how big of an area is needed to capture the true texture pattern so that within this area all the frequency components associated with the texture are present. The following discussion on some of the existing non-parametric texture synthesis algorithms will highlight that scale dependence is a problem associated with many of them.

Causal Cluster Based Synthesis [158], (1993)

Non-parametric approaches implicitly model the new texture image using statistics obtained from heuristic measurements over the training sample texture. Popat and Picard [158] proposed a cluster-based probability model for texture synthesis. Their method can be classified as a non-parametric Markov chain synthesis algorithm which synthesises each pixel in the new image sequentially. Each pixel to be synthesised is assigned a pseudo-random value that is generated according to its conditional probability mass function (PMF), where the conditional values

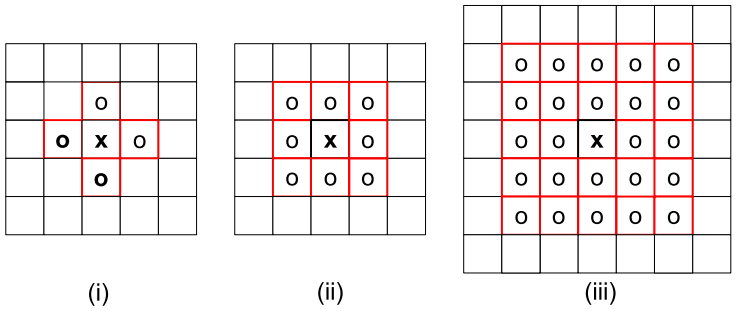


Figure 2.12: *Non-causal neighbourhood structures used in the Paget and Longstaff [151] texture synthesis algorithm. Neighbourhood (i) is first-order, (ii) is second order and (iii) is eighth order.*

are a subset of the previously generated pixels. Before pixels can be synthesised, the model is trained using the example texture image. Taking the neighbourhood around the pixel to synthesised, each pixel in the example image is concatenated onto this neighbourhood vector of pixels. Figure 2.4 illustrates two example *causal conditional neighbourhoods* of the type used in [158]. A clustering based model is then trained on these vectors using a clustering process described in [136]. This clustering process is semi-supervised where the number of clusters must be specified beforehand and will be chosen based on the computational resources available together with the desired fidelity of the model. Each cluster is modelled as a Gaussian process and so the overall model is a Gaussian mixture model.

To apply the model to synthesise each pixel, a recursive procedure based on previously synthesised pixels is used for computing the conditional PMFs of each pixel in the training image. These PMFs are then used in generating the new pseudo-random pixel value. The above procedure is non-hierarchical where synthesis is performed in a single resolution image only. A hierarchical version of the algorithm is also presented in [158]. In the hierarchical algorithm, texture is synthesised in several stages, beginning with a coarse resolution version of the texture (establishing macroscopic structure) and proceeding to progressively finer resolutions (filling in microscopic detail). The hierarchical approach is simple, where each finer resolution is obtained by first up sampling the current resolution, then filling in the missing pixel values using the cluster based model.

The algorithm works well on stochastic textures but fails to accurately capture the structural components associated with deterministic textures. Their approach is causal and so synthesis is performed in a sequential manner beginning with a "seed" taken from the sample texture and gradually moving away. The problem with this type of approach is that if the previously synthesised pixels start to deviate too far from true texture pattern, the synthesis algorithm can get lost in a space where only garbage pixels that do not resemble the texture pattern are produced. The causality of the algorithm severely limits its ability to synthesise areas of texture.

Non-Causal Synthesis [151], (1998)

The problem with the sequential approach of [158] was addressed by Paget and Longstaff [151] in their non-causal non-parametric multi-scale Markov random field approach to texture synthesis. In [151] a top down approach is used, where frequency components of a texture are gradually introduced into the synthetic texture from low to high frequency. The same multi-scale representation is used in [151] as was used in [158]. Figure 2.12 illustrates the non-causal neighbourhood structures used in [151]. Texture synthesis is performed via a multi-scale synthesis algorithm incorporating local annealing in the form of a pixel temperature function. This temperature function controls the size of the neighbourhood at each resolution. Beginning at a coarse level and moving progressively toward finer resolutions, pixels are synthesised from the non-parametric MRF model using either the Gibbs sampler [82] or the more computationally attractive deterministic iterative conditional modes [20] algorithm. Both of these optimisation techniques will be discussed in more detail in chapter 5. The algorithm described in [151] is tested on both stochastic and deterministic textures and the order of improvement over [158] is considerable. However, their synthesised results still contain patches which are not in line with the true texture pattern, thus compromising the visual quality. Results of the algorithm are shown in [151].

Non-Parametric Pyramid Based Synthesis [26], (1997)

DeBonet [26] removed the iterative requirement of the parametric Heeger and Bergen [100] algorithm and rather enforced a similar but non-parametric coarse to fine resolution approach. This allowed the frequency components of the sample texture to be gradually introduced into the synthesised texture from low to high frequency. In a two-phase process, the input texture is first analysed by measuring the joint occurrence of texture discrimination features at multiple resolutions. In the second phase, a new texture is synthesised by sampling successive spatial frequency bands from the input texture, conditioned on the similar joint occurrence of features at lower spatial frequencies. The texture structure is also better preserved by further restricting the sampling procedure to pixels that fall within a threshold determined by the texture features. The results obtained using [26] outperformed those obtained using the Heeger and Bergen approach. However, the tuning of threshold parameters is difficult and the success of the results is very much dependent on these parameter values.

A Shannon Inspired Approach [68], (1999)

While the non-parametric approaches described above generated some good results especially with stochastic textures, their ascendancy over parametric approaches when applied to deterministic textures was still in question. However, the power and strength of the non-parametric modelling technique was finally demonstrated by Efros and Leung [68] in their non-parametric

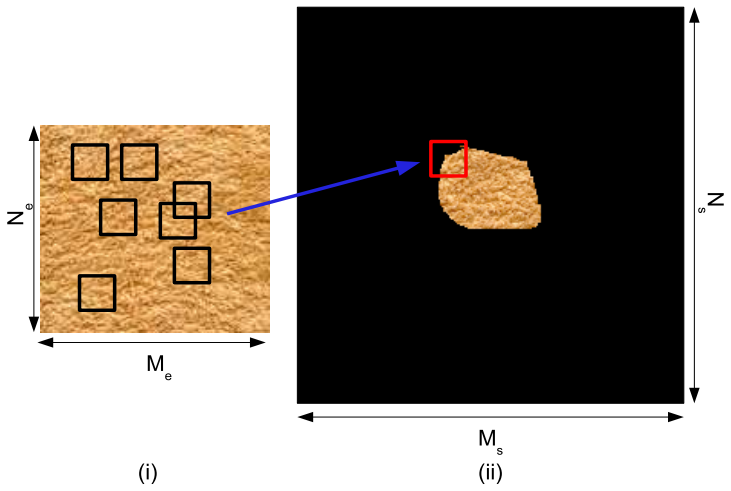


Figure 2.13: *Block matching in the Efros and Leung [68] texture synthesis algorithm. Image (i) is the example texture* $\mathbf{I_e}$ *and image (ii) is the new texture image* $\mathbf{I_s}$ *to be synthesised. For each pixel* $I_s(\mathbf{x})$ *to be synthesised at site* $\mathbf{x}$ *in* $\mathbf{I_s}$*, its neighbourhood (red block)* $N(\mathbf{x})$ *is compared to all possible neighbourhoods* $N(\mathbf{p}), \forall \mathbf{p} \in \mathbf{X_e}$ *in the example texture. The set of best matching neighbourhoods (black blocks) is found and the centre pixels of these neighbourhoods form an approximation to the probability density function (pdf) of* $I_s(\mathbf{x})$*, i.e.* $p(I_s(\mathbf{x}))$*. This pdf is sampled randomly and the new chosen value is assigned to* $I_s(\mathbf{x})$*.*

texture synthesis algorithm. The approach taken by Efros and Leung modifies the algorithm proposed by Popat and Picard by introducing an exhaustive nearest neighbour searching process. In addition, a Shannon inspired heuristic is proposed where the image to be synthesised is initialised with a "seed" texture taken from the example texture. Figure 2.13 illustrates the exhaustive block searching process used by Efros and Leung.

The Efros and Leung algorithm proceeds as follows. Beginning with an example texture image $\mathbf{I_e}$ and the image to be filled with texture $\mathbf{I_s}$, the algorithm is initialised by placing a "seed" taken from the example texture and placing it in the centre of the new image to be synthesised. For each pixel $I_s(\mathbf{x})$ to be synthesised in $\mathbf{I_s}$, its $w \times w$ neighbourhood block $N(\mathbf{x})$ centred on $\mathbf{x}$ is taken and compared to every possible $w \times w$ neighbourhood block $N(\mathbf{p})$ in $\mathbf{I_e}$. This block is weighted by a Gaussian kernel to emphasise local structure. As will be demonstrated later, the success of the algorithm is very much dependent on choosing a suitable neighbourhood size $w \times w$ for the given sample texture. Figure 2.4 illustrates two typical neighbourhood structures used in [68].

When comparing two neighbourhood blocks, the distance measure is given as the normalised sum of the squared differences. Using this metric, the set of best matching neighbourhoods is found and the centre pixels of these neighbourhoods form an approximation to the probability density function $p(I_s(\mathbf{x}))$ of the pixel to be synthesised. This pdf is sampled randomly and the chosen value is assigned to $I_s(\mathbf{x})$. This process is repeated for all unknown pixels in $\mathbf{I_s}$. Figure

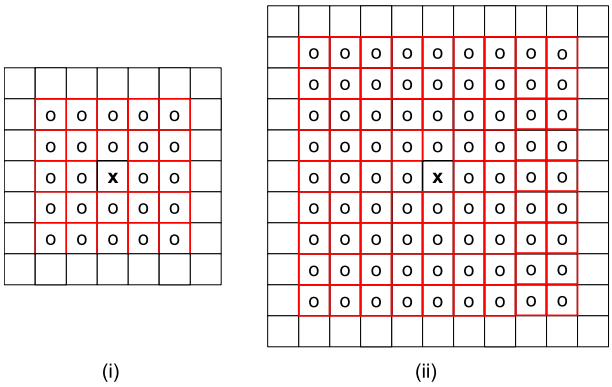


Figure 2.14: *Neighbourhood structures used in the Efros and Leung algorithm. Neighbourhood (i) measures* 5×5 *pixels while neighbourhood (ii) is composed of* 9×9 *pixels.*

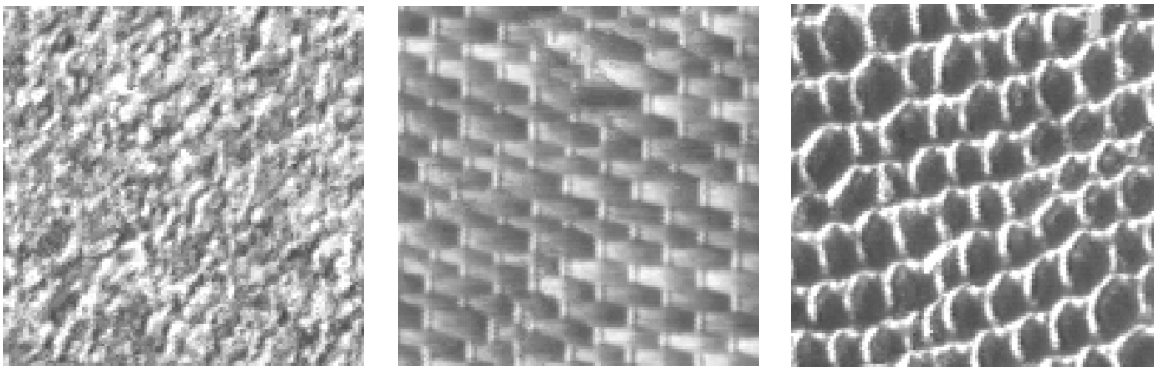


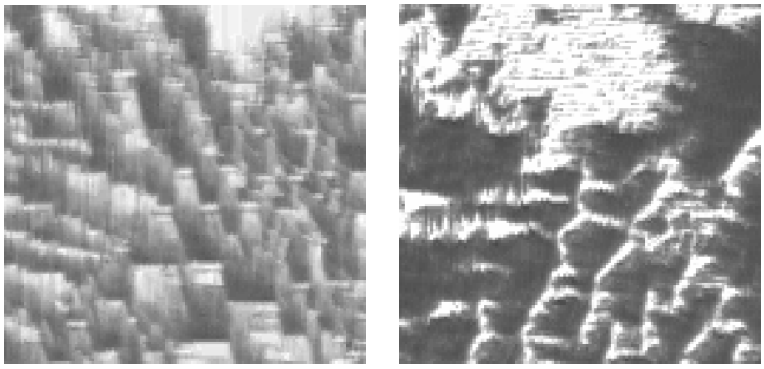
Figure 2.15: *Synthesised textures obtained using the Efros and Leung [68] algorithm. In the block matching process the neighbourhood sizes used were:* 7×7 *for image (i) and* 17×17 *for image (ii) and (iii).*

2.15 shows some of the results obtained using the Efros and Leung algorithm.

Note that in the Efros and Leung algorithm the image to be synthesised was initialised with just a 3×3 "seed" taken from the example texture. However, in the implementation carried out as part of this work, it was found that this small "seed" was insufficient to initialise the synthesis process given that a neighbourhood of 17×17 was needed. Hence, the "seed" used is of size 16×16 and is placed in the centre of the new 128×128 synthesised image.

Results obtained using the method proposed in [68] demonstrate that it works on stochastic (i) and the more complicated deterministic (ii) and (iii) texture images. Initial indications would suggest that this non-parametric modelling technique developed in [68] appears to have solved the texture synthesis problem. However, this is not the case and Figure 2.16 illustrates two inherent weaknesses associated with [68].

The synthesised texture in Figure 2.16 (i) shows that while the pixel synthesis algorithm



(i) (ii)

Figure 2.16: *Synthesised textures obtained using the method presented in [68]. The example images were taken from Figure 2.6 (ii) and (iii). The synthesised results shown here demonstrate how the quality of the synthesised result is very much dependent on the correct interpretation of texture scale. In this case, the neighbourhood sizes used were* 5×5 *pixels and the synthesised result fails to resemble the original example texture.*

starts off well, the stability of the algorithm decreases as the synthesis process moves further away from the "seed". At the boundaries, the texture pattern fails to resemble the true texture pattern in the example texture. The same initial conditions were used as in Figure 2.15 (ii), that is the example texture measures 128×128, the synthesised texture image measures 128×128 and a 16×16 pixel "seed" was used to kick start the pixel "growing". However, for the synthesised results in Figure 2.16, the neighbourhood size used is 5×5 pixels. This block size fails to capture the regularity of the example texture so both results are unstable. These results illustrate how the success of the algorithm is very much dependent on the correct choice of neighbourhood size. In theory the neighbourhood size $w \times w$ should capture the largest feature present in the example texture. Taking too small a neighbourhood means the large features present in the example texture will not be represented in the synthesised result. Similarly, if w is too large, the randomness of the example texture will not be replicated in the synthesised result. Given the variability of texture behaviour and the infinite set of possible texture patterns, there is no one value of $w \times w$ that will remain suitable for all texture types. Therefore, w needs to be chosen according to the input texture. This implies that the algorithm proposed in [68] is scale dependent.

As well as failing to replicate the regularity of the example texture, Figure 2.16 (i) and (ii) illustrate another problem with the Efros and Leung approach. This problem is also associated with the Popat and Picard algorithm. When estimating the pdf for a given pixel, its surrounding spatial neighbourhood is compared to all possible neighbourhoods in the sample image. However, at the time of synthesising, not all pixels in the neighbourhood may be known and in [68] the neighbourhood structure is adjusted to compare only known pixel values. The distance between

two individual neighbourhoods is then normalised by the total number of known pixel values when computing the conditional pdf for the pixel. This heuristic approach offers no guarantee that the pdf for the pixel will stay valid as the rest of the neighbourhood is filled in, however in most cases it appears to be a good approximation. Similarly, in [158] a causal neighbourhood is used and synthesis is performed in a raster scan order and so the pdf can change as additional pixels are synthesised. Regions where the synthesis process grows *garbage* illustrates when the constant pdf approximation is exorbitant. However, if a large enough neighbourhood size is used then the heuristic measurement is more stable and the approximation will remain plausible.

In [68] for each pixel to be synthesised, its spatial neighbourhood has to be compared to every possible neighbourhood in the sample image. The difference between two neighbourhoods is taken as the normalised squared sum of individual pixel differences. In addition, this neighbourhood is also weighted with a Gaussian kernel so that the error for nearby pixels in greater than for pixels further away. An exact breakdown of the computational cost associated with the Efros and Leung technique will be given later in chapter 4 but intuitively the computational cost associated with synthesising one pixel is high. Extend this to an entire image and the cost becomes even more significant.

Notwithstanding the scale dependency and computationally expense associated with the Efros and Leung algorithm, to date it offers the most powerful and robust means to capture texture behaviour. If the scale dependency and the computational burden could be overcome under the same powerful modelling framework, then the resulting algorithm would provide a complete solution to texture synthesis problem.

Fast Synthesis Using Image Pyramids [213], (2000)

Around the same time that Efros and Leung published their algorithm, Wei and Levoy [213] published a similar algorithm which as the authors point out[3] was not derived from [68] but rather extended from Popat and Picard [158] and therefore should be considered as a concurrent work of [68], with hard evidence given in an earlier paper by Wei and Levoy [212]. Similar to [68], the algorithm proposed in [213] is based on MRF texture models and generates textures through a heuristic searching process where the example texture image is used to implicitly model the new texture image to be synthesised. However, there are a number of subtle differences between [68] and [213]. Firstly, rather than placing a sample "seed" texture in the centre of the new image to be generated as in [68], in [213] the new texture image $\mathbf{I_s}$ is initialised with white Gaussian noise. To improve this initialisation, histogram equalisation is performed between the noise sample and the example texture image. Beginning in a raster scan order, each pixel is synthesised in turn by comparing its neighbourhood with all possible neighbourhoods in the example image $\mathbf{I_e}$. The similarity between two neighbourhoods is given as the sum of squared difference between individual pixels. The most similar neighbourhood is found and the centre

[3] http://graphics.stanford.edu/papers/texture-synthesis-sig00/similarity.html

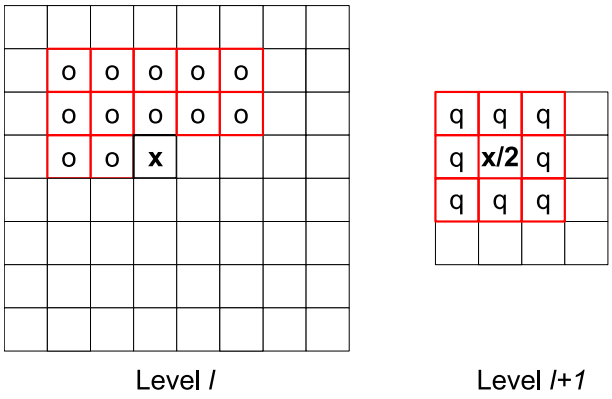
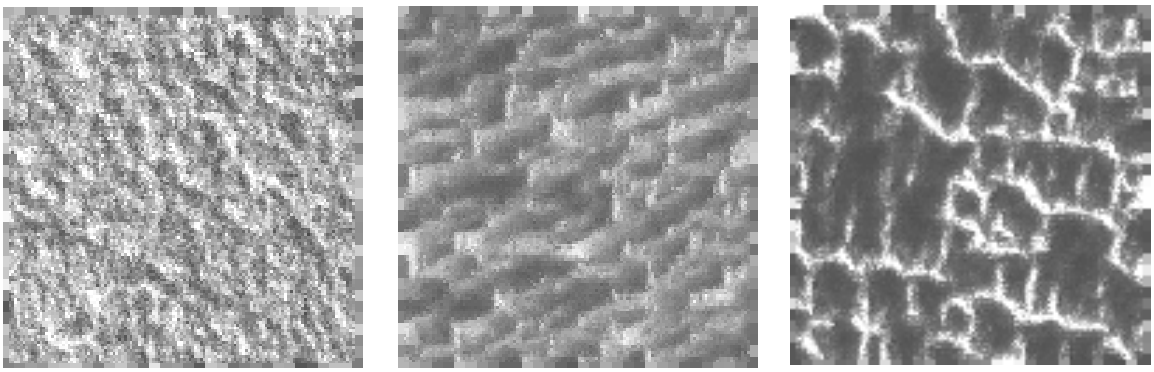


Figure 2.17: *A causal multi-resolution neighbourhood with size* $[5\times5, 2]$ *as used by Wei and Levoy [213]. The current level* l *of the pyramid is shown on the left and the next lower resolution level* $l+1$ *is shown on the right. The pixel to be synthesised at site* $\mathbf{x}$ *at level* l *must use only a causal neighbourhood as only the preceding pixels have already been synthesised and all pixels after* $\mathbf{x}$ *are unknown. The position of the parent pixel of* $\mathbf{x}$ *is* $\mathbf{x}/2$ *at level* $l+1$*. Since the level of the parent is completely synthesised, the parent neighbourhood can contain pixels around* $\mathbf{x}/2$*, marked* $\mathbf{q}$*. When searching for a match for the pixel at* $\mathbf{x}$*, the neighbourhood vector that includes all pixel locations marked* $\mathbf{o}$*,* $\mathbf{q}$ *and* $\mathbf{x}/2$ *is constructed.*

pixel of this neighbourhood is assigned to $I_s(\mathbf{x})$. This is slightly different to the approach taken in [68] where the set of best neighbourhoods is found and the centre pixels of these neighbourhoods form an estimation of the pdf for the pixel to be synthesised. This pdf is sampled randomly and the chosen value is assigned to the unknown pixel location. In addition, because of the raster scan order taken by Wei and Levoy, the neighbourhood structure used is causal. In addition, unlike [68] where synthesis is performed at the highest resolution image space, in [213] synthesis is performed across each level of a Gaussian pyramid structure [38].

The multi-resolution approach proposed in [213] proceeds as follows. Two L level Gaussian pyramids $\mathbf{G_e}$ and $\mathbf{G_s}$ are built from $\mathbf{I_e}$ and $\mathbf{I_s}$, where L is the highest level of the pyramid. The algorithm synthesises each level of $\mathbf{G_s}$ from coarse to fine resolution, such that each higher resolution level is constructed from the already synthesised lower resolution levels. This coarse to fine type synthesis allows lower frequency information to be added at the coarser resolutions and then higher frequency information to be added at the finer resolutions. Within each synthesised pyramid level $\mathbf{G_s}^l$, the pixels are synthesised in a manner similar to the single resolution case where the pixels are generated in raster scan ordering. The only modification is that for the multi-resolution case, each neighbourhood $N(G_s^l(\mathbf{x}))$ contains pixels in the current resolution l as well as those in the lower resolution $l+1$. The similarity between two multi-resolution neighbourhoods is measured by computing the sum of the squared difference of all pixels within them. These lower resolution pixels constrain the synthesis process so that the added high frequency details will be consistent with the already synthesised low frequency structures. An example multi-resolution neighbourhood is shown in Figure 2.17.



(i) (ii) (iii)

Figure 2.18: *Synthesised texture images obtained using the Wei and Levoy [213] algorithm.*

The Wei and Levoy algorithm removes some of the scale dependency associated with Efros and Leung. However as results in Figure 2.18 will show, it does not remove it entirely. Rather it uses texture scale as a control (coarse resolution guides finer resolution levels) rather the exploiting it directly. With [213] it is necessary to synthesise each level of the pyramid of the Gaussian pyramid individually. This is an even larger computational burden than that associated with Efros and Leung [68]. To reduce the computational expense, tree-structured vector quantisation (TSVQ) [83] is introduced as the searching algorithm [212]. This avoids the exhaustive nearest neighbour searching process by considering neighbourhoods as points in a multi-dimensional space and casting the neighbourhood matching process as a nearest-point searching problem. One disadvantage of the TSVQ acceleration is the memory requirement necessary to obtain accurate texture representation. High texture accuracy requires large memory allocations. Figure 2.18 shows some results obtained using the Wei and Levoy algorithm [213]. The original example textures images are as shown in Figure 2.6. Each new synthesised image measures 128×128 pixels. Synthesis was performed over three levels of a Gaussian pyramid and a causal multi-resolution neighbourhood similar to that shown in Figure 2.17. The synthesised images capture the texture properties of the original but are not very realistic. The synthesised results appear incoherent and the structure features of textures Figures 2.6 (ii) and (iii) are not replicated in the synthesised images in Figure 2.18.

Order Based Synthesis [98], (2001)

Harrison [98] extends the algorithm proposed in [68] to include order based synthesis. The specific order in which pixels are generated is based on the local interactions of closely neighbouring pixels. Since non-parametric modelling defines no explicit model, often large features present in the example texture will not be replicated in the synthesised result unless the spatial neighbourhood used to search the example image is large enough to capture them. This results in a scale dependent algorithm whereby the neighbourhood size needs to be varied with input

texture. In [98] this problem of scale dependency is addressed by enforcing order based synthesis. By synthesising pixels in a specific order, large features which may be larger than the chosen neighbourhood size can still be reproduced in the output image.

Similar to Efros and Leung [68], pixels are synthesised one at a time using a heuristic measurement taken from the input image. In [98] the stability of [68] is improved by introducing the following features. Firstly, the Manhattan distance is used when computing the similarity between two neighbouring blocks as it is more forgiving to outliers. Secondly, the order in which pixels are synthesised in the new image is changed. In the Efros and Leung algorithm, pixels are synthesised based on the number of known neighbours in their spatial neighbourhood. Pixels with a higher number of known neighbours are synthesised before those with a lower number. In [98] the synthesising order is determined by a priority value assigned to each location in the output image. Higher priority sites are synthesised before those of lower priority. To assign a priority value to each site, a normalised weighting is defined which indicates the relative amount of information a pixel gives about each of its neighbours. The priority of each empty location in the output image is then defined as the sum of the weightings from neighbouring pixels. To create the weightings, interactions between neighbouring pixels in the input image are analysed. This order based synthesis addresses to some extent the scale dependency of [68], however it does not remove it entirely. The computational burden associated with the exhaustive nearest neighbour searching inherent in [68] is also removed in [98]. Rather, a *kd-tree* [74] structure is introduced in order to reduce the search time. As with most nearest neighbour searching approximations, the reduction in search time comes at the expense of increased memory storage.

Synthesis using Coherent Searching [14], (2001)

Ashikhmin [14] reduced the computational burden associated with the non-parametric modelling by replacing the exhaustive nearest neighbour searching with the more computationally attractive coherent searching process. This algorithm was designed to synthesise natural textures (e.g. grass, leaves, clouds, etc.) only. Coherent searching is based on the observation that during the synthesis process, the neighbourhood of the pixel to be synthesised is likely to be similar to shifted versions of the neighbourhoods of the other previously synthesised pixels in its neighbourhood. This information is not used in any way by [68] or [213], where the searching process starts from scratch at each new pixel, ensuring that the best available choice is made. However, Ashikhmin uses this observation and assumes that pixels from the input sample that are appropriately "forward shifted" with respect to pixels already used in the synthesis are well suited to fill in the current pixel. This idea is shown in Figure 2.19.

The algorithm described in [14] is based on the single resolution version of the Wei and Levoy algorithm and so a causal neighbourhood (similar to that shown on the left of Figure 2.17) is used and synthesis is performed in raster scan order. The image to be synthesised is initialised with a noise sample which has been histogram equalised with the sample texture. The coherent

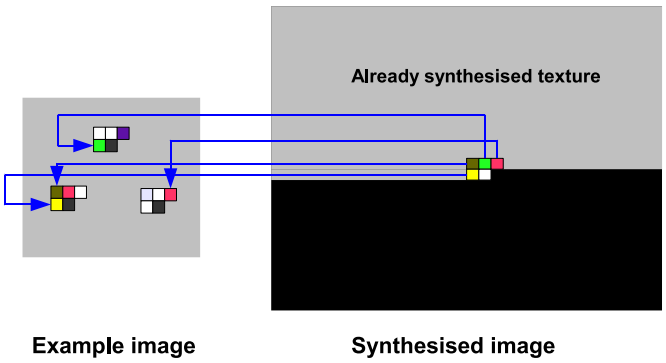
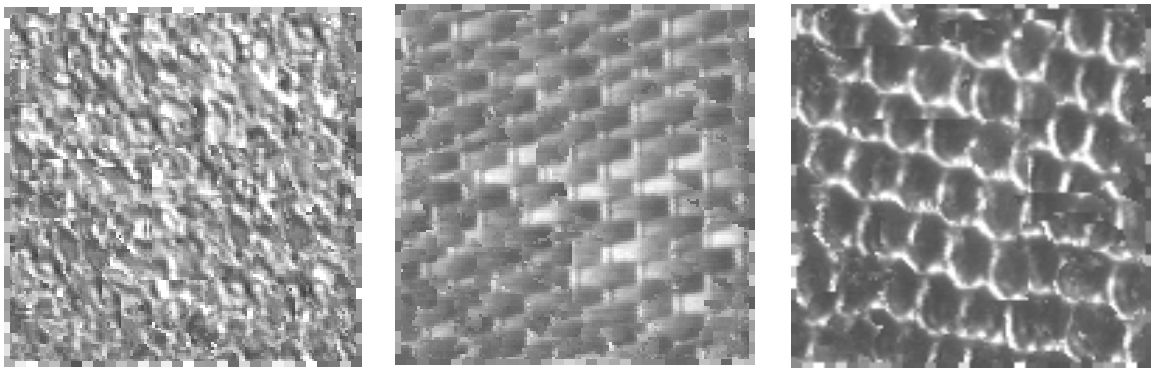


Figure 2.19: *Coherent searching used in [14]. For each pixel to be synthesised, its causal neighbourhood is constructed (right). Each pixel in this neighbourhood generates a "shifted" candidate pixel (black) set according to its original position in the input texture. The pixel to be synthesised is chosen from this set only.*

searching texture synthesis algorithm given in [14] can be summarised as follows.

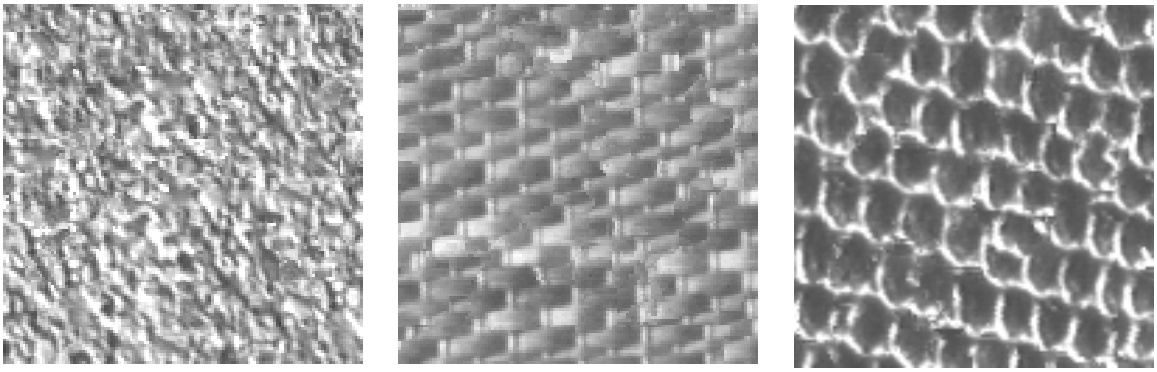
1. The array of original pixel positions is initialised to random valid positions in the input image.
2. To synthesise pixel $I_s(\mathbf{x})$, first construct its spatial neighbourhood.
3. For each pixel in this neighbourhood, use its original source in the example texture image (taken from an array) to generate a candidate pixel which is appropriately shifted.
4. Remove duplicate candidates and search the candidate list for a pixel with a neighbourhood most similar to the current $L-$ shaped neighbourhood in the output image. The current pixel value in the output image is copied from the position in the example image identified as the most similar by this search and this position is recorded in the array of the original positions.
5. In raster scan ordering repeat steps $2-4$ for all pixels to be synthesised. If necessary, modify the algorithm for the last few rows and re-synthesise the top rows to improve visual quality of the new texture.

The Ashikhmin algorithm works well and generates a visually pleasing coherent result which works well for natural textures. In addition, the computational burden associated with [68] and [213] has been much reduced. Figure 2.20 shows some synthesised textures obtained using the Ashikhmin algorithm. As before the example images were taken from Figure 2.6 and the new synthesised textures measure 128×128 pixels. A causal neighbourhood of size 15×15 pixels was used in the synthesis process. The synthesised images capture the original texture features but horizontal edges are visible in the synthesised result when patches reach the end of



(i) (ii) (iii)

Figure 2.20: *Synthesised textures using the Ashikhmin [14] algorithm.*



(i) (ii) (iii)

Figure 2.21: *Synthesised textures using the Hertzmann et al. [101] algorithm.*

the source image.

Combining Coherent and Exhaustive Searching [101], (2001)

To avoid the inclusion of artificial horizontal edges in the synthesised result, Hertzmann et al. [101] combines the exhaustive nearest searching proposed by Wei and Levoy with the coherent searching proposed by Ashikhmin [14]; the result being that for each pixel to be synthesised, two searching processes: exhaustive and coherent searching of the example texture image are evoked. In general the exhaustive searching will return a better match than that of the coherent searching, but intuitively the coherent value may give a more realistic result since the synthesised texture should be coherent by nature. To maintain this coherency, in [101] a weighting value is added to favour the coherent value whenever possible. The more weighting that is added to the coherent searching process, the more coherency is favoured over accuracy in the synthesised result. Similar to [213], the algorithm proposed in [101] performs the synthesis over each level of a Gaussian pyramid structure.

Figure 2.21 shows synthesised textures obtained using the Hertzmann et al. algorithm. The synthesised images offer a more accurate representation of the original texture images in Figure 2.6. A considerable improvement is evident over synthesised results obtained using the Wei and Levoy [213] and Ashikhmin [14] algorithms. However, computationally, the algorithm developed by Hertzmann et al. is excessive given that both coherent and exhaustive nearest neighbour searches are required for each pixel to be synthesised. To reduce some of the computational cost an Approximate Nearest Neighbour (ANN) [13] searching process is introduced, but nevertheless performing two types of searching processes over all multi-resolution levels of a Gaussian pyramid is cumbersome.

Because of the wide variability in texture behaviour, it is found that non-parametric approaches are better suited to modelling texture than their parametric counterparts. This is evident from some of the synthesised results presented as part of this review. However, on the downside, this improvement in perceptual similarity is at the cost of computational expense. In addition, many non-parametric algorithms suffer from scale dependence. The patch-based approaches, discussed in the next section, particularly aim to address the issue of computational burden.

2.5 Patch-Based Approaches

Parametric and non-parametric approaches can be considered as pixel based, where only one pixel is synthesised at a time. On the contrary, as its name suggests, patch-based texture synthesis algorithms synthesise entire patches or blocks of pixels at a time. There are two issues to be resolved in patch-based approaches: (i) which patches to extract and in what order and (ii) how to merge or "sew" patches together in the new image so that there is no visible boundary line between two adjoining patches. All of the patch-based approaches are based on the loose Markovian assumption that was discussed earlier when discussing non-parametric texture synthesis approaches (section 2.4). Given the computational benefits of synthesising entire patches rather than individual pixels, there have been a number of patch-based approaches developed. Some of the more popular approaches will be described here.

Chaos Mosaicking [218], (2000)

Xu et al. [218] proposed the use of chaos mosaicking as as a method of producing fast efficient texture synthesis. The algorithm can be summarised in the following manner. The area to be synthesised is initially tiled with a tile containing the largest texture feature in the example image. Inside each tile, a smaller randomly sized block of texture is chosen and placed over any of the seams introduced as a result of the tiling process. This process of choosing sub-blocks and placing them over seams is repeated for all tiles in the image. The size of the sub-block and where it will be placed in the image is determined by a Cat Map iteration. The Cat Map

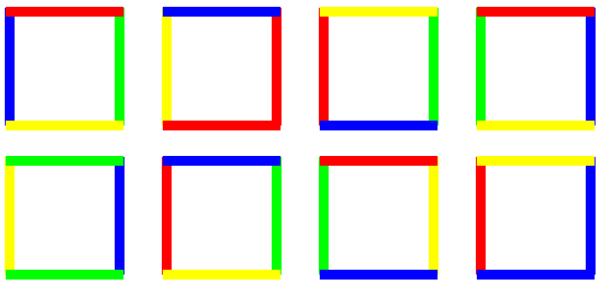


Figure 2.22: *A set of eight Wang tiles that can be used for the non-periodic surface required for texture synthesis.*

iteration is an iterative process taken from the field of deterministic chaos. Deterministic chaos denotes the state of disorder and irregularities generated by a nonlinear dynamic system in which the previous actions uniquely determine future behaviour of the system. It is not the intention of this work to discuss in detail the Cat Map iteration used in [218], however a good description of this process can be found in [12]. Once the sub-block tiling has been repeated a number of times over the entire image, the algorithm finishes by evoking a simple cross-edge filter across sub-block boundaries. This filtering process should remove any unwanted boundary edges and mis-matched features. Overall, chaos mosaicking is efficient and simple to implement but in reality it only works for stochastic textures.

Synthesis using Wang Tiles [52, 197], (1997, 2003)

Figure 2.5 shows eight example Wang tiles than can be used for non-periodic surface mapping. A Wang Tile set consists of square tiles with colour-coded edges. The squares cannot be rotated and a valid tiling of the infinite plane consists of any number of copies from the set laid such that all contiguous edges have matching colours. Stam [197] was the first to consider the use of square Wang tiles for texture synthesis. Based on a deterministic algorithm that uses a set of 16 individual square tiles with coloured edges, large non-repetitive textures were created by defining texture samples on the tiles following the border constraints introduced by the shared edges.

Cohen et al. [52] extends the idea presented in by Stam [197] to include a coding of the corners of the tiles and an automatic tile design technique. The corner-coding allows discrete objects to overlap on more than one edge. These give a more realistic result and avoids the introduction of visually repetitive patterns. These patterns can occur if for example an object or texture feature is placed at the corner of the tile. Without colour-coded edges, the vertical edge colour constraint associated with the original Wang Tiles enforces all the tiles with the same colour opposite the vertical edges to include the remainder of this object to ensure the fitting condition. As the same object is also on the horizontal edge, all tiles that contain the

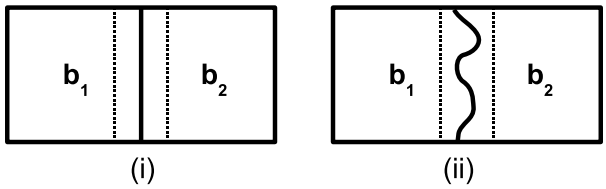


Figure 2.23: *Quilting texture as in [67]. Square blocks of predefined size are patched together to synthesise a new texture image. To hide the boundary line between adjacent blocks, an overlap region (1/6 size of neighbourhood) between two blocks is introduced (i) and the minimum cost path through this area is found and chosen as the new boundary (ii).*

corresponding horizontal edges have to be adapted similarly. The result is a repetitive pattern on the horizontal line. This problem is avoided with the introduction of colour-coded corners.

The automatic tile design technique fills each individual tile with texture from the example texture image. These tiles will then be used to aperiodically map the new image to be synthesised. Each tile is created by combining diamond shaped (squares rotated by 45^o) sample portions of the source image, one for each edge colour of horizontal and vertical edges. The manner in which each diamond block is chosen and merged with others is derived from the image quilting texture synthesis algorithm proposed by Efros and Freeman [67]. Using Wang tiles for texture synthesis works well on natural stochastic textures that contain little structure. However, for more structured textures, a larger set of tiles would have to be used thus reducing the computational simplicity that was the initial motivation.

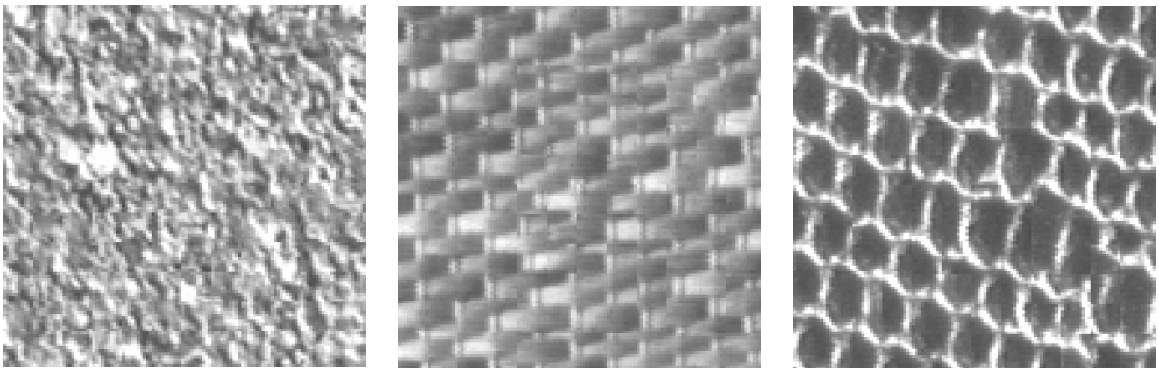
Image Quilting [67], (2001)

In [67], Efros and Freeman propose image quilting to solve the texture synthesis problem. Unlike the Efros and Leung algorithm, where the unit of synthesis is single pixel, Efros and Freeman synthesise square blocks of user-specified size. The size of these blocks is chosen on initialisation and remains constant throughout the algorithm. To avoid the appearance of boundary lines between blocks, an overlap in the placement of blocks onto the new image is incorporated. Before placing a chosen block into the new texture image, the error in the overlap region is examined and a minimum cost path through that error surface is found. This minimum cost path is then defined to be the boundary of the new block. Figure 2.23 illustrates the manner in which blocks are placed side by side.

The image quilting algorithm given in [67] can be summarised as follows.

1. Go through the image to be synthesised in raster scan order in steps of one block (minus the overlap).

2. For every location, search the input texture for the best matching block. The best matching block will be the one whose overlap area is most similar to the overlap area of the previously



(i) (ii) (iii)

Figure 2.24: *Synthesised texture image obtained using the Efros and Freeman [67] image quilting algorithm.*

synthesised block. The error in overlap regions is taken as the L_2 norm on pixel values.

3. Once the best matching block has been found, find all blocks that fall within some tolerance of this best match. Note this is similar to the heuristic measurements inherent with the approach taken in [68].

4. This set of most similar blocks form an approximation to the probability density function of the block to be synthesised. By selecting a block at random (uniform) from this list a sample is generated for this implicit distribution over the blocks.

5. Using the chosen block and the previously synthesised block, compute the error surface in the overlap region between them. Find the minimum cost path along this surface and make that the boundary of the new block.

6. Paste the block onto the texture. This procedure is repeated until the new image is fully tiled or "quilted".

Figure 2.24 shows some synthesised textures obtained using the image quilting algorithm given in [67]. The 128×128 pixel Brodatz images given in Figure 2.6 are used as the example images and each synthesised image measures 128×128 pixels. A block of size 12×12 was used as the unit of synthesis and the overlap region between blocks was 2×2 pixels. The algorithm generates impressive, stable results that certainly resemble the original texture examples. However, some horizontal and vertical seams are present where patches overlap. Computationally the algorithm is attractive as entire blocks are synthesised at a time.

One of the problems of patch based approaches thus far is that the size of the patch has to be defined *a priori*. There are two issues with this predefined patch size. Firstly, the patch size needs to be large enough to capture the largest feature present in the example texture. Too small a patch and these features will not be replicated in the output image while too large a

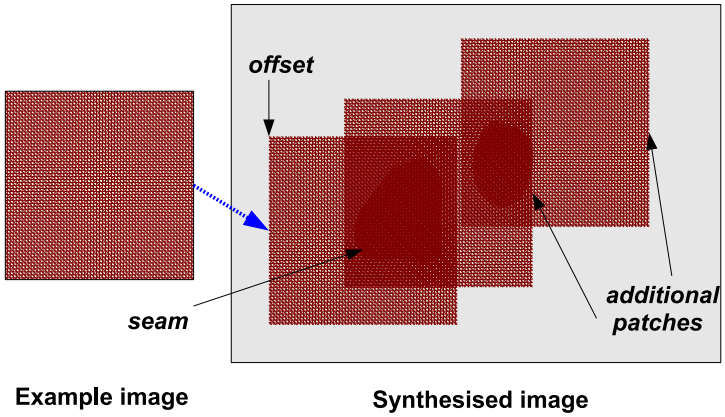


Figure 2.25: *Graph-cut texture synthesis [131]. Patches from the example texture are placed in the area to be synthesised as different offsets (sites). These patches will overlap with those placed previously. The graphcut technique is used to find the minimum cost boundary between the overlapping patches. Texture inside this boundary is then pasted onto the synthesised image.*

patch will result in an output texture which is too structured and loses some of the stochastic nature of the texture. Since the size of texture features will vary from one texture to another, the prerequisite of choosing a suitable patch size is dependent on the particular input texture. Thus the algorithm is scale dependent. The second issue with constant patch size is that hiding the boundaries between patches is difficult and often horizontal and vertical edges are easily detected between patches because of patch homogeneity.

Graphcut Textures [131], (2003)

Kwatra et al. [131] proposed an algorithm that avoids the need to define the patch size *a priori* and instead the size of the patch transfered from the example texture to the synthesised image is defined using the *graph-cut* technique [32]. In order to incorporate the graph-cut technique into the synthesis problem, the texture is modelled as a Markov Random Field. Under this model, the output texture can be visualised as a grid of nodes, where each node refers to a pixel or a neighbourhood of pixels in the example texture. The marginal probability of a pair of nodes depends on the similarity of their pixel neighbourhoods so that pixels from similar neighbours in the example texture end up as neighbours in the generated texture. Under this model, the problem of texture synthesis can be reformulated as the solution for the nodes of the network that maximises the total likelihood. Maximising the total likelihood over the graph is equivalent to minimising the energy cost function over the graph. The problem of energy minimisation over a graph is common in machine learning. Chapter 5 outlines some popular techniques that are used for finding the optimal minimum energy field. One such energy minimising process is the graph-cut technique. Essentially, the idea behind the graph-cut technique is to construct a

specialised graph for the energy function to be minimised such that the minimum cut on the graph also minimises the energy (either locally or globally). This will be discussed in greater detail later in chapter 5.

Figure 2.25 illustrates how the graph-cut is used to determine the boundary between patches. The patch copying process is performed in two stages. Firstly, a candidate rectangular patch (or patch offset) is selected by performing a comparison between the candidate patch and the pixels already synthesised in the output image. Second, an optimal (irregularly shaped) portion of this rectangular patch is computed and only these pixels are copied to the output image. The portion of the patch to be copied to the output image is determined using a graph-cut algorithm. For stage one, the manner in which candidate patches are chosen is based on the type of texture that is being synthesised. For random textures, *random placement* of patches is chosen. That is, the entire example image is translated to a random site in the output image. This is shown in Figure 2.25. For structured or semi-structured textures, *entire patch matching* is preferred. Entire patch matching involves searching for translations of the input image that match well with the currently synthesised output. The set of best matching patches form a probability density function (pdf) which is sampled randomly and the chosen patch is assigned to the output image. The third type of patch selection techniques is *sub-patch matching*. This is used for stochastic and video (3D) texture. In sub-patch matching, a small sub-patch (significantly smaller than the input texture) is chosen from the output texture. The algorithm then searches the example texture for a set of patches that matches the output sub-patch. Similar to entire patch selection, this set of patches form a pdf which is sampled randomly and the new patch assigned to the output texture. In entire patch matching and sub-patch matching, the placement of patches in the synthesised image is governed by a cost function whereby the cost of existing seams is used to quantify the error in the particular region of the image. The region with the largest error will be synthesised first and the patch selection algorithm is forced to choose only those patch locations that completely overlap the error-region.

The graph-cut approach to texture synthesis works well and avoids the problem of choosing a suitable patch size based on the input texture. This approach has been applied to both 2D and 3D textures. It works well on stochastic textures but for deterministic textures some visual seams can be seen along patch boundaries. Results for the graph-cut method can be found in [131].

Comparing patch-based approaches to the parametric and non-parametric (typically pixel) based approaches discussed previously, it can be said that patch-based approaches are good at preserving the global structure of the texture and work well on stochastic type textures. They are computationally efficient and therefore attractive for synthesising large areas. On the downside, often the boundary edges between individual patches is visible in the synthesised result.

This concludes the discussion on patch-based texture synthesis algorithms. Note that the parametric, non-parametric and patch-based algorithms were described in terms of synthesising gray-scale texture. In general most of the algorithms synthesise colour texture by carrying out

a similar modelling process on the three RGB (Red-Green-Blue) channels (3D) rather than the single gray-scale channel (1D). Alternatively, some algorithms translate the colour values into a different image space (e.g. YUV) and perform synthesis on the luminance component only. The colour components follow the movement of the luminance component. Some of the commonly used colour spaces are discussed further in chapter 5.

2.6 A New Texture Synthesis Algorithm

The accuracy of any new synthesis process will be highly dependent on the accuracy of the underlying texture model. Looking ahead, this texture model will also form the basis for the work on example based segmentation. The idea behind the segmentation approach will be to characterise objects based on their texture components. Thus the image to be segmented can be interpreted as a mixture of different textures, each of which represents a particular object. In order to isolate and recognise objects, it will be necessary to model and define each texture component. The image segmentation problem will be discussed in chapters 5 and 6 but at this stage it is worth noting that the choice of model used in the synthesis algorithm will be influenced by the applicibility of that modelling process to the segmentation problem.

In general, patch-based approaches work well in the texture synthesis domain because the example texture is composed of only one type of texture. However, if there is more than one type of texture present (as in the segmentation case) then patch-based modelling and analysis becomes complicated given that each patch may contain more than one texture. Therefore, for a more robust modelling process, it is more sensible to consider each pixel individually.

Because of the wide variability in image texture behaviour, non-parametric approaches have by far out-performed parametric approaches. This is evident from the results given in sections 2.3 and 2.4. However, associated with non-parametric processes are the inherent limitations of scale dependency and large computational expense. The scale dependency is introduced as a result of the Markovian assumption on which the non-parametric modelling process is built. Results in this chapter demonstrated that when the texture scale was misinterpreted and incorrectly specified, the algorithm failed to capture the basic features of the texture and so the modelling process failed. Conversely, if the correct texture scale is specified, the synthesised result and hence the implicit modelling process is accurate in capturing the underlying texture behaviour.

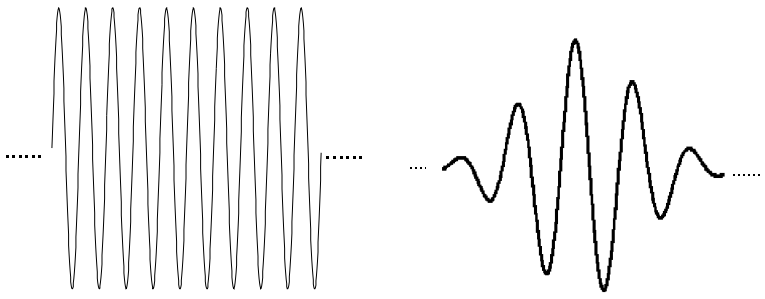
Motivated by the strength of the non-parametric modelling technique and the need to overcome the computational costs and scale dependency typically associated with it, a new algorithm has been developed as part of this work on example based image processing. This algorithm is non-parametric and exploits the advantages of wavelet analysis by performing synthesis in the wavelet domain. This algorithm will be discussed in chapter 4 but before that some aspects of the wavelet transform are introduced.

3

Aspects of Wavelets

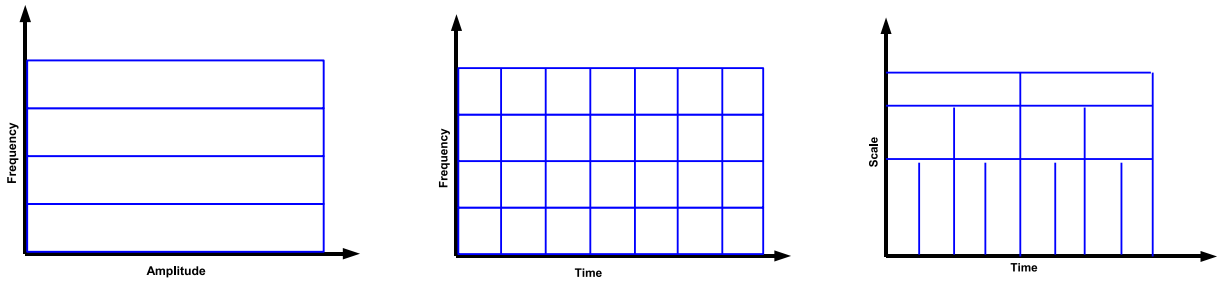
The suitability of the wavelet transform (WT) for use in image analysis has been well established. This is due to its ability to enable the localisation of functions in space, scaling and orientation [140]. In signal analysis, the WT can be interpreted as a signal decomposition onto a set of basis functions called *wavelets*. These *wavelets* can be described as small finite wavelike functions or ripples that have their energy concentrated in time. This finite oscillating wavelike characteristic associated with wavelets makes them suitable for the analysis of transient, non-stationary or time varying phenomena associated with most signals [37, 139]. Figure 3 shows an example wave function (sinusoid) and a wavelet function (Morlet Wavelet). The wave function is smooth, periodic and infinite, whereas in contrast the wavelet function is finite and non-periodic.

Wavelet analysis was developed to perform signal analysis in the time-frequency domain.



(i) Wave (Sinusoid) (ii) Wavelet (Morlet)

Figure 3.1: *Representation of a wave (i) and a wavelet (ii) function.*



(i) Fourier Transform (Frequency)

(ii) Short Time Fourier Transform (Fixed Time-Frequency)

(iii) Wavelet Transform (Variable Time-Frequency)

Figure 3.2: *The domains within which the (i) Fourier , (ii) Short-Time Fourier and (iii) Wavelet transforms operate.*

Prior to the development of wavelet theory, signals were commonly analysed within the frequency domain using the Fourier transform (FT). Fourier analysis uses waves (sinusoids) as deterministic basis functions for the expansion of functions (signals) that are assumed to be time invariant or stationary. In reality most naturally occurring signals, of which images are an example, cannot be considered stationary. Therefore, assuming stationarity and performing analysis in only the frequency domain has serious limitations. So even though the FT gives an accurate representation of the frequency content of a signal, it cannot offer both the time and frequency localisation necessary for most naturally occurring (non-stationary) signals.

To address the limitations of the standard FT, the Short Time Fourier Transform (STFT) was introduced by Gabor [75]. The STFT represents a sort of compromise between the time and frequency based views of a signal. Essentially, the STFT moves a fixed window over the length of the signal. Inside this window, the signal is assumed to be stationary and so the Fourier analysis of the windowed signal is performed. This combined windowing and frequency analysis of the signal provides some information about both when and at what frequencies a signal event occurs. On the downside, this time-frequency information can only be given with limited precision which is governed by the size of the window which remains constant for all frequencies. However, many signals require a more flexible approach in which the window size can be varied to introduce more accuracy in either time or frequency. Logically, it is sensible to have a small window for high frequencies and a large window for low frequencies. The wavelet transform provides such a tool as it was designed to offer flexible time-frequency analysis. Figure 3.2 illustrates the three domains within which the Fourier, Short-Time Fourier and Wavelet transforms operate.

Wavelet signal analysis is initialised by adopting a specific wavelet prototype function. Temporal analysis is performed using a contracted, high frequency version of the prototype wavelet, while frequency analysis is performed with a dilated, low frequency version of the same wavelet [86]. Since the frequency responses of the bandpass bands are scaled down by a factor of two at each level, their impulse responses become longer by the same factor but their shapes

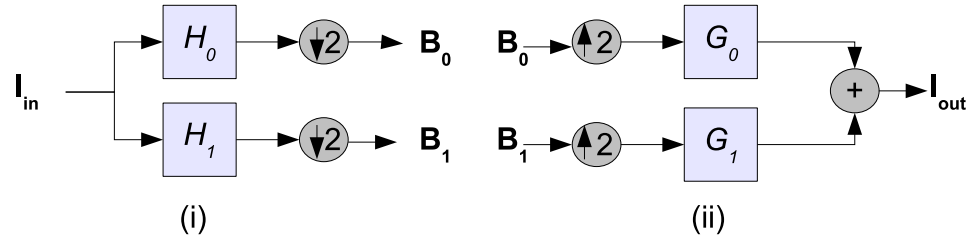


Figure 3.3: *Two-band filter banks for analysis (i) and reconstruction (ii).*

remain very similar. The basic impulse response wave shape is almost independent of scale and is known as the *mother wavelet*.

The WT has found many applications in image processing due to its ability to localise functions in space, scaling and orientation [56, 59, 140, 169]. Some example applications of the WT are in image compression [8, 9, 106, 204], image de-noising [61, 90, 175], digital watermarking of images [137], object segmentation [50, 186, 198] and texture analysis and synthesis [33, 59, 77, 78, 160].

There are several types of wavelet transform (e.g. continuous, discrete, complex) and depending on the particular application, one type of transform may be preferred over the others. In the case of image analysis, the discrete wavelet transform (DWT) initially proved the most popular. Recently, and as a result of limitations imposed by the DWT, the Complex Wavelet Transform (CWT) and in particular the Dual-Tree Complex Wavelet Transform (DT-CWT) has become the favoured transform in image analysis. This section will describe both of these transforms. The DT-CWT will form the domain in which the example based image processing algorithms developed as part of this work will be conducted. Since the wavelet transform is used as a tool for signal representation, the focus of this discussion will not digress into an in-depth discussion of wavelet theory. Therefore, topics such as the mathematical foundations of the transform, and wavelet filter design will not be discussed here. However, detailed descriptions of these topics can be found in [118, 121, 128, 139, 169, 211].

3.1 The Discrete Wavelet Transform

The basic building blocks of the one dimensional (1D) Discrete Wavelet Transform (DWT) are shown in Figure 3.3. During the analysis stage of the DWT, the input signal $\mathbf{I_{in}}$ is decomposed into its high frequency and low frequency components using the analysis low pass H_0 and high pass H_1 filters. To prevent redundancy in the combined output of the transform, the outputs from each filter are down-sampled by a factor of two. The outputted low pass signal $\mathbf{B_0}$ is known as the *scaling coefficients* while the high pass signal $\mathbf{B_1}$ which contains the fine details is known as the *wavelet coefficients*. A full multi-scale wavelet transform is constructed by repeating this basic splitting and down-sampling operation a number of times, where at each level the basic splitting operation is performed on the scaling coefficients generated at the previous level. This

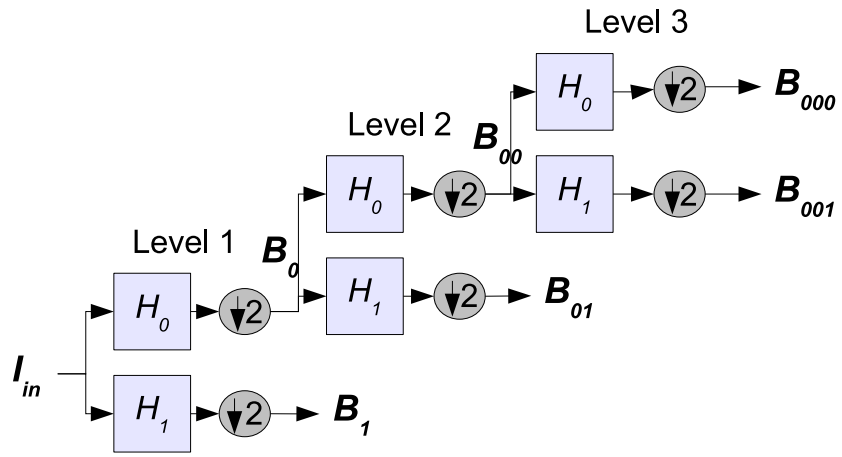


Figure 3.4: *Three-level DWT for a 1D signal.*

multi-scale decomposition is shown in Figure 3.4. The ratio of the number of outputs to the number of inputs is known as the redundancy of the transform. The redundancy of the DWT is $1:1$.

Figure 3.3 (ii) illustrates the steps involves in the inverse wavelet transform, where $[G_0, G_1]$ denote the reconstruction low pass and high pass filters respectively. The reconstruction operation is the exact inverse of the analysis operation whereby the signals $\mathbf{B_0}$ and $\mathbf{B_1}$ are up-sampled by a factor of two and filtered by the low pass G_0 and high pass G_1 reconstruction filters respectively. The two filtered signals are then summed together to produce the output signal. If the output signal $\mathbf{I_{out}}$ is identical to the original input signal $\mathbf{I_{in}}$ then the transform satisfies the perfect reconstruction condition (PR). In practice all transforms are constructed to adhere to this condition.

For two dimensional signals (2D), e.g. images, the binary tree given in Figure 3.4 can be extended into the quad-tree structure shown in Figure 3.5. In a separable implementation, each level of the quad-tree comprises of two stages of filtering. The first stage filters and sub-samples the rows of the image, generating a pair of horizontal low pass and high pass sub-images. The second stage of the transform filters and sub-samples the columns of the filtered row signal to produce four sub-images, denoted $\mathbf{B_0}, \ldots, \mathbf{B_3}$. This separable filtering implementation is the most efficient way to perform the 2D DWT [119]. Similar to the 1D case, for the multi-level transformation $\mathbf{B_0}$ the low pass image obtained from the previous level becomes the new input at the next level of the transform. Figure 3.6 shows a two-level DWT decomposition of an image.

The sub-band images $\mathbf{B_1}, \mathbf{B_2}$ and $\mathbf{B_3}$ represent the detailed or higher pass wavelet coefficients representing the horizontal, diagonal and vertical components of the input signal. The transform can be further extended to high dimensions by applying filters to each dimension in turn. For m dimensional signals 2^m sub-band images are produced at each level.

The DWT decomposition given in Figure 3.6 and all subsequent DWTs given here will use

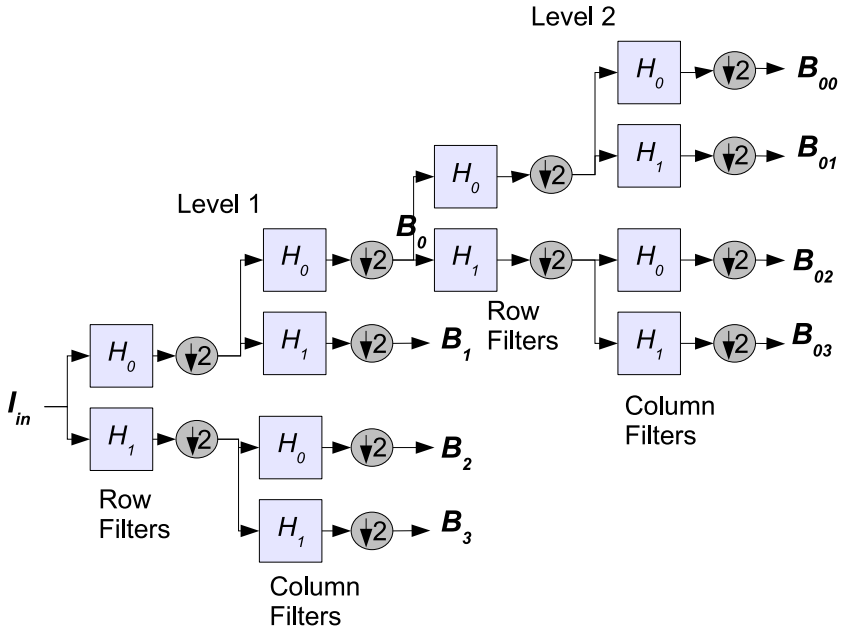
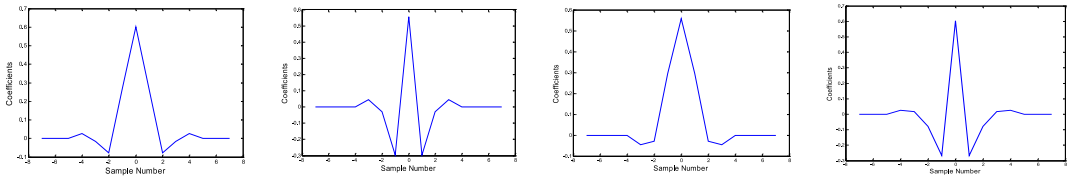


Figure 3.5: *Two-level DWT for a 2D signal.*

(i) Input Image (ii) Two-level DWT decomposition.

Figure 3.6: *Two-level DWT decomposition (ii) of the input image (i) using Antonini* $(7, 9)$*-tap filters. In image (ii), the upper right and bottom two quadrants are the level one sub-band images while their associated "child" sub-band images are nested inside them. The low pass image of scaling functions is located at the top left corner.*



H_0 H_1 G_0 G_1

Figure 3.7: *The Antonini* $(7,9)$*-tap filters [119, 120].* $[H_0, H_1]$ *and* $[G_0, G_1]$ *are the analysis and reconstruction low pass and high pass filters respectively.*

the Antonini $(7,9)$-tap filters [119,120]. The Antonini filters are bi-orthogonal meaning that the frequency responses of the analysis filters are not the same as that of the reconstruction filters. These filters are the most favoured in image compression and are used in the FBI fingerprint compression system [119]. Figure 3.7 shows the analysis and reconstruction filters associated with the Antonini $(7,9)$-tap filters.

Although the DWT is widely used in image compression [106], its application to other image processing problems has been hampered by two principle disadvantages. These are:

- *Lack of shift invariance:*
  A process is shift invariant if its output is independent of the absolute location of the data within the input to the process. For example if the process gives an output $\mathbf{I_{out}}$ when given an input $\mathbf{I_{in}}$, then the process is shift invariant if it generates a translated version of $\mathbf{I_{out}}$ when given a translated version of $\mathbf{I_{in}}$. With reference to the wavelet transform, the transform is shift invariant if the total energy in the sub-band image is unaffected by translations applied to the input. The shift dependency occurs as a result of the aliasing that is introduced by the down-sampling that follows each filtering operation. The un-decimated (remove the down sampling after the filtering) form of the DWT solves this problem but this is at the expense of large redundancy and increased computational expense.

  Figure 3.8 illustrates the shift dependence of the DWT. The input signal is a 1D step response shifted 16 times. Four levels of the DWT are taken and the sub-band signals associated with each is shown below the input signal. The level four scaling functions are shown on the bottom of the image. The shift dependence of the transform is evident from the varying energy in each of the sub-band signals.

  In image processing applications, the shift dependence of the DWT has limited its suitability for texture analysis. This is because in any given image, texture may present itself under any shift. If texture is to be characterised by its sub-band decomposition, then this decomposition needs to remain constant irrespective of the localisation of the texture within the image. As a result, any transform used to analyse texture should be as close to shift invariant as possible.

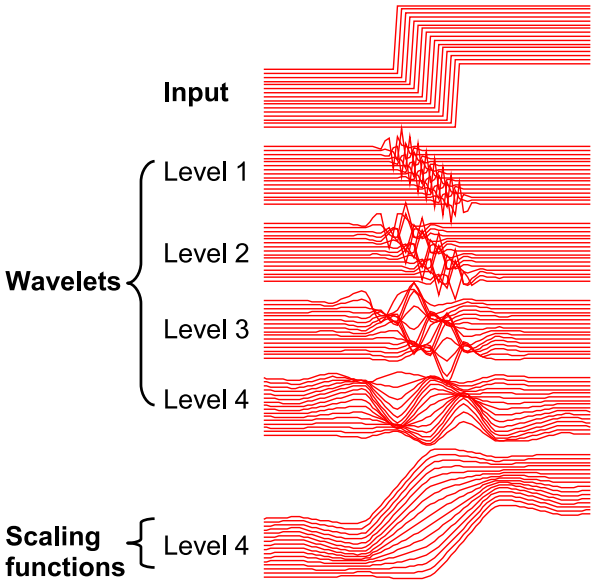
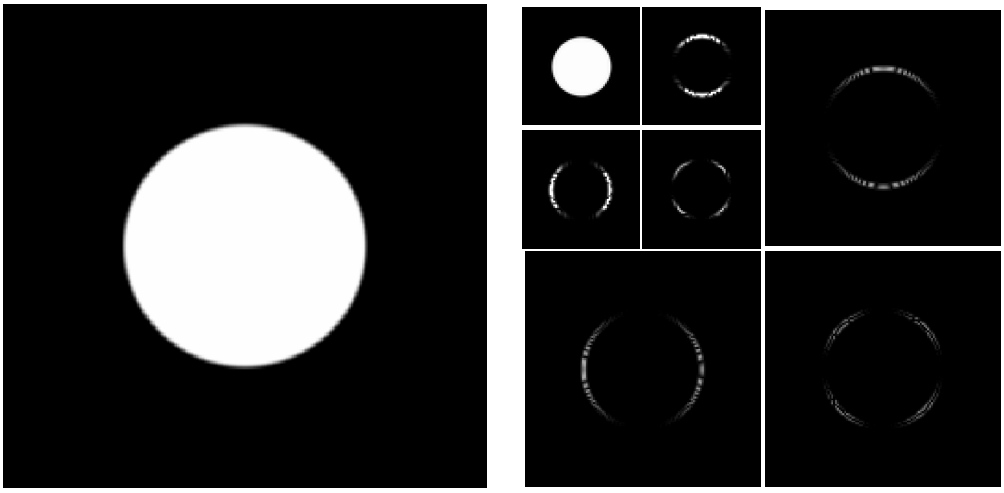


Figure 3.8: *Shift dependence of the DWT. Input signal is* $1D$ *step function at sixteen different shifts. The sub-band and low pass signals associated with the four-level DWT decomposition are also shown. The shift dependence is evident from the fact that the energy in each sub-band at any given level varies with the shift in the input signal.*

- *Poor directional selectivity:*
  Separable filtering of the image rows and columns produces four sub-images at each level. These sub-band images are obtained using real filters which cannot distinguish between positive and negative frequency components. Therefore, each sub-band contains both positive and negative frequency components resulting in poor directional selectively of the DWT. This inability to distinguish between positive and negative edge orientations increases the DWT unsuitability for texture analysis, given that textures are generally characterised by their frequency components.

  Figure 3.9 shows the poor directional selectivity of the DWT. The two level DWT of the 'circle' image (i) is shown in (ii). Level one sub-band images are placed on the boundary while the corresponding level two sub-band images are nested inside them. The final level scaling functions are shown at the top left hand corner of (ii). The intensity value in each sub-band images corresponds to the magnitude of the wavelet coefficient at that particular site. Each sub-band highlights either the horizontal, vertical or diagonal edge components of the input 'circle' image. The poor directional selectivity of the DWT is especially evident in the sub-band that contains the diagonal components of the circle (bottom right quadrant, bottom right of top left quadrant). These sub-bands contain both the diagonal edges of the circle making it impossible to distinguish between them.



(i) Input 'circle' image (ii) Two-level DWT decomposition.

Figure 3.9: *The poor direction selectivity of the DWT. The input "circle" image (i) and a two-level DWT decomposition (ii). Both positive and negative frequency components are represented in the diagonal sub-band (bottom right corner of (ii)) making it is impossible to distinguish between them resulting in poor directional selectivity.*

To address the lack of shift invariance and poor directional selectivity associated with the DWT, Kingsbury [118–120] developed the Dual Tree-Complex Wavelet Transform (DT-CWT). This will be discussed next.

3.2 The Dual-Tree Complex Wavelet Transform

The DT-CWT replaces the single tree structure of the DWT shown in Figure 3.4 with a dual tree of real-valued filters shown in Figure 3.10. These two parallel trees filter and down-sample the input signal in the same way as DWT but because there are two rather than one, the aliasing that resulted in shift dependency of the DWT is removed. At each level one tree produces the so called "real" part of the complex wavelet coefficients, while the other tree produces the "imaginary" part. The filters in each tree are real-valued and the notion of complex coefficients only appears when outputs from the two trees are combined. The addition of the second filter bank increases the redundancy of the transform to $2:1$ for a 1D signal. For an mD signal, the redundancy of the transform is $2^m:1$.

As well as removing the shift dependence of the transform, the two tree structure also allows the positive and negative frequencies present in the original signal to be treated separately. At each level of the 2D DT-CWT, a low pass image $\mathbf{B_0}$ and six sub-band images ($\mathbf{B_1}, \ldots, \mathbf{B_6}$) are produced. Each of the sub-band images contains the wavelet coefficients whose magnitude is proportional to any one of the $\pm 15^o$, $\pm 45^o$ and $\pm 75^o$ directional edges in the original image. Thus, the DT-CWT has associated with it good directional selectivity. This good directional selectivity

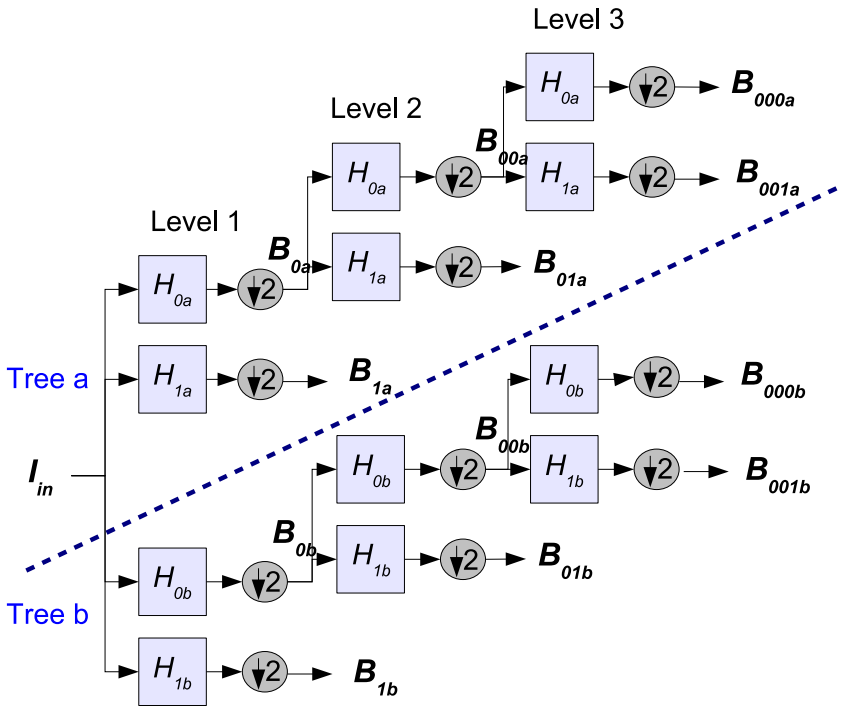


Figure 3.10: *Three-level DT-CWT for a 1D signal.*

is advantageous for texture representation inherent with many image processing applications [59, 78, 160, 185].

Figure 3.2 shows a two-level DT-CWT decomposition of the sample image given in Figure 3.6 (i). The intensity value of each of the sub-band images is obtained from the absolute value of the complex wavelet coefficients in each sub-band image. Thus bright pixels in any of the sub-band images indicate a large frequency content for that particular orientation. Note the $4:1$ redundancy for the 2D transform. This extra redundancy enables the properties of shift invariance and good directional selectivity to be associated with the DT-CWT.

To demonstrate the shift invariance of the DT-CWT, an input signal consisting of a step function at sixteen different shifts is decomposed using a four level DT-CWT. Recall that if the transform is shift invariant, the energy in each sub-band should remain constant regardless of the shift in the input. Figure 3.12 shows the sub-band signals associated with the DT-CWT. Since the energy with each sub-band signal at any given level remains constant regardless of shift, the DT-CWT is therefore shift invariant.

The good directional properties of the DT-CWT are shown in Figure 3.13. The 'circle' image shown earlier (Figure 3.9) is this time decomposed using the DT-CWT. Unlike the DWT which combines positive and negative frequencies and produces three sub-band images at each level, the DT-CWT treats positive and negative frequencies separately and produces six sub-band images at each level. Each sub-band contains wavelet coefficients whose magnitude are proportional

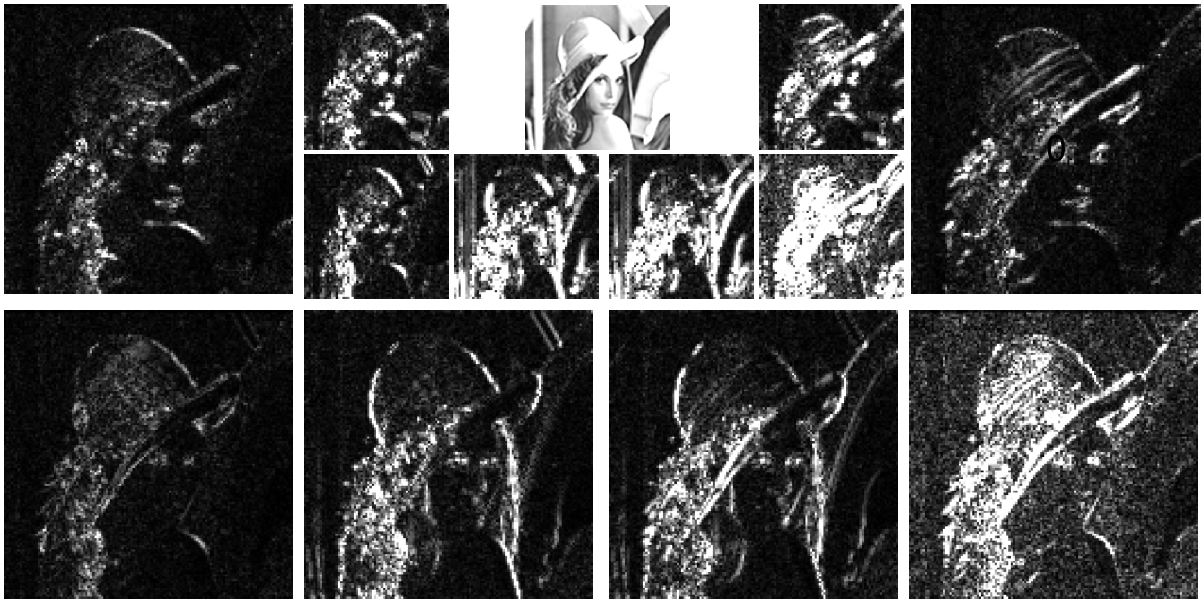


Figure 3.11: *Two-level DT-CWT decomposition of the image given in Figure 3.6 (i). The level one sub-band images are located around the boundaries and the level two sub-band images are nested inside them. The level two low-pass image of scaling coefficients is located in the middle of row one.*

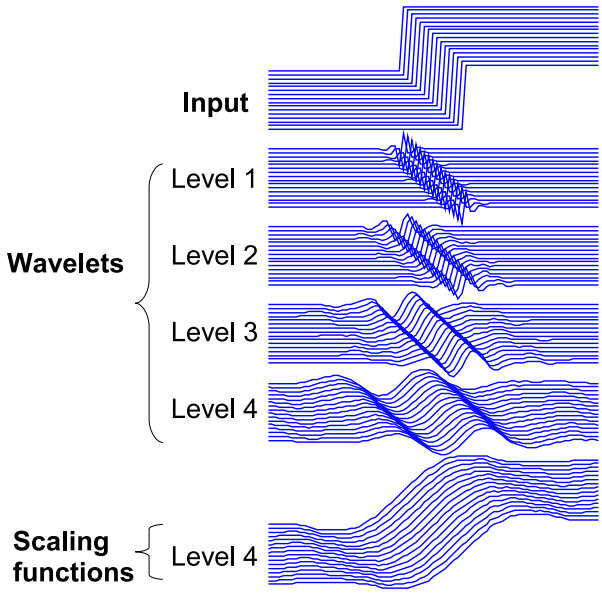


Figure 3.12: *Shift invariance of the DT-CWT. Input signal consists of a step function at sixteen different shifts. The four-level DT-CWT decomposition is show below. At each level, the energy in each sub-band remains constant regardless of shift. This invariance of energy to shift implies that the transform is shift independent.*

to one of the $\pm 15, \pm 45, \pm 75$ directional orientations of the input signal. Because positive and negative orientations are treated separately, the DT-CWT provides greater directional selectivity than the DWT.

3.2.1 *Q-shift* DT-CWT

In the initial version of the DT-CWT [119] the filter trees were designed to contain odd and even length filters alternatively from level to level. This odd/even placement of filters ensured that the sampling offsets between the filters in the two trees were evenly spaced. However, this system suffered from a complicated sampling structure and the corresponding filter tree responses were not identical. These properties of the transform limited its suitability for use in hierarchical algorithms which rely on a simple sub-sampling relationships between successive resolutions, e.g. quad-tree relationship. To address this issue, the *Q-shift* DT-CWT was introduced [120].

In the *Q-shift* DT-CWT two sets of filters are used: bi-orthogonal filters for level one and the q-shift filters for all higher levels. The introduction of the q-shift filters for levels ≥ 2 improves the overall sampling structure of the transform so that each wavelet coefficient at level l is now parent to four "child" wavelet coefficients at the lower resolution level $l-1$. This quad-tree relationship is illustrated in Figure 3.14 for the 15^o sub-band image obtained from the DT-CWT decomposition of the 'circle' image. Note that, while the *Q-shift* version of the transform improves the overall sampling structure, the good shift invariance and directional properties of

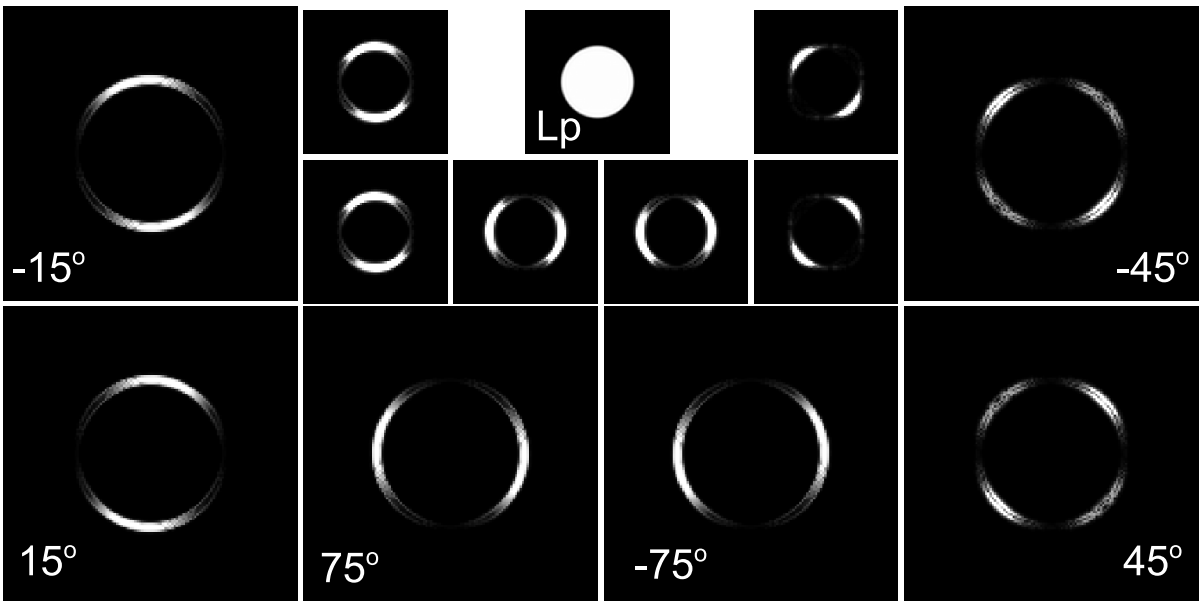


Figure 3.13: *The good directional properties of the DT-CWT are shown in the two-level DT-CWT decomposition of the 'circle' image shown in 3.9 (i). The level one sub-band images are located around the boundary and level two sub-band images are nested inside them. The level two low pass image of scaling functions is located at the top centre.*

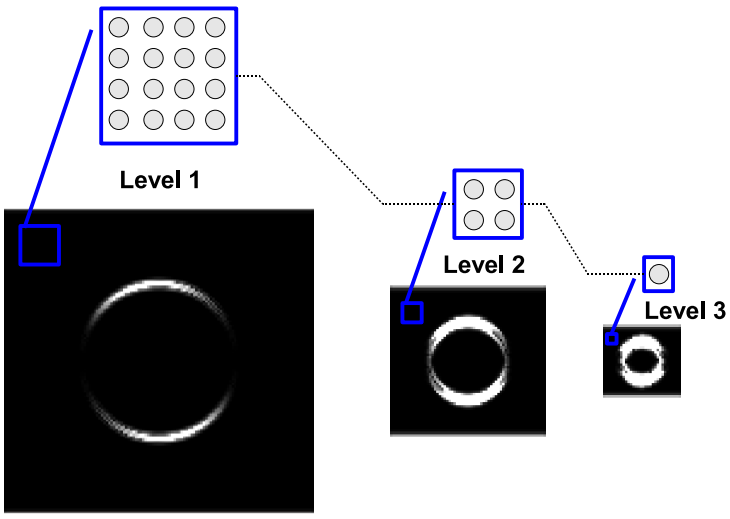


Figure 3.14: *Quad-tree relationship associated with Q-shift DT-CWT. At each level, each wavelet coefficient is parent to four children wavelet coefficients at the preceding level.*

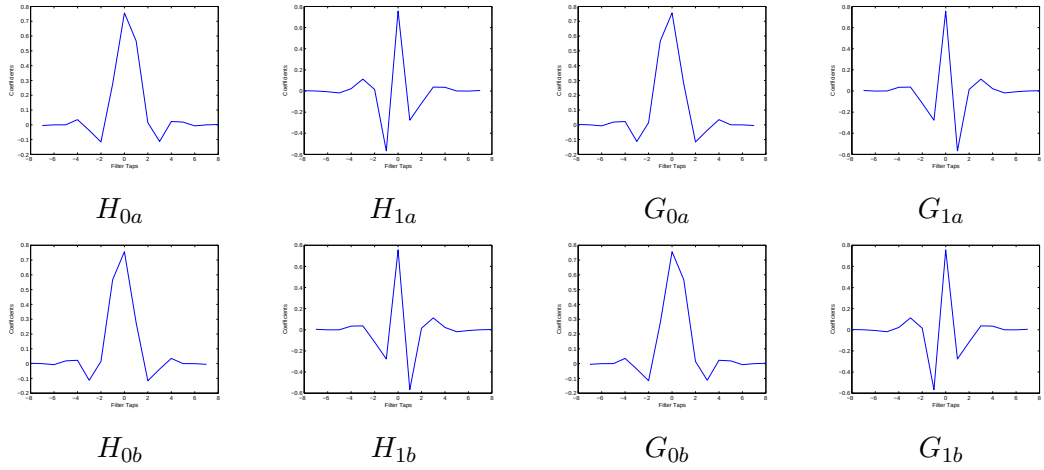


Figure 3.15: *Q-Shift* $16, 16$ *tap filters [120]. These filters are used for levels* ≥ 2 *in the Q-shift DT-CWT.* $[H_{0a}, H_{1a}]$ *and* $[G_{0a}, G_{1a}]$ *are the analysis and reconstruction filters for tree a, while* $[H_{0b}, H_{1b}]$ *and* $[G_{0b}, G_{1b}]$ *are the analysis and reconstruction filters for tree b.*

the original transform are also preserved.

The synthesis and segmentation algorithms developed as part of this work on example based image processing are hierarchical in nature and so the *Q-shift* version of the DT-CWT is used. Therefore, all further references to the DT-CWT will refer to the *Q-shift* version unless explicitly stated otherwise. For convenience the *Q-shift* name will be omitted and the transform will be referred to as the DT-CWT. As with the DWT decompositions given earlier, the Antonini $(7, 9)$ tap filters shown in Figure 3.7 are used for all the DT-CWTs presented here. However, unlike the DWT, these bi-orthogonal filters will only be used at level one since q-shift filters will be used for all levels greater than one. Kingsbury provides a range of suitable q-shift filters in [120]. In this work, the q-shift "c" $(16, 16)$-tap filters were used. These filters are shown in Figure 3.15, where $[H_{0a}, H_{1a}]$ and $[G_{0a}, G_{1a}]$ are the low pass, high pass analysis and reconstruction filters for tree a. Similarly, $[H_{0b}, H_{1b}]$ and $[G_{0b}, G_{1b}]$ are the low pass and high pass analysis and reconstruction filters for tree b.

This concludes the discussion on some of the aspects of wavelets which are relevant to this work. The next chapters will describe how the DT-CWT has been used in the example based image processing algorithms that have been developed as part of this thesis.

4

Exampled Based Synthesis with Wavelets[1]

4.1 Introduction

Recall that the aim of a texture synthesis process is to use an example texture image as a guide in creating a new (typically larger) texture image that is perceptually similar to the example texture image. The concept of texture synthesis and some existing approaches to the problem were discussed in chapter 2. It was found that while existing approaches all offer contributions to the texture synthesis problem, an algorithm that is stable, computationally efficient and robust enough to work for all texture types has yet to be realised.

To address this deficiency a new texture synthesis algorithm has been developed as part of this work on example based image processing. In the review in chapter 2, it was found that the non-parametric modelling approaches offered the most suitable means to capture the varying statistical behaviour associated with texture images. However, while this non-parametric framework provides an accurate means to model the texture behaviour, it has associated with it some limitations which have hampered its use. *The algorithm proposed here exploits the strengths of the non-parametric modelling framework within the wavelet domain.*

This chapter will begin by describing the single resolution synthesis algorithm developed by Efros and Leung [68]. Their algorithm is non-parametric and forms the basis on which the new Dual Tree-Complex Wavelet Transform Texture Synthesis (DT-CWT TexSyn) algorithm will be based. A description of this new algorithm will be given as well as some of the synthesised images obtained using this approach. The chapter will conclude by highlighting the strengths

[1] Results from this chapter have been published in [76–78].

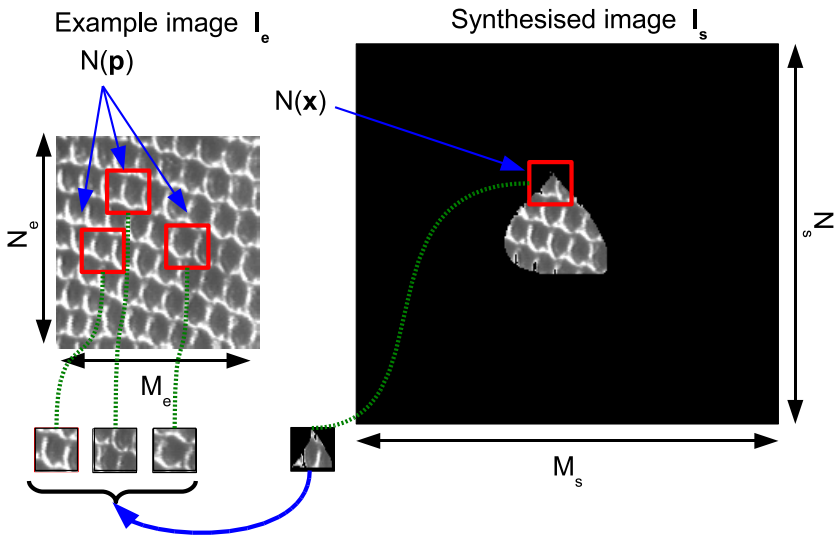


Figure 4.1: *Given an example texture image* $\mathbf{I_e}$*, a new image* $\mathbf{I_s}$ *is synthesised one pixel at a time. For each pixel* $I_s(\mathbf{x})$ *to be synthesised at site* $\mathbf{x}$*, the algorithm finds all neighbourhoods (boxes on the left) in* $\mathbf{I_e}$ *that are perceptually similar to the neighbourhood* $N(\mathbf{x})$ *of* $I_s(\mathbf{x})$ *(box on the right). The centre pixels of these similar neighbourhoods form an approximation to the pdf* $p(I_s(\mathbf{x}))$ *of* $I_s(\mathbf{x})$*, which is randomly sampled and the new value assigned to site* $\mathbf{x}$*.*

of the new algorithm and illustrating some possible directions for future work.

4.2 Single Resolution Synthesis

Let $\mathbf{I_e}$ denote the example texture image that will be used to generate a new texture image $\mathbf{I_s}$ of similar content. The example image $\mathbf{I_e}$ is defined on a $M_e \times N_e$ lattice $\mathbf{X_e}$. Each site in $\mathbf{X_e}$ can be indexed using the spatial vector $\mathbf{p} = [x, y]^T$, and $I_e(\mathbf{p})$ denotes the pixel at site $\mathbf{p}$ in $\mathbf{I_e}$. The new image to be synthesised is defined on a similar but typically larger $M_s \times N_s$ lattice $\mathbf{X}$. Each site in $\mathbf{X}$ can be indexed using the spatial vector $\mathbf{x} = [x, y]^T$ such that $I_s(\mathbf{x})$ denotes the pixel at site $\mathbf{x}$ in $\mathbf{I_s}$. Since $\mathbf{I_s}$ is generated using only values from $\mathbf{I_e}$, each $I_s(\mathbf{x})$ will be taken from the set $\{I_e(\mathbf{p}), \forall \mathbf{p} \in \mathbf{X_e}\}$.

The aim of the texture synthesis algorithm is to generate a new image $\mathbf{I_s}$ that is perceptually similar to the example image $\mathbf{I_e}$ thus giving the overall impression that both images were generated from the same underlying process. In probability terms this can be expressed as finding the image $\mathbf{I_s}$ that maximises the probability distribution $p(\mathbf{I_s}|\mathbf{I_e})$. Estimating and sampling from this probability distribution is computationally intractable given the range of possible configurations of $\mathbf{I_s}$. To reduce the computational burden and based on the observation that texture is composed of repeated homogeneous regions all of which have similar statistics, the texture is modelled as a realisation of a *Markov Random Field* (MRF). The MRF property was introduced

earlier in chapter 2 and states that the value of the pixel $I_s(\mathbf{x})$ at any site $\mathbf{x}$ is dependent only on the values of the pixels that are located in the spatial neighbourhood $N(\mathbf{x})$ surrounding $I(\mathbf{x})$. The advantage of the MRF model is that it enables the joint distribution of the entire image $p(\mathbf{I_s})$, to be uniquely determined by its individual *Local Conditional Probability Density Functions* (LCPDF) $p(I_s(\mathbf{x})), \forall \mathbf{x} \in \mathbf{X}$ [20]. That is,

$$p(\mathbf{I_s}) = \prod_{\mathbf{x} \in \mathbf{X}} p(I_s(\mathbf{x})) \tag{4.1}$$

Under this MRF property, the problem of texture synthesis is now that of estimating $p(I_s(\mathbf{x}))$ for each pixel in the new image to synthesised $\mathbf{I_s}$. Efros and Leung proposed an empirical method of estimating an approximation to the probability density function (pdf) of each pixel to be synthesised. Their approach is non-parametric and so does not enforce any explicit model on the new texture image to be synthesised. Instead the new image is implicitly modelled using empirical measurements taken only from the example texture. These empirical measurements are based on information from: (i) the spatial $w \times w$ neighbourhood centred around the pixel to be synthesised and (ii) the set of all possible $w \times w$ neighbourhoods in the example image $\mathbf{I_e}$. This is shown in Figure 4.1. Efros and Leung use a $w \times w$ block neighbourhood structure and the width of this neighbourhood w is a user defined parameter which specifies how stochastic or random the user believes the texture to be. Results at the end of this chapter will demonstrate how the quality of the overall synthesised texture depends heavily on the the correct interpretation of the texture scale and hence the correct choice of the parameter w.

4.2.1 Neighbourhood Searching

To obtain an approximation to the pdf $p(I_s(\mathbf{x}))$ of $I_s(\mathbf{x})$, the $w \times w$ neighbourhood around $I_s(\mathbf{x})$, denoted $N(\mathbf{x})$, is constructed and compared to the set of all possible neighbourhoods in the example texture image, $N(\mathbf{p}), \mathbf{p} \in \mathbf{X_e}$. The perceptual similarity between two neighbourhoods $N(\mathbf{x})$ and $N(\mathbf{p})$, denoted $D(N(\mathbf{x}), N(\mathbf{p}))$, is defined to be the sum of squared intensity differences between individual pixels in each neighbourhood. To preserve the local structure of the texture, the error value for sites adjacent to the centre of the neighbourhood receives more weighting than that for sites located close to the boundaries. To facilitate this variable weighting scheme, the distance measure $D(\cdot, \cdot)$ is multiplied by a 2D Gaussian kernel of variance $w/6.4$. Given this distance measure, the best matching or most similar neighbourhood denoted N_{best} to $N(\mathbf{x})$ is found from the example image. That is,

$$N_{best} = \arg \min_{\mathbf{p} \in \mathbf{X_e}} D(N(\mathbf{x}), N(\mathbf{p})) \tag{4.2}$$

Using the metric $D(N(\mathbf{x}), N_{best})$ as a reference, the set of neighbourhoods in $\mathbf{I_e}$ that fall within some threshold value of this this metric are collected. This threshold value is given as,

$$D(N(\mathbf{x}), N(\mathbf{p})) \leq (1+\epsilon) D(N(\mathbf{p}), N_{best}) \tag{4.3}$$

where the constant ϵ is a scalar value used to control the randomness of the synthesised texture. A high value of ϵ will result in a large number of neighbourhoods satisfying the criterion in (4.3). Since the pdf of the pixel to be synthesised is formed from the set of neighbourhoods that satisfy this criterion, a larger set will imply a more widely distributed pdf. This will result in a greater choice for $I_s(\mathbf{x})$ and so a more random texture. Conversely, a small value of ϵ will result in a smaller set of suitable neighbourhood candidates, a narrower pdf and thus a more structured texture. In this application ϵ remains constant for all textures and is set at $\epsilon = 0.1$. Figure 4.4 shows the images obtained using three different values of ϵ.

Using the set of most similar neighbourhoods satisfying (4.3), $p(I_s(\mathbf{x}))$ is approximated by taking the centre pixels of each of these neighbourhoods. This is given in (4.4). The new pixel to be synthesised $I_s(\mathbf{x})$ is obtained by sampling randomly from $p(I_s(\mathbf{x}))$.

$$p(I_s(\mathbf{x})) = \{I_e(\mathbf{p}), \forall \mathbf{p} \in \mathbf{X_e} \text{ such that } D(N(\mathbf{x}), N(\mathbf{p})) \leq (1+\epsilon) D(N(\mathbf{p}), N_{best})\} \tag{4.4}$$

4.2.2 Synthesising Pixels

The algorithm given in the previous section works for synthesising a pixel when all the neighbourhood pixels are already known. In practice when synthesising an image, some of the values of the spatial neighbours will yet have to be synthesised and so will be unknown. In theory, the correct solution for synthesising pixels would then be to consider the joint probability of all pixels together. However, this is computationally intractable for images of realistic size. To overcome this Efros and Leung proposed a Shannon [187] inspired heuristic whereby the new texture image is grown outward in layers from an initial 3×3 seed taken randomly from the example image (in the case of region *filling* in or *in-painting*, the synthesising proceeds from the boundary or edge of the region to be filled in).

To accommodate unknown pixel values in the neighbourhood of $N(\mathbf{x})$, the pixel synthesis algorithm is modified by only matching the known values in $N(\mathbf{x})$ and normalising the error by the total number of known pixels when comparing two neighbourhood structures. It should be noted that this heuristic approach offers no guarantees that the pdf of $I_s(\mathbf{x})$ will remain valid as the rest of $N(\mathbf{x})$ is filled, but Efros and Leung maintain that it appears a good approximation. However, it should also be noted that the stability of this approximation is very much dependent on the correct choice of neighbourhood size. In cases where the wrong neighbourhood size is chosen, the synthesis process enters a loop whereby the new generated texture does not resemble the original seed texture. Evidence of this is shown in the results presented in Figure 4.3.

The image to be synthesised is initialised by placing a "seed" of the example texture in its centre. In the Efros and Leung algorithm a 3×3 seed is used, however in this implementation it was found that this "seed" was too small to kick start a stable synthesis process. In practice,

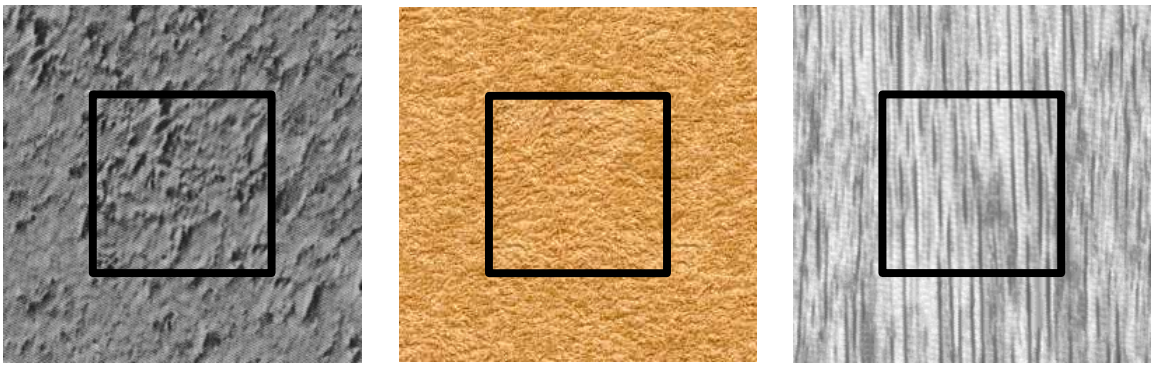


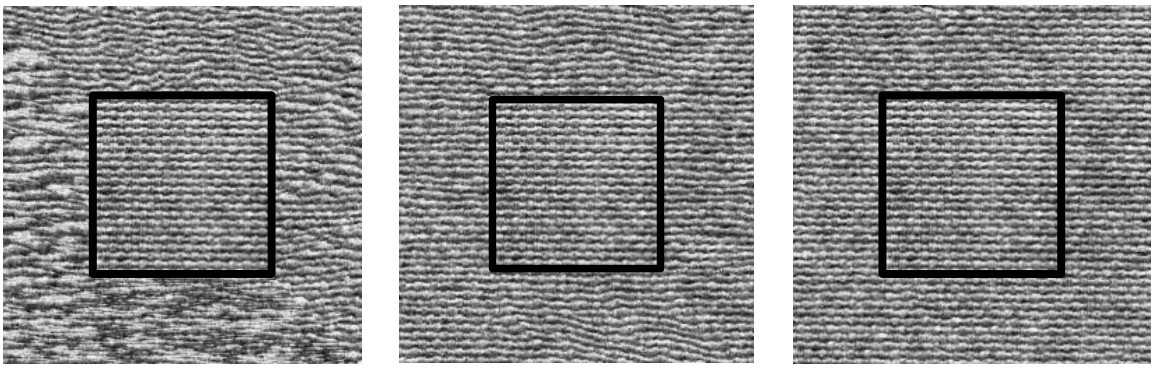
Figure 4.2: *Synthesised textures obtained using the Efros and Leung algorithm. The synthesised images are of size* 256×256 *pixels and the example textures (shown inside the black square) are of size* 128×128 *pixels. The algorithm parameters of neighbourhood width and randomness were set to* $w = 9$ *and* $\epsilon = 0.1$.

it was found that the "seed" needs to be at least of size $w \times w$. In this implementation when the images to be synthesised are significantly larger than the example images, it seems sensible to use the entire example texture images as a "seed". As well as providing a stable initialisation point, the large seed speeds up the synthesis process by reducing the number of pixels to be synthesised.

The order in which pixels are synthesised is determined by the number of known neighbours in each individual pixel neighbourhood. Pixels with a higher number of known neighbours will be synthesised before those with a lower number of known neighbours. Boundary neighbourhoods are treated in a manner similar to the partially synthesised neighbourhoods whereby only known pixels in the neighbourhoods are matched and the difference in neighbourhoods is normalised by the number of known pixels.

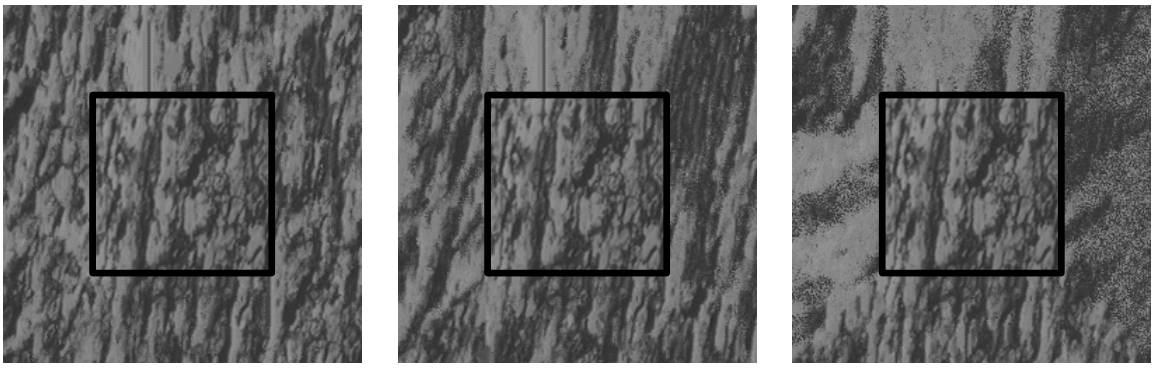
Results obtained using the Efros and Leung algorithm are shown in Figures 4.2, 4.4 and 4.3. The synthesised images are of size 256×256 and were generated from the 128×128 example texture images. These example images were used as a seed to initialise the synthesis process and are shown inside the black square of each new image. These textures can be classified as being mainly stochastic and in all three implementations a neighbourhood size of 9×9 and a randomness parameter of $\epsilon = 0.1$ was used.

The results shown in Figure 4.2 demonstrate that under the correct conditions the Efros and Leung algorithm will generate impressive results which provide an accurate representation of the original example texture. However, it should be noted that stochastic textures are the easiest to replicate accurately. This is because their random nature implies that neighbourhoods are less structured and so their is greater choice in suitable pixel values. In general for stochastic textures the algorithm parameters of ϵ and especially w can remain constant at approximately $w = 9$, $\epsilon = 0.1$.



(i) (ii) (iii)

Figure 4.3: *The scale dependence of the Efros and Leung algorithm is demonstrated by examining the results obtained using different values of neighbourhood width w. The neighbourhood size $w \times w$ used during the synthesis process must be large enough to capture the underlying scale of the texture. In this case three different values of w were used; (i) $w = 5$, (ii) $w = 9$ and (iii) $w = 17$. Widths $w = 5$ and $w = 9$ were not large enough to capture the scale of the example texture and so the synthesised texture does not resemble the example texture (inside the black box). The neighbourhood width $w = 17$ is large enough but this large width adds to the computational burden of the synthesis process.*



(i) (ii) (iii)

Figure 4.4: *Varying the value of ϵ will effect the randomness of the synthesised texture. In (i) $\epsilon = 0.1$, (ii) $\epsilon = 0.5$ and (iii) $\epsilon = 1$. Images (ii) and (iii) illustrate how taking too large a value of ϵ will reduce the accuracy of the result.*

However, for more rigid and deterministic textures, it is necessary to vary these parameters (particularly w) to suit specific texture types. This is shown in Figures 4.4 and 4.3.

Figure 4.4 illustrates the effect of varying ϵ. The algorithm parameter ϵ controls the randomness of the synthesised texture; a large value of ϵ will result in a widely distributed pdf for each pixel to be synthesised and hence a more random synthesised texture. Figure 4.4 shows the synthesised images when (i) $\epsilon = 0.1$, (ii) $\epsilon = 0.5$ and (iii) (i) $\epsilon = 1$. Overall it is found from observation that $\epsilon = 0.1$ gives the best results and this parameter can remain constant for all texture types.

The value of the neighbourhood width w is much more fundamental to the success of the algorithm. In addition it is found that there is no one value of w that is suitable for all texture types. The effect of taking different values of w is shown in Figure 4.3. The example textures used in Figure 4.3 are deterministic and so are more difficult to synthesise given there is less margin for error. Therefore, if the incorrect neighbourhood size is used the effect on the quality of the synthesised result is visibly obvious. In Figure 4.3 the neighbourhood widths were set as: (i) $w = 5$, (ii) $w = 9$ and (iii) $w = 17$. The synthesised results obtained using $w = 5$ and $w = 9$ fails to resemble the example texture shown inside the black box. This is because the neighbourhood sizes of 5×5 and 9×9 are not large enough to capture the structural features intrinsic with this texture pattern. In addition, toward the edges of the texture, the growing garbage phenomena discussed earlier is shown. This garbage texture is produced when the approximation to the pdf of the pixel to be synthesised is far from the true pdf and thus it enters a loop whereby all preceding pixels in the neighbourhood to be synthesised are not a true representation of the texture pattern. The neighbourhood size of 17×17 is large enough and so the synthesised texture is perceptually similar to the example texture. The randomness parameter ϵ remained constant for all three implementations at $\epsilon = 0.1$. Choosing the correct value of w is related to interpreting the underlying texture scale.

Scale dependence and computational inefficiency are the two main limitations of the Efros and Leung approach. These will be discussed next.

- *Scale Dependence:*
  The quality of the synthesised image $\mathbf{I_s}$ depends on the accuracy with which the underlying pdf $p(I_s(\mathbf{x}))$ is approximated at each site $\mathbf{x} \in \mathbf{X}$. Since $p(I_s(\mathbf{x}))$ is constructed using a nearest neighbour searching technique, the neighbourhood $N(\mathbf{x})$ and primarily the size of the neighbourhood $w \times w$ will directly influence the accuracy with which the approximated pdf represents the true pdf. As mentioned earlier, the width of the neighbourhood w is a user controlled parameter and specifies how stochastic the user believes the texture to be. If the texture is presumed to be deterministic then the size of the window should be on the scale of the largest regular feature in the texture. Taking too small a value for w will result in this regular feature not being replicated in the synthesised texture. On the other hand, taking too large a value of w can result in some of the randomness of the texture

being lost. The texture scale and the corresponding correct neighbourhood size will vary for each texture. The result is a parameter that has to be adjusted for each texture thus implying a scale dependent algorithm.

- *Computational Inefficiency:*
  Intrinsic to this algorithm is the need to search and compare each neighbourhood in the example image to the neighbourhood of the pixel to be synthesised. Since neighbourhood comparison is based on the normalised sum of square pixel intensities, the overall computationally liability introduced as a result of such an exhaustive search is cumbersome to say the least. It will be shown later that to synthesise a $M_s \times N_s$ image from a $M_e \times N_e$ example image, the computational load is of the order of $4M_eN_ew^2$ for one pixel and $4M_sN_sM_eN_ew^2$ for the entire image. This large computational burden implies that for realistically sized images, the synthesis process is slow. In addition and as a result of the scale dependence of the algorithm, the neighbourhood width w needs to be big enough in order to capture the largest feature present in the texture. For example, for some of the 128×128 pixel example images shown later in the results section, a neighbourhood size of $w \geq 11$ pixels is required in order for the synthesised result to replicate the sample texture. A large neighbourhood size further increases the computational burden.

In order to speed up the process an alternative implementation of the single resolution texture synthesis algorithm was carried out. This implementation exploits the extra processing power available via the Graphics Processing Unit (GPU) and speeds up the synthesis process by performing the computationally intensive nearest neighbour searching on the GPU. This work was done in collaboration with Francis Kelly and was published in [76, 116]. A description of this GPU implementation of the single resolution synthesis algorithm is given in Appendix A.

The strength of this non-parametric modelling technique is evident from the synthesised results and indeed the number of algorithms that have evolved from this technique [14,27,98,101, 213]. The means by which the algorithms described in [14, 27, 98, 101, 213] extend and alter the Efros and Leung algorithm were discussed in chapter 2. It should be noted that while they offer significant improvements, none of these approaches can be classified as fully scale independent, computationally efficient and suitable for all texture types. As part of this work on example based processing a new texture synthesis algorithm has been developed. This algorithm is non-parametric and exploits the strength of the DT-CWT representation by performing synthesis in the wavelet domain. This algorithm will be described next.

4.3 DT-CWT TexSyn

The Dual-Tree Complex Wavelet Transform (DT-CWT) developed by Kingsbury [119] has found many applications in image processing [50, 76, 160, 186] due to its strong shift invariance, good directional sensitivity and relatively low redundancy. Aspects of the wavelet transform were

discussed in chapter 3. In the discussion on texture features in chapter 2 it was found that two commonly used features to describe a given texture are resolution and regionality. The resolution property refers to the frequency components present in a given texture and so it is a natural progression to represent texture in a multi-resolution domain so that the various frequency components of the texture can be analysed. The DT-CWT provides such a suitable domain since the DT-CWT is a multi-resolution transform that allows simultaneous spectral and spatial analysis of a given image. At each level or resolution of the transform, six directionally sensitive sub-band images and one low pass image are produced. These directionally sensitive sub-band images are orientated at $\pm 15, \pm 45$ and ± 75, implying that the DT-CWT has an overall redundancy of $4:1$ for a $2D$ image.

Figure 4.5 shows the sub-band and low pass images produced in the DT-CWT decomposition of an image taken from the Brodatz collection [34]. The sub-band images are made up of complex coefficients and as a result the intensity values of the images shown in Figure 4.5 represent the absolute value of these wavelet coefficients. The idea behind the DT-CWT TexSyn algorithm developed as part of this work is to perform synthesising at each level of the wavelet tree using a non-parametric modelling technique derived from the Efros and Leung algorithm. Modelling the texture at different resolutions allows the different frequency components of the texture to be analysed and thus synthesised separately. Because of correlation among the sub-band images at any given level and to speed up the process, each of the sub-bands will be synthesised in parallel. The algorithm will be described in three stages: (i) *algorithm initialisation*, (ii) *multi-resolution neighbourhood searching* and (iii) *synthesising the wavelet tree*. Before describing each of these stages, the symbols that have been defined so far and those that will be used in the description of the wavelet based algorithm are summarised in Table 4.1.

4.3.1 Algorithm Initialisation

Similar to the single resolution algorithm described earlier, the input parameters of the DT-CWT TexSyn algorithm are the example texture image $\mathbf{I_e}$ and the dimensions $M_s \times N_s$ of the new image to be synthesised. The algorithm begins by taking the L-level DT-CWT of $\mathbf{I_e}$. An example three-level decomposition is shown in Figure 4.5. The input image measures 128×128 pixels and due to the sub-sampling inherent with the DT-CWT, sub-band images at levels 1, 2 and 3 measure 64×64, 32×32 and 16×16 pixels respectively. The level 3 low pass image measures 16×16 pixels. The idea behind the wavelet approach is to analyse the dominant frequency components present in the example texture and perform synthesis at each level of the transform. Beginning at the coarsest level and moving toward the finer resolution, each level of the wavelet tree will be synthesised using a technique derived from the nearest neighbour searching process discussed earlier. Given the correlation among the sub-band levels and orientations, the synthesis process is constrained in two ways. Firstly, correlation among the sub-band levels is acknowledged by the fact that at any level l the synthesis process will be

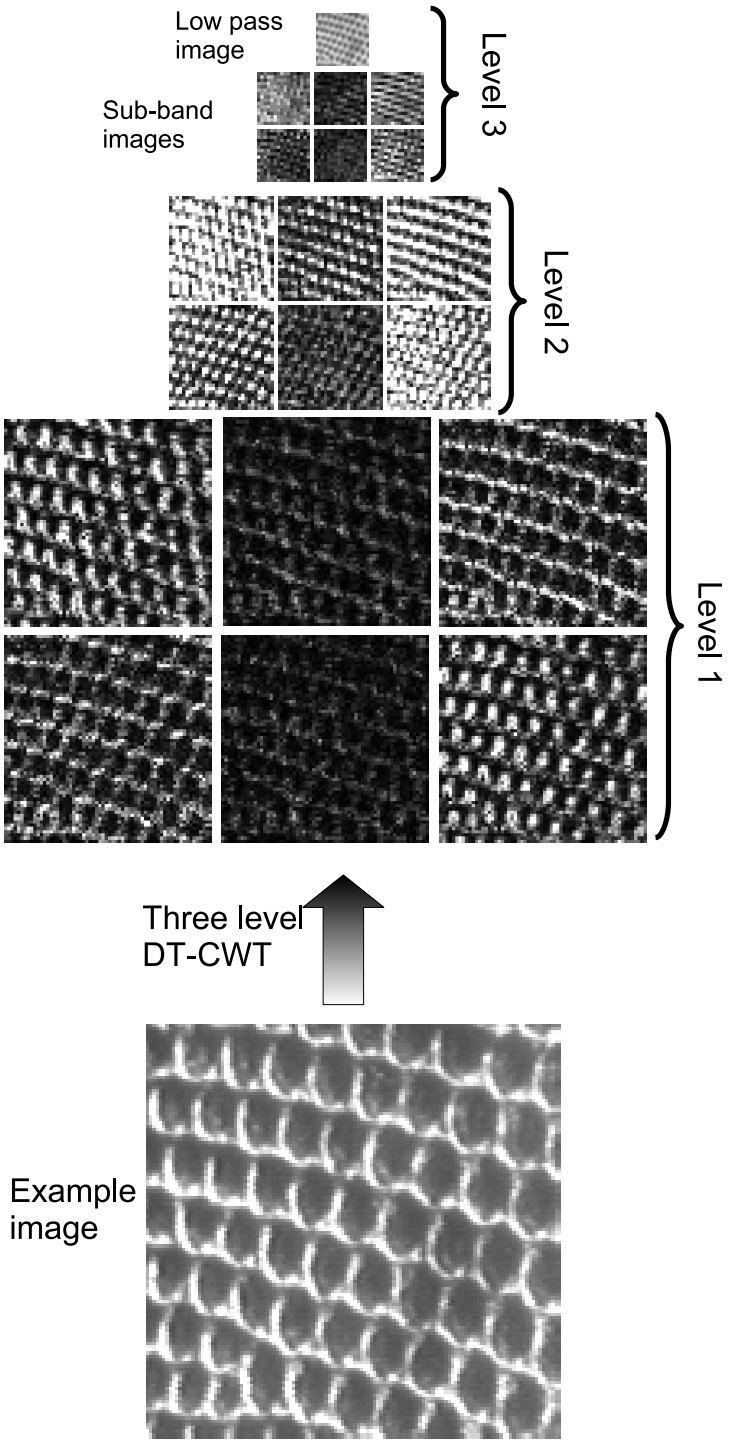


Figure 4.5: *Three level DT-CWT decomposition of texture image showing six sub-band images produced at levels* $1-3$ *and the level* 3 *low pass image.*

Symbol	Meaning
$\mathbf{I_e}$	Example Image of size $M_e \times N_e$
$\mathbf{X_e}$	Lattice on which $\mathbf{I_e}$ is defined
$\mathbf{I_s}$	Image to be synthesised of size $M_s \times N_s$
$\mathbf{X}$	Lattice on which $\mathbf{I_s}$ is defined
$I_e(\mathbf{p})$	Pixel at site $\mathbf{p}$ in $\mathbf{I_e}$
$I_s(\mathbf{x})$	Pixel at site $\mathbf{x}$ in $\mathbf{I_s}$
$\mathbf{B_e^l}$	Example set of six sub-bands at level l
$\mathbf{X_e^l}$	Lattice on which each image in $\mathbf{B_e^l}$ is defined
$B_e^l(\mathbf{p})$	Set of six wavelet complex coefficients at site $\mathbf{p}$ in $\mathbf{B_e^l}$
$N(\mathbf{p})$	Neighbourhood around site $\mathbf{p}$
$\mathbf{B_s^l}$	Synthesised set of six sub-band at level l
$\mathbf{X^l}$	Lattice on which each image in $\mathbf{B_s^l}$ is defined
$B_e^l(\mathbf{p})$	Set of six wavelet complex coefficients at site $\mathbf{p}$ in $\mathbf{B_e^l}$
$N^6(\mathbf{p})$	6D Neighbourhood around site $\mathbf{p}$
$\mathbf{G_e}$	Final level example low pass image
$\mathbf{G_s}$	Final level synthesised low pass image

Table 4.1: *Symbols used in description of wavelet based texture synthesis algorithm.*

guided by the result obtained at the immediate coarser level $l+1$. Secondly, correlation among the sub-band orientations within each level is accounted for by performing the neighbourhood searching process in the six sub-band images in unison. That is, rather than considering each wavelet coefficient at site $\mathbf{x}$ in each sub-band image, the wavelet coefficients in all six sub-band images at site $\mathbf{x}$ will be synthesised together.

Given the dimensions $M_s \times N_s$ of the image to be synthesised, the DT-CWT wavelet tree is initialised by calculating the dimensions of each of the L level sub-band images and the final low pass image. This is easily done given the simple sub-sampling structure under which the DT-CWT operates. That is, the dimensions of the images at level l are double those at level $l+1$ and so one pixel at level $l+1$ is "parent" to four "child" pixels at level $l+1$. Each of the $6L$ sub-band images, $\mathbf{B_s^l} \forall l = 1, \cdots, L$ and the coarse level low pass image $\mathbf{G_s}$ in the wavelet tree to be synthesised is initialised with zeros to denote unknown wavelet coefficients to be synthesised.

To kick start the synthesis process a "seed" from each example sub-band image is placed in the centre of each sub-band and low pass image in the wavelet tree to be synthesised. The source and location of this seed should be consistent throughout the transform. For example if the "seed" placed in each of the sub-band images in the set $\mathbf{B_s^l}$ is centred around site $\mathbf{x}$, then the size of the "seed" placed in the sub-band images at the coarser resolution $\mathbf{B_s^{l+1}}$ should be be centred around $\mathbf{x}/2$ and quarter the size of that placed at the previous level. Correspondingly, if the "seed" placed in $\mathbf{B_s^l}$ is sourced from $\mathbf{B_e^l}$ centred at site $\mathbf{p}$, then the seed placed in $\mathbf{B_s^{l+1}}$

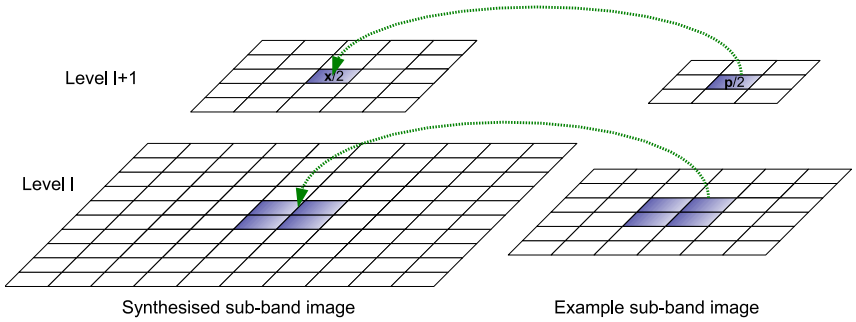


Figure 4.6: *Sourcing the "seed" texture from the example sub-band image to the synthesised sub-band image at different resolutions.*

should be sourced from $\mathbf{B_e^{l+1}}$ and centred on $\mathbf{p}/2$. This idea of consistent "seed" placing is illustrated in Figure 4.6 for one sub-band image at levels l and $l+1$.

Using the quad-tree relationship illustrated in Figure 4.6, this seeding process should be repeated for all sub-band images $\mathbf{B_s^l}\ \forall\, l = 1, \cdots, L$ and the coarse level low pass image $\mathbf{G_s}$ in the wavelet tree to be synthesised. Once each image has been initialised with a "seed", the algorithm proceeds to the next step, neighbourhood searching.

4.3.2 Multi-directional Neighbourhood Searching

The idea behind the DT-CWT TexSyn algorithm is to synthesise a wavelet tree similar to the example wavelet tree by considering a non-parametric synthesis procedure at each level of the transform. At the coarsest level L of the transform, large features in the texture are represented by fewer pixels and so much of the intensive block matching process will be performed at this level. Using estimates obtained at the coarse level, the finer resolution levels will be updated using one of three methods described in the next section.

Note that this section is titled multi-directional rather than multi-resolution as it will describe the manner in which each of the coarse level six sub-band images of different directional orientation in the set $\mathbf{B_s^L}$ are synthesised using a variant of the non-parametric synthesis process given in section 4.2.

Because of correlation between the sub-band images at each orientation, each of the sub-band images in the set $\mathbf{B_s^L}$ will be synthesised in unison thus implying that the set of six wavelet coefficients $B_s^L(\mathbf{p})$ at site $\mathbf{p}$ in $\mathbf{B_s^L}$ will be synthesised together. Similar to the single resolution synthesis algorithm, a Markovian assumption over each of the sub-band images will be introduced. Under this assumption, the wavelet coefficient at any site will be dependent only on the set of wavelet coefficients in its predefined spatial neighbourhood. This neighbourhood, $N(\mathbf{x})$, will be a square block of size $w_1 \times w_1$ and centred on the site $\mathbf{x}$ of the wavelet coefficient to be synthesised. The low pass image is not included in this neighbourhood search and scaling functions in the low pass image will be synthesised based only on information taken from the

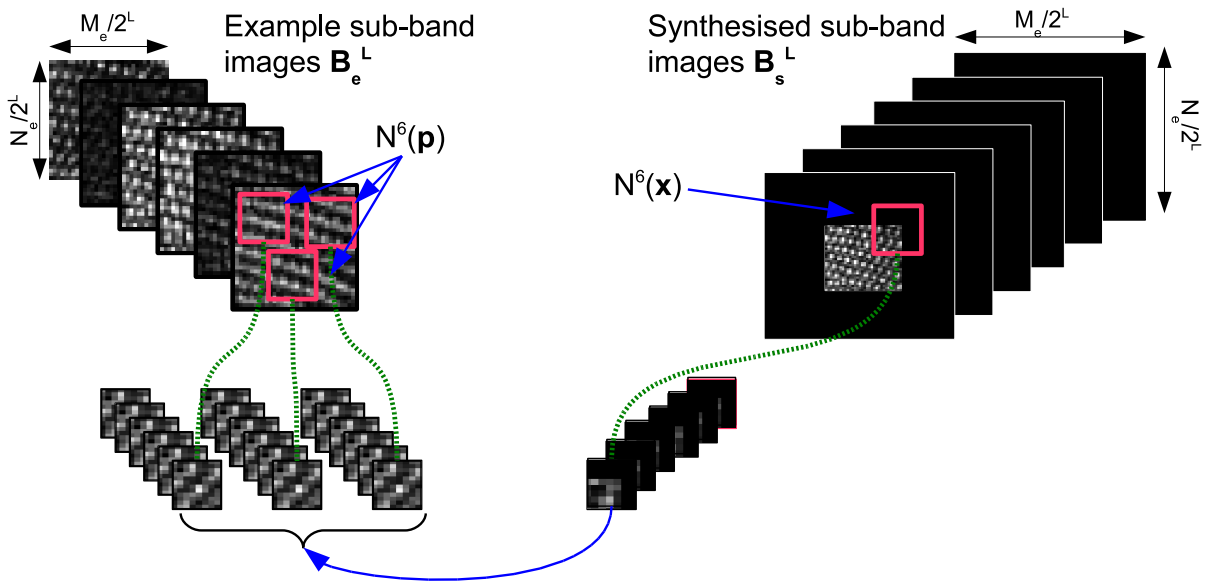


Figure 4.7: *Multi-directional neighbourhood searching. The coarse level L sub-band images are synthesised using a variant of the single resolution algorithm discussed previously. Block matching is 6D given that each neighbourhood is made up of individual neighbourhoods from each of the six sub-band images.*

sub-band search.

To synthesise each set of six wavelet coefficients $B_s^L(\mathbf{x})$ at site $\mathbf{x}$ in level L, it is necessary to construct an approximation to the pdf of $B_s^L(\mathbf{x})$, denoted $p(B_s^L(\mathbf{x}))$. To estimate $p(B_s^L(\mathbf{x}))$ the $w_1 \times w_1$ 2D neighbourhood $N(\mathbf{x})$ centred at site $\mathbf{x}$ in each of the six sub-band images in $B_s^L(\mathbf{x})$ is constructed. The width of this neighbourhood remains constant for all texture types. This implies that the algorithm is scale independent. Initial testing of the algorithm found that $w_1 = 5$ is large enough to capture both the deterministic and stochastic features of most natural textures. The set of six 2D neighbourhoods taken from each sub-band image in $\mathbf{B_s^L}$ is combined to form one 6D neighbourhood structure $N^6(\mathbf{x})$.

Similar to the single resolution case $N^6(\mathbf{x})$ is compared to the set of all possible $w_1 \times w_1$ 6D neighbourhoods in the example sub-band image set $\mathbf{B_e^L}$. The similarity between two 6D neighbourhoods is given as the sum of the absolute value of differences between the corresponding complex wavelet coefficients. Under this metric, the most similar neighbourhood to $N^6(\mathbf{x})$ is found and given as,

$$N^6_{best} = \arg \min_{\mathbf{p} \in \mathbf{X_e^L}} D(N^6(\mathbf{p}), N^6(\mathbf{x})) \tag{4.5}$$

Using the metric $D(N^6_{best}, N^6_{\mathbf{x}})$ as a reference, the set of most similar 6D neighbourhoods that fall within some threshold of this metric is collected. This set is composed of neighbourhoods that satisfy the following criterion:

$$D(N^6(\mathbf{x}), N^6(\mathbf{p})) \leq (1+\epsilon) D(N^6_{best}, N^6(\mathbf{x})) \tag{4.6}$$

where ϵ is a constant that controls the randomness of the synthesised texture. As with the single resolution case, ϵ remains constant at $\epsilon = 0.1$. The set of neighbourhoods that satisfy (4.6) are used to form an approximation to the 6D pdf of $B^L_s(\mathbf{x})$ by collecting the six wavelet coefficients at the centre of each of the 2D neighbourhoods that make-up the 6D neighbourhood. The 6D pdf $p(B^L_s(\mathbf{x}))$ is given as,

$$p(B^L_s(\mathbf{x})) = \{B^L_e(\mathbf{p}), \forall \mathbf{p} \in \mathbf{X^L_e} \text{ such that } D(N^6(\mathbf{x}), N^6(\mathbf{p})) \leq (1+\epsilon) D(N^6_{best}, N^6(\mathbf{x}))\} \tag{4.7}$$

The pdf $p(B^L_s(\mathbf{x}))$ is sampled randomly and the six chosen wavelet coefficients are assigned to the corresponding sub-band image at location $\mathbf{p}$. Figure 4.7 illustrates the multi-directional neighbourhood searching process that takes place at the coarse level. On the right are the coarse level sub-band images to be synthesised and on the left are the coarse level example sub-band images. The 6D neighbourhood of the wavelet coefficients to be synthesised are shown together with the set of most similar neighbourhoods from the example sub-band image set.

A Note on the DT-CWT and DWT

It is worth at this point reinforcing why the DT-CWT was favoured over the discrete wavelet transform (DWT). Inherent with both transforms is the separation of an input signal (image) into its high and low frequency components. Using the high frequency information, the DWT creates three sub-band images whose wavelet coefficients are proportional to the $0^o, 45^o$ and 90^o directional components of the input signal. In contrast, the DT-CWT creates six sub-band images with wavelet coefficients proportional to the $\pm 15^o, \pm 45^o$ and $\pm 75^o$ directional components of the input signal. The DWT is non-redundant while the DT-CWT has a redundancy of $2^m : 1$ for an mD signal. In the DT-CWT TexSyn algorithm, wavelet coefficients are synthesised by considering the set of six coefficients at any level l. By summing the neighbourhood matches of the six individual sub-band coefficients, all the high frequency information is in fact combined. Thus, it could be argued that the directional information gained through the separation of high frequency components is lost and so in theory synthesising texture using the DWT should give a similar result to that obtained using the DT-CWT.

The direction orientation information given in the DT-CWT could be exploited by introducing a weighting between sub-band images. The weight assigned to each sub-band image would be dependent on the energy contained in that image. High energy in a sub-band image implies that there exists a large occurrence of that particular directional orientation in the input signal. A weighting function would give matches from these high energy sub-bands priority in the neighbourhood searching process. This weighting function will be discussed later in chapter 6 as it used in the DT-CWT segmentation algorithm developed as part of this work. However, it was

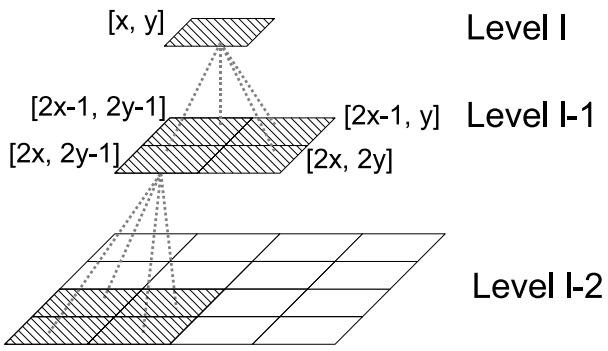


Figure 4.8: *Quad-tree relationship where one wavelet coefficient at level l is "parent" to four "child" coefficients at preceding higher resolution level $l-1$.*

found that in the case of texture synthesis, adequate results are obtained without this weighting function and so it was not necessary to incorporate it into the DT-CWT TexSyn algorithm.

So, if not due to the directional qualities of the DT-CWT, then why was it favoured over the DWT? The answer to this question is that the DT-CWT is shift invariant. Since texture and texture features (*textons*) can present themselves under any shift, it is important that a transform is shift invariant to these shifts so that features can be identified, analysed and replicated independent of shift. The shift dependency of the DWT was discussed earlier in chapter 3 where it was shown that the energy in each sub-band varied with the shift of the input signal. In contrast, the shift invariance of the DT-CWT implied that the energy in each sub-band remained constant regardless of shift. Phase discontinuities introduced as a result of the shift dependence of the DWT would result in structures not joining up and ultimately an incoherent synthesised texture. In contrast, the shift invariance of the DT-CWT avoids this problem and results in a more coherent realistic synthesised texture.

The block matching process described here generates wavelet coefficients at the coarsest level L of the DT-CWT. The next stage in the algorithm is to synthesise the coefficients at the higher resolution levels $l < L$ aswell as the scaling functions in the coarse level low-pass image. To synthesise these, one of the following methods can be used.

4.3.2.1 Copy

Wavelet coefficients at higher resolution levels $\mathbf{B_s^l}$, $\forall\, l < L$ and scaling coefficients in the coarse level low pass image $\mathbf{G_s}$ are synthesised according to the translated source location of the synthesised wavelet coefficients at the coarse level. Under the quad-tree relationship which the DT-CWT obeys, the wavelet coefficient at location $\mathbf{x}$ will be "parent" to four "child" wavelet coefficients at next higher resolution level, $l-1$. This is shown in Figure 4.8. To illustrate, consider the scenario where the six wavelet coefficients $B_s^L(\mathbf{x})$ at location $\mathbf{x} = [x', y']^T$ at the coarse level L were sourced from the site $\mathbf{p} = [x, y]^T$ in the example sub-band image set $\mathbf{B_e^L}(\mathbf{x})$.

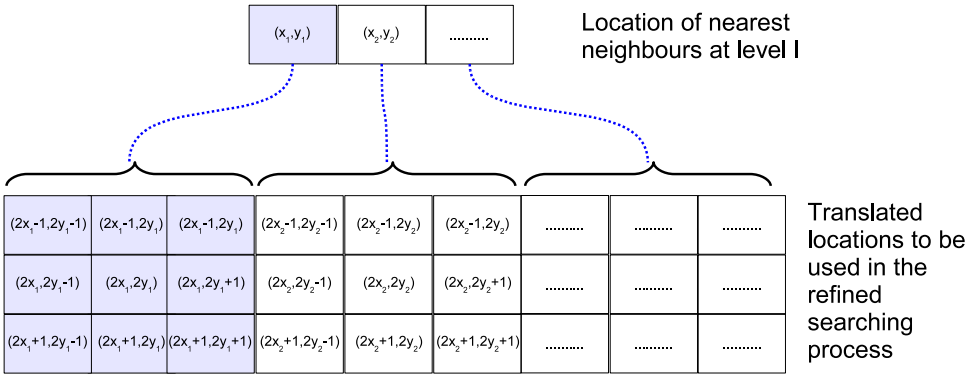


Figure 4.9: *In the refined searching method coefficient locations at level l are transformed and passed down to the higher resolution level $l-1$ where they are used as a guide for performing a reduced nearest neighbourhood search.*

The simplest choice for the four coefficients at $[2x', 2y']^T, [2x'-1, 2y']^T, [2x', 2y'-1]^T$ and $[2x'-1, 2y'-1]^T$ in the preceding finer resolution level $L-1$ is to directly source and copy the wavelet coefficients from locations $[2x, 2y]^T, [2x-1, 2y]^T, [2x, 2y-1]^T$ and $[2x-1, 2y-1]^T$. Similarly, the scaling functions in $\mathbf{G_s}$ follow the same relationship and can be sourced directly from $\mathbf{G_e}$. This process of using the source location of adjacent coarser resolution l wavelet coefficients as a reference for sourcing coefficients at higher resolution levels $l-1$ is repeated at all levels of the wavelet tree.

Using the copy method of syntheising the wavelet tree is efficient given that the computationally intensive block matching is only conducted at the coarse resolution where the data set is much reduced. The source and fetch process by which higher resolution level coefficients are found is fast given that the computational demands are light. On the downside, performing the block matching at the coarse resolution only means that finer detail, which may discriminate between individual neighbourhoods at a higher resolution, is not considered. While finer texture detail is added as the tree is synthesised, there is no guarantee that the higher resolution levels would be synthesised in the same manner if a more refined higher resolution neighbourhood searching process was evoked at each level. In theory this efficiency could compromise the accuracy and sharpness of the synthesised results. However, observable textures contain a large stochastic element and in practice this copy updating method is adequate for synthesising these textures.

4.3.2.2 Refined Searching

At the coarse level L the set of best matching 6D neighbourhoods is sought and their centre wavelet coefficients form an approximation to the pdf of the unknown set of six wavelets to be synthesised. This pdf is sampled and the chosen coefficients are assigned at the specified location. At the preceding higher resolution level $l = L-1$, the refined searching method uses

the information provided by the set of best matching neighbourhoods as a guide for performing a reduced neighbourhood search. By noting the coordinates of the centres of the best matching neighbourhoods at the coarser level and translating these coordinates using the quad-tree relationship, a new set of neighbourhoods is constructed and searched. This new set of neighbourhoods is composed of translated locations taken directly from the coarse resolution search as well as neighbourhoods centred in the 3×3 window around these translated coarse level locations. This idea is illustrated in Figure 4.9. The neighbourhood of the wavelet coefficients to be synthesised is compared to all neighbourhoods centred at each new location. For practical purposes the neighbourhood width is increased as the resolution increases. In this implementation, the neighbourhood width at level l is set to $w_1 + 2(L - l)$, where $w_1 = 5$ is the width at the coarsest resolution. The set of best matching neighbourhoods are found from this reduced neighbourhood search and as before the centre wavelet coefficients form an approximation to the pdf of the wavelet coefficient to be synthesied. This pdf is sampled and the new values assigned to the wavelet coefficients. The pdf at level l or correspondingly the set of centre wavelet coefficients from the best matching neighbourhoods is then passed down to the preceding higher resolution level and used as a basis for a further refined search. This process is repeated at all levels of the transform.

Performing neighbourhood searching at higher resolutions allows finer details in the texture to be considered when finding the best match. This results in a sharper synthesised texture. It should be noted that this increased sharpness comes with extra computational burden. Given the stochastic nature of many real world textures, such an improvement is not visually obvious and so the extra expense associated with the refined searching process is only justified for very deterministic fine detailed textures, e.g. text.

4.3.2.3 Single Resolution Synthesis using Coarse Resolution Searching

This method is only concerned with synthesising pixels in the single resolution image $\mathbf{I_s}$ and uses the nearest neighbourhood results obtained from the coarse level as a basis for performing a reduced neighbourhood search over $\mathbf{I_e}$. This updating method is similar to the refined searching process discussed previously with the exception that rather than synthesising the wavelet tree, only pixels in $\mathbf{I_s}$ are synthesised. For each unknown pixel in $\mathbf{I_s}$, the algorithm finds the parent wavelet coefficient at the coarsest level. This is done using the quad-tree relationship shown earlier. The set of neighbourhoods that are most similar to the neighbourhood of this parent wavelet coefficient is found and the centre locations of these neighbourhoods are then transformed back to image space using the inverse quad-tree relationship. This list of best matches is passed down to the single resolution synthesis process and the neighbourhoods around each of these sites is compared to the neighbourhood of the pixel to be synthesised. In addition, sites in the 5×5 block surrounding each site in the best matches list is also considered. The idea is to obtain a rough estimate of where exactly the best match for $\mathbf{x}$ lies using information obtained at

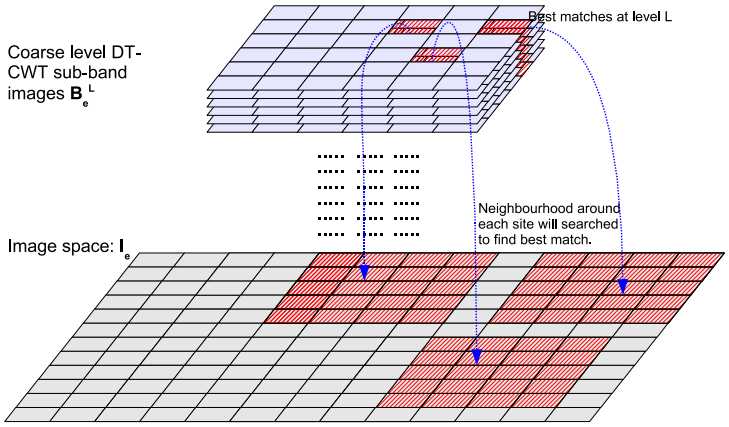


Figure 4.10: *Nearest neighbour searching is performed at the coarse level and the best matches found are used as a basis for a reduced neighbourhood search in image space.*

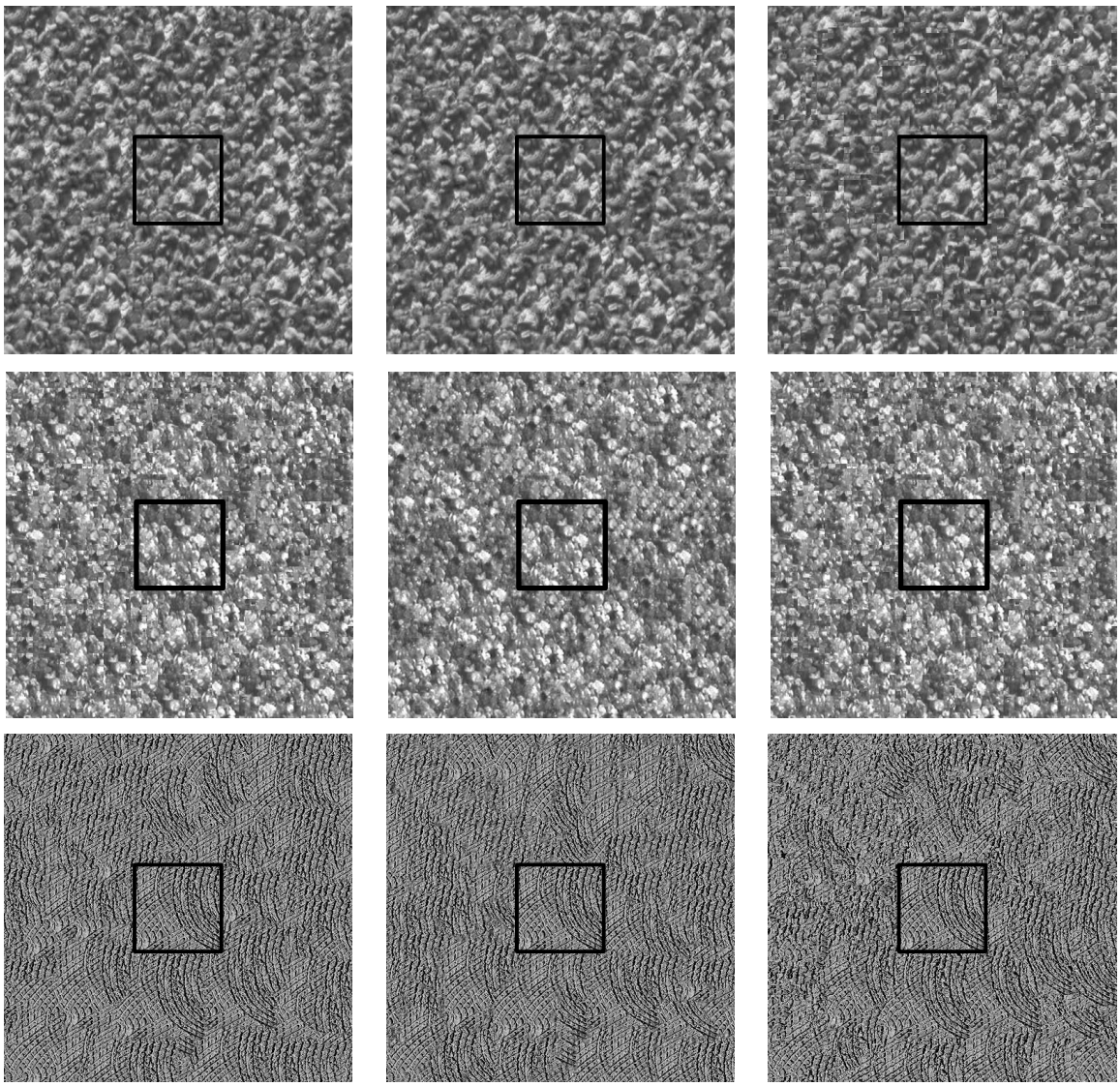
the coarse level. This speeds up the synthesis process by allowing the computationally intensive exhaustive nearest neighbourhood search to be performed at the coarse level. Using this set of possible best matches, a reduced neighbourhood search is conducted at the highest resolution to find the optimal value of the pixel to be synthesised. This process is repeated for all unknown pixels in $\mathbf{I_s}$. Figure 4.10 illustrates how the best matches at the coarse level can be used for a reduced neighbourhood search in image space.

As was the case with the refined searching method, the width of the neighbourhood used at the high resolution image space is increased from that used at the coarse level. In this implementation, the neighbourhood width used in the single resolution searching process is set to $2w_1+1$, where $w_1 = 5$ is the width used at the coarse level. This method has the advantage of reducing the computational burden of the Efros and Leung algorithm. In addition, the growing garbage problem associated with the single resolution algorithm is removed as the pdf remains more stable because of the coarse level initialisation of the searching process.

This adaptation of the DT-CWT TexSyn algorithm should in theory give the sharpest results and still remain more efficient than the Efros and Leung algorithm. However, it is the most expensive of the three variants of the DT-CWT TexSyn algorithm and is only necessary for the most deterministic textures, e.g. text. Overall, it was found that for most observable textures the overall improvement is not visually obvious and similar to the refined searching process, the extra computational burden is unnecessary.

For update methods 1 and 2, once all of the wavelet coefficients have been generated, the sub-band $B_s^l, \forall l = 1, \cdots, L$ and low pass $\mathbf{G_s}$ images are transformed using the inverse DT-CWT and should result in a new texture image $\mathbf{I_s}$ that is perceptually similar to the example texture. For update method 3, pixels in the single resolution are synthesised directly and so there is no need to inverse transform.

To illustrate the visual differences obtained using the three variants of the wavelet algo-



(i) Copy (ii) Refined Searching (iii) Single Resolution using Coarse Searching

Figure 4.11: *Comparing the three variants of the DT-CWT TexSyn algorithm: (i) Copy, (ii) Refined Searching and (iii) Single Resolution Synthesis using Coarse Resolution Searching. The synthesised images are of size* 512×512 *and the original texture "seed" is shown inside the black square and is of size* 128×128*. The algorithm parameters were set to* $L = 3$ *and* $w_1 = 5$*.*

rithm, Figure 4.11 presents three different textures that were synthesised using the (i) *Copy*, (ii) *Refined Searching* and (iii) *Single Resolution Synthesis using Coarse Resolution Searching* implementations. The copy and refined methods give very similar results and so the extra computation expense associated with the refined method makes it inefficient given that the overall improvement in the results in not visually obvious. The synthesised textures obtained using the single resolution method are sharp in patches. However, the overall quality of the result is compromised by the visual presence of artificial edges introduced as a result of the coherent searching process. These edges could be removed by further refining the estimate at the single resolution but this would add to the computational burden.

In general, because variants (ii) and (iii) perform neighbourhood searching and hence pdf estimation at finer resolutions in addition to the coarse level approximation, the resultant synthesised textures should in theory expected to be sharper. This is due to the fact that more high frequency information will be included in the nearest neighbour searching process. However, numerical analysis of the number of suitable neighbourhoods found during the neighbourhood searching stage of the algorithm reveals that in the majority of cases there is only one choice for the best neighbourhood. That is to say once a pixel has been synthesised, it is highly likely that all its spatial neighbours will be sourced from an adjacent translated location in the example texture. This implies that although a refined searching process is evoked, the chances are that the chosen wavelet coefficient will be the same as that obtained using only the coarse level approximation. This observation formed the original motivation behind the Efros and Freeman [67] patch based and the Ashikhmin pixel-based coherent searching [14] algorithms discussed in chapter 2. By extension, the copy method may be considered a type of patch based synthesis.

In-depth visual analysis of the generated results obtained using each of the variants of the wavelet based algorithm indicates that in all three cases the copy method performs as well if not better than the refined and single resolution coherent searching methods. Therefore, because of the increased computational burden associated with methods (ii) and (iii), method (i) is the preferred update method of choice.

4.3.3 Synthesising Colour Texture

The single resolution synthesis and DT-CWT TexSyn algorithms described so far have been aimed at synthesising gray-scale texture images where each pixel $I_e(\mathbf{p})$ and $I_s(\mathbf{x})$ in the example and synthesised images take their value from the set $\lambda = \{0, 1, \cdots, 255\}$. To extend these algorithms to synthesise colour textures as in the case of results shown in Figure 4.11, one of the following two methods can be used.

1. Three Channel Synthesis:

   Unlike gray-scale images which are mapped in a 1D space, colour images are represented in a 3D space. There have been many different colour spaces designed for colour represen-

tation but the most popular is the Red-Green-Blue (RGB) colour space. As a result of the popularity of RGB colour representation, many hardware devices are designed to work within this colour space. In RGB colour space, each pixel is given as $I(\mathbf{x}) = [r, g, b]$ where the r, g and b components take their value from the set $\lambda = \{0, 1, 2, \cdots, 255\}$. To synthesise a pixel in RGB colour space using three channel synthesis, the neighbourhood searching is performed across the three R, G, B channels. That is the neighbourhood around the r, g and b components pixels are constructed and compared to all possible r, g and b neighbourhoods in the sample texture. Neighbourhood similarity is then taken as the combined differences of neighbourhoods in the RGB channels. Similar to the gray-scale synthesis process, the set of best neighbourhoods is found and the centre pixels of these neighbourhoods form an approximation for the pixel to be synthesised.

Performing neighbourhood searching across the three colour channels is computationally expensive and increases the work load of the already slow gray-scale algorithm by a further factor of three. In addition, searching the three colour channels is wasteful given that the human eye is more sensitive to the brightness of a pixel rather than colour changes. The next method described uses this observation to speed up the synthesis process.

2. Luminance then Chrominance Synthesis:

   The sensitivity of the human eye to luminance or brightness fluctuations in an image has been well established [108] and exploited for years through black and white images and video representation. To isolate the luminance components of an image, the image is moved from RGB colour space to a colour space which has a dimension associated with the luminance components. Some of these colour spaces are discussed in chapter 5. In this application, the YUV colour space is chosen because of the linear transformation between RGB and YUV colour spaces. In this colour space, the Y component corresponds to the luminance, the U represents the chroma or more specifically the blue component and V denotes the other chroma red component. The transformation between RGB and YUV is given by,

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \mathbf{C} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \; where, \mathbf{C} = \begin{bmatrix} 0.3 & 0.6 & 0.1 \\ -0.15 & -0.3 & 0.45 \\ 0.4375 & -0.3750 & -0.0625 \end{bmatrix} \tag{4.8}$$

   The inverse relationship is given by,

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \mathbf{C}^{-1} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \tag{4.9}$$

   The range of values for Y,U and V are defined as $Y = 16, \ldots, 255 - 16$ and $U, V = -128, \cdots, 0, \cdots, 128$. Performing texture synthesis in YUV space is very similar to the

gray-scale process discussed earlier. To synthesise the pixel $I_s(\mathbf{x}) = [y, u, v]$, the nearest neighbourhood search is performed in the Y channel only. A best match is found and assuming that the best match is sourced from site $\mathbf{p}$ in the sample image, then the corresponding u and v components of $I_s(\mathbf{x})$ are also sourced from site $\mathbf{p}$ in the U and V channels of the sample image.

This method is the most efficient since the fetch and copy methods used to update U and V are computationally inexpensive. This means that the computational expense associated with colour synthesis is similar to gray-scale synthesis. All DT-CWT TexSyn synthesised textures shown here were generated using the luminance then chrominance method of texture synthesis. The next section will present some of these results.

4.3.4 Synthesised Textures

To demonstrate the robustness and scale independence of the DT-CWT TexSyn algorithm it has been applied to a wide variety of observable textures. The synthesised images presented in this section were all grown from a "seed" composed of the entire example texture image. This "seed" or equivalently the example texture image is located in the centre of each synthesised image and is highlighted by surrounding it with a black square.

Figures 4.12 and 4.13 show synthesised textures obtained using the *copy* variant of the DT-CWT TexSyn algorithm. Synthesised images in Figure 4.12 measure 512×512 pixels and were sourced from a 128×128 pixel example image. The images in Figure 4.13 measure 1024×1024 pixels and were taken from a 256×256 pixel example texture. The example textures used to synthesise the images in Figure 4.12 (i)-(vi) and 4.13 (i)-(iii) fall under the observable category discussed previously in chapter 2. These observable textures contain a mixture of stochastic and deterministic features. The example texture used as the source in Figure 4.13 (iv) is an artificial texture composed of a series of thick black horizontal lines on a white background. The reason for the inclusion of this synthesised result is to illustrate the efficiency and accuracy with which the best match at a coarse level can be translated down to the finer resolution levels by means of the quad-tree relationship. If the transform did not obey such a relationship or the best match at the coarse level was not a close approximation to that at the high resolution then the crisp horizontal lines would be replaced by a more blurry version of the example texture. All results in Figures 4.12 and 4.13 where generated using the following algorithm parameters: $L = 3, w_1 = 5, \epsilon = 0.1$. The DT-CWT decomposition used the bi-orthogonal Antonini $7, 9$ tap and the Qshift 'C' $16, 16$ tap filter sets [120]. The frequency responses of these filters was given earlier in chapter 3.

Close inspection of the images in Figures 4.12 and 4.13 indicate that the new synthesised texture is perceptually similar to the original sample texture and gives the overall impression of being generated from the same underlying process. The wide variety of observable textures generated demonstrates the suitability of the algorithm to both deterministic and stochastic

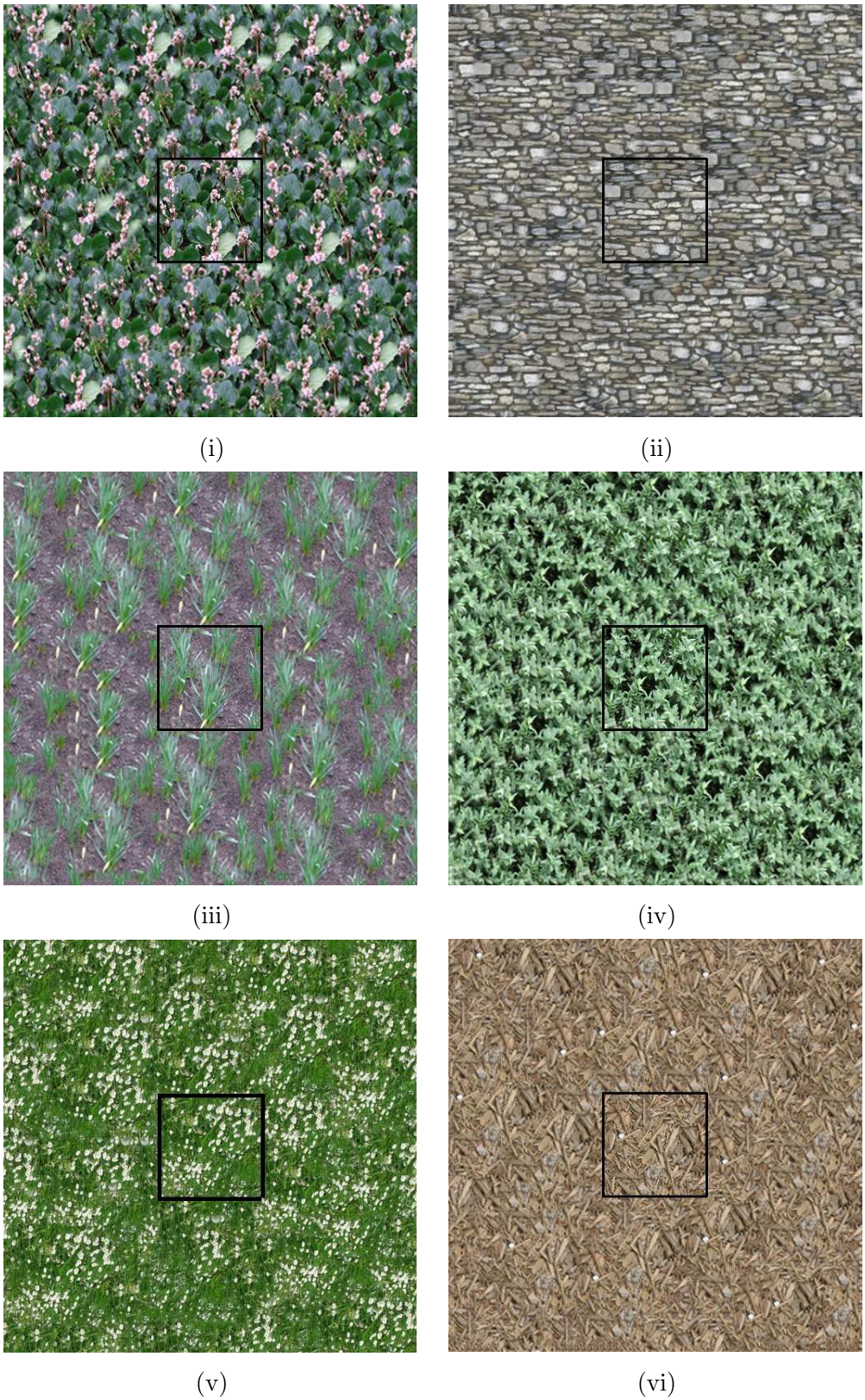


Figure 4.12: *Synthesised textures of size* 512×512 *pixels obtained using the copy variant of the DT-CWT TexSyn algorithm. The example image used as the "seed" to initiate the process is of size* 128×128 *and is shown inside the black square.*

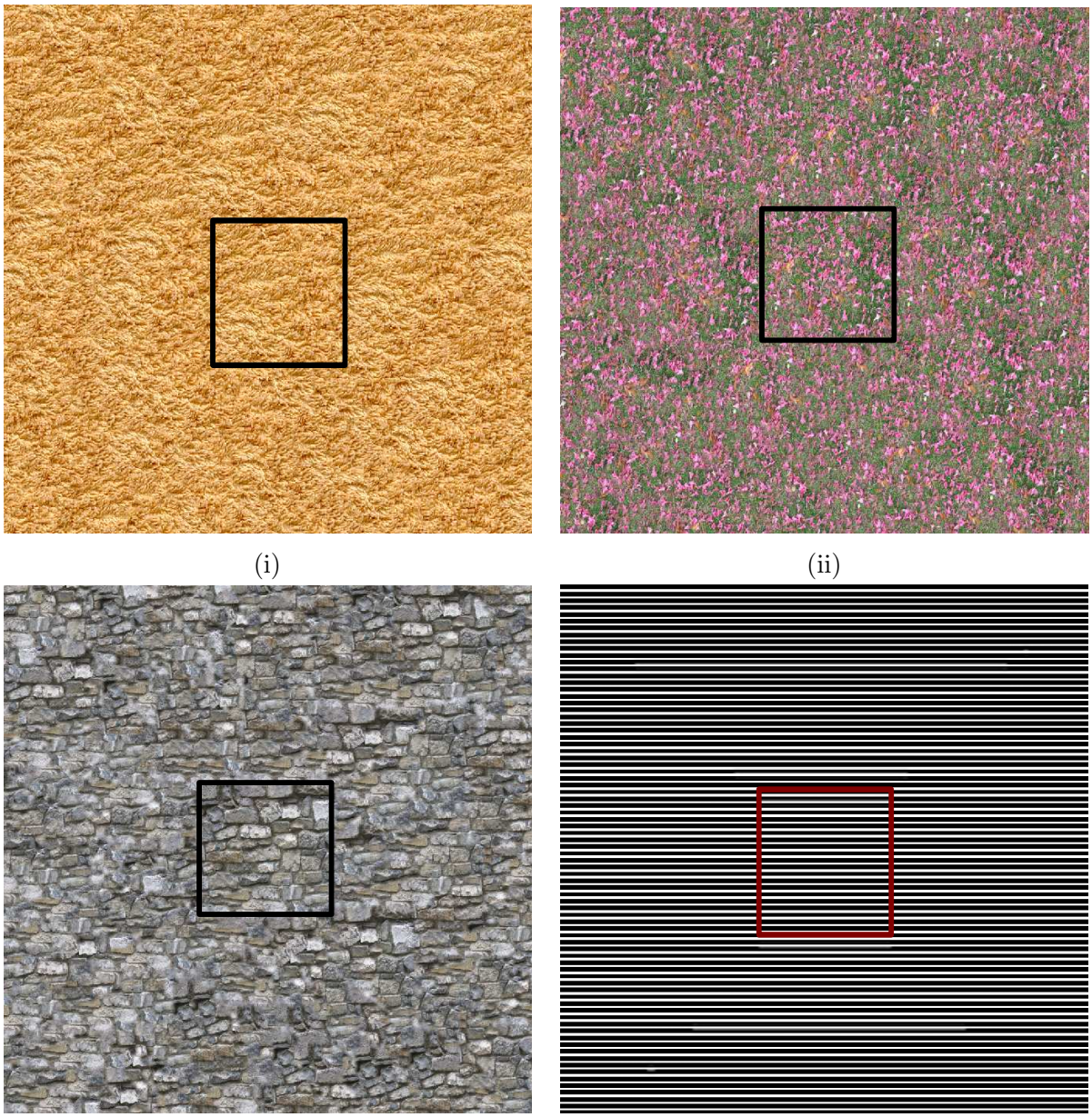


Figure 4.13: *Synthesised textures of size* 1024×1024 *pixels obtained using the copy variant of the DT-CWT TexSyn algorithm. The example image used as the "seed" to initiate the process is of size* 256×256 *and is shown inside the black square.*

Algorithm	One Pixel $I_s(\mathbf{x})$	Whole Image $\mathbf{I_s}$
Efros and Leung synthtesis ($nhood = w$)	$4M_eN_ew^2$	$4N_sM_sM_eN_ew^2$
DT-CWT TexSyn ($nhood = w_1$)	$6\frac{M_eN_e}{2^{2L}}w_1^2 \times 6$	$6\frac{M_sN_sM_eN_e}{2^{4L}}w_1^2 \times 6$

Table 4.2: *Comparing the computational load of the single resolution [68] and the wavelet based algorithms.*

texture types. Furthermore, the scale independence of the algorithm is demonstrated by the fact that the algorithm parameters (neighbourhood size especially) remained constant for all types of texture.

Before comparing the visual quality of the results to previous approaches, the efficiency of the DT-CWT TexSyn algorithm will be highlighted by discussing the computational loads associated with both the DT-CWT TexSyn and Efros and Leung algorithms.

4.3.5 Computational Load

Table 4.2 summarises the computational expense associated with the DT-CWT TexSyn and the Efros and Leung algorithms. The new image to be synthesised and the example image are of size $M_s \times N_s$ and $M_e \times N_e$ pixels respectively. The neighbourhood width used in the Efros and Leung algorithm is denoted w while that used in the wavelet based algorithm is w_1. It is assumed that the *Copy* method of updating the wavelet tree was used in the wavelet based algorithm. The computational load for each algorithm is given in terms of performing neighbourhood searching in one channel only and so it is suitable for both gray-scale and colour (luminance then chromiance) texture generating. The load for performing the DT-CWT decomposition of the example image is $O(M_e \times N_e)$ and similarly to invert the new synthesised wavelet tree the computational load is $O(M_s \times N_s)$ [119]. These loads are not included in the table as they are minuscule when compared to the overall computational expense associated with the synthesis process.

To synthesise a pixel $I_s(\mathbf{x})$ using the Efros and Leung method, it is necessary to perform an entire search of the example image $\mathbf{I_e}$. This searching process involves comparing the neighbourhood of $I_s(\mathbf{x})$ with the $M_e \times N_e$ possible neighbourhoods in the example image. When comparing neighbourhoods, the square of the individual differences in pixel values is calculated ($2w^2$ operations). This difference is multiplied by a $2D$ Gaussian Kernel in order to add more weight to differences at the centre of the neighbourhood (w^2 operations). Finally the differences over the neighbourhood are summed to give the total neighbourhood difference (w^2 operations). Thus, the load for comparing two neighbourhoods is given as $4w^2$ algebraic operations. Since the neighbourhood of the pixel to be synthesised will be compared to all possible neighourhoods

in $\mathbf{I_e}$, the total load for synthesising one pixel is given as $4M_e \times N_e w^2$ operations. To synthesise an entire image $\mathbf{I_s}$ of $M_s \times N_s$ pixels, $4M_s N_s M_e N_e w^2$ operations are needed.

In order to capture the underlying texture characteristics, the neighbourhood size $w \times w$ needs to be big enough to accommodate the largest visual feature present in the texture. Generally, a neighbourhood width of $w \geq 11$ is needed for most observable textures. This large neighbourhood compounds the computational burden of the algorithm resulting in a slow synthesis process.

Conversely, the wavelet based algorithm performs block matching at a low resolution level where the data set to be analysed is reduced considerably over the single resolution equivalent. To synthesise a wavelet coefficient at the coarsest resolution L, its neighbourhood is compared to the set of all possible neighbourhoods in the example sub-band images. This neighbourhood comparison occurs over the six sub-band images where is made up of 6 individual $2D$ neighbourhood searches. When comparing two $2D$ neighbourhoods, the absolute value of individual complex wavelet coefficient differences is calculated ($5w_1^2$ operations). These differences are summed over the entire 2D neighbourhood (w_1^2 operations). This neighbourhood process is undertaken in each of the six sub-band images and the differences at each site are summed together. Therefore the load for comparing two 6D neighbourhood is given as $6w_1^2 \times 6$ algebraic operations. At the coarsest level L, there are $\frac{M_e N_e}{2^{2L}}$ wavelet coefficients to be searched and so to synthesise one wavelet coefficient $6\frac{M_e N_e}{2^{2L}} w_1^2 \times 6$ operations are needed. To synthesise the entire level $6\frac{M_s N_s M_e N_e}{2^{4L}} w_1^2 \times 6$ operations are needed.

Under the *Copy* method, wavelet coefficients at higher resolution levels are synthesised using a copy and fetch process, no additional algebraic operations are needed. It also should be noted that because neighbourhood searching is performed at the coarse level where large features are represented by a smaller number of wavelet coefficients, the neighbourhood size needed is much smaller than the single resolution equivalent. In this application a constant neighbourhood width of $w_1 = 5$ was used and the number of levels used in the DT-CWT decomposition was set to $L = 3$.

In comparing the number of operations needed with each approach, the DT-CWT TexSyn algorithm is more efficient than the Efros and Leung algorithm by a factor of *approximately* $\frac{2^{4L} w^2}{9 w_1^2}$. Given $w = 11$, $w_1 = 5$ and $L = 3$, the DT-CWT TexSyn process is approximately 2203 times faster than the Efros and Leung algorithm. Using a C++ implementation of the algorithms on a computer with an Intel Pentium 2.4GHz processor and 1GB RAM, to synthesise a 256×256 pixel gray-scale image from a 128×128 pixel example image, the wavelet synthesis process took roughly 0.81 seconds. In comparison, the Efros and Leung algorithm took approximately 1276.8 seconds (21.28 minutes) to synthesise an equivalent image. It should be noted also that the "seed" used to initialise the synthesis process was the same in both algorithms and consisted of the entire example image.

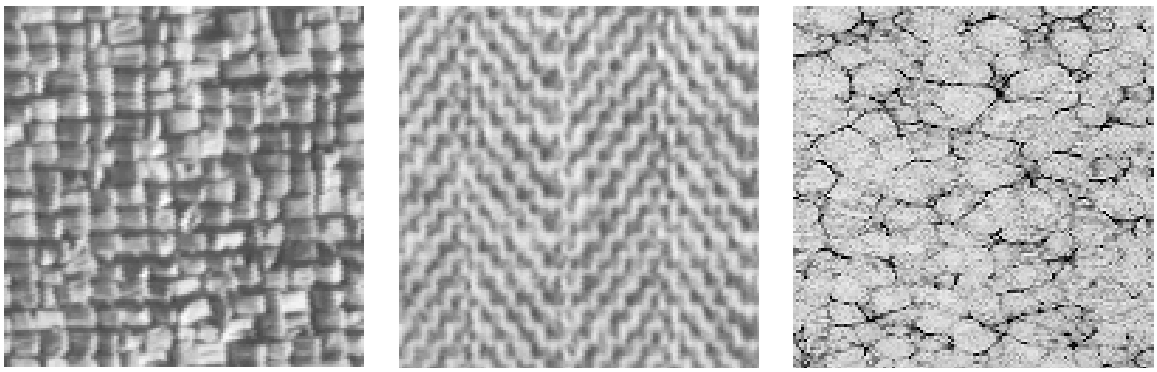


Figure 4.14: *Example texture images from the Brodatz collection [34].*

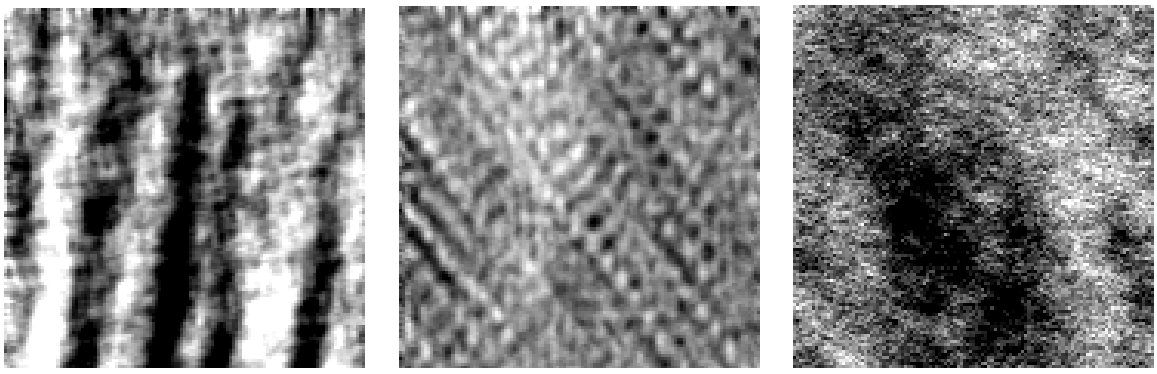
4.4 Comparison to Other Algorithms

To demonstrate the accuracy of the DT-CWT TexSyn algorithm, its synthesised results were compared to those obtained using some of the more saliant previous approaches. These algorithms were discussed in chapter 2. The example texture images used as a source for the synthesis process are taken from the Brodatz album [34] and are shown in Figure 4.14. These images are of size 128×128 pixels and are classified as observable textures. In the taxonomy of texture synthesis algorithms presented earlier in chapter 2, texture synthesis algorithms were classified as being either parametric, non-parametric or patch-based. The comparison given here will follow that classification and present results obtained from algorithms from each of the three categories.

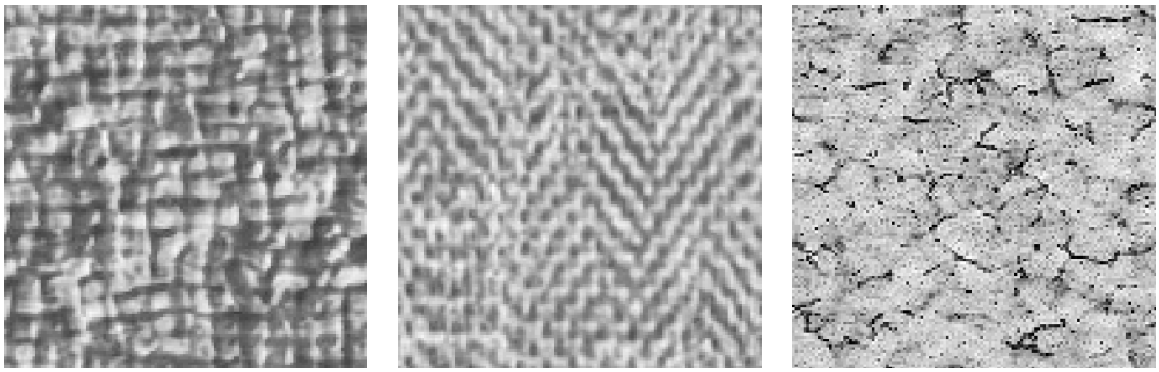
4.4.1 Parametric Results

Synthesised texture obtained using the 2D AR process are shown in Figure 4.4 (i). This algorithm can be classified as single resolution, parametric and MRF based. The 2D process used in this implementation is an adaptation of the *in-painting* algorithm proposed by Kokaram in [123]. Under the 2D AR model, each pixel is given as a linear combination of pixels in the neighbourhood around that pixel plus an excitation or residual error that is assumed Gaussian. The model parameters are made up of coefficients which govern the influence each neighbouring pixel will have on the pixel to be synthesised and the variance of the residual error process. In the implementation used here, a 9×9 causal neighbourhood was used and the residual error at each point was modelled as a Gaussian distribution with mean and variance determined by the residual error calculated from the example image.

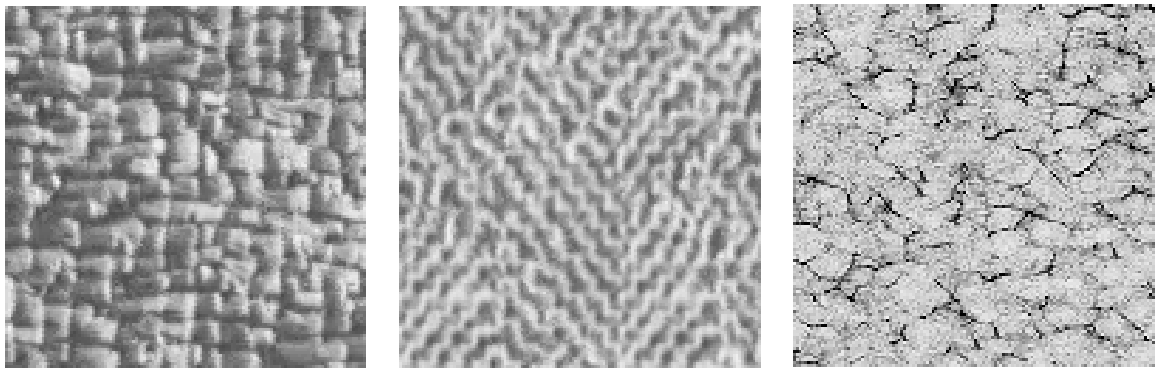
All of the synthesised results obtained using the 2D AR process fail to capture the underlying features of the original texture samples. Reasons for such poor results may be due to the fact that the model order was invariant to input texture. In addition, the residual error was modelled as a Gaussian process and it is widely acknowledged that most images cannot be considered Gaussian. Chellappa and Kashyap [47] offer a more accurate means to approximate the residual



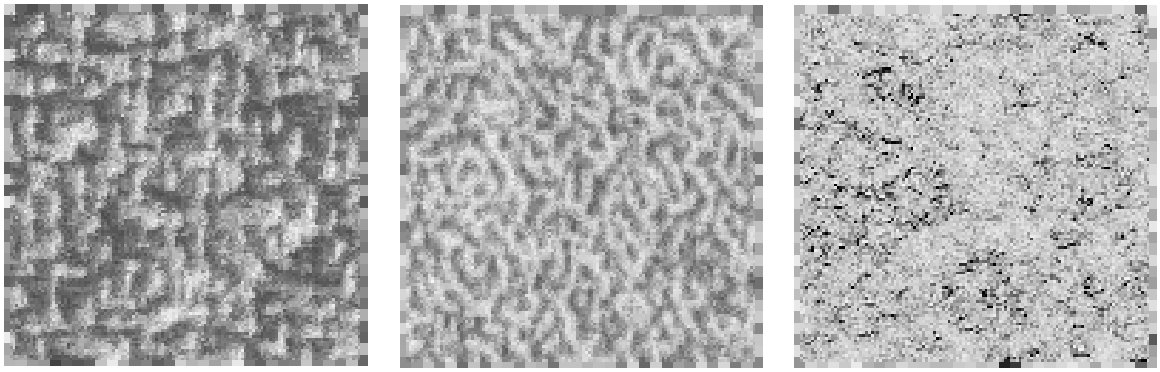
(i) Kokaram [123]



(ii) Portilla and Simoncelli [160]

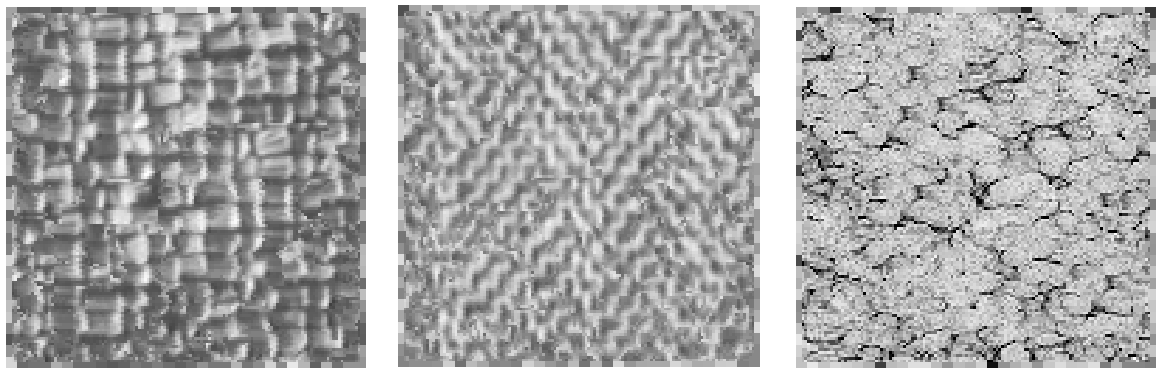


(iii) Efros and Leung [68]

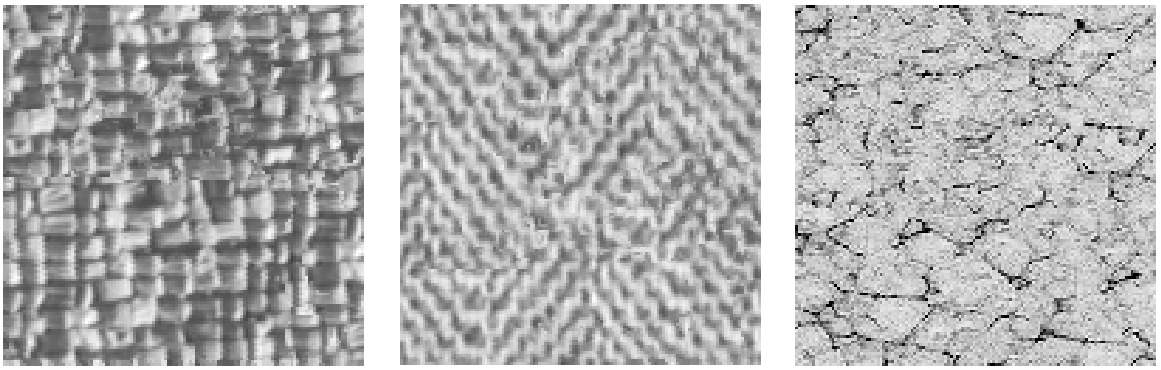


(iv) Wei and Levoy [213]

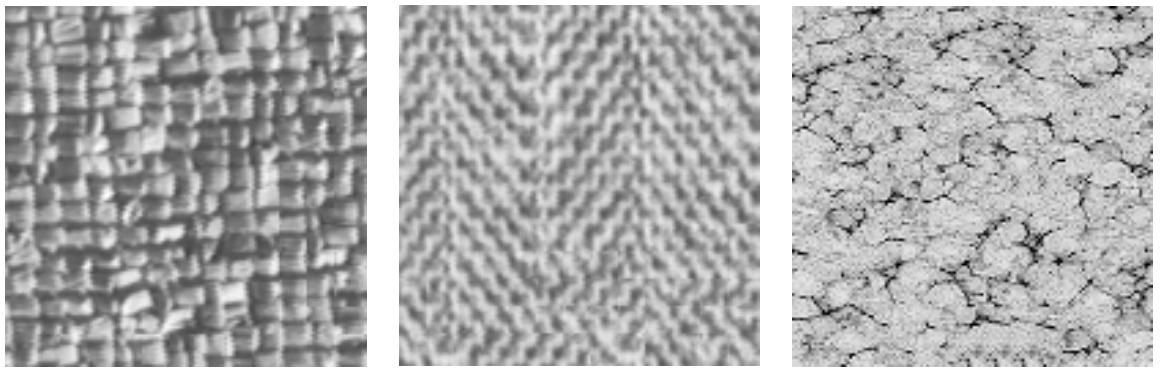
Figure 4.15: *Comparing texture synthesis algorithms (1).*



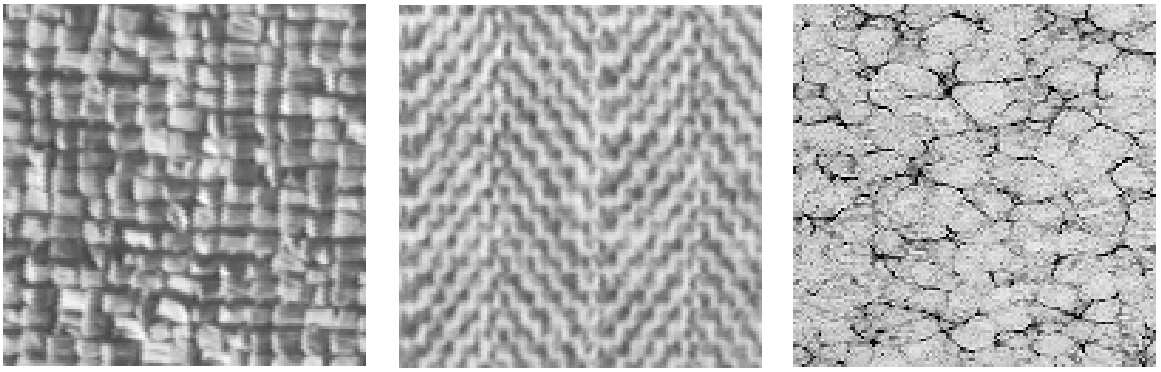
(v) Ashikhmin [14]



(vi) Hertzmann et al. [101]



(vii) Efros and Freeman [67]



(viii) DT-CWT TexSyn

Figure 4.16: *Comparing texture synthesis algorithms (2).*

error and better synthesis results can be found in their paper.

In a process that can be classified as parametric, multi-resolution and MRF based, Portilla and Simoncelli [160] synthesise texture in the wavelet domain using a parametric process. The deterministic and stochastic aspects of texture are captured by a finite set of parameters whose value is dependent on: 1) the local spatial correlation of wavelet coefficients within each sub-band, 2) the local spatial correlation of wavelet coefficient magnitudes, 3) the cross-correlation between coefficient magnitudes at adjacent scales and orientations, and 4) the first few moments of the pixel histogram. Synthesised images obtained using this approach are given in Figure 4.4(ii). The initial impression of these images suggests that both the synthesised and example images come from the same underlying statistical process. The deterministic and stochastic features of the example image are replicated in the synthesised result and overall the synthesised image is very realistic. However, close inspection of each image (especially the middle image) reveal regions where the texture pattern is not well defined and fails to resemble the example image. These regions compromise the accuracy of the result.

4.4.2 Non-Parametric Results

Parametric approaches aim to model and synthesise texture using a definable process and a finite number of parameters estimated over the example image. During the review of existing texture synthesis algorithms, it was found that because of the wide variability of texture behaviour and texture types, it is impossible to define a processes and a list of parameters suitable for all texture. As a result, non-parametric process which offer no such model and rather aim to measure texture statistics using only information taken from the example texture have achieved the most impressive results. Figure 4.4 (iii) and (iv) shows some of the results obtained using some non-parametric approaches.

Figure 4.4 (iii) shows the results obtained using the Efros and Leung [68] algorithm. This algorithm was revolutionary in demonstrating the strength of the non-parametric modelling approach. Their algorithm can be described as non-parametric, single resolution and MRF based. This algorithm was described in detail at the beginning of this chapter and forms the foundation on which the DT-CWT TexSyn algorithm developed as part of this work was based. The synthesised images are realistic and similar to the example image on which they are based. In each case a neighbourhood size of 9×9 was used to generate the results. Similar to Portilla and Simoncelli, the middle texture proves problematic. This is due to the fact that the neighbourhood size used is not big enough to capture the largest feature present in this texture. This scale dependency of the algorithm is its main limitation since the accuracy with which the texture is modelled is very much dependent on the correct interpretation of texture scale. Another large drawback of the approach which was highlighted in section 4.3.5 is the computational burden associated with synthesing realistically sized images.

The next method to be compared is that of Wei and Levoy [213]. The authors state that

this method is not a direct extension of the Efros and Leung algorithm but rather a concurrent process developed around the same time. The Wei and Levoy algorithm can be classified as non-parametric, multi-resolution and MRF based. Texture is synthesised at each level of a Gaussian pyramid using heuristic measurements taken from the equivalent level of the example texture pyramid. The process begins at a coarse resolution and updates each level using information from the previously synthesised coarse resolution and the present level to be synthesised. The neighbourhood used is causal on the present level and non-causal on the coarse level. An illustration of this neighbourhood structure was given earlier in chapter 2.

Synthesised textures obtained using the Wei and Levoy algorithm are shown in Figure 4.4 (iv). A neighbourhood size of 9×9 was used in each case and the Tree Structured Vector Quantisation (TSVQ) [213] implementation of the algorithm was used. This TSVQ reduces the computational burden associated with synthesising each level of the Gaussian pyramid by avoiding an exhaustive nearest neighbour search at each point. The downside of the introduction of TSVQ is that the quality of the synthesised results is decreased. This is evident from the synthesised images in Figure 4.4 (iv), where visual speckle of mis-matched pixel values are rampant throughout the images. In addition, the middle and right most images fail to capture the texture behaviour of the example texture. Similar to Efros and Leung, this failure to capture the texture pattern can be attributed to the scale dependency of the algorithm. Although Wei and Levoy address this scale dependency problem by performing synthesis at different levels or resolutions of a Gaussian pyramid, they use scale as a control and do not attempt to measure it directly.

In an algorithm derived directly from the single resolution version of the Wei and Levoy algorithm, Ashikhmin [14] replaces the computationally expensive nearest neighbour searching process associated with the Wei and Levoy algorithm with an intuitive coherent searching process. This coherent searching process is based on the observation that the source from which a pixel is taken in the example image will be governed by its immediate neighbours. Therefore, it is highly likely that all pixels in the same neighbourhood will be taken from approximately the same location in the example image. So in theory, to synthesise a pixel, just look where its neighbours were sourced from. This method of coherent searching is much faster than performing exhaustive searching.

The Ashikhmin algorithm can be classified as non-parametric, single resolution and MRF based. The results presented in Figure 4.16 (v) were synthesised using a 9×9 causal neighbourhood. The algorithm was originally designed for synthesising natural textures which are largely stochastic by nature. As a result, it performs very well on the right most texture. In the left and middle images the result is not as convincing as the algorithm does suffer from some scale dependency. In addition, the coherent searching seems to have difficulty restarting effectively when a neighbourhood being copied ends (e.g. runs off the end of an image). This problem produces abrupt discontinuities in the form of visual lines between texture patterns.

To avoid the inclusion of artificial lines between texture patterns, Hertzmann et al. [101]

combined the power of exhaustive nearest neighbour searching with that of coherent searching. The result of this combination is that for each pixel to be synthesised, both exhaustive and coherent searching processes are evoked. In general the exhaustive searching will return a better match than the coherent searching process but from observation the coherent match may give a more realistic result since the texture should be coherent by nature. To maintain this coherency a weighting favouring the coherent result is introduced. Similar to Wei and Levoy, Hertzmann et al. perform synthesis at each level of a Gaussian pyramid. The Hertzmann et al. algorithm can be classified as non-parametric, multi-resolution and MRF based.

Synthesised images obtained using an implementation of the Hertzmann et al. algorithm are shown in Figure 4.16 (vi). These results offer an improvement over those obtained using the Ashikhmin approach. Note that there still exists some visual lines and the middle texture image does not accurately resemble the example texture from which it was generated. In addition, the algorithm is slow as performing two searching process for each pixel to be synthesised is cumbersome. To reduce some of the computational cost an Approximate Nearest Neighbour (ANN) [13] searching process can evoked.

4.4.3 Patch Based Results

The third class of synthesis algorithms are patch based approaches. Rather than sourcing and copying individual pixels from the example texture, these approaches work by copying entire patches at a time. Synthesised textures obtained using the Efros and Freeman [67] image quilting algorithm are given in Figure 4.16 (vii). This algorithm can be classified as patch based, single resolution and MRF based. The motivation behind the Efros and Freeman algorithm is similar to that of the Ashikhmin algorithm described earlier. That is, in the majority of cases the pixel to be synthesised and its spatial neighbours will have been sourced from the same location in the example image. It therefore seems logical and more efficient to synthesise entire patches rather than individual pixels. In the Efros and Freeman algorithm, patches are placed in the region to be synthesised and joined to other patches using a minimum distance path. This minimum distance should avoid the introduction of unwanted visual lines at patch boarders. In this implementation a patch size of 12×12 and an overlap of 2×2 pixels was used.

The results obtained using the image quilting algorithm are impressive and give a good representation of the true texture pattern. The texture pattern of the middle image is almost consistent but there is some evidence of the visual lines between individual patches. The synthesis process is fast given that entire patches rather than individual pixels are synthesised at a time. However, there sill exists an element of scale dependency whereby if the wrong patch size is chosen then features which characterise the example texture may not be replicated in the synthesised result.

4.4.4 DT-TexSyn Results

Results obtained using the *copy* variant of DT-CWT TexSyn algorithm are given in Figure 4.16 (viii). In all variants of the algorithm (*copy*, *refined* and *single resolution reduced searching*), the algorithm parameters of number of levels and neighbourhood width remained constant at $L = 3$ levels and $w_1 = 5$ pixels. This implies that the algorithm is scale independent. All three texture images are perceptually similar to the example texture from which they were generated. Comparing the synthesised results to those obtained using previous approaches (i)-(vii), it can be said that the results obtained using the DT-CWT TexSyn process are the most accurate and stable. In addition to the visual quality of the results, the algorithm is the most computationally efficient. This high efficiency is due to the fact that most of the computational intensive neighbourhood searching is performed at the coarse level. At this level the data set is much reduced and so the computational burden associated with the exhaustive searching process is much lighter than the single resolution equivalent.

4.5 Final Comments

This chapter presented a new texture synthesis algorithm, DT-CWT TexSyn. This algorithm can be classified as multi-resolution non-parametric. It builds upon the strengths of previous non-parametric modelling techniques by exploiting the manner in which heuristic measurements taken from a training example image can be used to implicitly model a new texture image. Associated with these previous approaches were the problems of scale dependency and large computational expense. The algorithm proposed here overcomes these difficulties by performing synthesis in the wavelet domain. The wavelet domain provides a natural environment in which to analyse and model the frequency characteristics of texture images. By analysing the texture over different frequency bands and orientations, the dominant features of the texture can be ascertained and replicated in the new image. The new algorithm also has the benefit of being fast given that much of the computational intensive neighbourhood searching is performed at a coarse scale where the data set is much reduced.

The chapter began by presenting the single resolution non-parametric algorithm proposed by Efros and Leung. Using this non-parametric modelling framework as a basis, the new algorithm DT-CWT TexSyn was described in detail. To demonstrate the efficiency with which this new algorithm synthesises texture, its computational expense was compared to that of the Efros and Leung algorithm. The strength and robustness of the DT-CWT TexSyn algorithm was illustrated by the wide range of texture images that were synthesised. Synthesised images were presented and a detailed comparison with some of the more pertinent previous approaches was given. This comparison demonstrated that results obtained using the DT-CWT TexSyn algorithm are a more accurate representation of the true texture pattern of the example image.

This concludes the discussion on texture synthesis. More results of the DT-CWT TexSyn

algorithm can be found in appendix B. The next chapters will focus on the problem of segmentation and discuss how the modelling process inherent with the DT-CWT TexSyn can be adapted and applied within a segmentation framework.

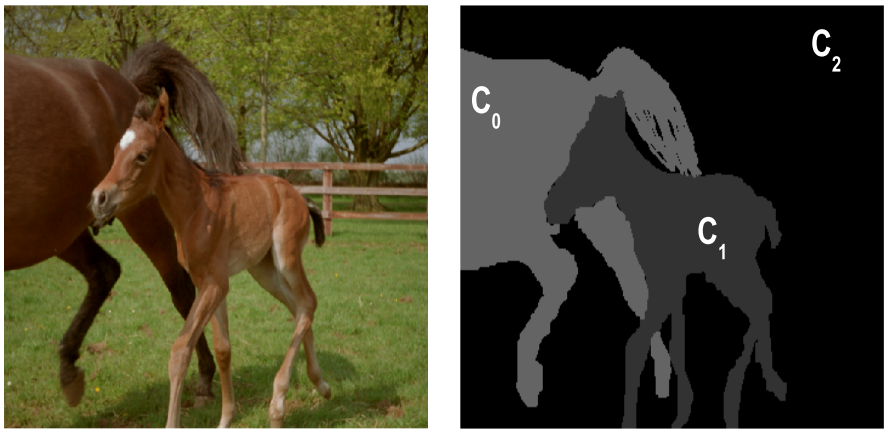
5

An Overview of Image Segmentation

5.1 Introduction

The segmentation of an observed image into an unknown number of distinct and in some way homogeneous regions remains a fundamental issue in low-level image analysis. Image segmentation is the research area that addresses this problem. Segmentation may be interpreted as the process of partitioning an observed image into some non-intersecting regions such that each individual region is homogeneous and the union of no two adjacent regions may be considered homogeneous [152]. The homogeneity constraint that defines a region is a low-level image feature which, over the region, may be considered uniform. There have been many different types of image features used and the success of the segmentation will be very much dependent on identifying the most suitable feature by which regions can be characterised and ultimately the image can be modelled.

Direct applications of a successful automated segmentation algorithm are broad and varied. For example, in medical image analysis [18, 72, 207] accurate and reliable segmentation of medical images such as X-ray and Magnetic Resonance (MR) images can enable the automatic and efficient detection of medical abnormalities such as tumors and lesions. In satellite image analysis [63, 105, 157, 183], segmenting or delineating regions of different texture provides information on the geographical topology of the land. Another large example application of segmentation algorithms is object recognition. By automatically detecting and separating objects in images, large databases can be searched without the need for manual intervention. This is an important issue in content-based retrieval [10, 43, 185] and in image compression [39, 167].



(i) Observed image (intensities) (ii) Segmented image (labels)

Figure 5.1: *Segmentation is the process of separating an observed image into its homogeneous or constituent regions. The homogeneity is captured in the meaning of the scene portrayed by those pixels.*

This chapter will present the image segmentation problem and discuss the complexities associated with estimating a label field that in some way summarises the content of an observed image. The chapter will begin by describing the uncertainties associated with segmentation and the low level features that can be used to characterise regions within a given image. A taxonomy of segmentation algorithms will then be given. In this taxonomy, it was found that many approaches could be unified within the Bayesian framework. The segmentation algorithm that has been developed as part of this work is based within this framework and so the taxonomy of segmentation algorithms will concentrate on Bayesian segmentation algorithms.

5.2 Segmentation: The User's Interpretation of a Scene

To demonstrate the result of an image segmentation process, consider the colour image in Figure 5.1 (i) and an example label field or segmentation given in (ii). At each site in the observed image there exists a RGB (Red-Green-Blue) value used to represent the exact colour of the pixel at that site. At each site in the segmentation image there exists a label indicating to which class the pixel at the equivalent site in the observed image belongs. In this case, the observed image is assumed to be composed of three regions of interest. The labels c_0, c_1 and c_2 are used to denote the three homogeneous regions (*of interest*) present in the observed image. Note that the exact *meaning* of each ensemble of class labels is drawn from the semantic interpretation of the image scene by the user. That is, segmentation may be considered as the division of a given image into separate regions that are either individually or collectively *meaningful* to the user [185].

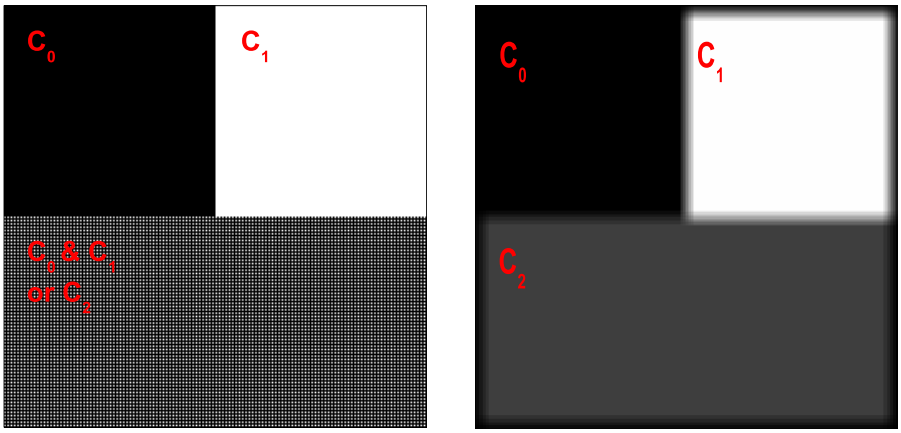
In the case of Figure 5.1, the segmentation highlights each of the two horses, labelled c_0 and c_1, from a background of trees and grass, labelled c_2. To isolate each of the horses in the image, the intensity and texture features associated with each horse is identified. The segmentation

process then finds all pixels in the image that satisfy these colour and texture constraints and assigns them the appropriate label. Any pixels that do not satisfy both the intensity and texture constraints are then labelled as the background region. However, this segmentation is successful if and only if the overall presence of horses are of interest in the picture. To illustrate, consider the following two scenarios. What if the trees, grass and fence regions of the image were of interest? In such circumstances the segmentation would fail given that all of these regions have been collectively assigned the one label c_2. Thus, because they have not been labelled individually they are lost in the overall solution. Another possible circumstance in which the segmentation would be inadequate is if the solution required is one which represents some of the finer details in the image, e.g. the various anatomical areas of each horse. In this case should the segmentation isolate the legs, head, tail of each horse etc? Since each horse has been assigned one overall label, some of the detail present has been lost in the result. To achieve the desired segmentation in each of these two scenarios, additional image features would be needed in order to identify the segments of interest. These two example cases where the segmentation would be inadequate demonstrate that the nature of the segmentation problem is very much determined by the particular application.

5.3 Uncertainty in Image Segmentation

As with many problems in computer vision, while the solution may be visually obvious, it can be difficult to devise a computational algorithm whose performance is comparable to that of the human visual system (HVS) [215]. In reference to the problem of segmentation, one of the major impediments to the development of an accurate automated algorithm has been the tendency to underestimate the complexity of the problem at hand, given that human performance on segmentation is both prodigious and mediated by processes that are primarily subconscious [142].

To demonstrate the uncertainty present in segmentation, consider the gray-scale image given in Figure 5.2 (i). Initial perception of the image insinuates that there exists three distinct homogeneous regions or classes, labelled c_0, c_1 and c_2. Each of these regions is spatially distinct and has associated with it a specific homogeneous constraint. However, closer inspection of each individual region reveals pixels whose gray-level intensity are are more in line with the gray-level intensities of the other regions. To illustrate, in the top left quadrant, pixels coloured black are labelled as belonging to the region labelled c_0. Likewise for the top right quadrant, pixels coloured white are assigned to region labelled c_1. However, consider the pixels in the bottom half of the image. These are coloured black and white respectively. Is this a new region requiring a new label c_2? Or should the black pixels be labelled c_0 and the white pixels be labelled c_1? Intuition and logic suggests that these pixels combine to form a new region which should be assigned a new label c_2. This suggests that segmentation performed by the HVS is not simply a pixel-by-pixel classification of the image but rather some form of local averaging of gray-level intensities is evoked at each site.



(i) High resolution intensity image (ii) Low resolution intensity image

Figure 5.2: *Uncertainty in image segmentation. Image (i) is the original image to be segmented. The uncertainty arises as a result of the trade-off between spatial and class properties. Looking at only spatial intensity values, then image (i) contains two classes c_0 and c_1. On the other hand, if the global properties such as a texture are considered then image (i) is composed of three classes c_0, c_1 and c_2. Image (ii) is the lower resolution version of image (i) obtained by filtering image (i) with a low-pass filter. At a lower resolution the class properties are more obvious and so the image can be seen to be composed of three regions c_0, c_1 and c_2. However, the boundaries are now less well defined. At a lower resolution there exists a trade off between spatial and class properties.*

As a result of this averaging process, there exists an uncertainty as to the representative value of the homogeneous property which defines the region. If gray-level intensity is the feature used to characterise a region then the resulting segmentation would contain only two labels, c_0 and c_1, representing the two distinct regions. However, if texture is the homogeneous constraint which characterises a region then the image is composed of three distinct regions and the segmentation will therefore contain three labels, c_0, c_1 and c_2. To compound the problem further, consider the location of the boundary lines between regions labelled c_0, c_2 and c_1. Close inspection of the image indicates that there exists an ambiguity as to the exact location of the boundary lines. For example looking at the boundary line between the region labelled c_0 and that labelled c_2, it is impossible to determine exactly where the region c_2 begins and where the region c_0 ends. This difficulty is a direct consequence of the local spatial averaging process evoked by the HVS. The result is that in certain locations the boundary lines appear blurred and their exact location cannot be ascertained.

From these two observations, it is clear that two types of uncertainties exist in the segmenting of an image. One is uncertainty about the representative value of the relevant property (e.g image intensity) in each region due to fluctuations in this property. The second uncertainty relates to the exact location of the boundary. The location is not well defined because of the spatial averaging conducted at each point. Given that these uncertainties are observable in a

simple two tone gray-scale image, then it is clear that they are likely to be further compounded when more complex region properties, such as complex texture, shape or colour are introduced. To worsen the situation, the two uncertainties are inversely linked. That is, precision in class properties can only be achieved at the expense of precision in boundary location. The result of this is an inverse relationship between the gain in feature separation between classes and the loss of accuracy in boundary position. To illustrate this inverse relationship consider the image in Figure 5.2 (*ii*). This image represents the lower resolution version of the image in Figure 5.2 obtained by passing image (*i*) through a 10×10 averaging filter. The result is a lower resolution image where some of the high frequency information has been removed. Note that this image is not sub-sampled. In the low resolution image, large homogeneous regions can be detected more easily given that there are less fluctuations in the homogeneous constraint which defines that region. As expected the properties of the regions labelled c_0 and c_1 remain constant given that their intensity values are uniform and thus they contain no high frequency information. The task of labelling the region in the bottom half of the image has now become easier given that the high frequency or fine detail which complicated the process has been removed and this region can now be clearly labelled as c_2.

The downside of the averaging process is that the boundaries between regions c_0, c_1 and c_2 become less well defined. In the original image the boundary between the regions label c_0 and c_1 was sharp and well defined. By definition, boundaries or edges are characterised by their high frequency content, removing this high frequency information will result in the location of the boundary line becoming unclear. This difficulty in determining boundary locations is the compromise which must be undertaken in order to simplify the task of region identification.

Mathematically, the uncertainty in segmentation arises as a consequence of the difficulty in combining the signals of which images are composed and the resulting symbolic descriptions that are the output of the segmentation process. It is found that this uncertainty can be linked to the uncertainty principle of signal theory [153], which arises because of a lack of compatibility between local and global descriptions of a signal. Wilson and Spann document and illustrate this uncertainty clearly and concisely in their book titled *"Image Segmentation and Uncertainty"* [215].

5.3.1 Multi-resolution framework: A means to reduce the uncertainty

Given that this uncertainty is a direct consequence of the relationship between signals that are defined locally and symbols that are defined globally, to reduce the overall uncertainty a compromise is needed between the spatial and class resolutions in the segmented image. Figure 5.2 (*ii*) illustrates how increased class resolution can be achieved by removing high frequency information. Reintroducing high frequency information increases the spatial resolution. The multi-resolution or multi-scale approach to image analysis is an ideal way to achieve this compromise between spatial and class resolutions.

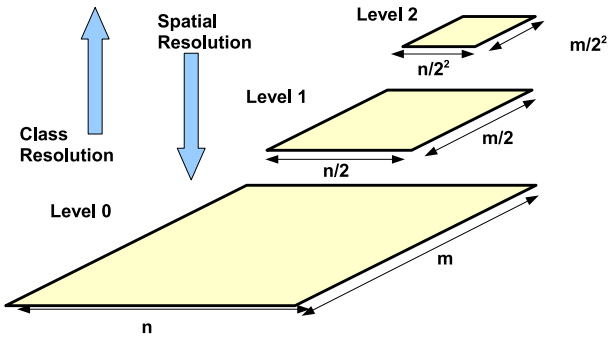


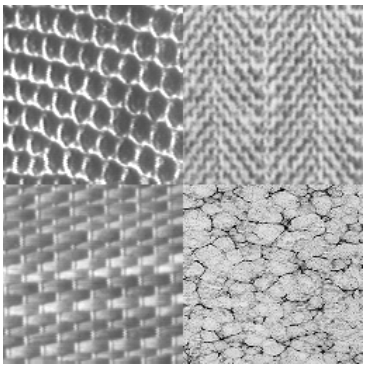
Figure 5.3: 3−*level multi-resolution image pyramid. Level* 0 *represents the original image and images at levels* $k > 0$ *are obtained by low pass filtering and sub-sampling the image at level* $k - 1$ *by a factor of two.*

A multi-resolution representation of an image allows both class and spatial information to be distributed across various scales. This results in the provision of a simple hierarchical framework for detailed image analysis of image information [122]. Typically, a multi-scale representation will reduce the spatial resolution of the image by a factor of two at each scale resulting in a pyramid of successively smaller but similar images. A simplified illustration of a multi-resolution transform is shown in Figure 5.3.

Performing image analysis within a multi-resolution domain allows increased class resolution to be achieved at the higher (coarse resolution) levels of the transform while greater positional resolution can be achieved at the lower (fine resolution) levels. In practice, this implies that at the coarse resolution large features present in the image will dominate because many of the *impulsive* intensities associated with finer features will have been averaged out. By performing the analysis at a coarse level and moving gradually to a high resolution level, large homogeneous regions can be detected more readily considering there are less fluctuations in image intensities. Such coarse-to-fine strategy is widely used in many pattern recognition problems [17, 50, 115, 185, 215] and forms the basis of the approach adopted in this work. The multi-scale transform used in this work is the wavelet transform, which essentially performs image decomposition into low and high frequency components. This splitting of an image into low and high frequency components allows the global class and local positional information to be analysed separately. That is, local positional information is more accurate in the high frequency components while low frequency components can be used to determine the homogeneity constraints of class membership. Since the combination of this positional (local) information and class constraints (global) is the root of the mathematical uncertainty in segmentation, separating these components is advantageous for minimising this uncertainty [114, 215]. Aspects of the wavelet transform were presented in chapter 3.

The next section will discuss some of the low-level image features that are generally used to characterise the regions present in an image. The selection and accurate description of these

(i) Natural scene

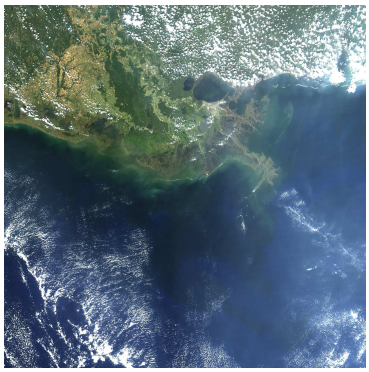


(ii) Texture image

(iii) Satellite (urban) image

(iv) MRI of human head

(v) Sporting image



(vi) Satellite (weather) image

Figure 5.4: *Image feature selection: the success of the segmentation will be determined the correct choice of image feature on which the image can be modelled.*

features is fundamental to the success of segmentation. Once suitable features have been selected, they will form the basis for the underlying image model around which a segmentation algorithm will be built.

5.4 A Note on Features

Figure 5.4 shows six example images to be segmented, each of which is composed of very different content. When performing the task of segmentation on each individual image the HVS automatically identifies and locates a particular low-level image feature or features that can be used to discriminate between the various regions or objects present in the image. This process is largely subconscious and attempting to do this manually is the problem of feature selection. The notion of feature selection is a large issue in content based image retrieval (CBIR). In CBIR systems, it is necessary to find given objects in large databases and to do so, image features are normally collected and labelled. Once a particular feature set has been chosen, a suitable segmentation algorithm will be developed using the chosen feature(s) as a basis for the underlying image model. As will be seen later, if the feature set chosen to model a given image is unsuitable

for the particular image content, then it will not matter how strong or sound the algorithm is, the resultant segmentation will be poor given that the underlying model is not suitable for that particular image content.

To illustrate the wide variability in image behavior and the importance of suitable feature selection, consider again the set of images given in Figure 5.4. Image (i) is based on a natural scene and each region can be characterised by its underlying colour intensity. For example, to isolate the horse, a thresholding technique could be used to find all dark coloured pixels in the image. Pixels satisfying this colour constraint would then be assigned to the same class. However, in the case of the image in (ii), intensity information alone would prove insufficient to define each region. Given that the gray-scale intensity of the pixels in this image all fall within a very close range, a feature which is representative of a group of sites rather than an individual site feature would be more suitable. Texture is such a feature and for the image in (ii), each region could be adequately defined by its texture. How to define and describe texture then becomes an issue. Similarly, in image (iii) the satellite image of the city of Rome, including spatial information in the modeling process would provide a more accurate representation of the underlying image content.

Images (iv), (v) and (vi) are more complicated in content and as a result, a combination of image features will need to be included in each region description. For example, in image (iv), the MRI of a human head, a combination of gray-scale intensity and shape information would provide the most accurate representation of the underlying image content. Similarly, in image (v) and (vi) both colour and texture would be the most descriptive.

As can be seen from just this small set of images, no particular combination of image features may be considered suitable for every possible image type. Therefore, the problem of feature selection arises and the resultant segmentation will be very much dependent on the correct choice of feature. In general, feature selection is made up of two principal issues:

1. What are the best features on which to model the image?

2. How should these features be represented?

A good segmentation will try and use as much information as possible in the underlying modeling process. The most commonly used visual features are colour, texture and geometric information. These will be discussed next.

5.4.1 Colour

Colour is probably the most expressive of all visual features and possesses a high visual impact on the semantic interpretation of image. Given the volume of information obtained from colour, it is surprising to observe that up until recently colour information as a suitable feature to model images has been widely ignored by the computer vision and pattern recognition community [196]. This indifference to colour information could be attributed to the additional computational

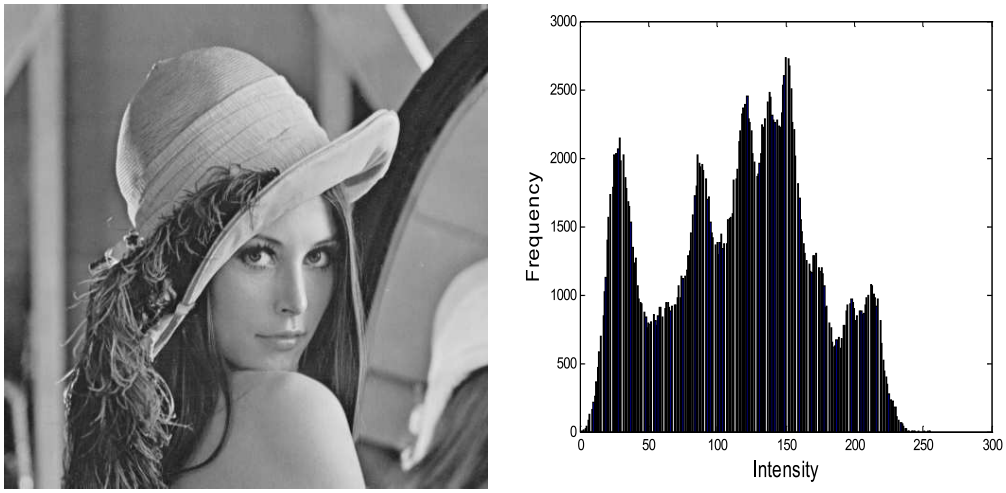
burden and difficulty associated with working within the 3D colour space, as opposed to the 1D luminance (brightness intensity) space which was previously favoured. It has been assumed that many images could be identified and classified using luminance values and thus the extra colour information become redundant. However, recently the additional information provided by colour has been found to strengthen the analysis process and yield better results than approaches using only luminance values [49]. As a result many traditional gray scale algorithms have been extended to work with colour information [49, 79, 91, 196, 219].

The first factor to be considered when using colour information is to decide which colour space to base the information in. The traditional and most commonly used colour space is the Red-Green-Blue (RGB) colour space. The popularity of the RGB colour space partly stems from the manner in which the human visual system (HVS) perceives colour. In the HVS, colour is interpreted using receptors (cones) in the retina and these receptors correspond to the three broad colour channels in the region of red, green and blue. All other possible colours: red, orange, yellow, green, blue, indigo, violet (ROYGBIV) are created through various combinations of the RGB components. Based on this observation most hardware devices are designed to work within the RGB colourspace. The numerical values for the intensities are chosen such that equal increments in value result in approximately equal apparent increases in brightness. However, one of the limitations of the RGB colour space is that it is not perceptually uniform. That is the Euclidean distance between two colour triples does not equate to the human semantic perceived difference between the colours represented by the triples.

There have been a number of colour spaces developed to address the lack of perceptual uniformity evident in the RGB color space format. For example, the Hue Saturation Value (HSV) colour space provides an intuitive representation of colour approximating the manner in which humans perceive and manipulate colour. The transformation from RGB to HSV is non-linear and reversible. The hue (H) represents the dominant spectral component colour in its pure form, i.e. one of ROYGBIV. Adding white to the pure colour changes the colour; the less white, the more saturated the colour is. This corresponds to the saturation (S) component. The value (V) corresponds to the brightness of the colour. Because the HSV format provides good perceptual uniformity through its colour space representation, and its non-linear transform is easily invertible, the HSV colour space format has been widely used in many image analysis applications [192].

An alternative colour space that is popular in content based image retrieval systems is the $L^*a^*b^*$ [24] colour space. Similar to HSV, the transformation from RGB to L^*a^*b is a non-linear and the L^*a^*b colour space has been explicitly designed to achieve perceptual uniformity. The luminance channel L^* represents the intensity content and the a^* and b^* channels capture the colour content from yellow to blue and green to red respectively.

The colour space that will be adopted in this work is the YUV colour space. This colour space was introduced as part of the work on texture synthesis (chapter 4). Similar to the HSV and $L^*a^*b^*$ colour spaces, the YUV colour space exploits the fact that the eye is much



(i) Gray-scale image (ii) Histogram representation

Figure 5.5: *Observed gray scale image (i) and its representative histogram (ii) showing the frequency of the observed pixel values in the image.*

more sensitive to luminance than to colour changes. In YUV, the Y components corresponds to the luminance value and U and V represent the chrominance or colour information. The transformation between RGB and YUV colour spaces is linear and was given earlier in chapter 4. The YUV transformation mapping was widely used in the 1950's in order to enable black and white television sets to view colour television signals. The main advantage of the YUV colour space algorithms is that they are compatible with both gray-scale and colour information. When used in television, another primary advantage was that the YUV format required less bandwidth than the traditional RGB format.

Once the colour space has been decided upon, the next issue to be addressed is to define a suitable measurement that will correctly characterise the colour properties. Undoubtedly, the most commonly used descriptor for colour information is the histogram. Given a discrete colour space defined by some colour axes (e.g. red, green, blue), the colour histogram is obtained by discretising the image colours and counting the number of times each discrete colour occurs in the image array. Figure 5.5 shows a gray-scale image in (i) and its resultant histogram plot in (ii). Once the histogram has been computed, various first-order statistics can be obtained, for example the mean and variance of the image intensities, and the energy and entropy of the image. In a basic segmentation, regions in an image can be located through peaks in the histogram. These peaks can be used to decide a threshold value for each class. Pixels that fall under this threshold will then be assigned to that particular class. Thresholding will be discussed in more detail in section 5.6.

Histograms are invariant to translation and rotation about the viewing axes. In general they change very little under varying scale and occlusion. Swain and Ballard [200] use the colour histogram for object indexing. Their algorithm is designed specifically for content based

retrieval where it is necessary to locate and compare similar objects in large image data sets. Their approach works well but is sensitive to noise interference in the form of illumination changes and quantisation errors. To overcome this, the concept of *Fuzzy Colour Histograms* was proposed in [91]. In contrast to conventional hard colour histograms that assign each pixel to one bin only, fuzzy colour histograms considers the colour similarity information by spreading the membership value of each pixel to all of the histogram bins. A high membership value will indicate that the pixel is likely to belong to that particular intensity bin. This technique draws inspiration from the popular fuzzy clustering algorithms. One large disadvantage of the histogram approach is the high dimensionality generally associated with histograms. Smith and Chang [195] proposed binary colour sets as an alternative to histograms. These binary sets reduce the dimensionality of the representation and also allow the ability to localise colour information spatially within images.

In more complicated images, colour alone is insufficient to define a region and as a result, other features have been defined as a basis to characterise regions. These will be discussed next.

5.4.2 Geometry

To describe regions using spatial based features, metrics based on object area and maximum / minimum shape encompassing the region have been used. For example, many systems use the ellipse to describe region shape. The feature descriptors used could then take the form of the ratio of the length of the major and minor axes or/and the orientation of the major axis with the $x-$axis [43, 104, 194]. Other spatial methods sometimes used include moments, typically in the first (centre of gravity) or second (size) [191]. In general, using spatial features on its own will only work for the simplest of images. A more robust and powerful feature description is based on using texture information.

5.4.3 Texture

The concept of texture was first introduced and discussed earlier in chapters 2 and 4 where the different categories and features of visual texture are discussed. Texture is an area feature that can be defined according to its statistical and structural properties. It is an intrinsic property of most surfaces and objects and therefore forms an important visual feature by which the identification of regions can take place.

There have been many different approaches proposed to extract textural information from images. Review papers on texture feature extraction [94, 166, 209] conclude that because of the wide variability in texture behavior, no one method may be considered suitable for all texture types [208]. Traditionally, the first approaches to texture analysis were statistical based techniques which aimed to capture the spatial distribution of pixel gray level values within the textured image. Haralick [93] proposed the use of second-order pixel statistics of the image by considering interactions between pixels. Pixel pairs are defined according to their relative

displacement and angular spatial relationship. Depending on the displacement chosen, a co-occurrence matrix is formed and texture features such as uniformity of energy, entropy and maximum probability can be extracted from this matrix. Despite the success of this method, this form of texture analysis is very much dependent on the correct choice of neighbourhood over which the analysis takes place. This correct interpretation of image scale is difficult to predicate given the wide variability of texture behavior. In reality, the texture analysis should be scale invariant and thus be suitable for all texture types.

An alternative approach to texture analysis is to model the texture using the Markov Random Field (MRF) model [54, 99, 216]. In particular the Gaussian Markov Random Field model (GMRF) has been widely used in texture analysis [46, 214]. The MRF and its variants form a probabilistic model for a set of variables that interact on a lattice structure. It facilitates the distribution of a single variable at a particular site on the lattice to be conditional on the configuration of a predefined neighbourhood surrounding that site [20]. A comprehensive examination of Markov Random Fields is given in [80].

The size and shape of the neighbourhood which each pixel is dependent on is captured by the order of the model. In practice, computational constraints normally dictate the size of the neighbourhood and for all intensive purposes, it should be be kept small. To illustrate how the MRF can be used for texture analysis, consider the approach proposed by Zhu et al. [220]. Their approach first applies a set of filters to the samples of a given texture. The histograms of the resulting filtered images are then extracted as texture features. These histograms provide estimates of the marginal distributions of the underlying probability distribution, which can be used to describe the given texture. By evoking a maximum entropy principle, a new distribution is generated that has the same marginal distributions as those extracted via the filtering procedure. This maximum entropy distribution provides an approximation of the underlying texture distribution.

More recently and in order to address the computational burden associated with the MRF based techniques, much of the research into texture description has been focused on multi-scale techniques. Of these multi-scale approaches, those based on the wavelet transform have been prevalent. The multi-resolution framework provides the most natural setting for texture analysis given that by definition textures are composed of various frequency components. Multi-resolution analysis exploits this inherent feature of textures and provides a basis on which analysis can be performed at different resolutions. This multi-resolution analysis will allow the more dominant frequencies present in the texture to be determined.

Based on the Discrete Wavelet Transform (DWT) domain Smith and Chang [193] used the mean and variance statistics extracted from each of the wavelet bands as a way to describe the texture. To improve this method and introduce better directional selectivity and shift invariance, the Dual-Tree Complex Wavelet Transform has been used as a basis for texture analysis [60]. Markovian structures have also been included in the multi-resolution modeling framework to capture any form of interaction between wavelet coefficients. In particular for the case of the

wavelet transform, the Hidden Markov Tree (HMT) [50, 55] has been widely used.

A more expensive alternative to the wavelet transform are *Gabor* decompositions. In essence, Gabor filters are Gaussians modulated by complex sinusoids. Each filter has associated with it a set of parameters that determine the location and orientation of the area of the frequency domain which it analyses. In texture applications, banks of these filters are chosen to analyse the particular spatial frequencies and orientations present in the texture [29, 110]. Although Gabor filters have shown considerable success in texture analysis, their widespread use has been hampered by two large inherent limitations associated with their use. Firstly, the filter parameters have to be set intuitively by the analyst and secondly their computational inefficiency in analysing low frequency information [133].

5.4.4 The Symbiosis between Texture Analysis and Synthesis

The structure of texture analysis methods is very similar to the texture synthesis algorithms discussed previously in chapter 2. The goal of a texture synthesis algorithm is to accurately model the sample texture in an attempt to replicate this texture on a much larger scale. Chapter 2 listed some of the more popular approaches that have been developed to address the texture synthesis problem. Algorithms that synthesise one pixel at a time were classified as either parametric or non-parametric based approaches. In comparing both types of algorithms it was found that non-parametric methods offer the most accurate means by which the sample texture can be modelled. In particular the results presented at the end of chapter 4 demonstrated the strength of the wavelet based non-parametric algorithm developed as part of this work. Importantly for image analysis, this algorithm was scale independent and therefore suitable for a wide range of images. Given the success of this technique in relation to texture modeling, it is attractive to consider using such an approach inside a segmentation process. This notion of "example based image segmentation" is powerful and will be explored and expanded upon in chapter 6.

However, in order to gain a more in-depth understanding of the nature of the problem of segmentation, some of the existing segmentation methods will be reviewed. These algorithms all aim to segment a given observed image using low-level information such as the colour intensity, texture and spatial information described previously.

5.5 A Taxonomy of Segmentation Algorithms

As a result of the magnitude of potential applications for an accurate automated segmentation algorithm, there have been many different approaches developed over the years. Due to the sheer volume of literature focused on image segmentation, it would be almost impossible and certainly impractical to provide a full review of all existing segmentation algorithms available. However, this review will focus on those techniques that have had the biggest influence on the

work carried out in this thesis. A more detailed review of existing segmentation algorithms can be found in [64, 96, 109, 149, 152, 184].

Note that because of the wide variability in image types, it has been found that no one approach is suitable for all image types. As a consequence of this general observation, a successful segmentation algorithm will generally combine more than one technique within a suitable framework. As a result, categorising the various segmentation algorithms developed is an arduous task. However, in 2005 the editorial panel for the IEEE Transactions on Image Processing attempted to classify many of the broad areas of image and video segmentation [6]. Nine categories were defined in an attempt to create a "people friendly" classification. These categories are: (i) edge or colour segmentation, (ii) texture segmentation, (iii) active-contour and level-set based methods, (iv) morphological-based methods, (v) clustering-based methods, (vi) model-fitting-based methods, (vi) statistical methods, (vii) video object segmentation and tracking, (viii) video shot / scene segmentation and (ix) other.

For the purposes of this review, algorithms will be categorised based on the dominant processing strategy employed. Thus, algorithms will be categorised under the headings: *Thresholding*, *Edge detection*, *Region growing*, *Clustering* and *Bayesian Framework* methods.

5.6 Thresholding

Thresholding is one of the oldest, simplest and most popular techniques for image segmentation [152]. It is generally based on colour or gray scale intensity features and be performed on global information (e.g. gray level intensity histogram of the image) or it can be performed on local information (e.g. co-occurrence matrix). In histogram thresholding, dominant regions in the image are characterised by crests in the histogram of image intensities. These crests are used to determine a suitable threshold intensity value for class membership. An example of an image histogram is shown in Figure 5.5. An example of an algorithm based on thresholding is that proposed by Nakagawa et al. [147]. Here, the authors assume that the object and background populations are distributed normally with distinct means and variances. The optimum thresholding for assignment of class labels is then chosen by minimising the total misclassification error.

While simple to implement, the selection of an optimum threshold remains a difficult task. In addition, the lack of spatial information present in the estimation means that thresholding methods have limited use. Saha et al. [181] addressed this by proposing a thresholding method that accounts for both intensity-based class uncertainty (histogram based property) and region homogeneity (image morphology based property). Their method included a scale-based formulation for region homogeneity computation. At any threshold, intensity based class uncertainty is computed by fitting a Gaussian to the intensity distribution of each of the regions to be segmented. Based on the observation that pixels with high uncertainty are located around region boundaries, they define a thresholding energy criteria based on class-uncertainty and region ho-

(i) Gray-scale image (ii) Edge field

Figure 5.6: *Observed gray scale image* (i) *and its corresponding edge field* (ii).

mogeneity such that, at any image location a high energy is created when both class uncertainty and region homogeneity are high or both are low. The optimum threshold value is that which corresponds to the minimum energy. This method was applied to medical image segmentation.

Overall the need to include other image features in the modeling process has meant that thresholding techniques in themselves have limited use. However, for simple images thresholding provides an effective and efficient means to obtain a basic segmentation.

5.7 Edge Detection

In any image to be segmented, the various regions of interest are assumed to be homogeneous and separated by well defined distinct boundaries. These boundaries may be considered as discontinuities in the image and edge detection is concerned with locating and identifying these discontinuities within the image. Edge detection approaches may be broadly divided into *local* and *global* techniques. Local techniques are based on identifying edge points using only information in the local spatial neighbourhood around the point of interest. These methods approach segmentation from a position perspective using high frequency information to determine where edges occur. On the other hand, global techniques are based on first finding the homogeneous regions within the image and then calculating where the boundaries lie between individual regions. These methods lead to a global energy minimisation problem and will be discussed further in section 5.10.

One approach to local edge detection is based on identifying the locations of discontinuities by convolving the image with an edge operator. The operator should provide maximum response in areas of the image where the intensity is rapidly changing. Supplementary processing steps are then introduced to connect these edges into edge chains that correspond better with borders

in the image [84]. Figure 5.6 gives the result obtained from an edge detection process using the Wilson and Spann edge detector [215]. Many forms of detectors have been developed and in general it can be said that methods can be characterised by the strategy used in the final border construction and the nature of the prior information used [114].

One of the simplest forms of edge detection is based on measuring gradients. For example, the Sobel and Prewitt [84] operators are based on the first derivative and use a 3×3 convolution mask to approximate the absolute value of the gradient at each site in the image. Canny [42] developed a similar approach where the first derivate of a Gaussian was used to locate edges using the maximum gradient magnitude. To reduce the effect of noise on the segmentation, the raw image is first convolved with a Gaussian mask. An edge in an image may point in a variety of directions, so the Canny algorithm uses 4 masks to detect horizontal, vertical and diagonal edges. The results of convolving the original image with each of these masks is stored and for each pixel, the largest result and its direction of the mask which produced the edge is noted. A map of intensity and direction gradients are thus produced. The algorithm then uses thresholding to determine which points are edges by following edge lines through the image. Marr and Hildreth [142, 143] found that improved results can be achieved using the second derivative of the image. These Laplacian operators locate zeros based on the rate of zero crossing in the second derivative.

By definition, an edge is characterised by an abrupt change in intensity. Thus they contain significant energy at high spatial frequencies. Unfortunately, this is also a characteristic of noise. As a result, distinguishing between the presence of an edge and that of noise is difficult at times. Rakesh et al. [165] proposed a statistical approach to local edge detection which aims to be more robust to noise. Their methodology is based on the following steps. Initially, the image surface is extracted using a bivariate smoother. The gradient vector at each site is calculated and variation in the gradient vector at each site is estimated using the variance covariance matrix. The covariance is then used to standardise the gradient vector at each site. The resultant statistic is used to extract regions in the image and edge points are found by locating sites with significantly large magnitudes. This algorithm is robust and works well on a large range of images using a fixed set of parameters. The statistical nature of the analysis makes it robust to noise.

In general, because edge detection algorithms approach the segmentation problem from a local positional perspective only, there is no class property information included in the solution. The result of this indifference to class property information can mean that the accuracy of the segmentation is compromised. As with all segmentation techniques, the uncertainty can be reduced by introducing a multi-resolution transform into the framework but this is at the expense of increased difficulty in detecting edges at coarser resolutions.

5.8 Region Growing

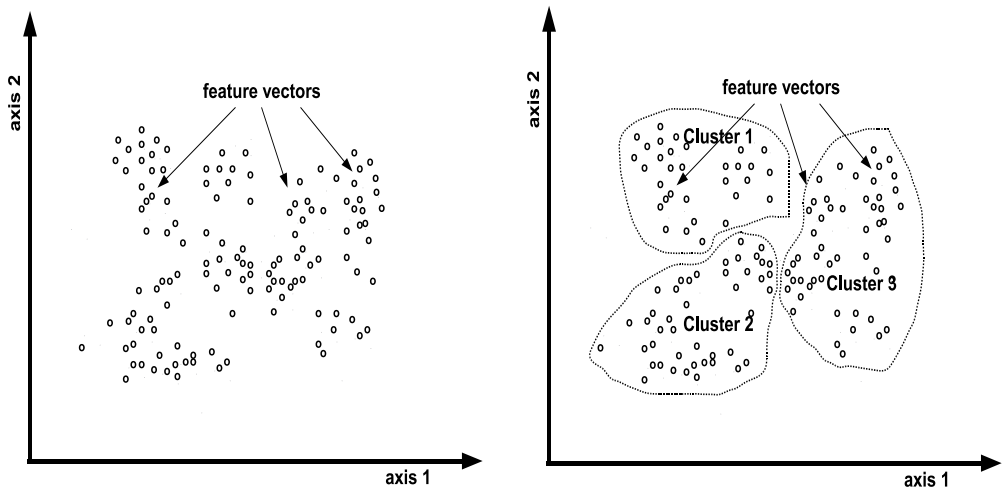
The region-based approach combines the features of both thresholding and edge detection and thus takes class and positional information into account in the estimation of the segmentation. Regions are generally "grown" based on some particular homogeneity criteria, for example colour intensity, texture or shape. The most natural method of region growing is to initialise a region at pixel level by planting a "seed". Adjacent pixels are then sampled and assigned the region class if their particular feature vector satisfies the particular homogeneity predicate which defines the region. The feature vector will be based on a description of the particular low-level image features on which the image is modelled. A method for evaluating the similarity measure between regions satisfying some weak constraints is given by Fiorio et al. [71]. Here they give a generic predicate for determining whether two regions are to be merged. The predicate is composed of six features and can be applied to any region growing algorithm.

An alternative type of region based algorithm was suggested by Buecher [36]. It is based on morphological operations drawn from a topographic analogy of the image. The idea is to imagine the gray-level intensity of the image to be segmented as a topographic relief, where minima can be identified and "pierced". The entire relief is immersed in water so that areas adjacent to the piercing points become flooded. As the relief goes down, flooded areas tend to merge. This merging can be prevented by raising infinitely tall dams along the "watershed" lines. When finished, the resulting network of dams defines the watershed of the image. These non-intersecting areas or *catchment basins* form the segmentation of the image [114]. The watershed approach was originally proposed for gray-scale images [36, 145] but has been extended to work for colour information [156] and texture information [102]. A full description of the watershed approach and some of the variants of the watershed approach can be found in [170]. As with other segmentation techniques, the watershed algorithm has also been extended to work within a multi-scale environment [102].

In general, because region based algorithms use a non-linear optimisation approach, both the final solution and the number of iterations required to obtain such a solution are highly dependent on the initial conditions. There is no guarantee that a given image would be segmented in the same way by the same algorithm with two different sets of initialisation points.

5.9 Clustering

Clustering techniques are based on the classification of similar objects into different groups. More precisely, clustering may be considered as the partitioning of a given data set into subsets "*clusters*", so that the data in each subset (ideally) share some common homogeneity predicate - often proximity according to some defined distance measure. Clustering takes place in feature space and the chosen feature space is dependent on the low-level features which are used to model the underlying intensity image. These features were described previously and are normally



(i) 2D Feature space before clustering (ii) 2D Feature space after clustering

Figure 5.7: *2D feature space showing the position of image feature vectors (i) and an example clustering (ii).*

composed of colour intensity or/and texture or/and spatial information. If intensity information is used alone then clustering is performed in image space.

Figure 5.7 illustrates a 2D feature space (left) with the feature vectors spread over the domain. After clustering has been performed, feature vectors are not moved but rather a cluster boundary placed over each set of features associated with that cluster. In general, clustering algorithms may be classified into one of four groups: (i) *exclusive*, (ii) *overlapping*, (iii) *hierarchical* and (iv) *probabilistic* clustering.

5.9.1 Exclusive Clustering

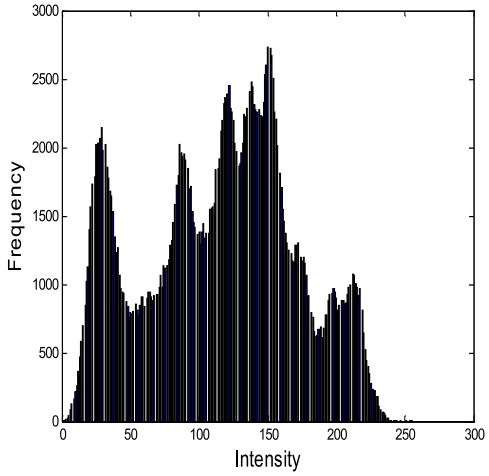
Exclusive clustering or hard clustering implies that data are grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it cannot be included in another cluster. One of the most popular hard clustering algorithms is the $K-$means algorithm [138]. $K-$means clustering is a simple and easy to implement technique for classifying a given data set into a certain number of clusters (assume K clusters) fixed *a priori*.

The algorithm is developed based on an iterative minimisation of the following objective function given in equation 5.1.

$$U(\mathbf{f}) = \sum_{i=1}^{K} \sum_{j=1}^{n} D(\mathbf{f}_j^{(i)}, \mathbf{u}_i) \tag{5.1}$$

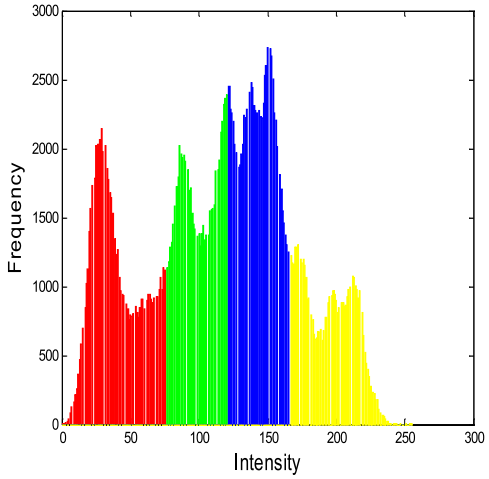
where $\mathbf{f} = \{\mathbf{f_1}, \mathbf{f_2}, \cdots, \mathbf{f_n}\}$ represents the n feature vectors representing the image to be clustered, $\mathbf{f_j^{(i)}}$ indicates that the feature vector $\mathbf{f}_j$ has been assigned to cluster i, $\mathbf{u} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u_K}\}$ denotes the K cluster centres and $D(\mathbf{f_j}^{(i)}, \mathbf{u}_i)$ is the chosen distance measure between the clus-

(i) Gray-scale image



(ii) Feature space (grayscale histogram)

(iii) Clustered intensity values



(iv) Feature space showing cluster locations.

Figure 5.8: *K-means clustering. (i) Gray-scale image to be clustered, (ii) its gray-level values plotted in feature space in the form of an image histogram, (iii) the segmentation obtained using $K-$means clustering with $K = 4$, and (iv) the label map transfered to the original histogram illustrating which intensity values were assigned to each cluster.*

ter centre $\mathbf{u_i}$ and the feature vector $\mathbf{f_j^{(i)}}$. Normally, the L_2 Euclidean distance is used, i.e. $D(\mathbf{f}_j^{(i)}, \mathbf{u}_i) = ||\mathbf{f}_j^{(i)} - \mathbf{u}_i||^2$.

The $K-$means algorithm begins by intuitively initialising the centroid value of each of the K cluster centres. In step 1 each pixel in the image is assigned to one of the clusters based on a minimum distance measure from the data point to the cluster centre. When no point is pending, the first step is completed. In step 2 the cluster centroids are recalculated based on the assigned data points from step 1. Step 1 and 2 are repeated with the centroids of each cluster centre changing their location at each iteration. The algorithm terminates when the centroids remain static between iterations or some maximum number of iterations has been obtained. Figure 5.8 (iii) illustrates the segmentation obtained using the $K-$means clustering algorithm on the gray scale intensity values of the image in (i). The $K = 4$ classes in the image are highlighted by the different colours. The original feature space in the form an image histogram is shown in (ii) and the location of the clusters is shown in this feature space.

While the $K-$means algorithm is simple to implement and generates impressive results even in its most basic form, it suffers from some limitations that have hampered its widespread use. These limitations include, (i) the number of clusters K must be fixed *a priori* making the algorithm unsuitable for unsupervised segmentation, (ii) the overall result of the algorithm is very much dependent on the initialisation position of the cluster centres, (iii) the algorithm does not include any positional information in its estimation of the segmentation and (iv) the algorithm is computationally expensive and has difficulty in handling irregular spaced clusters. Alasbti [11] improved the efficiency of the $K-$means approach by introducing a $K-d$ structure that is suitable for fast pattern matching. This reduces the computational burden without compromising the accuracy of the results.

5.9.2 Fuzzy Clustering

Overlapping or fuzzy clustering methods use fuzzy sets to cluster data, so that each data point may belong to one or more clusters with different degrees of membership. In this case, data will be associated with an appropriate membership value. The fuzzy $c-$means (FCM) algorithm [22, 66] is the most widely used example of an overlapping clustering method. The algorithm is developed based on an iterative minimisation of the following objective function:

$$U(\mathbf{g}, \mathbf{f}) = \sum_{i=1}^{c} \sum_{j=1}^{n} g_{ij}^m D(\mathbf{f_j}^{(i)}, \mathbf{u}_i), \quad \forall \ 1 \leq m < \infty \tag{5.2}$$

where m is any real number greater than 1, $\mathbf{f}$ and $\mathbf{u}$ are as before for the $K-$means algorithm, c is the number of cluster centres fixed *a priori* and $\mathbf{g} = [g_{ij}]$ is a membership function which determines the strength by which each feature vector is associated with each cluster. $\mathbf{g}$ is a $c \times n$ matrix, where n is the number of feature vectors and g_{ik} is the i^{th} membership value of the k^{th} feature vector $\mathbf{f_j}$. $\mathbf{g}$ must satisfy the following conditions: $0 \leq g_{ij} \leq 1$, $\sum_{i=1}^{c} g_{ik} = 1$; and

$0 \leq \sum_{j=1}^{n} g_{ij} \leq n \forall \ i = 1, 2, \cdots, c; \ \ j = 1, 2, \cdots, n$. The fuzzy partitioning process is repeated for a finite number of iterations. A general termination criterion is when the distance between cluster centres at successive iterations is less than 1.

The FCM algorithm has proved popular in the literature but again its overall use has been hindered by the need to specify the number of clusters (c) *a priori*. This makes the algorithm unsuitable for unsupervised segmentation. In addition, similar to the $K-$means approach, the success of the algorithm is highly dependent on the initial location of the cluster centres. Initialising the algorithm with incorrect or unsuitable cluster centres can result in a failure to converge.

5.9.3 Hierarchical Clustering

The third type of clustering algorithm falls under the heading of hierarchical clustering [112]. This type of clustering is based on the union between the two nearest clusters. Algorithms are initialised by setting every datum as a cluster. At each iteration, clusters are merged and the algorithm ends when all the clusters have merged into one. Of course there is no point in having the N items grouped in a single cluster but, the output of the algorithm is a hierarchical tree so if k clusters are required, it can be fetched from the output tree.

5.9.4 Probabilistic Clustering

Probabilistic clustering is a model-based approach that consists of introducing certain models for clusters and attempting to optimize the fit between the observed data and the chosen model. In practice, each cluster may be mathematically represented by a parametric or non-parametric distribution and the entire data set is therefore modelled by a mixture of these distributions. The parametric approach is generally based on the assumption that the underlying data density can be modelled using some definable process. For example, Carson et al. [44] use a mixture of k Gaussian densities modelled in an 8D feature space. The feature space is composed of a mixture of components from colour, texture and positional features. By estimating and refining the parameters associated with the Gaussian distributions, data can be partitioned based on their proximity to the nearest cluster space. The Expectation-Maximisation (EM) algorithm [57] is then used to determine the maximum likelihood parameter estimates for each Gaussian in the combined feature space.

Probabilistic clustering algorithms can generate accurate segmentations of most image types. However, the number of clusters still needs to be specified *a priori* thus implying that the algorithm cannot be used for unsupervised type problems. In addition, it is found that many real world signals are not perfectly Gaussian and making such an assumption compromises the accuracy of the result.

Overall, clustering methods work well for segmentation and this fact is supported by the volume of literature focused on clustering methods. However, clustering suffers from a number of

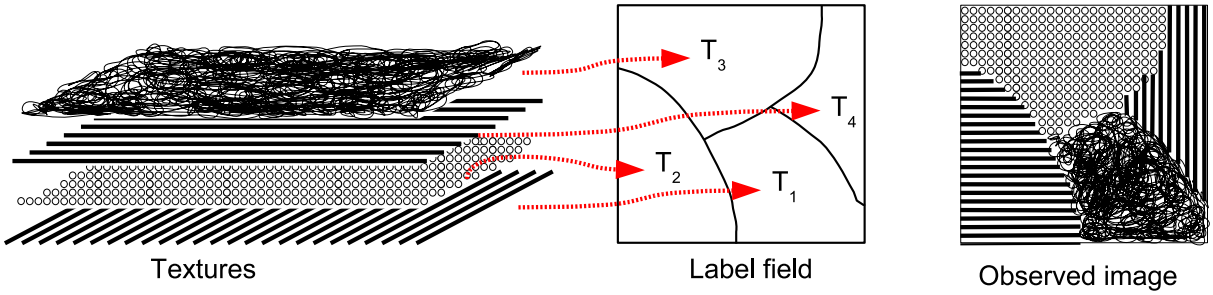


Figure 5.9: *An observed image may be interpreted as a mixture of textures, which are combined in a specific topology. The goal of the segmentation is to characterise and identify each of these texture regions and estimate where they occur in the observed image.*

limitations which have hampered its widespread use. Firstly, many of the clustering algorithms developed are suitable for semi-unsupervised segmentation only, thereby requiring the number of regions present in the image to be specified *a priori*. Another disadvantage is that in order to attain accurate representation of the image, a mixture of colour, textural and spatial information is needed. This results in high dimensionality of the feature space and overall increased computational burden. Finally, because of the wide variability in image behavior it is impossible to define a feature space that will be suitable for all image types. Therefore, a given clustering algorithm could work well for one image type and completely fail for a different image type.

The next category of algorithms is more robust and considered suitable for all image types. This class of algorithms can be derived from a Bayesian framework and assume that the underlying image model can be specified as a Markov Random Field model. The Bayesian framework introduces some kind of stratification of segmentation processes and will be presented next.

5.10 The Bayesian Segmentation Framework

Let $\mathbf{I}$ denote the observed image to be segmented and $\mathbf{L}$ denote the segmentation to be estimated or the label field. Both $\mathbf{I}$ and $\mathbf{L}$ are defined on an identical rectangular $M \times N$ lattice $\mathbf{X}$. Each site in the lattice can be indexed using the spatial coordinate vector $\mathbf{x}$, such that $\mathbf{x} = [x, y]^T$. The goal of the segmentation is to assign to each pixel $I(\mathbf{x})$ in $\mathbf{I}$ a label $L(\mathbf{x})$, indicating to which region or class that pixel belongs. Each class label $L(\mathbf{x})$ is taken from the set $\lambda = \{c_1, c_2, \cdots, c_K\}$, where K is the number of class labels present in $\mathbf{L}$.

Each of the K regions in the observed image has a particular homogeneous constraint which will be used to characterise that particular region. From the discussion on image features (section 5.4), texture is the most powerful and robust feature by which to characterise a region. This is because by definition texture is composed of a mixture of intensity and spatial information. Using texture as a region identifier implies that each region has a distinct texture pattern associated with it. This idea is shown in Figure 5.9 where the observed image is interpreted as a mixture

of textures. Let $\mathbf{T}_{\mathbf{L(x)}}$ denote any texture associated with the class label $L(\mathbf{x})$. Each pixel in the observed image can then be given as,

$$I(\mathbf{x}) = T_{L(\mathbf{x})}(\mathbf{x}) \tag{5.3}$$

where $T_{L(\mathbf{x})}(\mathbf{x})$ represents the intensity value associated with the texture labelled $L(\mathbf{x})$ at position $\mathbf{x}$. The problem of segmentation now becomes that of:

1. Estimating the textures $\mathbf{T_i}$ for $i = 1, \cdots, K$ which make up the observed image.
2. Mapping the corresponding label field $\mathbf{L}$ using both the estimated textures and the observed image.

By its nature segmentation is an ill-posed problem in that the observed image itself is insufficient to unambiguously define the segmentation problem [146, 176]. As a result it is necessary to introduce suitable prior information regarding the solution in order to constrain the nature of the problem. Let $[\theta^L, \theta^I]$ denote two parameter vectors that have been introduced in order to add some prior information into the image model. The parameter θ^I is associated with the *likelihood* that characterises the relationship between the label field $\mathbf{L}$ and the observed data $\mathbf{I}$. Similarly, θ^L is the parameter vector associated with the prior. This prior is introduced in order to relate some physical understanding as to the nature of textures. Generally a prior will act as a smoothness constraint on the overall solution.

Hence it is required to estimate both the texture parameters and the label field given the observed data. This is best done by maximising the following probability distribution,

$$p(T_{L(\mathbf{x})}(\mathbf{x}), L(\mathbf{x}) | I(\mathbf{x}), \theta^I, \theta^L) \tag{5.4}$$

To manipulate this distribution, a Bayesian approach is natural. The Bayesian framework provides a basis to allow identification of unknown model parameters from noisy observations of a given process, given certain prior information about those parameters [176]. Under the Bayesian framework, expression (5.4) can be decomposed into a mixture of conditional and marginal distributions.

$$p(T_{L(\mathbf{x})}(\mathbf{x}), L(\mathbf{x}) | I(\mathbf{x}), \theta^I, \theta^L) \propto \underbrace{p(I(\mathbf{x}) | L(\mathbf{x}), T_{L(\mathbf{x})}(\mathbf{x}), \theta^L)}_{likelihood} \underbrace{p(T_{L(\mathbf{x})}(\mathbf{x}) | \theta^I) p(L(\mathbf{x}) | \theta^L)}_{prior} \tag{5.5}$$

where $p(T_{L(\mathbf{x})}(\mathbf{x}), L(\mathbf{x}) | I(\mathbf{x}), \theta^I, \theta^L)$ is the posterior distribution, $p(I(\mathbf{x}) | L(\mathbf{x}), T_{L(\mathbf{x})}(\mathbf{x}), \theta^L)$ is the likelihood distribution, $p(T_{L(\mathbf{x})}(\mathbf{x}) | \theta^I)$ is the prior distribution for textures and $p(L(\mathbf{x}) | \theta^L)$ is the prior over the label field.

The posterior distribution expresses the state of knowledge about the model *after* the parameters have been estimated and the aim of the segmentation is to estimate the parameters that will maximise this distribution. This is known as Maximum-A-Posterior (MAP) estimation. If

the prior information is omitted, the estimation process is known as Maximum Likelihood (ML) estimation. However, since segmentation is ill-posed, prior information is necessary in order to constrain the solution and so MAP estimation will be used here. MAP will involve both the likelihood and prior distributions. The manner in which these distributions influence the solution is as follows:

- The *likelihood* distribution is the conditional probabilistic model linking the unknown parameters to the observations **I**. It connects the texture regions and assignment of labels to the observed image data through the image model given in (5.3).
- The *prior* distributions are marginal probabilistic models which introduce some physical understanding or intuition into the nature of images. The label field prior provides information on the expected organisation of the label field, perhaps by introducing a bias toward certain types of configurations that are intuitively more likely than others. For example some priors may impose a smoothness or regularisation constraint on the solution. The prior on the likelihood model is concerned with the mechanism by which the texture regions present in the observed image can be modelled. Thus a suitable model is chosen and each texture feature in the image is described using this chosen model.

Note the likelihood and the prior on the likelihood are very closely related given that they are both based on the underlying image model given in (5.3). The means by which the likelihood prior distribution can be incorporated within the likelihood distribution is described next.

5.10.1 The Likelihood and Texture Prior

The relationship between the observed gray-scale values $I(\mathbf{x})$ and the unknown parameters $[L(\mathbf{x}), T_{L(\mathbf{x})}(\mathbf{x})]$ is given by the likelihood. The likelihood should discourage label and texture region estimates that are not supported by the observed data. To incorporate the element of uncertainty associated with the modelling process, the image model given earlier in (5.3) can be re-expressed as:

$$I(\mathbf{x}) = T_{L(\mathbf{x})}(\mathbf{x}) + e(\mathbf{x}) \tag{5.6}$$

where $e(\mathbf{x})$ represents any error in the image model. This expresses the idea that every pixel in the image is an observed manifestation of some "hidden" texture $T_{L(\mathbf{x})}(\mathbf{x})$. The residual term $\mathbf{e}$ is formed by combining the noise components in the image formation process together with errors in the underlying modelling of the original signal. In most applications, it is assumed that this residual term can be modelled by a Gaussian process with zero mean and variance σ_e^2, i.e. $e(\mathbf{x}) \sim \mathcal{N}(0, \sigma_e^2)$ [176].

Given that the observed image is composed of texture regions T_k, $k = 1, \cdots, K$ each of which must be modelled, the manner by which these underlying textures will be modelled will

be dependent on the choice of expression for $p(T_{L(\mathbf{x})})$. This is a key point in this thesis. There are two broad choices for texture descriptions: *parametric* and *non-parametric*. A good parametric model is the Gaussian Markov Random Field (GMRF). Under the GMRF model, the texture is assumed Gaussian and the intensity at any site $\mathbf{x}$ can be given by the following difference equation,

$$T_{L(\mathbf{x})}(\mathbf{x}) = \sum_{k=1}^{P} a_k (T_{L(\mathbf{x})}(\mathbf{x}+\mathbf{q}_k) + T_{L(\mathbf{x})}(\mathbf{x}-\mathbf{q}_k)) + \epsilon(\mathbf{x}) \tag{5.7}$$

where $\epsilon(\mathbf{x})$ is a zero mean stationary Gaussian sequence, $\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma_\epsilon^2)$. The P coefficients of the model are denoted a_k for $k = 1, \cdots, P$. The pixels used in the prediction are known as the support or neighbourhood of the model and are mapped by the P spatial offset vectors $\mathbf{q}_k$.

However, it was shown in chapter 4 that non-parametric techniques for texture synthesis are vastly more powerful than parametric approaches. This has important implications for "example based image segmentation", a term that is coined here to describe the new class of techniques presented in this thesis. The non-parametric means by which texture can be modelled will be discussed later. For the moment however, consider that $p(T_{L(\mathbf{x})})$ is described with a GMRF. More specifically, the model used here is the non-causal 2D Autoregressive (AR) model which is a sub-set of the GMRF. The 2D AR process is given by,

$$T_{L(\mathbf{x})}(\mathbf{x}) = \sum_{k=1}^{P} a_k T_{L(\mathbf{x})}(\mathbf{x}+\mathbf{q}_k) + \epsilon(\mathbf{x}) \tag{5.8}$$

Using this 2D AR model, any particular pixel in the texture region $T_{L(\mathbf{x})}$ at site $\mathbf{x}$ can be predicated by a linear combination of pixels also associated with the texture region plus the added excitation or residual error which is assumed to be Gaussian, $\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma_\epsilon^2)$. As with the general GMRF model, the P coefficients of the model are denoted a_k and the spatial offset vectors or support are given by $\mathbf{q}_k$ for $k = 1, \cdots, P$. The 2D AR model parameters are given by $\theta^I = [\sigma_\epsilon^2, \mathbf{a}]$ (arranged in a vector of P elements).

This prior on the likelihood model (5.6) can be incorporated into the initial image model (5.8) to give the overall image model,

$$\begin{aligned} I(\mathbf{x}) &= \sum_{k=1}^{P} a_k T_{L(\mathbf{x})}(\mathbf{x}+\mathbf{q}_k) + \underbrace{e(\mathbf{x}) + \epsilon(\mathbf{x})}_{v(\mathbf{x})} \\ &= \sum_{k=1}^{P} a_k T_{L(\mathbf{x})}(\mathbf{x}+\mathbf{q}_k) + v(\mathbf{x}) \end{aligned} \tag{5.9}$$

where $\mathbf{v}$ represents the overall residual error in the modelling process, and $\mathbf{v}$ is obtained by summing the two Gaussian functions $\mathbf{e}$ and ϵ. The noise $\mathbf{v}$ is Gaussian with $v(\mathbf{x}) \sim \mathcal{N}(0, \sigma_v^2)$. Since the noise is independently identically distributed (i.i.d.) over the data, the likelihood

$p(I(\mathbf{x})|\cdots) \propto p(v(\mathbf{x}))$. Therefore, the likelihood expression required in the assembly of the posterior may now be expressed as:

$$\begin{aligned} p(I(\mathbf{x})|L(\mathbf{x}),\theta^L) &= \left[\frac{1}{\sqrt{2\pi\sigma_v^2}}\right]^{M\times N} exp\left(-\frac{\sum_{\mathbf{x}} v(\mathbf{x})^2}{2\sigma_v^2}\right) \\ &= \left[\frac{1}{\sqrt{2\pi\sigma_v^2}}\right]^{M\times N} exp\left(-\frac{\sum_{\mathbf{x}}(I(\mathbf{x})-T_{L(\mathbf{x})}(\mathbf{x}))^2}{2\sigma_v^2}\right) \end{aligned} \tag{5.10}$$

Having already discussed how the texture prior can be incorporated into the likelihood model, the following section will introduce some priors that have been developed in order to add smoothness to the solution.

5.10.2 Regularisation Priors

This probability distribution is used to introduce constraints on the label field. Similar to the texture prior discussed previously, the MRF model will be evoked in order to form a probabilistic model for the labels at each individual site on a lattice. From the definition of the MRF model, the label at each site will be dependent only on the labels at the sites of its predefined neighbourhood. The size and structure of this neighbourhood will be dependent on the model type and order chosen. In segmentation, many types of MRF models have been used but three types have remained dominant. These are the *Gaussian Markov Random Field* (GMRF) [46], the *Potts model* [161] and more recently the *Hidden Markov Model (HMM)* [163]. The GMRF was discussed earlier as the prior on the texture model and so will not be discussed in this section. The Potts model can be considered a classical Markov model, which aims to model only one variable or feature set at a time. The HMM model was introduced to model more than one variable at any one time. The manner in which these MRF models have been used in segmentation will be discussed next.

5.10.2.1 The Potts Model

The Potts model is used to evoke a smoothness constraint over the label field. It can be said that images are composed of smooth homogeneous regions; the corresponding segmentation or label field should mirror this smoothness by containing large areas of labels assigned to the same class. The Potts model provides a method by which the label at each site can be influenced by the labels at neighbouring sites [58, 82, 103]. The order of the model is determined by the neighbourhood size over which a pixel is modelled. Figure 5.10 illustrates a first and second order neighbourhood structure. Higher order MRF models include more sites in the estimation process and so result in a more accurate modelling framework. It should be noted that this increased accuracy brings with it an increase computational load. In practice and as a trade-off between accuracy and computational expense, most segmentation algorithms use second-order MRF models.

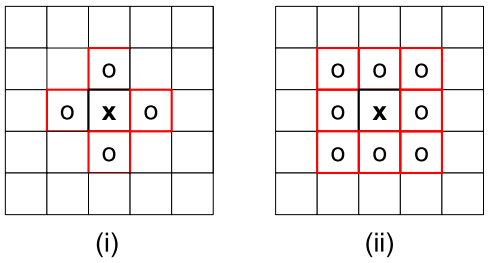


Figure 5.10: *Neighbourhood structures used in MRF models,* **x** *is the site in question and* **o** *denotes sites on which the value of* **x** *will be dependent. (i) is a first order and (ii) is second order neighbourhood system.*

Under the Potts model, the prior can be expressed as,

$$p(L(\mathbf{x})|\theta^L) \propto exp\left(-\sum_{k=1}^{P}\beta_k[1-\delta(L(\mathbf{x}),L(\mathbf{x}+\mathbf{q}_k))]\right) \tag{5.11}$$

where the neighbourhood labels are obtained using the P offset vectors $\mathbf{q}_k$, the P coefficients of the model are denoted by β_k for $k = 1, \cdots, P$ and $\delta(.)$ is the delta Kronecker function defined such that, $\delta(L(\mathbf{x}), L(\mathbf{x}+\mathbf{q}_k)) = 1$ for $L(\mathbf{x}) = L(\mathbf{x}+\mathbf{q}_k)$. The regularisation model parameters are given by $\theta^L = [\beta]$. The parameter β represents the weight that is attached to each neighbouring site in the estimation process of the centre site.

Choosing suitable values for β is by no means a trivial task, given that choosing values that are too large will result in over smoothing of the label field [81]. Similarly choosing values that are too small will result in a weak prior which will not penalise labels that are unrealistic given the already labelled local neighbours. For example, consider the following values of β suitable for a second order Potts model.

$$\beta_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}; \quad \beta_{\mathbf{2}} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & 0 & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{bmatrix}. \tag{5.12}$$

The parameter matrix $\beta_{\mathbf{1}}$ applies equal weighting to all sites in the neighbourhood structure. The result of this equal weighting is block type smoothness constraint on the label field. These blocks are sometimes evident in the overall solution, making it look artificial and patchy. An alternative approach is to apply less weighting to sites located diagonal to the centre site; $\beta_{\mathbf{2}}$ is an example of such a parameter which will applied this variational weighting. Giving less emphasis to diagonal sites seems natural given that physically diagonal sites are further away from the centre site and so should have less influence than those sites located directly above or beside the centre site. This variation in weighting also decreases the blocky appearance of the segmentation. Figure 5.11 illustrates the subtle differences obtained by slightly varying the parameter vector β.

(i) (ii)

(iii) (iv)

Figure 5.11: *Illustrating the smoothness obtained using two different smoothness parameters:* β_1 *and* β_2*. Images (i) is the observed intensity image to be segmented, image (ii) is an example segmentation obtained using* K *means clustering with* $k = 4$*. Images (iii) and (iv) are the segmentations obtained after the Potts prior constraint has been applied. Note the subtle differences introduced as a result of the different parameter vectors, in (iii)* β_1 *was used and in image (iv)* β_2 *was used.*

5.10.2.2 The Hidden Markov Model

In the classical Markov model, there is only one variable being modelled at any one time and the state of this variable is directly visible to the observer. The Potts model described previously can be categorised as a classical Markov model, where the label alone is the variable being modelled. The state of the label is directly visible to the observer and the state transition probabilities β are the only model parameters that needs to be estimated. However, in some cases it is found that a more powerful model can be obtained if more than one variable is modelled. These variables represent different types of features that are all related but do not characterise the same feature. That is, there may exist variables that are not directly visible to the observer but the parameters that are associated with these variables will influence other variables that are visible to the observer. Thus, the state of the model may not be not be directly visible to the observer but variables influenced by the state are visible. The hidden Markov model (HMM) provides such a model.

HMMs were first used in speech recognition [164] but have recently have been applied to many image processing problems. To illustrate how HMMs can be used in segmentation, consider the approach taken by Chen and Kundu [48] which aims to solve the problem of unsupervised texture segmentation. In supervised texture segmentation, the number of textures present in the observed image is known *a priori*. The problem of segmentation then involves estimating the parameters which characterise each of these texture regions. In contrast, in unsupervised texture segmentation the number of textures present in the image is not known beforehand and so the segmentation problem involves estimating the number of texture regions and the parameters that are associated with each texture region. Estimating the number of texture regions present in the image is by no means a trivial task and there have been many attempts to solve this problem including [17,51,110,141]. Chen and Kundu propose a two stage segmentation procedure. Initially, a feature map of the observed image is formed by moving a square averaging window over the image. This window is used to extract the textural and directional properties at each site in the image. Each pixel in the feature map is then represented by a sequence of 4D feature vectors. The feature sequences belonging to the same textures are modeled using a HMM. Under this scheme, if there are K different textures present in an image, there will be K distinct HMMs to be found and trained. Consequently, the unsupervised texture segmentation problem becomes a HMM-based problem where the appropriate number of HMMs, the associated parameters and the discrimination among the HMMs is the focus of the algorithm.

Another example of HMMs in segmentation is that of Marroquin et al. [144]. Here the authors proposed the use of the HMM as a means to find the optimal estimator for the region parameter vector θ^I and the label field L. The classical MRF segmentation approach estimates the optimal θ^I and L using a two step Segmentation/Model Estimation (SM) process such as the Expectation Maximisation (EM) algorithm. During any SM estimation process, the best segmentation given the current estimate of the model parameters θ^I is found during the S step,

while the best estimate for θ^I given the current estimate for the segmentation is found during the M step. The problem with this approach is that it is computationally intractable to obtain an exact optimal segmentation in the S step. Therefore, approximations based on deterministic methods have to be introduced in order to make the problem more computationally manageable. However, to avoid some of these problems and obtain exact optimal estimators for the label field and model parameters, Marroquin et al. model the label field as a HMM. The label field is generated by a two step stochastic function. This doubly stochastic prior model for the label field is achieved by introducing a hidden Markov random vector field that is used as a discrete probability measure on the observable label state space. They applied this method to the segmentation of MRIs and motion segmentation.

Multi-scale Hidden Markov Models

In general, the main instance in which HMMs are found in segmentation algorithms is in those that incorporate a multi-scale approach. In section 5.3 it was shown that the some of the uncertainties inherent with the segmentation can be reduced by adopting a multi-scale approach to the problem. Of the different forms of multi-scale transforms available, the wavelet transform has been the most prevalent among the image segmentation literature. Aspects of the wavelet transform are presented in chapter 3. In order to model the statistical dependencies among wavelet coefficients at different scales, Crouse et al. [55] developed wavelet-domain HMMs. These models have associated with them two intrinsic properties which summarise the probabilistic structure of the coefficients of the wavelet transform. These are: (i) *mixture densities:* which model the marginal probability of each wavelet coefficient as a mixture density with a hidden state variable and (ii) *probabilistic graphs:* which characterise the key dependencies between the wavelet coefficients together with a Markovian dependency between the hidden state variables. Crouse et al. proposed three types of model based on different probabilistic graphs. Model 1. is the independent mixture model which leaves the state variables unconnected and hence ignores any inter-coefficient dependencies. Model 2. is the hidden Markov chain model which connects the state variables horizontally within each scale. Model 3. is the hidden Markov Tree (HMT) model which connects the state variables vertically across scale. The HMT model is the most powerful of these models given the strong dependences between coefficients at different scales. One drawback of the HMT model is the computational expense associated with the training of the HMT to a given data set which should have similar texture characteristics as the regions to be modelled.

Romberg [172] address the computational burden of the Crouse et al. HMT and apply this model to the problem of segmentation. Their reduced parameter HMT model is constructed using two empirical tertiary properties of image wavelet coefficients. These tertiary properties reflect the self-similar nature of images and their resulting generalised $1/frequency$ spectral behaviour [65, 177]. They can be summarised as the exponential decay of wavelet coefficients

across scale and the stronger persistence of wavelet coefficients at fine scales. Using these properties, Romberg et al. propose a reduced parameter set HMT which is independent of the size of the image and of the number of wavelet scales used. In [171], this algorithm is altered to work with the Dual Tree-Complex Wavelet Transform (DT-CWT).

Similarly, Choi and Baraniuk [50] also take the HMT model proposed by Crouse et al. as a basis for an image texture segmentation algorithm they call HMTSeg. The HMTSeg algorithm consists of three main steps. In step one the HMT model is trained using separate homogeneous images. Step two involves computing the likelihood at each scale of the wavelet tree and step three involves fusing these multi-scale likelihoods using a labelling tree to form the multi-scale MAP classification. This HMTSeg algorithm is applied to synthetic images.

Algorithms based on HMMs have proved popular among the segmentation literature, as they offer the means to model more than one feature at a time which can often lead to an improved segmentation. The strength of HMMs is especially evident in wavelet based segmentation algorithms as they offer a basis to model the dependences between multi-scale wavelet coefficients. However, this improved modelling framework is at the expense of computational load given the increased parameter set that has to be estimated. This computational burden is increased further by the fact that the HMMs need to be trained initially and the accuracy of the model will be determined by the data on which it is trained.

Moving on from this discussion on HMMs, the next section will illustrate how the likelihood and prior terms of the Bayesian framework combine to form an energy minimisation problem that is equivalent to the maximum-a-posterior estimator discussed earlier.

5.10.3 From Probability to Energy

MAP estimation is concerned with finding the label field $\mathbf{L}$ that will maximise the posterior probability function given in (5.5) based on the observed intensity image $\mathbf{I}$ and the model parameters $[\theta^I, \theta^L]$. This section will illustrate how maximising this posterior can be expressed as an energy minimisation problem between $\mathbf{L}$ and $\mathbf{I}$. Realising the problem as an energy minimisation problem avoids the computationally expensive likelihood prior product present in (5.5) and instead replaces it with a more computationally manageable likelihood prior summation. To achieve this, the natural logarithm (Ln) of (5.5) is taken giving,

$$\begin{aligned} Ln\left(p(T_{L(\mathbf{x})}(\mathbf{x}), L(\mathbf{x})|I(\mathbf{x}), \theta^I, \theta^L)\right) &\propto Ln\left(p(I(\mathbf{x})|L(\mathbf{x}), \theta^I)p(L(\mathbf{x})|\theta^L)\right) \\ &\propto Ln\left(p(I(\mathbf{x})|L(\mathbf{x}), \theta^I)\right) + Ln\left(p(L(\mathbf{x})|\theta^L)\right) \end{aligned} \tag{5.13}$$

The next step is to find the natural logarithm of the likelihood $Ln\left(p(I(\mathbf{x})|L(\mathbf{x}), \theta^I)\right)$ and prior, $Ln\left(p(L(\mathbf{x})|\theta^L)\right)$. Recall the likelihood distribution given in (5.10). Taking the natural logarithm of this distribution yields,

$$\begin{aligned} Ln\left(p(I(\mathbf{x})|L(\mathbf{x}),\theta^L)\right) &= Ln\left(\left[\frac{1}{\sqrt{2\pi\sigma_v^2}}\right]^{M\times N} exp\left(\frac{\sum_{\mathbf{x}}(I(\mathbf{x})-T_{L(\mathbf{x})}(\mathbf{x}))^2}{2\sigma_v^2}\right)\right) \\ &= Ln(2\pi\sigma_v^2)^{-\frac{MN}{2}} Ln\left(exp\left(-\frac{\sum_{\mathbf{x}}(I(\mathbf{x})-T_{L(\mathbf{x})}(\mathbf{x}))^2}{2\sigma_v^2}\right)\right) \\ &= -\frac{MN}{2} Ln(2\pi\sigma_v^2)\frac{\sum_{\mathbf{x}}(I(\mathbf{x})-T_{L(\mathbf{x})}(\mathbf{x}))^2}{2\sigma_v^2} \end{aligned} \tag{5.14}$$

Let $\Psi(I(\mathbf{x}),\mathbf{T}_{\mathbf{L}(\mathbf{x})}(\mathbf{x}))$ denote the energy between the observed image $\mathbf{I}$ and the texture region $\mathbf{T}_{\mathbf{L}(\mathbf{x})}$ such that,

$$\Psi(I(\mathbf{x}),T_{L(\mathbf{x})}(\mathbf{x})) = -\frac{MN}{(2\sigma_v)^2} Ln(2\pi\sigma_v^2)(I(\mathbf{x})-T_{L(\mathbf{x})}(\mathbf{x}))^2 \tag{5.15}$$

Similarly, taking the natural logarithm of the regularisation prior term given in (5.11) yields the following expression for $Ln\left(p(L(\mathbf{x})|\theta^L)\right)$,

$$\begin{aligned} Ln\left(p(L(\mathbf{x})|\theta^L)\right) &\propto Ln\left(-exp\left(\sum_{k=1}^{P}\beta_k(1-\delta(L(\mathbf{x}),L(\mathbf{x}+\mathbf{q_k})))\right)\right) \\ &\propto -\sum_{k=1}^{P}\beta_k(1-\delta(L(\mathbf{x}),L(\mathbf{x}+\mathbf{q_k}))) \end{aligned} \tag{5.16}$$

Let $\Phi(L(\mathbf{x}))$ denote the energy at site $\mathbf{x}$ over the label field $\mathbf{L}$ such that,

$$\Phi(L(\mathbf{x})) \propto \beta_k(1-\delta(L(\mathbf{x}),L(\mathbf{x}+\mathbf{q_k}))) \tag{5.17}$$

Combining the likelihood and prior energy terms given above yields the following energy term,

$$Ln\left(p(T_{L(\mathbf{x})}(\mathbf{x}),L(\mathbf{x})|I(\mathbf{x}),\theta^I,\theta^L)\right) \propto -\sum_{\mathbf{x}}\Psi(I(\mathbf{x}),T_{L(\mathbf{x})}(\mathbf{x})) - \sum_{k=1}^{P}\Phi(L(\mathbf{x})) \tag{5.18}$$

Maximising $p(T_{L(\mathbf{x})}(\mathbf{x}),L(\mathbf{x})|I(\mathbf{x}),\theta^I,\theta^L)$ over the entire lattice implies maximising $exp(-U(\mathbf{I},\mathbf{L}))$ which is equivalent to minimising $U(\mathbf{I},\mathbf{L})$, where the energy term $U(\mathbf{I},\mathbf{x})$ is given as:

$$U(\mathbf{I},\mathbf{L}) = \underbrace{\sum_{\mathbf{x}}\Psi(I(\mathbf{x}),T_{L(\mathbf{x})}(\mathbf{x}))}_{datadriven} + \underbrace{\sum_{k=1}^{P}\Phi(L(\mathbf{x}))}_{regularisation} \tag{5.19}$$

The data driven term tends to give a solution similar to the data, while the regularisation term tends to favor homogeneous regions. The MAP estimator is the configuration that maximises the posterior probability or minimises the energy function $U(\mathbf{I},\mathbf{L})$ across the entire lattice $\mathbf{X}$. Unfortunately this energy function has many local minima which makes it a difficult optimisation

problem to solve. In addition, the optimisation problem is further complicated by the high dimensionality of the lattice structure $\mathbf{X}$. Some of the approaches that have been developed to solve this optimatisation problem are discussed next.

5.10.4 MAP Optimisation

As shown the segmentation problem can be formulated as a maximum a posteriori (MAP) Markov random field (MRF) problem which is equivalent to minimising of the energy function in (5.19). There have been many methods developed to solve the MAP-MRF problem. At the heart of all of these methods is a step that locally perturbs the solution at a pixel or subset of pixels with the aim of minimising the global energy at each iteration. A comparative study of energy minimisation methods for MRFs is given in [201]. In general, energy minimisation techniques fall into two categories: *stochastic* and *deterministic* methods. One of the main class of stochastic techniques used are Markov Chain Monte Carlo (MCMC) methods, of which simulated annealing (SA) is one example. Deterministic methods such as Iterative Conditional Modes (ICM) [21] were developed in order reduce the computationally load of MCMC methods. In recent years, Graph-Cuts [30] and Bayesian Belief Propagation [154] have been widely used in MAP-MRF estimation.

5.10.4.1 Markov Chain Monte Carlo Techniques

For MAP-MRF estimation, the posterior distribution for the label field $\mathbf{L}$ to be estimated is given as,

$$\mathbf{L} \sim p(\mathbf{L}|\mathbf{I}) \tag{5.20}$$

where $\mathbf{L}$ represents the entire label image containing $M \times N$ unknown elements to be estimated. The aim of the segmentation algorithm is to manipulate this probability distribution to yield the best segmentation given the observed image $\mathbf{I}$. The essence of a stochastic solution is to draw samples of $\mathbf{L}$ from this distribution and then numerically evaluate the probability density function (pdf) for each $\mathbf{L}$. The optimal $\mathbf{L}$ will be the one which is most probable. For example, a possible approach would be to directly evaluate $p(\mathbf{L}|\mathbf{I})$ at each possible configuration of $\mathbf{L}$ and hence obtain the cumulative density function (cdf). A numerical method may then be used to draw a sample from L [162].

However, because of the high dimensionality of the problem[1], this technique is not computationally feasible. To make the problem more manageable, Geman and Geman [82] proposed *the Gibbs sampler*. The Gibbs sampler allows the problem of drawing samples from $p(\mathbf{L}|\mathbf{I})$ to be

[1] At each site $\mathbf{x}$, the label assigned will be taken from the set λ, therefore there are $|\lambda|$ choices of labels, where $|\lambda|$ is the number of elements in the set λ. Across the entire $M \times N$ label field, the number of possible configuration of $\mathbf{L}$ is $(M \times N)^{|\lambda|}$

broken down into a draw from a number of conditional distributions which are easier to sample from.

The Gibbs sampler [82] can be interpreted as a special case of the MCMC Metropolis-Hastings algorithm [203] and is a method for drawing samples from complicated high dimensional distributions. It does this by decomposing high dimensional distributions into a number of conditional distributions of smaller dimension. These techniques rely on the locality of the Markov based model which enables the probability density function of the entire label image to be defined in terms of the sum local conditional probability density functions. Thus, the multidimensional draw is decomposed into draws from local conditional distributions at every site $\mathbf{x_i} \in \mathbf{X}$, beginning with an initial estimate for the label field $\mathbf{L}$ defined as $\mathbf{L^0}$. This initial estimate may be a zero estimate or quite often it can be obtained using a simpler non-Bayesian method such as the K-means clustering algorithm discussed previously. With the Gibbs sampler, samples for each site $\mathbf{x_i}$ are drawn as follows,

$$\begin{aligned} L^1(\mathbf{x_1}) &\sim p(L(\mathbf{x_1})|\mathbf{I}, \mathbf{L^0}(-\mathbf{x_1})) \\ L^1(\mathbf{x_2}) &\sim p(L(\mathbf{x_2})|\mathbf{I}, \mathbf{L^0}(-\mathbf{x_1}, -\mathbf{x_2}), \mathbf{L^1}(\mathbf{x_1})) \\ &\vdots \\ L^2(\mathbf{x_1}) &\sim p(L(\mathbf{x_1})|\mathbf{I}, \mathbf{L^1}(-\mathbf{x_1})) \\ &\vdots \end{aligned} \tag{5.21}$$

where $\mathbf{L^0}(-\mathbf{x_1})$ denotes all the labels in the label field at iteration 0 that are *not* at site $\mathbf{x_1}$ and $L^i(\mathbf{x_j})$ denotes the ith sample of the label L at site $\mathbf{x_j}$.

The Gibbs sampler method proceeds by drawing samples for $L(\mathbf{x}), \forall \mathbf{x} \in \mathbf{X}$ given the current state of samples for the label field. Each new sample drawn at a site $\mathbf{x_j}$ then replaces the previous sample at that site. The process continues at the next site where a label sample is drawn based on the current state of the label field which now includes the sample just drawn at the previous site. This updating process is repeated iteratively until convergence is achieved. Thus the samples being generated form the multidimensional $p(\mathbf{L}|\mathbf{I})$ as required.

A number of methods can be used to draw samples from the conditional distribution $p(L(\mathbf{x})|.)$. As before, a direct method of evaluating all possible values of $L(\mathbf{x})$ to generate the cdf can be used. Alternatively, an approximation to the cdf can be obtained by evaluating $p(L(\mathbf{x})|.)$ on a subset of grid points using a technique known as the Griddy sampler [203]. A sample from the distribution can then be obtained by transforming a uniformly distributed random variable using the numerically evaluated cdf. The location of the grid points will dependent on the implemented used.

Once the step of random sample generation has been completed, the next step will involve manipulating these samples to yield the MAP estimate. Computationally this process is very expensive and slow. Even though stochastic methods are guaranteed to converge toward a global minimum, the computational burden associated with reaching this global minimum is unrealistic

for most practical applications

To address this computational burden, deterministic methods have been developed. These methods replace the global optimisation problem associated with the MCMC with a succession of local optimisations each of which is more computationally manageable.

5.10.4.2 Iterative Conditional Modes

The Iterative Conditional Modes (ICM) algorithm developed by Besag [21] is based on iteratively choosing the best estimate at each site using a local energy function rather than a global energy function. The ICM process works by maximising the local conditional probabilities at each site in the lattice and thus converges to some local minimum over all sites in the image. Its implementation can be linked to that of the Gibbs sampler discussed above, but in the ICM case the label chosen at each site is the one which maximises the local conditional probability, rather than drawing a value based on the conditional probability distribution.

The MAP estimation problem to find the label field that gives the maximum posterior probability is given as,

$$\hat{\mathbf{L}} = \max_{\mathbf{L}} p(\mathbf{L}|\mathbf{I}) \tag{5.22}$$

where $\hat{\mathbf{L}}$ is the optimum label field. To find the MAP, ICM chooses labels that will maximise the conditional distributions at each site and proceeds in a similar manner to that of the Gibbs sampler. That is,

$$\begin{aligned}
\hat{L}^1(\mathbf{x_1}) &= \max_{L(\mathbf{x_1})} p(L(\mathbf{x_1})|\mathbf{I}, \mathbf{L^0}(-\mathbf{x_1})) \\
\hat{L^1}(\mathbf{x_2}) &= \max_{L(\mathbf{x_2})} p(L(\mathbf{x_2})|\mathbf{I}, \mathbf{L^0}(-\mathbf{x_1}, -\mathbf{x_2}), L^1(\mathbf{x_1})) \\
&\vdots \\
\hat{L^2}(\mathbf{x_1}) &= \max_{L(\mathbf{x_1})} p(L(\mathbf{x_1})|\mathbf{I}, \mathbf{L^1}(-\mathbf{x_1}))
\end{aligned} \tag{5.23}$$

The ICM algorithm converges much faster than the stochastic simulated annealing algorithms. However, this convergence is to a local minimum and there is no guarantee that this will coincide with the global minimum. In addition, the success of the ICM algorithm is very much dependent on the initialisation estimate, therefore a good initial estimate is required.

5.10.4.3 Graph-Cut Technique

More recently and with particular application to the binary (two label) segmentation problem, techniques based on graph-cuts have been introduced into the segmentation framework to minimise the energy function given by (5.19) [30, 87, 127]. The idea behind the graph-cut technique is to construct a specialised graph for the energy function to be minimised such that the minimum cut on the graph also minimises the energy (either locally or globally). The minimum

cut can be computed using very efficient max flow algorithms. The two most popular graph-cut algorithms are the *swap move* and the *expansion move* algorithms introduced in [30].

To illustrate, let $[c_0, c_1]$ be the pair of symbols used to label the object and background pixels in the image. A swap move takes some subset of pixels currently given the label c_0 and assigns them the label c_1, and vice-versa. The swap move algorithms finds a local minimum such that there is no swap move, for any pair of labels c_0, c_1 that will produce a lower energy labelling. Similarly, during an expansion move the label c_0 will increase the set of pixels that are given this label at each iteration. The expansion move algorithm finds a local minimum such that no expansion move, for any label c_0, yields a labelling with lower energy. When deciding whether a pixel should be re-labelled in the expansion or swap move algorithms, the graph-cut technique is used to find the minimum energy. The criteria for a local minimum with respect to expansion or swap moves are so strong that there are much fewer minima in high dimensional spaces compared to standard moves.

The graph-cut technique for energy minimisation has been applied to many vision problems including image restoration [30], stereo and video motion [25], image synthesis [131], multi-camera scene reconstruction [126] and image segmentation [31]. The graph-cut technique offers a stable and efficient way to solve the binary segmentation problem. More recently algorithms have been developed that use the graph-cut technique for multiple label segmentation problem [125]. However, the computational burden and complexity associated with solving the multiple label energy minimisation problem using the graph-cut technique has limited its use in many segmentation applications.

5.10.4.4 Belief Propagation

The method of Belief Propagation (BP) [154] arises directly from the need to manipulate $p(\mathbf{L}|\mathbf{I})$, the probability of a particular segmentation solution $\mathbf{L}$ given the data $\mathbf{I}$. It is a form of marginal inference given that the algorithm requires the manipulation of $p(\mathbf{L}|\mathbf{I})$ at each site $\mathbf{x}$. Essentially, BP algorithms are reliant on the factorisation of $p(L(\mathbf{x})|\mathbf{I})$ into a number of probability distributions known as "messages". Each node (site or group of sites) sends each of its neighbours a message containing information about what state it believes that node should be in. After a certain number of iterations of message passing, each node calculates a "belief" based on its local data and the messages received from neighbouring nodes. This belief is a probability distribution over its own state. The state which maximises the belief at each node is chosen as the state for that node.

BP is an exact inference method if the network contains no loops, that is its is singularly connected. However, if the network contains loops, then the original BP algorithm proposed by Pearl is not guaranteed to converge [154]. Loopy Belief Propagation (LBP) is belief propagation that ignores the existence of loops in the network [199]. There are two variants of the LBP algorithm used. In max-product LBP, the algorithm is designed to find the lowest energy

solution while the sum-product algorithm will compute the marginal probability distribution at each node in the graph. However, in general LBP is not guaranteed to converge and may go into a infinite loop switching between two labellings.

This concludes the discussion on MAP optimisation. The following section will discuss the problem of model selection.

5.10.5 Model Selection

In any data analysis problem of which segmentation is an example, the overall problem can be sub-divided into two basic issues to be resolved. The first issue concerns model selection[2]. This involves choosing from a set of candidate models, a model that is best supported by the observed data. Once a suitable model has been selected, the second issue to be resolved is of parameter estimation. In the case of segmentation, the parameter set represents both the label field and the selected model parameters. The estimated parameters will be dependent on the observed data sets. Analogous to the parameter estimation problem, Bayesian evidence $p(\mathbf{I}|\mathcal{M}_k)$ can be used to select the signal models and the noise statistics [173, 176] appropriate to the observed data. Evidence is defined in the following multiple integral,

$$p(\mathbf{I}|\mathcal{M}_k) = \sum_{\mathbf{\Theta}} p(\mathbf{I}|\theta_k, \mathcal{M}_k) p(\theta|\mathcal{M}_k) d\theta_k \tag{5.24}$$

where $\mathbf{\Theta}$ is the model parameter space, θ_k is the vector containing the parameters of the signal model, $\mathcal{M}_k$ is the model structure and $\mathbf{I}$ represents the observed image. The term $\mathcal{M}_k$ is known as the data model and represents the joint assumption of *both* the noise statistics and the signal model. The probability of a given data model being correct given the set of M possible data models $\{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_M\}$ is given by the relative evidence for that model:

$$\begin{aligned} p(\mathcal{M}_k|\mathbf{I}) &= \frac{p(\mathbf{I}|\mathcal{M}_k)p(\mathcal{M}_k)}{\sum_{i=1}^{M} p(\mathbf{I}|\mathcal{M}_i)p(\mathcal{M}_i)} \\ &= \frac{p(\mathbf{I}|\mathcal{M}_k)}{\sum_{i=1}^{M} p(\mathbf{I}|\mathcal{M}_i)} \end{aligned} \tag{5.25}$$

assuming that all models are equally likely *a priori*, that is $p(\mathcal{M}) = 1|M$. It is important to note that in terms of the observed data, the correct data model may not be in the set of possible data models chosen. In such an instance, all that can be done is to compare the candidate models that have been defined and determine which model is the most plausible [173].

In general there exists a trade off between the number of parameters that can be used in the modelling process against the accuracy of the model. The more complex the chosen model, the more accurately it will fit the observed image. To illustrate, an image consisting of $M \times N$ observations can always be described exactly in terms of a model with $M \times N$ parameters. This

[2]Note that this discussion on model selection will cover the two concerning issues of model type and model order.

model will describe the observation data set with zero error. However, this model will be specific to that particular $M \times N$ image only. To model other observed images which may have different noise realisations and dimensions a new model would be needed in order to describe the new data with equivalent zero error. Very often, the new parameter estimates can be vastly different to the old parameter estimates. Based on this observation it is found that model simplicity is the key to maximising the degree of consistency between parameter estimates computed from independent realisations of the data [111, 176].

The computation of the model selection evidence is difficult as it involves integrating the product of the likelihood and the prior probability density functions over all the possible parameters in the model. This is a global energy minimisation problem over the model parameter space and similar to the parameter estimation problem, Monte Carlo methods can be used. Combining the problems of model selection and parameter estimation, results in the following optimisation criterion,

$$\hat{K}, \hat{\theta_k}, \hat{\mathbf{L}} = \max_{K,\theta_k,\mathbf{L}} p(\mathbf{L}, \theta_k, \mathcal{M}|\mathbf{I}) \tag{5.26}$$

where K is the number of classes present in the segmentation image $\mathbf{L}$. Estimating the optimum parameters in (5.26) constitutes the unsupervised segmentation problem, where both the number of classes K and the associated model parameters are unknown *a priori*. To solve this problem, Barker et al. [15, 16] proposes incorporating reversible jumps into the Markov Chain process. These reversible jumps allow movement between model spaces and reduces the optimisation process to a single annealing run. The reversible jump algorithm consists of a Metropolis Hastings sampler with a dimension balancing element. This dimension balancing is introduced in order to facilitate sampling from different model spaces, by incorporating proposals that increase or decrease model order. Barker et al. apply this reversible jump algorithm to both isotropic and Gaussian MRF models.

5.10.6 The Multi-resolution Approach

Multi-scale or multi-resolution algorithms were mentioned briefly in the discussion on hidden Markov models. In summary, multi-resolution or multi-scale algorithms are designed to represent signal information over a fixed number of resolutions or scales. This allows image features to be represented at a range of scales so that any distinctive behavior exhibited can be captured and modelled independently of scale. Section 5.3.1 illustrated how the uncertainty inherent with the segmentation process can be reduced by adopting a multi-scale approach to the problem. This is because the multi-resolution representation of an image introduces a trade off between high positional accuracy versus high class accuracy. In addition to the ability to providing this class positional trade off, multi-scale approaches have the following attractive features which has ensured they have remained popular among the segmentation literature.

1. They are computationally attractive. Given that multi-scale approaches allow rough pa-

rameter estimates to be obtained at coarser scales where the data set is much reduced, the computational demands are light. These rough estimates are refined with increasing resolution but overall much of the heavy computational effort is undertaken at the coarse level.

2. Many powerful modelling frameworks (e.g. Wavelet Based HMMs [55], Multi-scale Random Field Model [28], Multi-resolution Gauss Markov Random Fields [129]) have been developed to exploit the features provided by multi-scale representations.

3. Multi-scale representations allow a natural domain in which to model texture features. Unlike colour or gray-scale intensity which are point specific, texture is a localised phenomenon that occurs over many scales. Therefore, it makes sense to analyse texture in a multi-scale environment and since many objects can be identified by their textural component, they can be more readily identified.

Of the multi-scale representations used in image segmentation, the wavelet transform is by far the most prevalent. The popularity of the wavelet transform is partly due to the good directional selectivity and low redundancy of the wavelet transform. The Dual Tree-Complex Wavelet Transform developed by Kingsbury [117] also has the attractive feature of shift invariance which makes it especially attractive for texture analysis and thus image segmentation. Aspects of the wavelet transform were presented in chapter 3.

5.10.7 Parametric Bayesian Segmentation

Depending on the manner in which the observed $\mathbf{I}$ and label $\mathbf{L}$ images are modelled, Bayesian segmentation algorithms can be classified as parametric or non-parametric. Parametric techniques attempt to model the image using some definable process. For example, in the Bayesian segmentation model described previously, the observed image is considered to be composed of a mixture of texture features all of which are modelled using a parametric Gaussian Markov Random Field (GMRF). Under the GMRF model, texture features are described and identified using a finite set of parameters and the set of GMRFs that make up the observed image are then characterised by a Gaussian distribution. This allows the likelihood function to be calculated between the intensity and class values. An expression for this parametric likelihood function was given in (5.10). Similarly, to impose a regularisation on the label field, it is modelled using a parametric Markov Random Field (MRF). This may be in the form of the Potts model or the doubly stochastic hidden Markov model (HMM) discussed earlier.

In general, parametric approaches are advantageous given that they allow the observed image and its associated segmentation to be described using a finite set of parameters, each of which has to be estimated. By and large parametric algorithms are efficient. In addition, there have been many complex models developed which can be applied to a wide range of image types. On the downside and because of the wide variability in image behaviour, it is impossible to define a

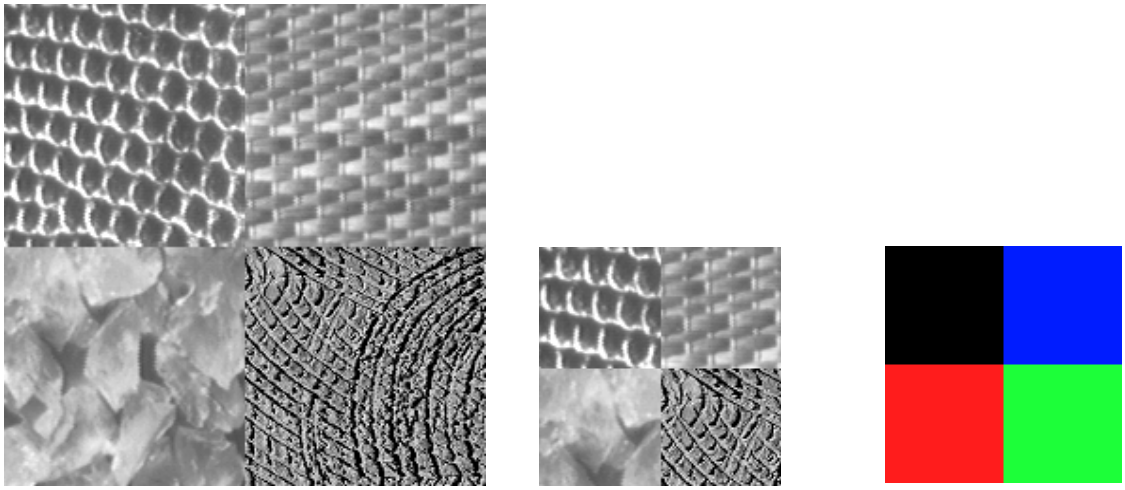
model or set of models that are suitable for all image genres. To illustrate, consider the Bayesian segmentation model given previously. Under this framework the observed image is assumed to be composed of independent texture regions, which are modelled using a GMRF. This GMRF model with its associated finite set of parameters is efficient for representing and analysing different texture types; however it is intractable to define a finite set of parameters that can accommodate all possible texture realisations. The limitations of this model were shown earlier in chapters 2 and 4, when parametric methods to synthesise new textures failed to accurately model some example texture types. As a result it can be concluded that while parametric methods may generate impressive results with some image types, they may completely fail for other image types. Another limitation inherent with the GMRF comes from its definition which states that the relationship between pixel intensity and class membership is governed by a Gaussian distribution. Image analysis has found that many images exhibit non-Gaussian behaviour [7, 134, 148, 180] and so assuming a Gaussian behaviour over all images types is unrealistic.

To illustrate another disadvantage of the parametric segmentation model, consider the label field which is generally modelled using a parametric Markov Random Field model. This MRF model offers a prior constraint on the label field by injecting a degree of smoothness into the segmentation result and section 5.10.2 outlines two MRF models used in segmentation. While these models have been popular in the literature and are robust to different image types, they do suffer from some serious limitations which hamper their use. For example, the task of parameter estimation is by no means trivial. Specifying or estimating the wrong set of parameters may result in the dependencies between labels being unsuitable or unrealistic for the desired segmentation. This can lead to *under segmentation* (over smoothing) or *over segmentation* (too many classes). Another limitation is that the size and shape of the model used is often dictated by computational constraints. Long range dependencies are difficult to implement due to the computational costs associated with high order MRF models. A third limitation is that complex configurations of the labeling process are difficult to define mathematically and so cannot be modelled using a (realistic) parametric model.

Motivated by the limitations associated with parametric methods and the success of non-parametric texture synthesis processes, Mignotte [146] proposed a non-parametric Bayesian segmentation model. This will be described next.

5.10.8 Non-Parametric Bayesian Segmentation

Unlike parametric approaches that model the image using some definable model, non-parametric approaches offer no such definable process and rather attempt to *measure* the image statistics using heuristic means. These heuristic measurements can be obtained from the image itself or some other training image that is similar in content to the image to be segmented. Efros and Leung [68] demonstrated the strength of this heuristic technique in their non-parametric texture synthesis algorithm which was discussed in chapters 2 and 4. To synthesise a new



(i) Image to be segmented $\mathbf{I}$ (ii) Example image $\mathbf{I_e}$ (iii) Example label field $\mathbf{L_e}$

Figure 5.12: *Non-parametric energy based modelling. Using the input parameters of (i) observed intensity image to be segmented* $\mathbf{I}$*, (ii) training example image* $\mathbf{I_e}$ *and (iii) example segmentation of image (ii)* $\mathbf{L_e}$*.*

texture image, Efros and Leung proposed the use of an implicit model obtained directly from heuristic measurements taken from an example image. This differed from previous approaches which were generally parametric and so used a definable explicit model for the image to be synthesised. Given the wide variability in texture behaviour, it was found that results obtained using the non-parametric approaches outperformed those obtained using parametric methods.

The symbiosis that exists between texture synthesis and analysis algorithms was discussed earlier in section 5.4.4 where it was found that if an algorithm is successful at texture synthesising, then it must be able to model the underlying example texture accurately. Because objects in an image to be segmented can be characterised and identified by their individual texture features, the success of a segmentation algorithm will be dependent on its ability to analyse and model texture regions accurately. Based on the success of the non-parametric texture synthesis algorithms, it makes sense to extend this non-parametric sampling technique to work within a segmentation framework. Mignotte [146] was the first to propose such an approach in his work on non-parametric multi-scale models. His approach may be considered as the reverse of the texture by numbers texture mapping process proposed in [35, 101].

Mignotte expressed the Efros and Leung sampling technique as a global cost function minimisation problem between the texture image to be synthesised and the given example texture image. To illustrate how this modelling technique can be applied to the segmentation problem, consider the image set given in Figure 5.12. Image (i) is the observed image $\mathbf{I}$ to be segmented and images (ii) and (iii) are the training example intensity image $\mathbf{I_e}$ and its *a priori* segmentation or label field $\mathbf{L_e}$. Both $\mathbf{I_e}$ and $\mathbf{L_e}$ are defined on an identical lattice $\mathbf{X_e}$ and this need not be of the same dimension as the lattice on which $\mathbf{I}$ is defined. A prerequisite of the training

image $\mathbf{I_e}$ is that it must contain texture samples similar to those of $\mathbf{I}$ and these samples must be of sufficient size to allow the underlying statistics of the texture to be ascertained. The idea behind the non-parametric approach is to implicitly model $\mathbf{I}$ using heuristic measurements taken from $\mathbf{I_e}$. Similarly, the segmentation to be estimated $\mathbf{L}$ will be implicitly modelled on the label field $\mathbf{L_e}$. Thus rather than defining explicit parametric models for the intensity image and label field, two non-parametric energy based models are proposed.

Under this non-parametric framework, the segmentation of $\mathbf{I}$ can be expressed as the minimisation of the following energy function,

$$U(\mathbf{I}, \mathbf{L}, \mathbf{I_e}, \mathbf{L_e}) = \underbrace{U_d(\mathbf{I}, \mathbf{I_e})}_{datadriven} + \alpha \underbrace{U_r(\mathbf{L}, \mathbf{L_e})}_{regularisation} \tag{5.27}$$

This energy function is similar to that defined earlier in (5.19) for MAP estimation. In the non-parametric energy function, the likelihood or data driven term $U_d(\mathbf{I}, \mathbf{I_e})$ represents the energy between the observed intensity image $\mathbf{I}$ and the training example image $\mathbf{I_e}$ and the regularisation constraint on the label field is obtained by minimising an energy function $U_r(\mathbf{L}, \mathbf{L_e})$ between the label field to be estimated $\mathbf{L}$ and the training label field $\mathbf{L_e}$. The scalar α allows a weighting to be given to the regularisation term.

The data driven and regularisation energy functions $U_d(\mathbf{I}, \mathbf{I_e})$ and $U_r(\mathbf{L}, \mathbf{L_e})$ are estimated using the non-parametric sampling technique proposed by Wei and Levoy [213]. The optimum segmentation is then found by minimising the energy function in expression (5.27) where energy minimisation was achieved ICM algorithm. To kick start the segmentation process the label field is initialised by considering only the data driven term $U_d(\mathbf{I}, \mathbf{I_e})$.

In order to exploit the advantages of the multi-resolution approach, Mignotte extends the energy based model to work within a multi-scale environment. Similar to the Wei and Levoy algorithm, a Gaussian pyramid is used. Moving from coarse to fine resolution, energy minimisation is conducted at each resolution using the previous level segmentation as an initialisation.

5.10.8.1 Observations on Non-parametric Energy Based Models

The non-parametric approach proposed by Mignotte offers a fresh approach to the segmentation problem. It avoids the complicated process of model selection and model parameter estimation. Rather, it utilises *a priori* segmented image(s) of similar content to implicitly model the observed image to be segmented. The strength of the underlying implicit modelling technique has been proven in texture synthesis applications, where it was tested on a wide range of image types. Because of the relationship that exists between synthesis and analysis, if the modelling technique is successful within the texture synthesis domain, then it will be suitable for the texture analysis and ultimately the segmentation problem.

As with any segmentation, in order to constrain the nature of the solution, some regularisation over the label field is needed. Mignotte proposes the use of a heuristic measurement

between the label field to be estimated and an *a priori* estimated label field. While this implicit model can offer the possibility of obtaining complex neighbourhood configurations that cannot be expressed using parametric models it does suffer from some serious limitations that question its effectiveness.

Experimental results (chapter 6) indicate that this regularisation prior does not impose a strong enough smoothness over the segmentation. There are a number of outliers present in the segmentation result which reduce the accuracy of the segmentation. In addition, in order for the non-parametric energy model over the label field to be suitable, the topology of texture regions in the training image must be similar to that of the observed image. To illustrate, if the texture region labelled c_1 is located beside the texture region labelled c_2, then the label field model will be biased toward configurations in which texture c_1 is beside c_2. However, consider the case in the segmented image where the texture region c_1 is located beside the texture region c_3. Since this labeling configuration does not exist in the training label field the regularisation term will not support it.

Since the algorithm proposed in [146] is based on the multi-scale texture synthesis algorithm developed by Wei and Levoy [213], the multi-scale representation is of the form of a Gaussian pyramid structure. From the work on texture synthesis (chapters 2 and 4), it was shown that the Wei and Levoy algorithm is scale dependent and computationally expensive. This computational expense can be reduced by evoking an Approximate Nearest Neighbour [13] searching process and Mignotte adopted such a reduced search technique. However, this compromises the accuracy of the modelling process and the scale dependence is still an inherent feature of the Gaussian pyramid approach.

The wavelet based texture synthesis algorithm developed as part of this work was shown in chapter 4 to overcome the problems of scale dependence and computational expense. The comparison of results obtained using the wavelet based algorithm and that of Wei and Levoy demonstrated that more accurate texture reproduction could be achieved using the wavelet based algorithm. Thus, this wavelet based algorithm provides a more stable domain in which to model and replicate the example texture. It thus seems a natural transition to extend or alter the wavelet synthesis process to work within a segmentation framework.

5.11 Towards Example Based Segmentation

Based on the work of existing segmentation approaches, a new algorithm is proposed which will combine the strengths of some of the previous segmentation approaches together with the advantages of the wavelet based texture modelling framework developed in chapter 4. Bayesian segmentation offers the best means by which to combine observed data with *a priori* information necessary to constrain the solution. Unlike previous algorithms which have either been totally parametric or non-parametric, the approach taken here will be a mixture of both. The observed image will be modelled implicitly using an example training image of similar content. This

approach will be similar to the data based energy model developed by Mignotte. However, to account for dissimilarities between the two images, an additional constraint will be imposed on the model. In contrast to the non-parametric modelling of the observed image, the segmentation to be estimated or label field will be modelled using a parametric Markov Random Field. This new example based segmentation algorithm will be discussed in the next chapter.

6

Example Based Segmentation

The problem of segmentation was introduced in chapter 5 and a taxonomy of some the existing approaches to the problem was given. It was shown that a Bayesian framework is able to unify many of the existing approaches. Building on the strengths of that framework, this chapter will describe a new segmentation algorithm that has been developed as part of this work on example based processing. This new algorithm entitled "Example Based Segmentation" exploits the strengths of the non-parametric modelling technique developed as part of the work on texture synthesis. Similar to the DT-CWT TexSyn algorithm described in chapter 4 the new segmentation algorithm will be based in the wavelet domain.

6.1 An Energy Based Model

Using the notation defined earlier in chapter 5, let $\mathbf{I}$ and $\mathbf{L}$ denote the image to be segmented and its estimated segmentation or label field respectively. Both $\mathbf{I}$ and $\mathbf{L}$ are defined on an identical $M \times N$ lattice $\mathbf{X}$ and each site in $\mathbf{X}$ can be indexed using the spatial vector $\mathbf{x} = [x, y]^T$. The goal of the segmentation is to assign to each pixel $I(\mathbf{x}) \in \mathbf{I}$ a label $L(\mathbf{x})$ indicating to which region or class that pixel belongs. The Bayesian segmentation framework was presented in chapter 5 and it was shown how the MAP problem can be distilled to an energy minimisation exercise. Energies relate observed pixel intensities to measurements of texture as well as label smoothness. This energy function $U(\mathbf{I}, \mathbf{L})$ can be expressed as,

$$U(\mathbf{I},\mathbf{L}) = \underbrace{\sum_{\mathbf{x}\in\mathbf{X}} U_d(I(\mathbf{x}), T_{L(\mathbf{x})}(\mathbf{x}))}_{datadriven} + \underbrace{\sum_{\mathbf{x}\in\mathcal{N}} U_r(L(\mathbf{x}),\mathbf{L})}_{regularisation} \tag{6.1}$$

where $\mathcal{N}$ denotes the neighbourhood of sites surrounding $\mathbf{x}$, $U_d(I(\mathbf{x}), T_{L(\mathbf{x})}(\mathbf{x}))$ denotes the energy derived from a relationship between the intensity image $\mathbf{I}$ and the texture regions $\mathbf{T}_{\mathbf{L}(\mathbf{x})}$ and U_r is a regularisation energy. The MAP segmentation will be the particular configuration of the label field that maximises the posterior probability or minimises the energy function $U(\mathbf{I},\mathbf{L})$ across the entire lattice $\mathbf{X}$.

To estimate the data driven and regularisation terms over the intensity image and label field respectively, it is necessary to impose a model on both of these images. The two classes of image models available are parametric and non-parametric and these were discussed in chapters 2 and 5. Recall that parametric models attempt to model an image explicitly using some definable process. They characterise image behaviour using some set of parameters, each of which have to be estimated. Conversely, non-parametric processes offer no such definite model and instead implicitly model the image using measurements taken from the image itself or some set of similar images.

Parametric methods have been the most widely used models in image segmentation. The strength of the non-parametric technique has been recognised and has recently been applied to the problem of segmentation [146]. To illustrate how the non-parametric technique used in the texture synthesis problem can be applied to segmentation, it is necessary to re-express the texture synthesis problem as that of a global cost function to be minimised.

6.2 Texture Synthesis: A Global Cost Minimisation Problem

Given the example texture image $\mathbf{I_e}$, the goal of the texture synthesis process is to generate a new texture $\mathbf{I_s}$ which is perceptually similar to $\mathbf{I_e}$ and gives the impression of being generated from the same underlying process. Both $\mathbf{I_e}$ and $\mathbf{I_s}$ need not be defined on the same lattice. Let $\mathbf{X_e}$ denote the $M_e \times N_e$ lattice structure on which $\mathbf{I_e}$ is defined. Each pixel in $\mathbf{I_e}$ can be indexed using a spatial vector $\mathbf{p} = [x, y]^T$ such that $I_e(\mathbf{p})$ denotes the pixel at site $\mathbf{p}$. Similarly, $\mathbf{I_s}$ is defined on an $M_s \times N_s$ lattice $\mathbf{X_s}$ and indexed using the spatial vector $\mathbf{x}$. In the non-parametric modelling process each pixel $I_s(\mathbf{x}) \in \mathbf{I_s}$ is modelled directly using measurements taken from $\mathbf{I_e}$. As a consequence both $I_e(\mathbf{p})$ and $I_s(\mathbf{x})$ will take their values from the same set λ, where λ may represent the gray-scale or RGB values associated with the image $\mathbf{I_e}$.

Let $U(\mathbf{I_e}, \mathbf{I_s})$ denote an energy function between the example and synthesised texture images. $U(\mathbf{I_e}, \mathbf{I_s})$ is defined as,

$$U(\mathbf{I_e}, \mathbf{I_s}) = \sum_{\mathbf{x}\in\mathbf{X_s}} \min_{\mathbf{p}\in\mathbf{X_e}} D(N(\mathbf{x}), N(\mathbf{p})) \tag{6.2}$$

where $N(\mathbf{x})$ and $N(\mathbf{p})$ are used to denote the neighbourhood of pixel values centred at site $\mathbf{x}$ and $\mathbf{p}$ respectively. These neighbourhoods can be either causal or non-causal and are of size $w \times w$ pixels. The spatial distance $D(.,.)$ between two neighbourhoods is defined to be the squared sum of individual pixel differences.

The goal of the texture synthesis algorithm is then to find the particular configuration of $\mathbf{I_s}$ that minimises (6.2). This can be done using any MAP opimisation procedure and some example techniques were discussed in chapter 5. The approach adopted by Efros and Leung was derived from the Iterative Conditional Modes (ICM) algorithm proposed by Besag [20].

Note that in the description of the Efros and Leung texture synthesis algorithm given in chapter 4 an additional parameter ϵ was included in the energy function calculation. With this parameter ϵ a sample was drawn from the group of possible pixels. This ensured a certain randomness was maintained in the synthesised texture. In the segmentation case only the exact minimum is of interest and so $\epsilon = 0$.

Results of the non-parametric texture synthesis process are shown in chapters 2 and 4. The quality and accuracy with which the synthesised texture captures the characteristics of the example texture highlight the strength of the non-parametric modelling technique. When compared to the rigid parametric modelling process it can be said that the success of the non-parametric modelling technique lies in the fact that rather than defining an explicit model and trying to fit this model to a given texture, the process rather attempts to implicitly model the texture by heuristically measuring image statistics from the example texture itself. The result is a more flexible modelling process which allows complicated image configurations which otherwise would be difficult to mathematically define using a parametric model.

6.3 From Synthesis to Segmentation

Figure 6.1 shows an observed image to be segmented $\mathbf{I}$ and 4 example texture examples which are representative of the textures which characterise the regions in $\mathbf{I}$. The idea behind the example based segmentation algorithm is to use these example training textures as a guide for obtaining the segmentation $\mathbf{L}$ of $\mathbf{I}$. The modelling process will be non-parametric and so $\mathbf{I}$ will be modelled implicitly using only information in the example training textures. For this implicit modelling technique to be viable the example training images must satisfy the following conditions:

- The example training textures must be statistically similar to the textures found in the image to be segmented.
- Each of the example training textures must be large enough to capture the underlying statistics of the infinite texture pattern.

Let $\mathbf{T_e} = \{\mathbf{T_{e_1}}, \mathbf{T_{e_1}}, \cdots, \mathbf{T_{e_K}}\}$ denote the set of K input example textures and let $\mathbf{L_e} = \{L_{e_0}, L_{e_1}, \cdots, L_{e_K}\}$ be the class labels associated with each of these textures. Each of the i

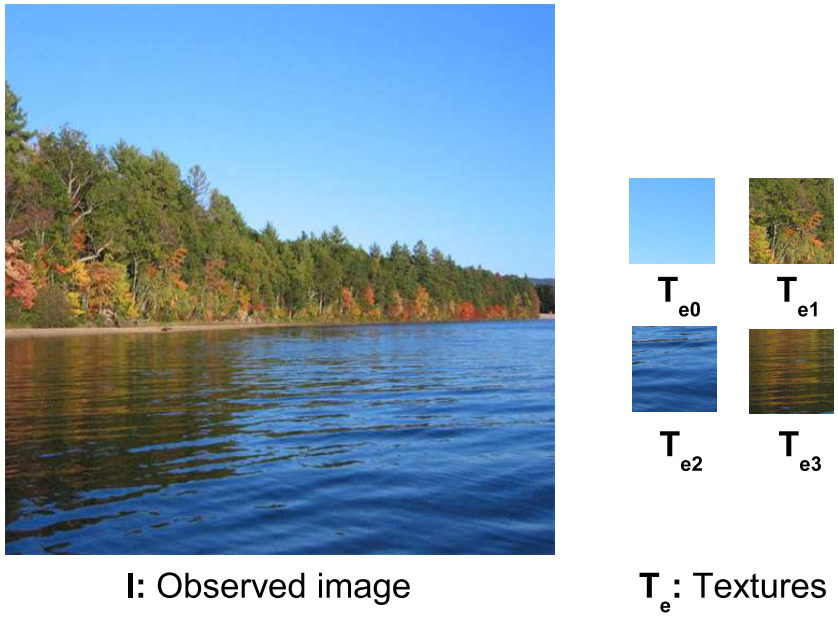


Figure 6.1: *The image to be segmented is considered as an ensemble of textures where each texture represents a region of interest. The idea behind example based segmentation is that given a set of some of these textures* $\mathbf{T_e}$*, the image* $\mathbf{I}$ *will be segmented by modelling the image* $\mathbf{I}$ *using information in* $\mathbf{T_e}$*.*

example textures is defined on its own separate $M_{e_i} \times N_{e_i}$ lattice $\mathbf{X_{e_i}}$ for $i = 1, \cdots, K$ and each site in $\mathbf{X_{e_i}}$ can be indexed using the spatial vector $\mathbf{p}$.

The estimate of the most likely label $L(\mathbf{x})$ for the pixel $I(\mathbf{x})$ is obtained using the following steps:

1. Construct the $w \times w$ non-causal neighbourhood $N(\mathbf{x})$ centred at site $\mathbf{x}$. This neighbourhood contains the intensity values of the pixels surrounding $\mathbf{x}$.

2. Compare $N(\mathbf{x})$ to all possible neighbourhood in each of the example textures. The distance between two neighbourhoods is denoted by $D(N(\mathbf{x}), \mathbf{N}(\mathbf{p}))$. There are a number of different choices for the perceptual similarity between two neighbourhoods and in this implementation the L_2 distance is used. The set of distances between $N(\mathbf{x})$ and $N(\mathbf{p}) \forall \mathbf{p} \in \mathbf{X_{e_i}}, i = 1, \cdots, K$ is constructed and given as,

$$D_x = \{D(N(\mathbf{x}), N(\mathbf{p})), \forall \mathbf{p} \in \mathbf{X_{e_i}}, i = 1, \cdots, K\} \tag{6.3}$$

3. The best matching neighbourhood is found and given as,

$$N_{best} = \min_{\mathbf{p} \in \mathbf{X_{e_i}}} D(N(\mathbf{x}), N(\mathbf{p})) \forall i = 1, \cdots, K \tag{6.4}$$

4. The label associated with the texture from which N_{best} is located is then taken as the most likely label for $I(\mathbf{x})$. This local energy minimisation at the site $\mathbf{x}$ is given by,

$$U(I(\mathbf{x}), \mathbf{T_e}) = \sum_{i=1}^{K} \min_{\mathbf{p} \in \mathbf{X_{e_i}}} D(N(\mathbf{x}), N(\mathbf{p})) \tag{6.5}$$

Minimising the local energy function given in (6.5) at each site is equivalent to finding the maximum likelihood (ML) segmentation of $\mathbf{I}$ given the example texture images. This ML segmentation is equivalent to minimising the following energy function,

$$U(\mathbf{I}, \mathbf{T_e}) = \sum_{\mathbf{x} \in \mathbf{X}} \sum_{i=1}^{K} \min_{\mathbf{p} \in \mathbf{X_{e_i}}} D(N(\mathbf{x}), N(\mathbf{p})) \tag{6.6}$$

To illustrate the strength of this non-parametric energy based model, ML segmentations obtained using it and an example parametric model are presented and compared in the next section.

6.4 ML Segmentation: Parametric versus Non-Parametric

Using the non-parametric energy based model defined in (6.6) and the 2D Autoregressive (AR) model discussed in chapter 5, two ML segmentation implementations were carried out, the results of which are shown in Figures 6.2 and 6.3.

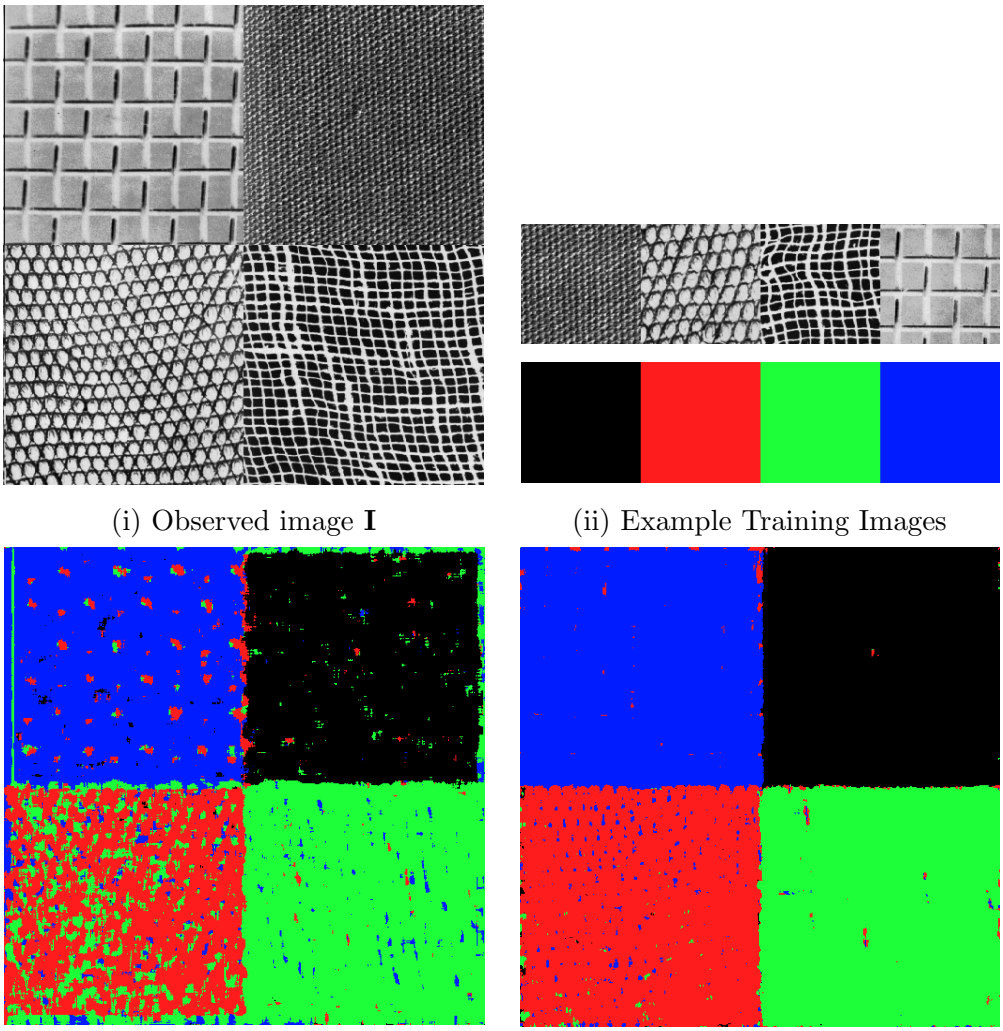
The original images to be segmented in Figures 6.2 and 6.3 are of size 512×512 and 768×768 respectively. The example training images used were all of size 128×128. These example textures were not sourced from the observed image and so although they are perceptually similar, they are not identical. Note that in the diagram each example texture is adjoined; however in the algorithm implementation each example texture is considered separately and so there is no ambiguity at boundary regions where two or more textures meet. All of the segmentations shown in 6.2 and 6.3 use a neighbourhood support of 9×9.

The ML segmentation obtained using the AR process was estimated using the following steps:

1. Estimate the model parameters $\mathbf{a}$ for each of the K example textures. A causal 9×9 neighbourhood is used as the neighbourhood support. The model parameters $\mathbf{a}$ are trained on the example textures and the means by which there were estimated is given in appendix C.

2. At each site the prediction error $e_i(\mathbf{x})$ for each texture example i is calculated. This error is based on the difference between the true pixel value $I(\mathbf{x})$ and the AR estimate $\hat{I}(\mathbf{x})$ and is given as,

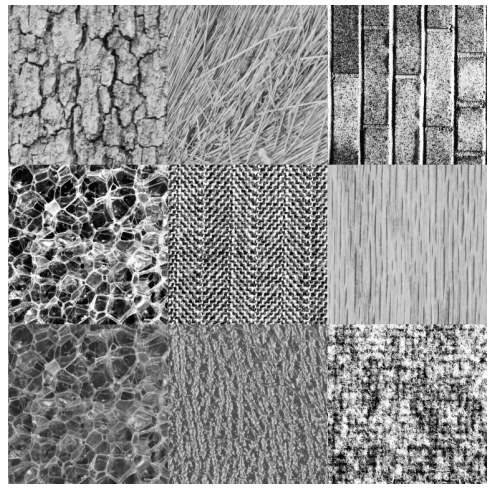
$$e_i(\mathbf{x}) = abs(I(\mathbf{x}) - \hat{\mathbf{I}}(\mathbf{x})) \tag{6.7}$$



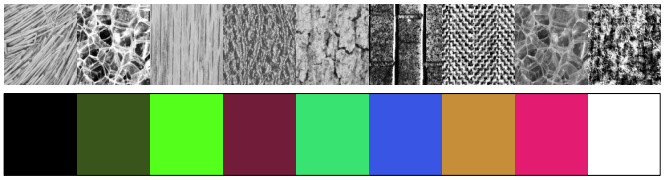
(i) Observed image **I** (ii) Example Training Images

(iii) Parametric ML Segmentation (iv) Non-Parametric ML Segentation

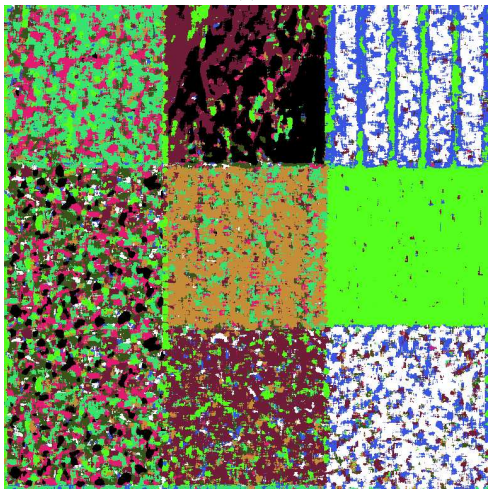
Figure 6.2: *Comparing parametric and non-parametric maximum likelihood (ML) segmentations. Images (i) and (ii) are the* 512×512 *intensity image to be segmented and the four* 128×128 *texture samples and their associated labels. The AR model parameters were trained on these textures and segmentation was performed on a* 9×9 *block-by-block basis (iii). The neighbourhood support used was a causal* 9×9 *neighbourhood. For the non-parametric method (iv), the neighbourhood used was non-causal and of size* 9×9.



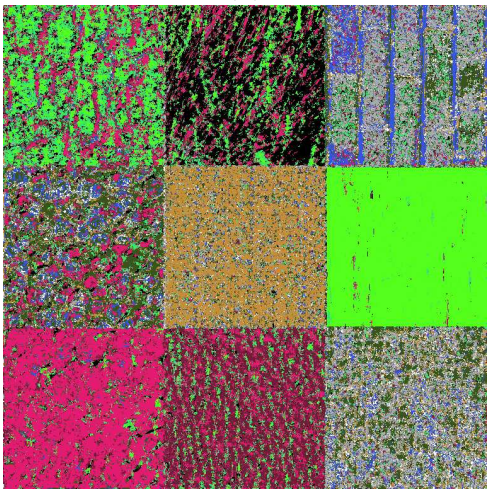
(i)



(ii)



(iii) Parametric ML Segmentation



(iv) Non-Parametric ML Segentation

Figure 6.3: *Comparing the Maximum likelihood segmentations of the 2D AR (parametric) and non-parametric energy models. Images (i) and (ii) are the* 768×768 *original image and the nine* 128×128 *example textures and their associated labels. The AR model parameters* a_k *were trained on these textures and segmentation was performed using a* 9×9 *block by block basis, (iii). The neighbourhood support used was a causal* 9×9 *neighbourhood. For the non-parametric method (iv), the neighbourhood used was non-causal and of size* 9×9.

3. To improve the segmentation, errors are minimisesd on a 9×9 block basis rather than at each individual site. Let $\mathcal{N}$ denote the block of 9×9 predication errors. At each distinct block $\mathcal{N}$ in $\mathbf{I}$, the label chosen for those set of sites will be that which minimises the sum of the predication errors at each site.

6.4.1 Efficiency

The AR model described here is relatively simple to implement and is based on a 9×9 neighbourhood support. More complicated models have been developed and a more accurate segmentation could be obtained. On the downside this would complicate the estimation process. Since the non-parametric model is simple it was felt that the AR model represented a similar simple parametric model. Before analysing results, it also worth mentioning the efficiency of both processes. One of the attractive features of parametric modelling is their efficiency and the AR segmentation process implemented as part of this work was much faster than the equivalent non-parametric segmentation. The non-parametric modelling process is computationally expensive given that for each pixel to be labelled, its neighbourhood has to be compared to every possible neighbourhood in the example texture image set. As a means to speed up this neighbourhood searching process, the computationally intensive neighbourhood searching associated with the non-parametric method could be implemented on the GPU. This would be very similar to the GPU texture synthesis implementation discussed in appendix A. This would speed up the process considerably and future work in this area will investigate a GPU implementation of the segmentation algorithm.

6.4.2 Accuracy

While high computation is an issue, the main focus here is on the accuracy of the modelling process. Considering Figure 6.2 initially it is clear that the segmentations obtained using the AR and non-parametric modelling processes both successfully detect all four texture regions. The boundary between each texture region is sharper in the non-parametric method and overall the number of misclassified pixels is larger in the AR based segmentation. This suggests that the non-parametric modelling process is more accurate in capturing the behaviour of the underlying texture characteristics.

Figure 6.3 is a more difficult problem to segment. Before analysing the segmentation, consider first the observed image. The texture regions on the centre left and the bottom left are very similar. The only difference between these textures is their intensity, their frequency characteristics are very similar.

Also, consider the top right brick texture. The intensity of this texture is lighter than the similar example texture which is located above the blue label in (ii). Since the non-parametric modelling process will be trained on this texture, it should provide an accurate representation of the texture found in the observed image. The obvious difference in intensity values between the

observed and example textures will compromise the accuracy of the non-parametric modelling process. This is one of the limitations of the non-parametric model whereby the accuracy of the modelling process will be dependent on the similarity between the example texture and the observed texture to be labelled. In general the AR model will offer a better generalisation over the texture and for cases when the example texture and observed texture differ the AR result will be better. However, in cases where the example texture and observed texture are perceptually similar, since the non-parametric model will better capture the example texture behaviour, the non-parametric segmentation will be better.

The strengths of both AR and non-parametric models are evident from the ML segmentations shown in Figure 6.3 (iii) and (iv). As expected segmentation (iv) fails to detect the brick texture given that the example texture and observed textures are not similar enough. Segmentation (iii) performs better in this case. For the two similar texture regions at the centre left and bottom, segmentation (iv) is better and both of these textures have been isolated. Segmentation (iii) fails to detect the difference between these two regions and the segmentation is very speckled here. As to which segmentation is better, it is arguable since both are good in different areas. However, because segmentation (iv) has identified each of the textures and each region contains more correctly labelled pixels than incorrectly labelled pixels, it is favoured. This dominance of correctly labelled pixels is important as it will allow the incorrectly labelled pixels to be re-estimated using a suitable smoothing function.

6.4.3 Towards Smoothness

Similar to the modelling of the data driven energy term, the regularisation energy is normally calculated by imposing a parametric model over the label field. There have been many different models developed and generally, because the label field should exhibit smooth behaviour, these parametric models perform well in capturing this behaviour. The regularisation used in the segmentation algorithm developed here will be based on the parametric potts model. The potts model is a class of Markov Random Field model which has been proven to work well in introducing some smoothness into the segmentation. Before discussing the combined parametric and non-parametric segmentation approach taken here, the fully non-parametric approach taken by Mignotte [146] will be considered.

6.5 A Non-Parametric Segmentation Approach

Using a non-parametric energy model similar to that given in (6.6), Mignotte exploited the non-parametric modelling technique further in order to introduce some regularisation into the segmentation. In addition to calculating the energy between intensity neighbourhoods, a regularisation energy was calculated between label neighbourhoods. The result is a fully non-parametric segmentation algorithm where both the intensity image and the label field are modelled implicitly

using heuristic measurements from the example training image and its *a priori* segmentation.

Similar to the Wei and Levoy synthesis algorithm, the segmentation algorithm proposed in [146] performs the non-parametric modelling process in a multi-resolution domain. Both $\mathbf{I}$ and $\mathbf{I_e}$ are represented at different resolutions using an L level Gaussian pyramid structure. Beginning at a coarse resolution and moving to the highest resolution, the segmentation at each level l which minimises the following non-parametric energy function is estimated.

$$U^l(\mathbf{I}, \mathbf{I_e}, \mathbf{L}, \mathbf{L_e}) = \sum_{\mathbf{x} \in \mathbf{X^l}} \min_{\mathbf{p} \in \mathbf{X_e^l}} \{D(N(\mathbf{x}), N(\mathbf{p})) + \alpha D(N_L(\mathbf{x}), N_L(\mathbf{p}))\} \tag{6.8}$$

where α is a scalar that provides a relative weighting between the data driven and regularisation energy terms and $D(N_L(\mathbf{x}), N_L(\mathbf{p}))$ is the distance between the two label neighbourhoods $N_L(\mathbf{x})$ and $N_L(\mathbf{p})$. The initial estimate of the segmentation at the coarsest level is obtained using the data driven energy only, i.e. $\alpha = 0$ and the segmentations at the higher resolution levels $l < L$ are initialised based on the estimates from the previous coarser level $l+1$. The energy minimisation process at each level is obtained using the ICM algorithm.

6.5.1 Details

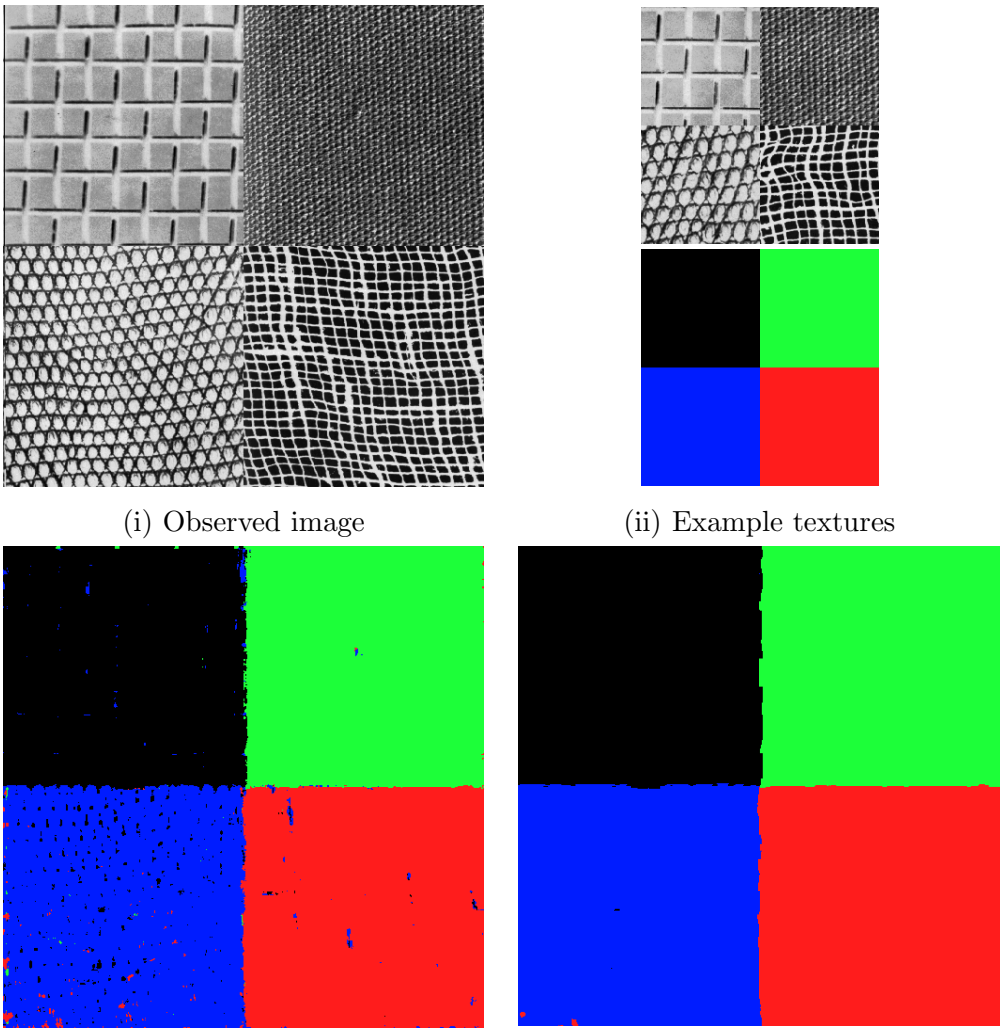
The data driven and regularisation energies were normalised so that their values lay within the interval $[0, 1]$. The distance between two intensity neighbourhoods was defined as the L_2 distance. This distance was normalised then to lie in the interval $[0, 1]$. The distance metric between label neighbourhoods was not specified by the author but in the implementation developed as part of this work, the distance between two label neighbourhoods N_{L_1} and N_{L_2} is defined as,

$$D(N_{L_1}(\mathbf{x}), N_{L_2}(\mathbf{x})) = \sum_{\mathbf{x} \in N_L} \delta(N_{L_1}(\mathbf{x}), N_{L_2}(\mathbf{x})) \tag{6.9}$$

$\delta(.)$ is the delta Kronecker function defined such that , $\delta(N_{L_1}(\mathbf{x}), N_{L_2}(\mathbf{x})) = 1 \Leftrightarrow L_1(\mathbf{x}) = L_2(\mathbf{x})$. Although this fully non-parametric algorithm was novel, it suffered from a drawback which limits its applicability. In order for the Mignotte algorithm to be suitable for a given segmentation problem, the example training image must be similar in content and configuration to the image to be segmented. Because the segmentation is trained on the *a priori* segmentation of the example training image, this modelling process will only support label configurations found in the example label field.

6.5.2 Mignotte Results

Figures 6.4 and 6.5 show two segmentations obtained using the Mignotte segmentation algorithm. Figure 6.4 was obtained using a single resolution implementation of the algorithm while the segmentation in Figure 6.5 was obtained using a multi-resolution implementation based on

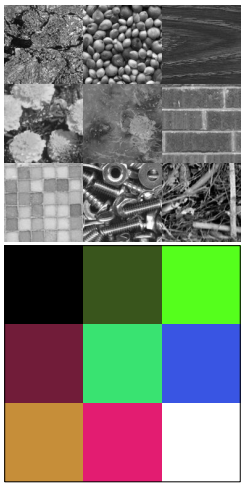


(i) Observed image (ii) Example textures

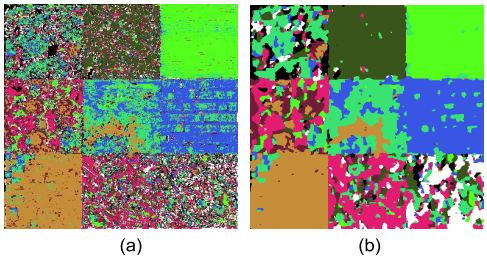
(iii)ML Segmentation (iv)MAP Segmentation

Figure 6.4: *Segmentation obtained using the Mignotte single resolution energy model. Images (i) and (ii) are the* 512×512 *observed image and the* 256×256 *example training images. Segmentation (iii) was obtained using* $\alpha = 0$ *and segmentation (iv) was obtained after* 3 *iterations of* $\alpha = 1$*. A neighbourhood width of* $w = 9$ *was used in the searching process.*

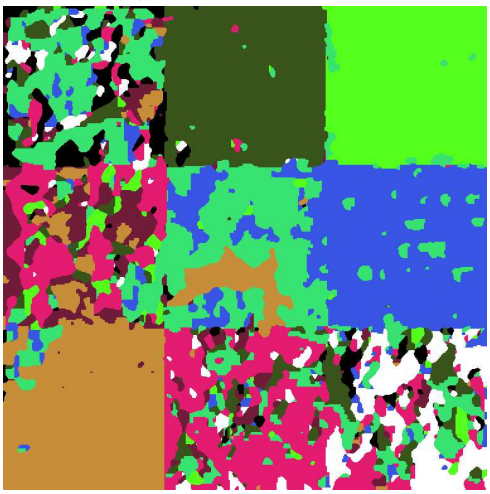
(i) Observed image



(ii) Training textures



(iii) Level 1 Segmentations



(iv) Final MAP Segmentation

Figure 6.5: *Segmentation obtained using the Mignotte multi-resolution energy model. Images (i) and (ii) are the* 768×768 *observed image and the* 384×384 *example training images. Images (iii) are the level 1 segmentation obtained at (a) the first* ($\alpha = 0$) *and (b) the third iteration. Image (iv) is the final segmentation obtained at the highest resolution (image) space using the level 1 segmentation as an initialisation.*

Gaussian pyramids. In both cases the observed images are composed of distinct texture regions and the example images contain textures of similar content and in the same configuration. In the non-parametric modelling process a neighbourhood width of $w_1 = 9$ was used and for the multi-resolution segmentation, the initial segmentation was obtained at $L = 1$ of the Gaussian pyramid. The weighting between the two energy terms was set to $\alpha = 1$ for all iterations > 1.

Note that to speed up the processes, Mignotte suggested using the Approximate Nearest Neighbourhood (ANN) [13] searching process in order to remove the exhaustive searching technique necessary for each pixel to be segmented. A similar approach was adopted by Wei and Levoy [213] and the result is a fast process but at the expense of reduced accuracy. The results shown here did not implement this ANN search and so are the optimum segmentation results obtained using the Mignotte algorithm.

Before analysing the segmentations, consider the observed and example textures. The observed image in Figure 6.4 contains four distinct textures which are all very different. The example textures given are similar to the observed texture and so the non-parametric model should perform well. The segmentation of Figure 6.5 (i) is a more difficult problem. For example consider the centre texture region in (i). As with the segmentation problem discussed in Figure 6.3, the example texture (centre (ii)) does not contain the same light intensity values that are associated with the observed texture. Therefore, the texture model will not support this light intensity region since it does not exist in the sample texture. Similarly, the top left example texture has a large black region. This black region is similar to the black regions found in the centre left texture and the bottom right texture. This could account for some inaccuracies in the modelling process. These observations summarise two issues or concerns of the non-parametric modelling process.

Firstly, is the example texture large enough and similar enough to the observed texture? The strength of the segmentation will be largely determined by the ability to accurately model the observed texture using information from the example texture. The second issue concerns the size of neighbourhood that should be used in the modelling process. The size of this neighbourhood is related to the notion of texture scale. For the modelling process to be accurate, this neighbourhood should be large enough to capture the biggest image feature present in the texture. This notion of texture scale and suitable neighbourhood sizes was discussed in chapter 2 and it was found that one of the main limitations of the non-parametric modelling technique is the scale dependence of the modelling process.

The segmentation shown in Figure 6.4 is very good and this suggests that the example textures are large and similar enough to capture the behaviour of the observed textures. As expected the segmentation in Figure 6.5 is more problematic. The final segmentation (iv) shows that the top centre, top right, centre right and bottom left textures have all been correctly segmented. As predicated the modelling process has difficulty with the light intensity values in the centre texture and mis-classifies it. Similarly, the darker regions in the centre left, bottom centre and right have been mis-classified and taken to be associated to the black labelled top left

texture. Overall, each region has more correctly labelled pixels than incorrectly labelled pixels and so the result is adequate

Looking at the segmentations it can be said that overall the Mignotte method performs well. Figure 6.4 (iii) is segmentation at the first iteration where $\alpha = 0$ and (iv) is the MAP segmentation obtained when the energy function in (6.8) has been minimised. This segmentation is smooth and each region has been correctly identified. There is some ambiguity at the internal boundaries between regions but overall the result is good. The segmentation shown in Figure 6.5 is not as smooth and there are a lot of misclassified pixels. The centre and top left regions have been confused and assigned the same label.

Some observations on the Mignotte algorithm were given in chapter 5 and it was found that although a fully non-parametric algorithm is intuitive and novel, it suffers from three limitations which the new segmentation algorithm developed here attempts to overcome.

- **A Flexible Prior Constraint**
  The regularisation term and indeed to a degree the data driven term require the example and observed images to have the same configurations. This severely limits the flexibility of the Mignotte algorithm. In the new segmentation algorithm developed as part of this work, each texture region and its associated segmentation will be considered separately. The image to be segmented will be modelled using the non-parametric model which will serve well in capturing the wide variability in image behaviour. Intuitively, the segmentation or label field should be smooth and so to capture these smooth regions, a parametric model will be imposed over the label field. This parametric model is simple to implement and flexible given that it does not depend on the *a priori* segmentations of the example texture images.

- **The Outlier Class**
  To make the approach more robust and suitable for object recognition, an outlier class has been introduced. The outlier class is assigned to any texture that is found in the image which is not considered to be similar enough to any of the input example textures. This outlier class is useful in cases when only certain regions or objects are of interest in the segmentation. All other regions can then be labelled as outliers.

- **Wavelet Analysis**
  The Mignotte algorithm was derived directly from the Wei and Levoy [213] texture synthesis algorithm and so the multi-resolution analysis takes place over a Gaussian pyramid structure. Chapters 2 and 4 showed some results of the Wei and Levoy algorithm and it was found that for all texture types the DT-CWT TexSyn algorithm developed as part of this work generated better results. This suggested that the wavelet based analysis process associated with the DT-CWT TexSyn was more suited for texture modelling. Based on this observation, the segmentation algorithm developed here will be based in the wavelet domain.

To establish the principles of this new example based segmentation algorithm, the data driven and regularisation terms will first be considered for the single resolution case.

6.6 Single Resolution Segmentation

Using the non-parametric energy based model given in (6.6) the next step in the algorithm development is to smooth the segmentation by imposing a regularisation energy over the label field. To do this the label field will be modelled using the Potts model. Under this model, the energy at each site $\mathbf{x}$ is given as,

$$U(L(\mathbf{x})) \propto \sum_{k=1}^{P} \beta_k (1 - \delta(L(\mathbf{x}), L(\mathbf{x} + \mathbf{q_k}))) \tag{6.10}$$

where the neighbourhood of labels around the site $\mathbf{x}$ are obtained using the P offset vectors $\mathbf{q}_k$, the P coefficients of the model are denoted by β_k for $k = 1, \cdots, P$ and $\delta(.)$ is the delta Kronecker function defined such that , $\delta(\mathbf{x}, L(\mathbf{x} + \mathbf{q}_k)) = 1 \Leftrightarrow L(\mathbf{x}) = L(\mathbf{x} + \mathbf{q}_k)$. The model parameter β denotes the matrix weights that are attached to each neighbouring site in the estimation process. The order of the model is governed by size of neighbourhood support used in the modelling process. A high order model will result is larger neighbourhood dependencies and so smoother regions. Too large a model and the segmentation can be overly smooth. In addition, a high order model signifies larger computational costs. There are many different options as to how the Potts model may be set up. In this implementation, both third order and fifth order models were considered and the parameter β was set according to the relative distance from each site to the centre site. For example, the third order model had a value of β given as,

$$\beta = \begin{bmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & 0 & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{bmatrix} \tag{6.11}$$

Combining this parametric prior with the non-parametric likelihood term given in (6.6) gives the following energy term which the segmentation will aim to minimise.

$$U(\mathbf{I}, \mathbf{L}, \mathbf{T_e}, \mathbf{L_e}) = \sum_{\mathbf{x} \in \mathbf{X}} \left(\sum_{i=0}^{K-1} \min_{\mathbf{p} \in \mathbf{X_e}} D(N(\mathbf{x}), N(\mathbf{p})) + \alpha \sum_{k=1}^{P} \beta_k (1 - \delta(L_e(\mathbf{p}), L(\mathbf{x} + \mathbf{q_k}))) \right) \tag{6.12}$$

The scalar α is introduced in order to provide a weighting between the data driven and regularisation terms. To obtain the minimum energy configuration, the ICM process is used. At each site $\mathbf{x} \in \mathbf{X}$, the label which minimises the local energy at that site is found and this process is repeated until convergence is achieved. Pixel labels are updated in a checkerboard scanning order. The initial segmentation is obtained using the data driven energy only, i.e. $\alpha = 0$.

6.6.1 Single Resolution Results

Figures 6.6, 6.7 and 6.7 show segmentations obtained using the single resolution energy model in (6.12). The colour information contained in the images in Figure 6.6 was included in the modelling process by first considering the image in YUV space. The transformation from RGB to YUV is linear and is given in chapter 4. The neighbourhood around each site $\mathbf{x}$ is now made up of the three Y, U and V neighbourhoods and the similarity between two neighbourhoods is taken as the sum of the distances between the Y, U and V neighbourhoods. Both ML and MAP segmentations are good and the number of misclassified pixels is low. The blue and black regions have been correctly segmented. However, looking at the red and green regions there are some groups of mislabelled sites. These mis-classifications occur at locations where the intensity value associated with these pixels are dark. In the example textures, the bark texture (green) has a dark region on the left and the segmentation process is taking the label for this dark region and assigning it to the other dark regions in the observed image. This mis-classification could be avoided in some cases by increasing the neighbourhood size or ensuring that the example textures are suitably representative of the observed textures.

In Figures 6.7 and 6.8 the result of two segmentations of an 256×512 observed image is shown. These segmentations are both good and this success may be attributed to the fact that both of the textures are considerably different and the 128×128 example texture samples are similar to the observed texture. It is interesting to note that in the segmentation in Figures 6.7, both the observed and example textures have the same orientation while in Figure 6.8 the observed textures are of different orientation to the example textures. In general the modelling process is not rotation invariant but since the two textures are so different this rotation does not affect the segmentation.

Building on the strength of the energy model given in (6.12), the next stage in the algorithm development was to introduce an outlier class.

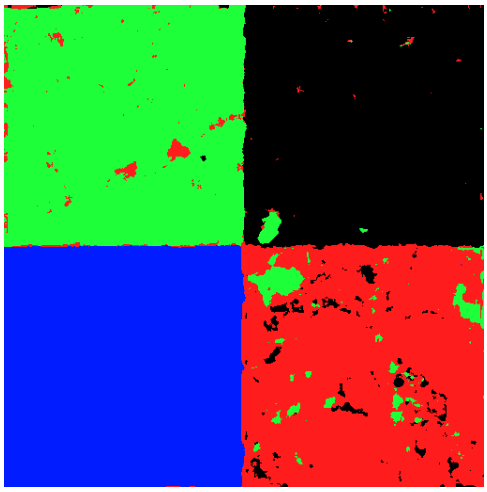
6.7 The Outlier Class

The outlier class label is assigned to any texture region found in the observed image that is not similar to those in the example texture training set. The addition of the outlier class allows the example based segmentation algorithm to be used in circumstances where only one of the regions or objects in the observed image are of interest. Finding objects in images is a large concern in content based retrieval systems and the addition of an outlier class would enable the algorithm to be used in applications where a given object is to be located in other images.

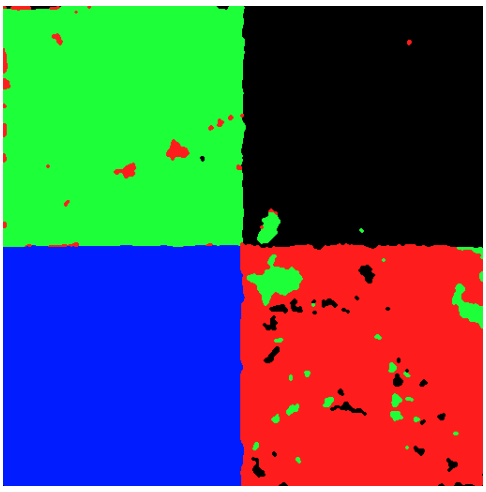
Inherent to the non-parametric modelling technique is a neighbourhood searching process whereby the neighbourhood of each pixel to be labelled is compared to all possible neighbourhoods in the example texture set. In order to introduce an outlier class, it necessary to specify an expected similarity distance between any two neighbourhoods which belong to the same class.

(i) Observed image

(ii) Example textures

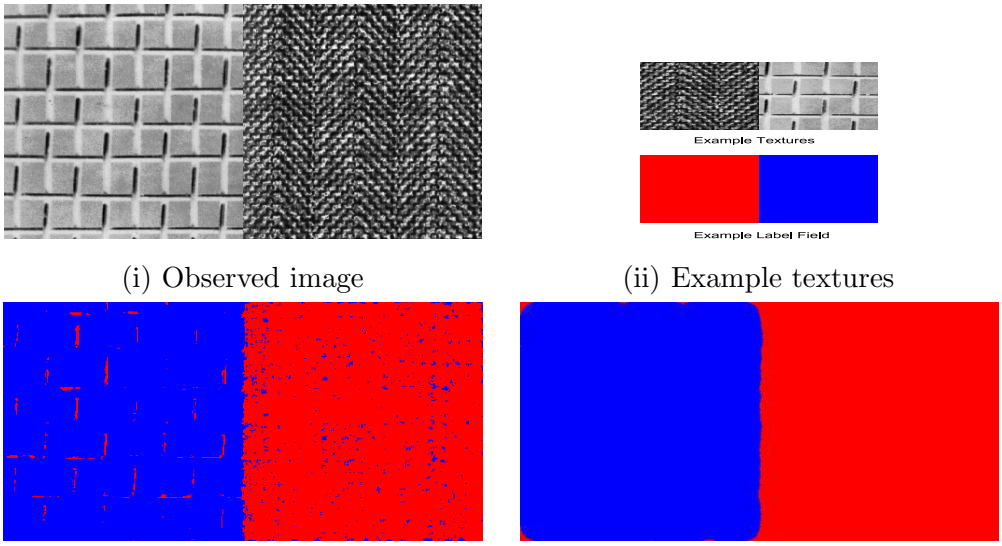


(iii) ML Segmentation



(iv) MAP Segmentation

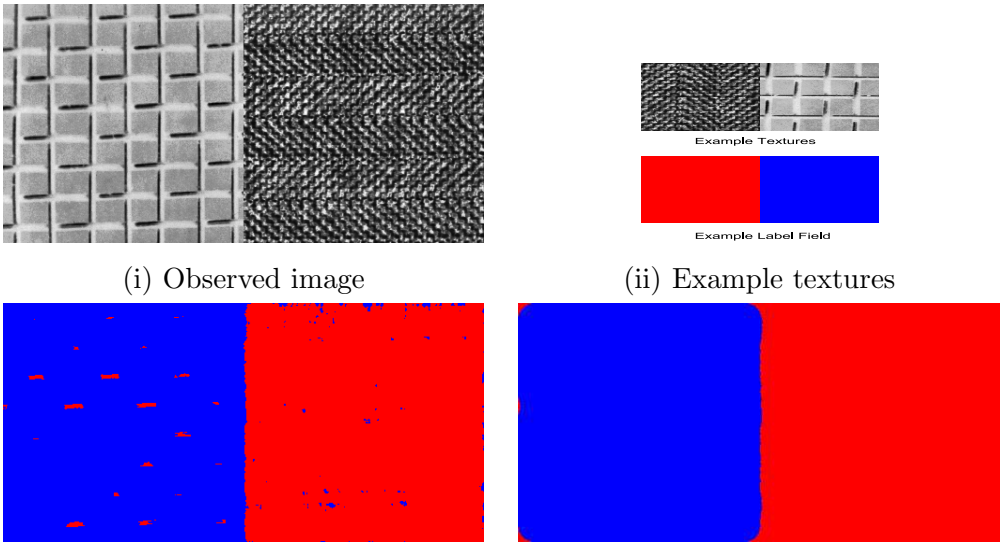
Figure 6.6: *Single resolution segmentation obtained by minimising the energy function given in (6.12). Images (i) and (ii) are the* 512×512 *observed image to be segmented and the set of four* 128×128 *example textures similar to those found in image (i). Images (iii) and (iv) are the ML* ($\alpha = 0$) *and MAP* ($\alpha = 1$) *segmentations of (i) obtained using a neighbourhood width of* $w_1 = 9$.



(i) Observed image (ii) Example textures

(iii) ML Segmentation (iv) MAP Segmentation

Figure 6.7: *Single resolution segmentation obtained by minimising the energy function given in (6.12). Images (i) and (ii) are the* 512×256 *image to be segmented and the* 256×128 *example textures and label images. Images (iii) is the ML segmentation* ($\alpha = 0$) *in (6.12) and image (iv) is the MAP segmentation obtained using* $\alpha = 1$. *Energies were calculated using a neighbourhood width of* $w_1 = 5$.



(i) Observed image (ii) Example textures

(iii) ML Segmentation (iv) MAP Segmentation

Figure 6.8: *Single resolution segmentation using the energy model given by (6.12). Images (i) and (ii) are the* 512×256 *image to be segmented and the* 256×128 *example textures and label images. Images (iii) is the ML segmentation* ($\alpha = 0$) *in (6.12) and image (iv) is the MAP segmentation obtained using* $\alpha = 1$. *Energies were calculated using a neighbourhood width of* $w_1 = 9$.

If the neighbourhood of the pixel to be labelled does not fall under this threshold, then it is assumed that this pixel does not belong to this class and so it is either labelled another class (if applicable) or the outlier class label.

To determine the threshold under which a class will be assigned the texture label, the distribution of distances of neighbourhoods which belong to the same class was examined. To calculate this distribution each possible neighbourhood in the example texture was compared to every other possible neighbourhood in the same example texture. Therefore, for an $M_e \times N_e$ sized texture, there will be $(M_e \times N_e)^2$ distances and this set is composed of the following entires,

$$\Upsilon = \{D(N(\mathbf{p}), N(\mathbf{x})), \forall \mathbf{p}, \mathbf{p} \in \mathbf{X_e}\} \tag{6.13}$$

where Υ is the distribution of neighbourhood distances and $\mathbf{x}$ and $\mathbf{p}$ are two spatial vectors used to index the lattice $\mathbf{X_e}$ on which the example texture is defined.

Figure 6.9 shows the distribution of neighbourhood distances for some example textures. From the plots of these distributions these are assumed to be Gaussian[1], and so the mean and variance of each distribution can be calculated from the following equations.

$$\mu_{T_e} = \frac{1}{(M_e N_e)^2} \sum_{\mathbf{p} \in \mathbf{X_e}} \sum_{\mathbf{q} \in \mathbf{X_e}} D(N(\mathbf{p}), N(\mathbf{q})) \tag{6.14}$$

$$\sigma^2_{T_e} = \frac{1}{(M_e N_e)^2} \sum_{\mathbf{p} \in \mathbf{X_e}} \sum_{\mathbf{q} \in \mathbf{X_e}} (D(N(\mathbf{p}), N(\mathbf{q})) - \mu_{T_e})^2 \tag{6.15}$$

Using these statistics and assuming for the moment that only one example texture T_e is provided as an input, the likelihood of any pixel being assigned that same label L_e as that texture can be calculated from the following Gaussian expression,

$$p(L(\mathbf{x}) = L_e | \mathbf{I}, \mathbf{T_e}) = \frac{1}{\sqrt{2\pi\sigma^2_{T_e}}} exp\left(-\frac{(D_{L_e} - \mu_{T_e})^2}{2\sigma^2_{T_e}}\right) \tag{6.16}$$

where D_{L_e} is distance between the neighbourhood of the pixel to be labelled $N(\mathbf{x})$ and the most similar neighbourhood in T_e, i.e.,

$$D_{L_e} = \min_{\mathbf{p} \in \mathbf{X_e}} D(N(\mathbf{x}), N(\mathbf{p})) \tag{6.17}$$

Conversely, the likelihood of the pixel being labelled as an outlier L_o is given as,

$$p(L(\mathbf{x}) = L_o | \mathbf{I}, \mathbf{T_e}) = \frac{1}{\sqrt{2\pi\sigma^2_{T_e}}} exp\left(-\frac{K^2}{2\sigma^2_{T_e}}\right) \tag{6.18}$$

where K is some constant.

[1] A long tail distribution would be a better assumption but for simplicity they will assumed to be Gaussian.

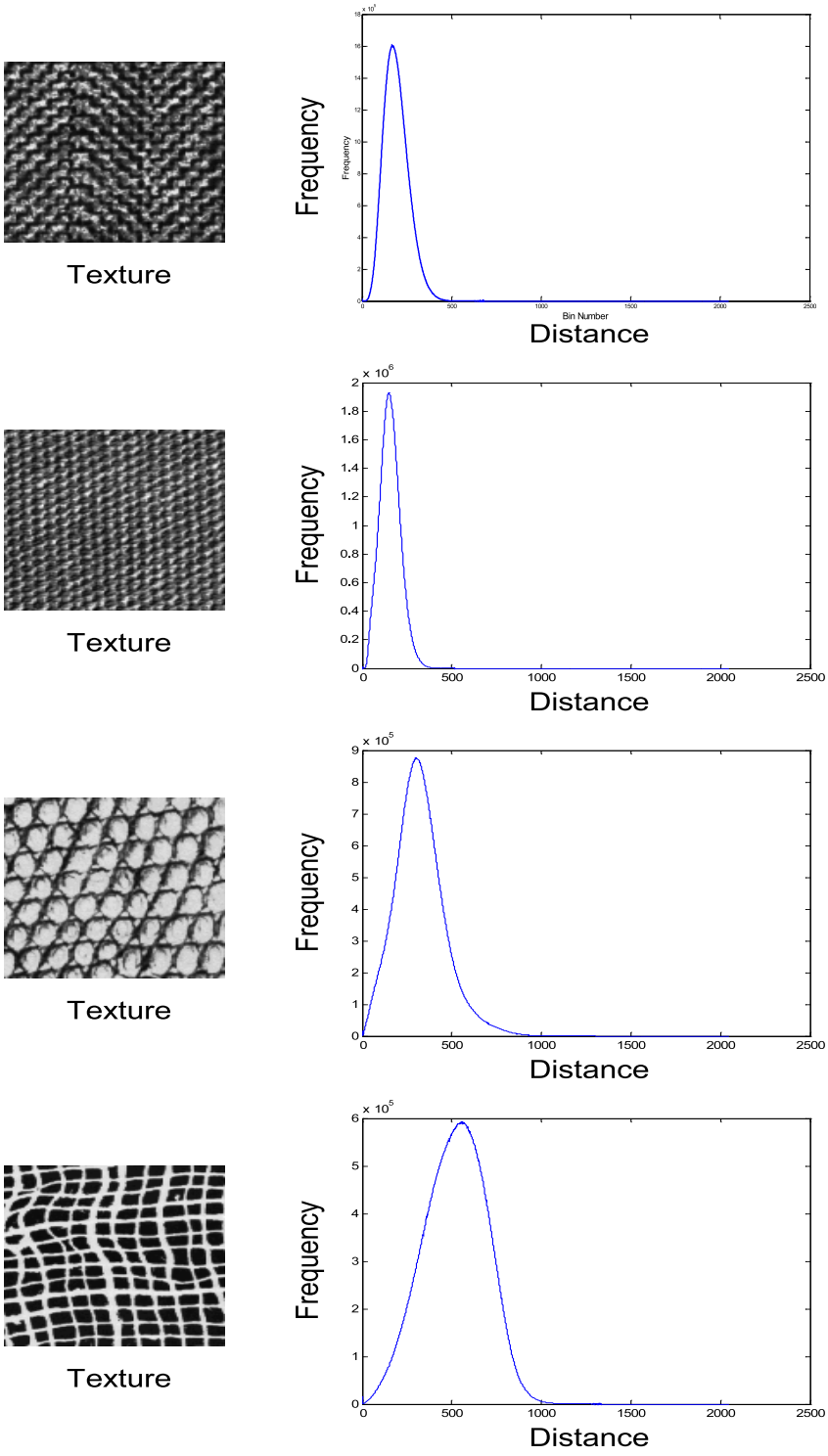


Figure 6.9: *Texture images and the distribution of neighbourhood distances* Υ *that occur within that image. A neighbourhood size of* 9×9 *was used during the neighbourhood comparison process.*

The next issue to how to estimate K. K is related to the upper limit at which distances which belong to that class can achieve. Any distance that exceeds this limit is assumed to be an outlier. Let,

$$K = \gamma\sigma_{T_e}^2 \tag{6.19}$$

Considering expression 6.18, the argument of the distance distribution is given as,

$$-\frac{\gamma^2\sigma_{T_e}^2}{2\sigma_{T_e}^2} \propto -\frac{\gamma^2}{2} \tag{6.20}$$

For a 99% confidence interval $\gamma \approx 2.76^2$. In this implementation γ was set to $\gamma = 2$. This corresponds to approximately a 95% confidence interval.

Combining this outlier estimation within the segmentation model will give the following probability expressions for the example label L_e and the outlier L_o.

$$p(L(\mathbf{x}) = L_e|\mathbf{I}, \mathbf{T_e}, \mathbf{L}) \propto exp\left(-(\frac{D_o}{2\sigma_{T_e}^2} + \alpha U_d(L_e, \mathbf{L}))\right) \tag{6.21}$$

and for the outlier class,

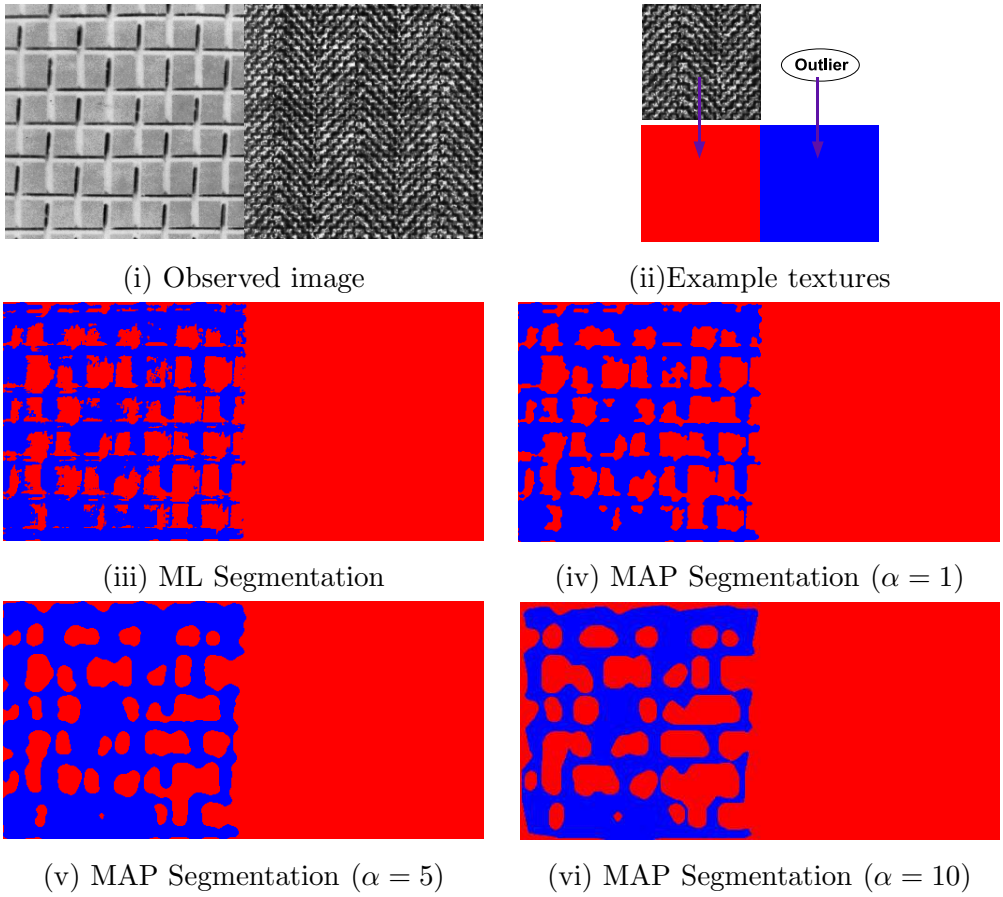
$$p(L(\mathbf{x}) = L_o|\mathbf{I}, \mathbf{T_e}, \mathbf{L}) \propto exp\left(-(\frac{\gamma^2}{2} + \alpha U_d(L_o, \mathbf{L}))\right) \tag{6.22}$$

6.7.1 Outlier Results

Figure 6.10 show the results of segmentation performed to test the outlier class. The observed image (i) is of size 256×512 and the example texture is of size 128×128. A plot of the energy distribution of neighbourhood distances for this texture is shown in Figure 6.9. A neighbourhood size of 9×9 was used in the estimation process and the outlier threshold γ was set to $\gamma = 2$. Pixels labelled blue indicate that they belong to the outlier class. Segmentation (iii) is the ML segmentation and segmentations (iv), (v) and (vi) were obtained using different potts models.

The ML segmentation shows that on the right, the algorithm correctly detected that all of these pixels belong to the example texture. The outlier region on the left contains a mixture of both example and outlier labels. However, it is fair to say that the algorithm does manage to capture a majority of outlier regions and after smoothing segmentation (iv) shows the dominance of the blue regions. Ideally, the entire region should be blue but segmentations (v) and (vi) demonstrate that these are not possible with the potts model or the type of optimisation used. Future work will involve introducing different prior models such as the Chien Model [62] and investigating other optimisation techniques such as graph cut [30].

It should be noted that the irregular behaviour at the external boundary regions is due way in which the segmentation was implemented here. External boundaries are zero padded and so any neighbourhood which overlaps the boundary will be padded with zeros. However, pixels

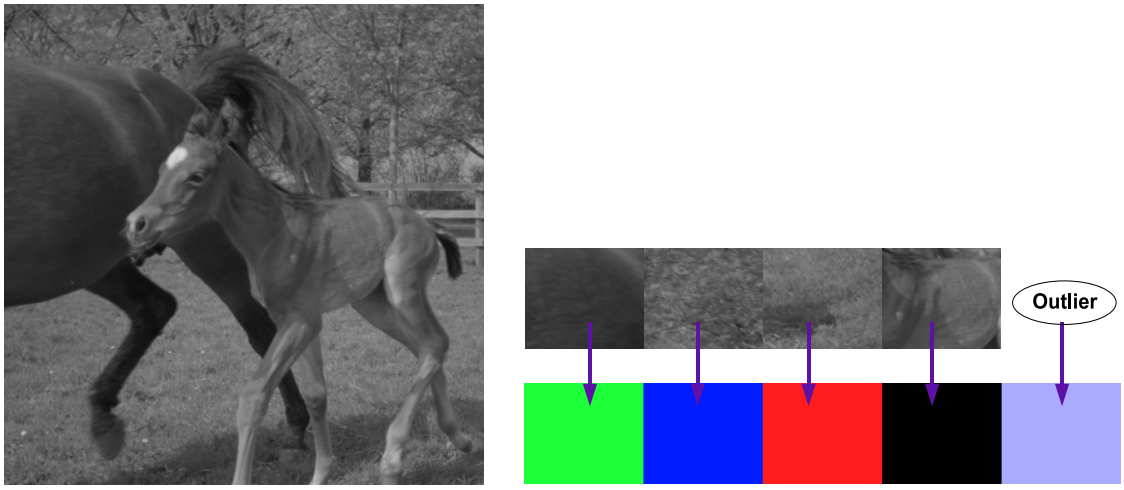


(i) Observed image (ii)Example textures

(iii) ML Segmentation (iv) MAP Segmentation ($\alpha = 1$)

(v) MAP Segmentation ($\alpha = 5$) (vi) MAP Segmentation ($\alpha = 10$)

Figure 6.10: *Segmentation showing the inclusion of the outlier class (blue region). Images (i) and (ii) are the* 256×512 *image to be segmented and the* 128×128 *example training texture. Segmentations (iii) and (iv) are the ML and the MAP segmentations. A neighbourhood width of* $w_1 = 5$ *was used and the value of* γ *used in the outlier detection was set to* $\gamma = 0.2$.



(i) Observed image (ii) Example textures, labels and outlier Label

Figure 6.11: *(i) A frame taken from a sequence which will be segmented using the example based segmentation method. The picture is of size* 576×576 *and the example textures (ii) were taken from a different frame in the sequence. The example textures are of size* 128×128.

which belong to the example texture are labelled the value 0. As a result, at sites close the boundary there is a larger probability of them being labelled zero. Future work will involve correcting this issue and investigating more intuitive ways to handle boundary regions.

Figure 6.13 shows some example segmentations of the observed image in Figure 6.11. The observed image was taken from a sequence of images and example textures given in (ii) were obtained from an earlier frame in the sequence. These example textures contain a sample of the trees, grass, foal and mare textures which are similar to those in the found in the observed image. It should be noted that each of the grass, tree, foal and mare regions in the observed image are characterised by different types of texture. For example the texture associated with the tail of the mare is very different to the texture on the back of the mare. Similarly, the head of the foal is very different from its body. These differences are compounded by lighting changes which take place over the sequence. Because the example textures were taken from an earlier frame in the sequence, the lighting conditions are different. Notwithstanding these differences, a MAP segmentation was performed and the outlier class condition was imposed on the segmentation.

Figure 6.12 shows the distribution of neighbourhood distances which were calculated for each of the sample textures. A neighbourhood width of $w_1 = 9$ was used in the non-parametric modelling process and a fifth order potts model is used in the segmentation. The weighting between the regularisation and likelihood energies was set to $\alpha = 1$.

ML and MAP segmentations of the horse image are shown in Figure 6.13. Textures in the observed image which are similar to the example textures are correctly identified. For example, the back of the foal, the mare and parts of the tree and grass regions. As expected the pixels associated with the tail of the mare have been mis-classified and these have incorrectly labelled

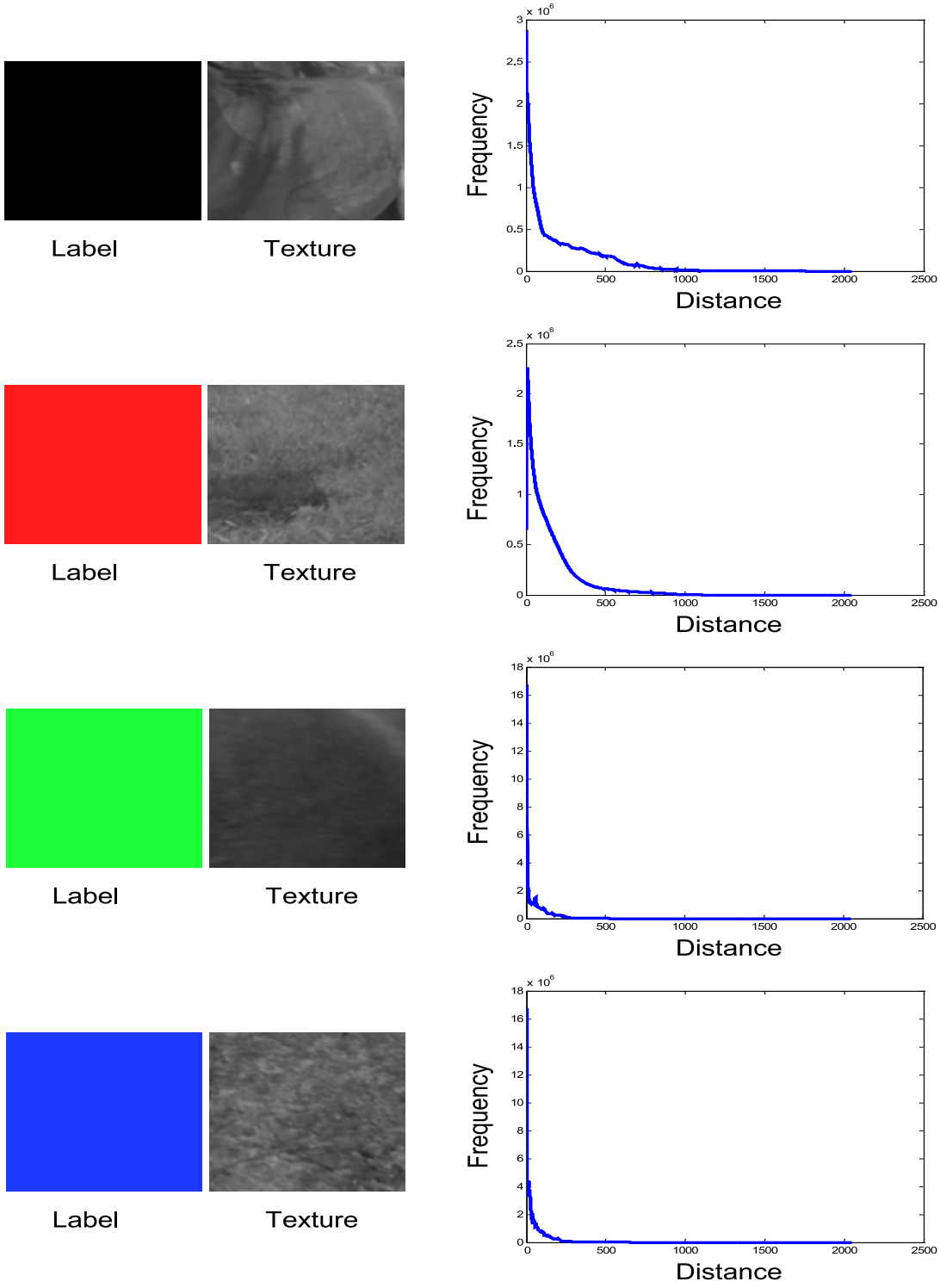
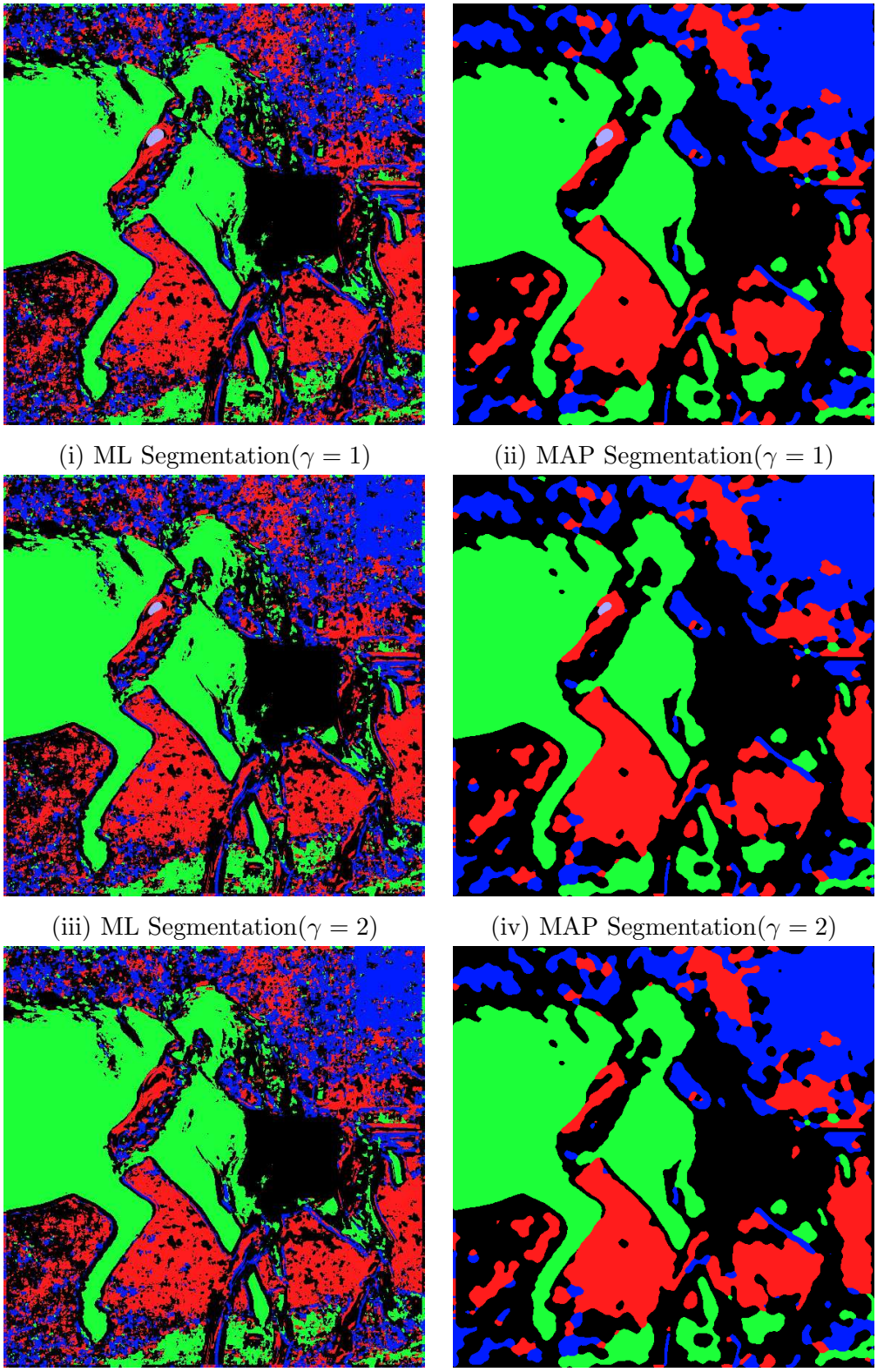


Figure 6.12: *Texture samples and the distribution of neighbourhood distances* Υ *that occur within each texture. A neighbourhood size of* 9×9 *was used during the neighbourhood comparison process.*



(i) ML Segmentation($\gamma = 1$) (ii) MAP Segmentation($\gamma = 1$)

(iii) ML Segmentation($\gamma = 2$) (iv) MAP Segmentation($\gamma = 2$)

(v) ML Segmentation ($\gamma = 3$) (vi) MAP Segmentation ($\gamma = 3$)

Figure 6.13: *Investigating the effects of using different values of γ. ML and MAP segmentation of the picture in Figure 6.11. A neighbourhood width of 9×9 was used in the likelihood estimation and a fifth order potts model was used in the smoothing process. Note the inclusion of the outlier class in the foals head in segmentation obtained using values of $\gamma \leq 2$. The input textures did not contain any texture similar to this and so it was labelled as outlier.*

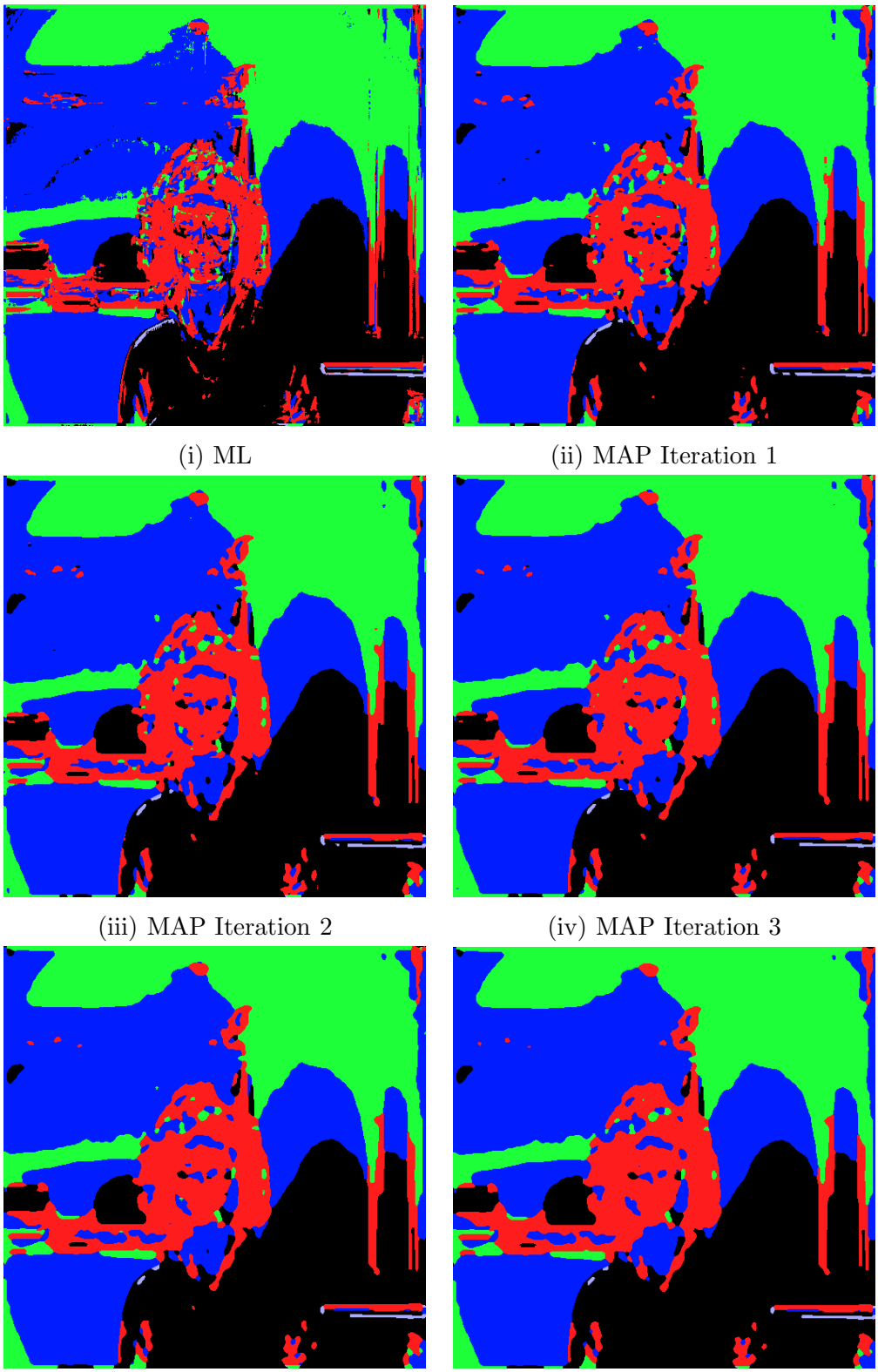
(i) Observed image (ii) Example textures

Figure 6.14: *(i) A frame taken from a sequence which will be segmented using the example based segmentation method. The picture is of size* 512×512 *and the example textures (ii) were taken from a different frame in the sequence. The example textures are of size* 128×128.

as belonging to the foal class. It is interesting to note that the only outlier pixels found were those associated with the head of the foal. This is because the white patch at the top of the head is very different to any of the input texture examples. Different values of γ were tested to illustrate the effect on the overall segmentation. A low value of γ will result in more outliers. This is seen in the head of the foal in segmentations (i) and (ii). A high value of γ results in fewer outliers and this is evident from segmentations (v) and (vi) where no outliers were detected. In this case the value of γ is too high.

Figure 6.15 shows the convergence of the MAP segmentation of the observed image in Figure 6.14. The outlier condition was imposed on the segmentation and as can be seen from the result some outlier pixels were detected. Since two example textures of the back ground region were inputted in the segmentation process, the overall number of outliers detected was relatively small given that both of these textures largely captured most of the back ground region. Note that further post-processing could merge class labels blue and green considering they belong to the background region and class labels black and red considering they belong to the person. The segmentation of the face region is interesting and shows the potential of the example based segmentation for use in face recognition systems. However, it is envisioned that scale will be a problem and further work in this area will investigate dealing with scale and can this method be extended to be approximately scale independent.

Moving on from the single resolution segmentation, the next section will describe the multi-resolution version of the algorithm that has been developed.



(i) ML (ii) MAP Iteration 1

(iii) MAP Iteration 2 (iv) MAP Iteration 3

(v) MAP Iteration 4 (vi) MAP Iteration 5

Figure 6.15: *Segmentations of the observed image shown in Figure 6.14. A value of $\gamma = 1$ was used in the outlier class estimation. Two example textures of the background region were given so by and large the number of outliers detected was small. Segmentations (ii)-(vi) are the MAP estimates obtained at iterations 1-5 and shows the convergence to a minimum energy configuration.*

6.8 DT-CWT Segmentation

Aspects of the wavelet transform were presented in chapter 3 and the advantages using the wavelet transform as a means for texture analysis were discussed in chapter 5 and demonstrated through the texture synthesis results in chapters 2 and 4. The idea behind the multi-resolution segmentation approach developed here is to model the image and example textures in the wavelet domain. Beginning at the coarsest level L and moving toward the highest resolution image space, the MAP segmentation will be estimated at each level by minimising an energy function derived from the single resolution energy model given in (6.12). The segmentation at each level will be initialised by the previous levels MAP estimate.

The good directional properties of the DT-CWT will be exploited by weighting the energies obtained from the sub-band images according to the dominant directional components of the texture in question. This will be achieved through a localised weighting function. This is slightly different to the modelling process used in DT-CWT TexSyn where each sub-band image was given an equal weighting. This equal weighting was adequate for the texture synthesis problem because only a single texture was considered and so it was not necessary to differentiate between different textures. However in the segmentation problem more than one texture is considered and so the modelling process used should extract as much information regarding each texture as possible.

In addition to this improved directional analysis, the energy minimisation process will be conducted over the six directional sub-band images and the intensity low-pass image. This differs again from the DT-CWT TexSyn algorithm, where only the six directional sub-band images were considered.

The DT-CWT Segmentation algorithm begins by taking the L level DT-CWT decomposition of the image to be segmented $\mathbf{I}$ and the example training texture set. Similar to the notation used in chapter 4, let $\mathbf{B^l}$ and $\mathbf{B^l_{e_i}}$ denote the set of sub-band images produced at level l in the DT-CWT decomposition of $\mathbf{I}$ and textures $\mathbf{T_{e_i}}, \mathbf{i = 1}, \cdots, \mathbf{K}$. Both $\mathbf{B^l}$ and $\mathbf{B^l_{e_i}}$ contain the six directionally selective images orientated at $\pm 15^o, \pm 45^o, \pm 75^o$. Each of the images in $\mathbf{B^l_{e_i}}$ are defined on the lattice $\mathbf{X^l_e}$ and can be indexed using the spatial vector $\mathbf{p}$. Similarly, each of the images in $\mathbf{B^l}$ is defined on the lattice $\mathbf{X}$ and can be indexed using the spatial vector $\mathbf{x}$. Let $\mathbf{G}$ and $\mathbf{G_{e_i}}$ denote the low pass image produced at the coarsest level L. Both $\mathbf{G}$ and $\mathbf{G_{e_i}}$ are defined on the lattices $\mathbf{X^{L-1}}$ and $\mathbf{X^{L-1}_e}$.

At the coarsest level of the DT-CWT, class resolution is high and positional resolution is low. Multi-resolution segmentation algorithms typically exploit this by performing segmentation in a coarse-to-fine manner; obtaining an initial segmentation at the coarse level and then refining the estimate with increasing resolution. This will be the approach take here. To extend the energy model given in (6.12) to work within the multi-resolution domain, the neighbourhood searching process which is inherent with the estimation of the data-driven energy will need to be adjusted in order to combine information from the six directional sub-band images $\mathbf{B^l} = \{\mathbf{B^{l,1}}, \cdots, \mathbf{B^{l,6}}\}$

and $\mathbf{B^l} = \{\mathbf{B_{e_i}^{l,1}}, \cdots, \mathbf{B_{e_i}^{l,6}}\}$. At the coarse level, this data driven term will also include information from the low pass images $\mathbf{G}$ and $\mathbf{G_e}$.

6.8.1 Multi-directional Data Driven Energy

Recall the DT-CWT TexSyn neighbourhood searching process discussed in chapter 4. In DT-CWT TexSyn the low pass image is not considered and the six directional sub-band images are considered in parallel. The distance between the neighbourhoods centred at sites $\mathbf{x}$ and $\mathbf{p}$ was given as,

$$D(N(\mathbf{x}), N(\mathbf{p})) = D(N^6(\mathbf{x}), N^6(\mathbf{p})) \tag{6.23}$$

where $N^6(.)$ indicates the 6D neighbourhood composed of individual neighbourhoods from each of the directional images. To further exploit the good directional characteristics of the DT-CWT decomposition, a localised weighting function between sub-band images is used in the DT-CWT Segmentation algorithm. This function applies a weight to the neighbourhood distance obtained from each of the 6 directional sub-band images. A large weighting will signify that that particular directional orientation is dominant in the neighbourhood of the site to be labelled. The distance between the neighbourhoods centred at sites $\mathbf{x}$ and $\mathbf{p}$ is now defined as,

$$D(N(\mathbf{x}), N(\mathbf{p})) = \sum_{k=1}^{6} \omega_k D(N^{(k)}(\mathbf{x}), N^{(k)}(\mathbf{p})) \tag{6.24}$$

where $N^{(k)}(.)$ is the neighbourhood of wavelet coefficients taken from the kth orientation sub-band image and ω_k is the weight assigned to that image. The weighting values are calculated by considering the 6 directional neighbourhoods around the pixel to labelled. The weight for the kth sub-band image is given as,

$$\omega_k = \sum_{\mathbf{x} \in N^k(\mathbf{x})} \frac{B^{l,k}(\mathbf{x})}{B^l(\mathbf{x})} \tag{6.25}$$

These weighting values satisfy the following conditions, $\omega_k < 1$ and $\sum_{k=1}^{6} \omega_k = 1$. For levels $l < L$, the data driven energy term to be minimised over the entire lattice is given as,

$$U_d^l(\mathbf{B^l}, \mathbf{B_e^l}, \mathbf{L}, \mathbf{L_e}) = \sum_{\mathbf{x} \in \mathbf{X^l}} \sum_{i=1}^{K} \min_{\mathbf{p} \in \mathbf{X_e^l}} D(N(\mathbf{x}), N(\mathbf{p})) \tag{6.26}$$

At the coarsest level L, the low-pass image is also included in the energy minimisation estimation and

$$U_d^L(\mathbf{B^L}, \mathbf{B_e^L}, \mathbf{L}, \mathbf{L_e}) = \sum_{\mathbf{x} \in \mathbf{X^L}} \sum_{i=1}^{K} \min_{\mathbf{p} \in \mathbf{X_e^L}} D(N(\mathbf{x}), N(\mathbf{p})) + \varphi D(N(\mathbf{x}/2), N(\mathbf{p}/2)) \tag{6.27}$$

where φ is a weighting over the neighbourhood distances in the low pass images. In both (6.26) and (6.27), $D(N(\mathbf{x}), N(\mathbf{p}))$ is given by (6.24).

Combining the data driven energy term with the Potts regularisation prior results in the following energy minimisation problem,

$$U^L(\mathbf{B^L}, \mathbf{B_e^L}, \mathbf{L}, \mathbf{L_e}) = \sum_{\mathbf{x}\in\mathbf{X^L}} \sum_{i=1}^{K} \min_{\mathbf{p}\in\mathbf{X_e^L}} D(N(\mathbf{x}), N(\mathbf{p})) + \varphi D(N(\mathbf{x}/2), N(\mathbf{p}/2)) + \alpha \sum_{k=1}^{P} \beta_k (1 - \delta(L_e(\mathbf{p}), L(\mathbf{x} + \mathbf{q_k}))) \tag{6.28}$$

at the coarse level and for levels $l < L$,

$$U^l(\mathbf{B^l}, \mathbf{B_e^l}, \mathbf{L}, \mathbf{L_e}) = \sum_{\mathbf{x}\in\mathbf{X^l}} \sum_{i=1}^{K} \min_{\mathbf{p}\in\mathbf{X_e^l}} D(N(\mathbf{x}), N(\mathbf{p})) + \alpha \sum_{k=1}^{P} \beta_k (1 - \delta(L_e(\mathbf{p}), L(\mathbf{x} + \mathbf{q_k}))) \tag{6.29}$$

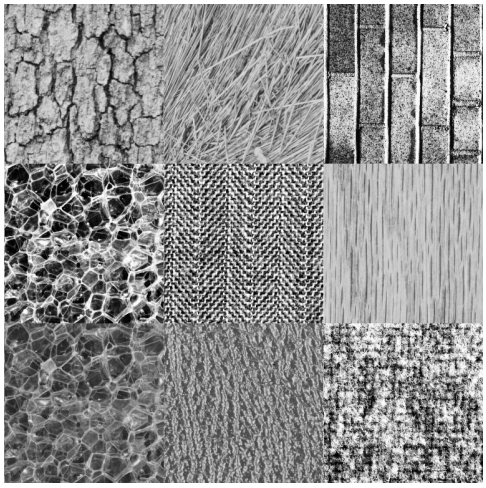
The ICM algorithm is used to minimise U^l and the segmentation at each level l is initialised using the estimate obtained from the previous level $l+1$. The local energy at each site is scanned in a checkerboard manner. MAP estimates of the segmentation are obtained at each level of the DT-CWT and these form an initialisation for a final single resolution segmentation. The initial segmentation at the coarse level was estimated using the data driven term only, i.e. $\alpha = 0$.

6.8.2 DT-CWT Segmentation Results

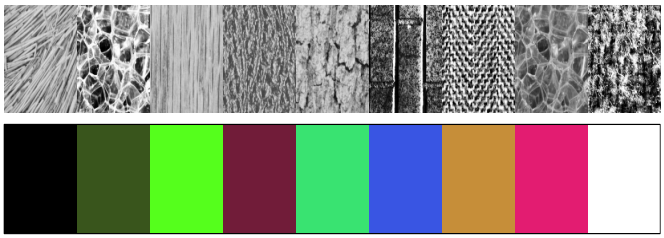
Figure 6.16 shows a segmentation obtained using the DT-CWT Segmentation algorithm. The initial coarse level estimate was obtained at level 1 and the corresponding ML and MAP segmentations were obtained using information from the directional sub-band images and the low-pass intensity image. The MAP segmentation at level 1 was then up sampled and used as an initial estimate for the segmentation at the single resolution. A neighbourhood size of 9×9 was used in the neighbourhood searching process and the weighting between the data driven and regularisation terms was set to $\alpha = 1$.

The observed and example textures are the same as those given in Figure 6.3 where the non-parametric modelling technique was compared to an AR (parametric) modelling technique. As mentioned previously, the differences between the example textures (in particular the blue texture) will make this a difficult problem to solve. In addition, the left centre and bottom textures are very similar and because they are located adjacent to each other, identifying the boundary between these regions is difficult.

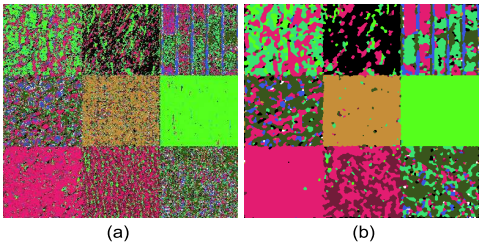
Considering the highest resolution MAP segmentation (iv) in Figure 6.16, the centre, centre right and bottom left are all been labelled correctly. The difficulty in modelling the upper right (blue region) is evident from the segmentation where there are a large number of misclassified



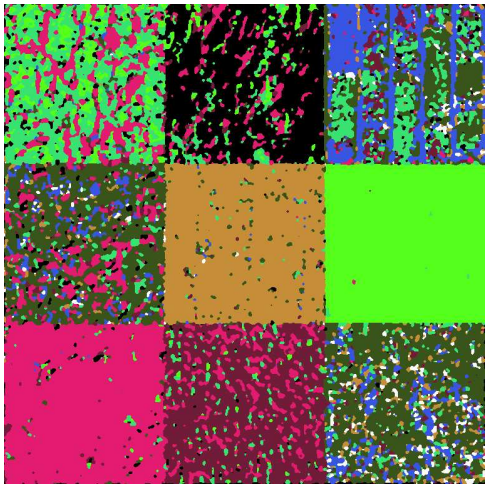
(i) Observed image



(ii) Example textures



(iii)Level 1 Segmentation



(iv) MAP Highest Resolution

Figure 6.16: *Segmentations obtained using the multi-resolution energy model given in (6.29) and (6.28). Images (i) and (ii) are the* 768×768 *image to be segmented and each of the 9* 128×128 *training example images. Segmentations (iii) (a) and (b) are the ML and MAP segmentations obtained at the coarsest level* $L = 1$*. Image (iv) is the MAP segmentation at the highest resolution image space. A neighbourhood width of* $w_1 = 5$ *was used in the segmentation and the potts weighting was set to* $\alpha = 1$*.*

pixels. In the bottom right region some of the sites which have associated with them a dark intensity have been incorrectly labelled as belonging to the blue region. This mis-classification could be reduced by using a larger neighbourhood size. Further work in this area will investigate if this scale dependence can be removed.

Figure 6.17 gives a segmentation of the colour image in (i). Similar to the single resolution segmentation discussed earlier, to include colour information in the segmentation process the image is first transformed from RGB colour space to YUV colour space. Performing a full neighbourhood searching process in each of the YUV channels would increase the computational load of the gray-scale algorithm by a fact of three and be largely inefficient given that much of the high frequency information is contained in the Y channel only. The U and V are relatively flat and based on this observation their directional sub-band images will not be considered in the multi-directional searching process. Only the Y sub-bands will be considered. However, all three Y, U and V channels will searched when considering the low pass image information.

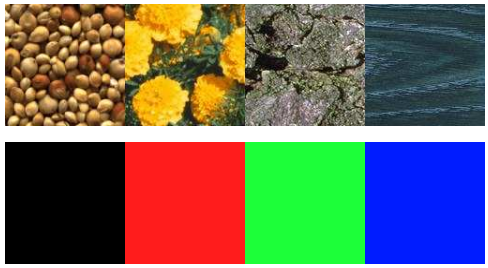
The segmentation shown in Figure 6.17 is good and each of the 4 textures have been correctly labelled. The boundaries surrounding each region are relatively sharp and the number of misclassified pixels within each region is low. As before the smoothness of the result could be improved by varying the regularisation parameters.

6.8.3 Discussion on the DT-CWT Segmentation Algorithm

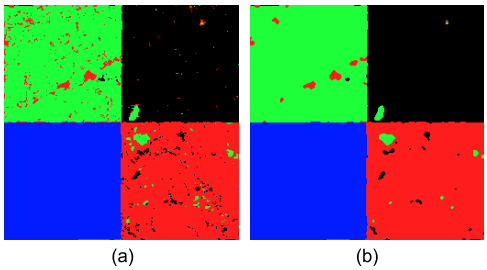
The segmentations given in Figures 6.16 and 6.17 show the potential of the DT-CWT Segmentation algorithm. Beginning at the coarse resolution and moving toward the single resolution, MAP estimates are obtained at each level of the DT-CWT and used as an initialisation for the next higher resolution level. The algorithm then moves into the single resolution level using the DT-CWT as an initialisation.

By initialising the segmentation at the coarsest level, the large regions present in the image are more easily identified since the high frequency information within these regions has been removed. Refining the segmentation at each resolution then re-introduces this fine detail. This coarse to fine estimation process is expensive and considering that the final segmentation is performed at the single resolution it is questionable whether some of the benefits of searching the lower resolutions are lost or not fully exploited. In this case, the multi-resolution analysis allows a good initialisation to be obtained at the single resolution level. This benefit of this good initialisation is evident by comparing the segmentations in Figure 6.18. The observed and example images associated with this segmentation are shown in Figure 6.6. The segmentations are similar and both attempts identify and classify correctly the four textures present in the observed image. The multi-resolution segmentation process performs the initial segmentation estimate at level 1 of the DT-CWT and then uses this estimate as an initialisation for the highest resolution image space segmentation. Comparing both segmentations it can be said that the multi-resolution segmentation is slightly better. Whether this increase in accuracy is worth the

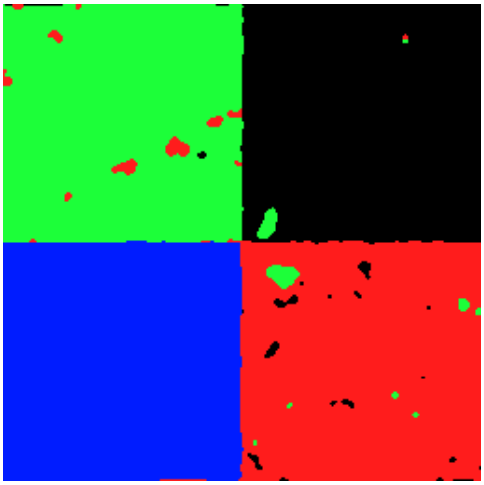
(i) Observed image



(ii) Example textures

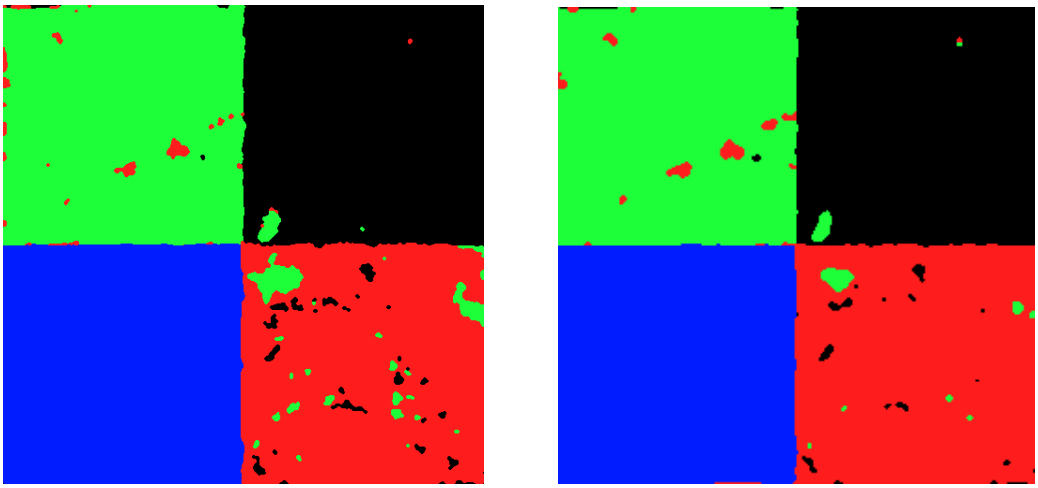


(iii) Level 1 Segmentation



(iv) MAP Single Resolution

Figure 6.17: *Segmentations obtained using the multi-resolution energy model given in (6.29) and (6.28). Images (i) and (ii) are the* 512×512 *image to be segmented and the* 128×512 *training samples used to guide the segmentation process. Images (iii) (a) and (b) are the first and last iterations of the segmentation obtained the coarsest level. Image (iv) is the final segmentation. A neighbourhood width of* $w_1 = 5$ *and the potts weighting was set to* $\alpha = 1$*. Initial segmentation was performed on the first level of the DT-CWT.*



(i) MAP Single Resolution Segmentation (ii) MAP Multi-Resolution Segmentation

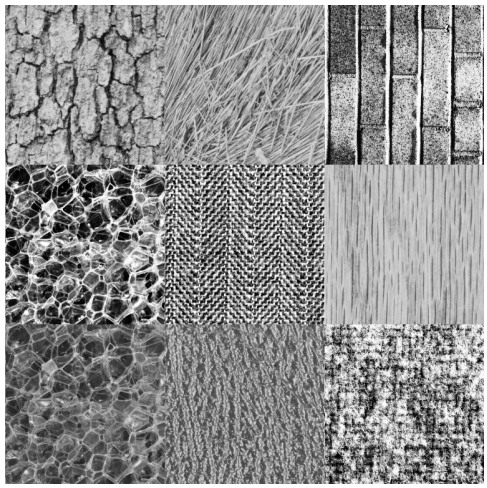
Figure 6.18: *Comparing the segmentations obtained using the single resolution and multi-resolution variant of the example based segmentation algorithm. Algorithm parameters were set to* $w_1 = 5$ *and* $\alpha = 1$ *and the multi-resolution segmentation analysis was performed using* $L = 1$ *levels of the DT-CWT.*

extra computational expense depends on the particular application where the segmentation will be used.

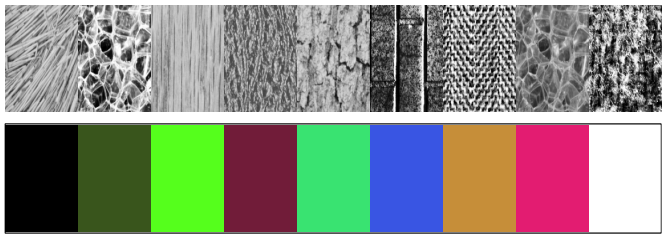
Future work in this area will investigate further ways of exploiting the multi-resolution information provided by the DT-CWT Segmentation. This will be used in the estimate of the outlier class condition and for increasing the accuracy of the segmentation. For example, considering each site at the highest resolution level, attach to each site a feature vector. This feature vector will contain the pixel value, the current label, sub-band wavelet coefficients at each directional orientation and resolution of the L level DT-CWT and the coarse low-pass scaling coefficient. These features vectors would then be considered in a non-parametric modelling process derived from that discussed above. While this approach would be thorough, it is envisaged the computational burden would be very large. A more manageable solution may be to determine different ways of exploiting the sub-band information and comparing sub-band neighbourhoods. The method outlined here used the L_2 distance between neighbourhoods. Other possible suggestions including comparing the energy between neighbourhoods or creating histograms and comparing neighbourhood histograms. The suitability of wavelet "jets" [132, 217] as a means of image representation will also be investigated.

6.8.4 Comparison with Mignotte

Figure 6.19 shows two segmentations of the observed image in (i). Segmentation (iii) is the MAP estimate obtained using the multi-resolution Mignotte algorithm described earlier. Segmentation (iv) is the MAP estimate obtained using the DT-CWT Segmentation algorithm developed as

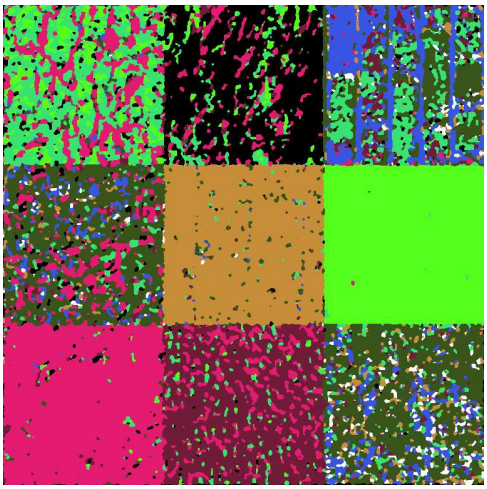


(i) Observed image



(ii) Example textures

(iii) MAP (Non-Parametric Gaussian)



(iv) MAP (DT-CWT Segmentation)

Figure 6.19: *Comparing the segmentations obtained using the multi-resolution Mignotte algorithm and the DT-CWT Segmentation algorithm. Both implementations use a neighbourhood width of* 9×9 *in the modelling process and the weighting between the data-driven and regularisation terms was set to* $\alpha = 1$*. A fifth order Potts model was used in the DT-CWT Segmentation and the number of levels over which the multi-resolution analysis takes place was set to* $L = 1$*. Note the poor segmentation at the boundary regions in segmentation (iii). This is because one of the requirements of the Mignotte algorithm is that the example image and the observed image be of the same configuration and content.*

part of this work. One of the requirements of the Mignotte algorithm is that the example image and observed image are of the same configuration and content.

The regularisation energy term used by Mignotte will tend to bias toward configurations that are found in the example label field and because in this case the textures are arranged differently in the observed image, the boundaries between textures will not be as sharp. This confusion in boundary location is evident in the segmentation in (iii) at the boundaries between the top centre and top right textures and the top left and centre left textures. In contrast, the segmentation obtained using the DT-CWT segmentation method shows the sharp boundaries between each texture.

The Mignotte segmentation is smoother and the centre column textures and the right centre texture were all accurately identified. The segmentation obtained using the DT-CWT Segmentation algorithm is not as smooth which suggests the smoothing using in the DT-CWT Segmentation could be increased. However, the Mignotte method fails to distinguish between the two similar textures in the left centre and left bottom of the observed image. Both approaches fail to correctly segment the top right and bottom right textures. This is because of the differences between the example and observed textures. Overall, the sharper boundaries and ability to separate each texture makes the DT-CWT Segmentation solution a more accurate approximation.

Before concluding this discussion on the DT-CWT Segmentation algorithm and as a means to speed up the segmentation, a refined DT-CWT Segmentation algorithm has also been developed.

6.9 Refined DT-CWT Segmentation

In the DT-CWT Segmentation algorithm presented before, MAP estimation is performed at each level of the DT-CWT and then at the highest resolution image space. While this coarse to fine MAP estimation will minimise the energy over each resolution, the computational expense associated with performing the segmentation at each resolution is large.

The Refined DT-CWT Segmentation algorithm addresses this large computational expense by offering an approximate minimisation of the energy function. The Refined DT-CWT Segmentation process begins by finding the MAP segmentation at the coarsest level of the DT-CWT. This MAP estimation is given in (6.28). The adjacent higher resolution level is then initialised with this MAP estimate in the normal manner. However, rather than recalculating the MAP segmentation at this new level, the new algorithm searches through the segmentation and only estimates the local energy for sites which have a regularisation energy term greater than zero. In homogeneous regions where all pixels have the same label, the regularisation energy will be zero. The Refined DT-CWT Segmentation process assumes that these sites have been correctly labelled and so their energies are not re-calculated. At boundaries there is confusion regarding pixel labels and so the regularisation energies will be greater than zero. The Refined DT-CWT Segmentation will recalculate energies in the same manner as as the DT-CWT Segmentation.

This process is repeated at each level.

6.9.1 Refined DT-CWT Segmentation Results

Figure 6.20 shows a detailed break down of the segmentation obtained at each of the 3 resolutions considered. The observed image and the example textures are of size 512×512 and 128×128 respectively. Beginning with the coarsest level 3 estimate, the segmentation is refined at each level.

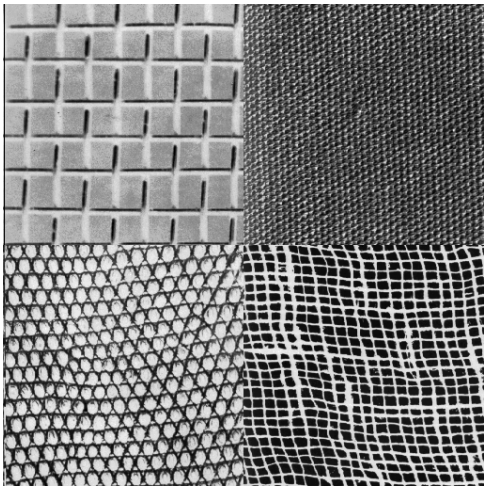
One of the limitations of the refinement process is that the processes assumes that the segmentation obtained at the coarsest level has correctly detected the large homogeneous regions and assigned them the correct label. In the case of bottom right quadrant, pixels assigned the green label are correctly labelled while those assigned the black label are incorrect. Note, the effects of the zero padded boundary issue discussed earlier is evident in this segmentation as black regions tend to grow into the solution. While these black pixels are incorrect, the algorithm will not re-estimate their energy given that there is a large homogeneous group of them and so it is assumed that they are stable and have been correctly assigned.

In Figure 6.21 the segmentations obtained using the refined and the full DT-CWT Segmentation are compared. The refined segmentation still identifies each distinct texture and as expected the result is much smoother. However, the number of mis-classified pixels is greater. Overall, the refined method offers a good approximation of the DT-CWT Segmentation at a much reduced computational cost.

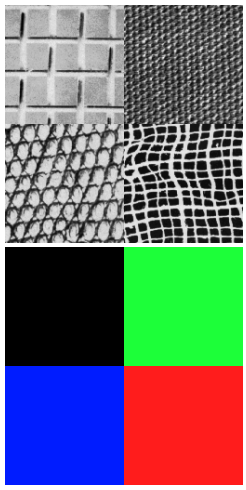
6.10 Concluding Remarks

Before closing this discussion on example based segmentation, the multi-resolution DT-CWT Segmentation algorithm has been applied to the landscape image shown at the beginning of this chapter. The results of this segmentation are shown in Figure 6.22. The example textures were taken from the original image. Gaussian white noise $n \sim \mathcal{N}(0, \sigma^2 = 4)$ was added to each of these example textures so that each of the texture images were not an exact match of those found in the image. The segmentation performs well in capturing the boundary between each of the tree, sky and sea regions. Each of these regions has been correctly classified and overall the segmentation is quite smooth.

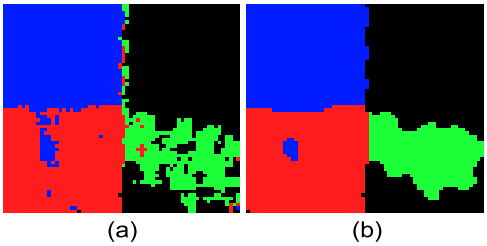
The results presented in this chapter demonstrate the potential of this example based segmentation technique. The next chapter will review some of this work and provide some guidelines for future work in this area.



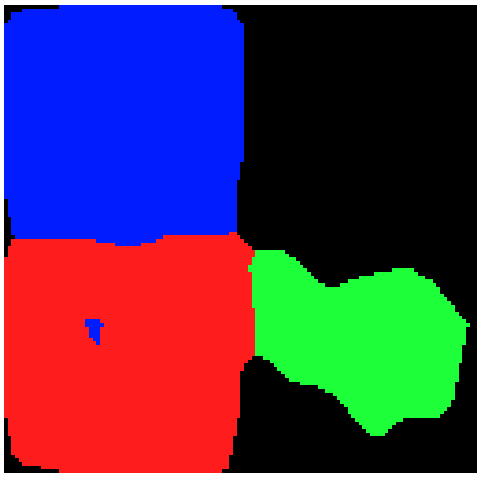
(i) Observed image



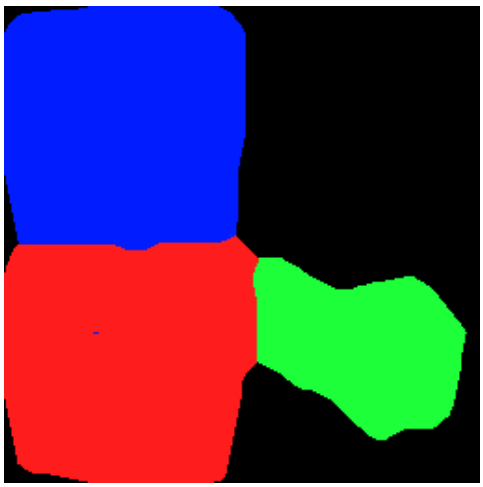
(ii) Example textures



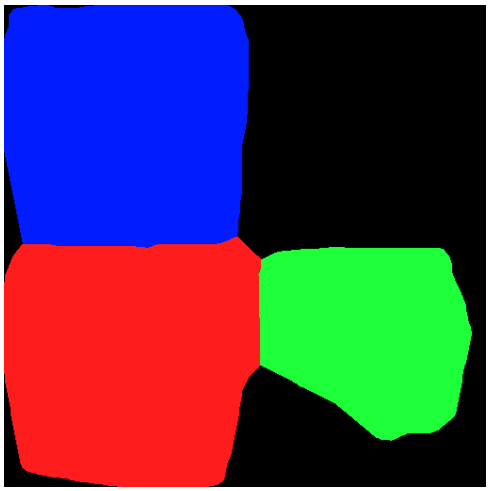
(iii) Level 3 Segmentation



(iv) Level 2 Segmentation

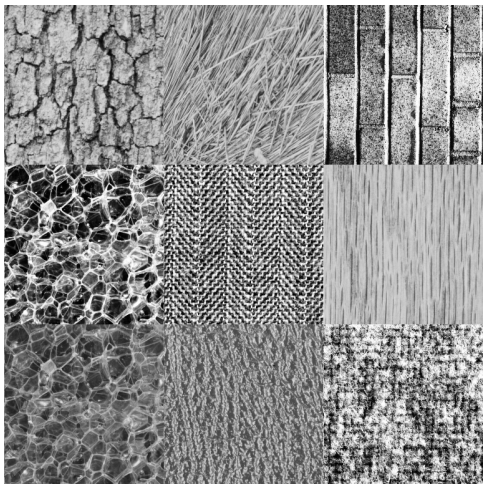


(v) Level 1 Segmentation

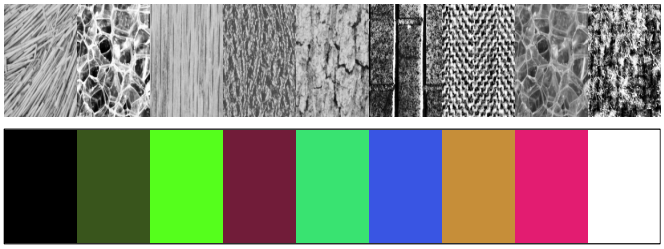


(vi) Single Resolution Segmentation

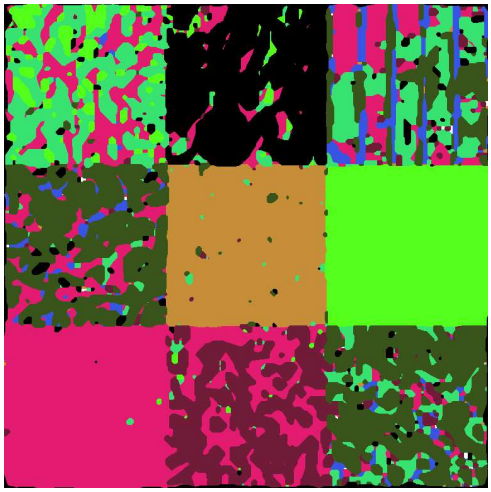
Figure 6.20: *Multi-resolution segmentations obtained using the Refined DT-CWT Segmentation method. Image (iii) is the true MAP segmentation obtained at level 3 of the DT-CWT. At each level this estimate is refined using the regularisation energy as a guide to recalculate posterior energies. A neighbourhood size of* 9×9 *was used and a fifth model with a weighting of* $\alpha = 1$ *was used.*



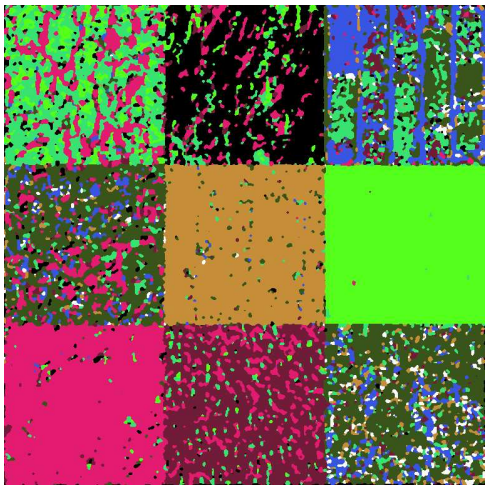
(i) Observed image



(ii) Example textures

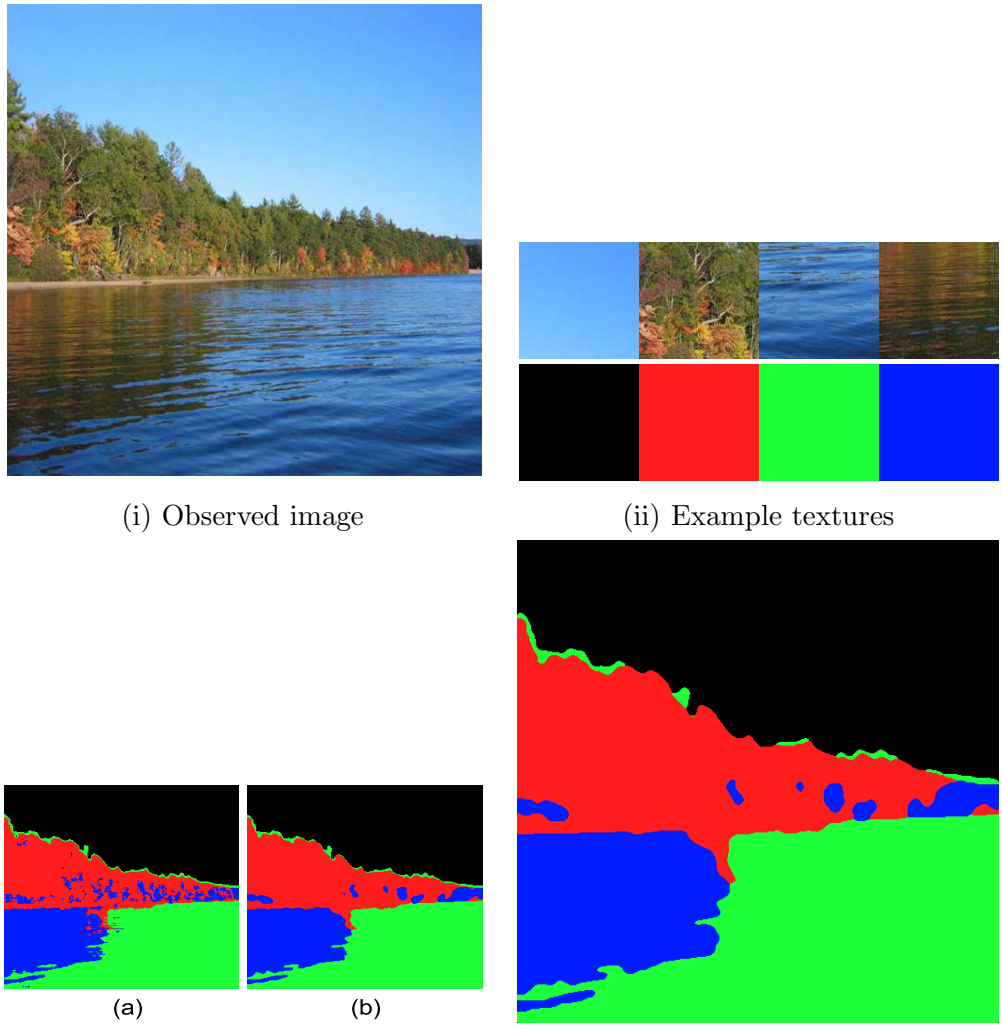


(iii) Refined DT-CWT MAP



(iv) DT-CWT MAP

Figure 6.21: *Comparing Refined DT-CWT Segmentation algorithm to the full DT-CWT Segmentation algorithm. Image (i) is* 768×768 *image to be segmented. For the MAP segmentation shown in (iii), full MAP estimation was performed at level* 2 *(the coarsest level) of the DT-CWT only. This low resolution segmentation estimate was then refined with increasing resolution only when the fifth order potts model detected some ambiguity in pixel labels at any given site. A neighbourhood width of* $w_1 = 9$ *was used and the weighting between the data driven and regularisation terms was set to* $\alpha = 1$*. The segmentation shown in (iv) was obtained using full MAP estimation at each level.*



(i) Observed image (ii) Example textures

(iii) Level 1 Segmentation (iv) Final MAP Segmentation

Figure 6.22: *Segmentation of the landscape image using the DT-CWT Segmentation algorithm. The initial estimate for the segmentation was obtained at* $L = 1$ *of the DT-CWT. A neighbourhood size of* 9×9 *and a fifth order potts model with weight* $\alpha = 1$ *was used in the segmentation process. The original image is of size* 512×512 *and the example textures are each of size* 128×128.

7

Discussion and Future Work

This thesis was concerned with example based image processing and demonstrated the strength of this technique by focusing on two particular applications, texture synthesis and image segmentation. While the problems of texture synthesis and image segmentation are very different, this thesis highlighted the symbiosis that exists between them. This symbiosis exists as a result of the image modelling requirement which is inherent with both synthesis and segmentation processes.

The first stage of this thesis was to develop a modelling process which would be suitable for both texture synthesis and image segmentation. To do this, the problem of texture synthesis was first considered since it provides a good means to illustrate the accuracy of the model.

7.1 Texture Synthesis

To become familiar with the existing types of modelling processes used in texture synthesis, a review of the state of the art in texture synthesis algorithms was given in chapter 2. In this review existing texture synthesis algorithms were categorised as being either patch based, parametric pixel based or non-parametric pixel based. As its name suggests patch based algorithms synthesise entire patches at any one time. These approaches are efficient but often the syntheised result can suffer from some visual inaccuracies at the patch seams.

In parametric pixel based approaches the example texture is modelled explicitly using some definable process. There have been many different types of parametric models developed and some of the more complex models work well in capturing the underlying characteristics of the

example texture. In general, these methods are efficient but because of the wide variability in texture behaviour it is impossible to define a model or set of models which will be suitable for all types of textures.

The third type of modelling process offers no such definable model and rather implicitly models the texture using measurements taken from the example texture itself or some similar example image. These non-parametric techniques are advantageous as they provide a means to capture the varying behaviour of the example texture which otherwise would be difficult to characterise mathematically using a parametric process. Because of the wide variability in texture behaviour, non-parametric methods have generated the most impressive results as demonstrated in chapters 2 and 4.

On the downside, these parametric approaches generally suffer from scale dependence and computational inefficiency. To address these limitations a new algorithm has been developed as part of this work on example based image processing. This new algorithm combines the strength of the non-parametric technique with the advantages of wavelet based analysis. Aspects of the wavelet transform were presented in chapter 3 and the shift invariance and good directional selectivity of the Dual Tree - Complex Wavelet Transform made it the sensible choice for texture analysis.

To decrease the computational expense of the non-parmetric modelling process, much of the heavy computational operations are undertaken at the coarse level of the transform. This coarse level modelling has two benefits. Firstly, the coarse level estimate can be used as approximation for the minimum energy configuration at higher resolution solutions. Secondly, at a coarse level large regions are represented by fewer pixels and so modelling texture at this region introduces a scale independence into the algorithm. Results of this new algorithm are impressive and when compared to some of the popular existing synthesis algorithms, this new algorithm outperformed them in terms of accuracy and computational efficiency. Three variants of the algorithm were presented. These offered a trade off between computational efficiency and high frequency analysis. It was found that the most efficient variant was sufficient for many real world textures.

Building on the strength of this non-parametric modelling process the focus of this thesis then turned to the problem of image segmentation.

7.2 Image Segmentation

By its nature image segmentation is an ill-posed problem and so some *a priori* information regarding the observed image is normally introduced as a means to constrain the solution. In example based image segmentation, this constraining information is given in the form of an example image set. These example images have already been segmented *a priori* and should be of similar content to the image to be segmented.

There have been many different attempts to solve the image segmentation problem and a

taxonomy of some of the existing algorithms was given in chapter 5. In this review it was shown that many of these existing approaches can be unified under a Bayesian framework. The Bayesian framework was presented in chapter 5 and it was shown that the Maximum a Posteriori (MAP) problem can be distilled down to an energy minimisation exercise. These energies relate the observed pixel intensities to measurements of texture as well as label smoothness. To estimate these energies it is necessary to impose a model over the observed image to be segmented and the label field to be estimated.

Similar to the texture synthesis problem, models can either be parametric or non-parametric. In general parametric methods have been the most popular due to their efficiency and good generialisation properties. However, given the success of non-parametric synthesis algorithms, the non-parametric technique has been recently applied to the problem of segmentation. These non-parametric approaches offer certain advantages over parametric approaches and this is illustrated in chapter 6 when a segmentation obtained using non-parametric modelling is compared to that using a parametric approach.

To exploit the strength of this non-parametric modelling technique, a new segmentation algorithm has been developed and this new algorithm is presented in chapter 6. This new segmentation algorithm models the complicated observed image non-parametrically, using measurements taken from the example image. However, since by nature the label field will be composed of smooth regions, it was felt that such smoothness would be best captured by a parametric model. This new algorithm is considered both in the single resolution case and, similar to the texture synthesis algorithm, the advantages of the wavelet expression are exploited by performing the texture analysis at each resolution of the DT-CWT.

To make the algorithm more flexible and suitable for object recognition, an outlier class condition has been included in the estimation process. This outlier class can be assigned to regions which the algorithm deems are not similar enough to any of the inputted example texture images. Results of this new segmentation algorithm demonstrate the potential of this example based segmentation technique.

Finally a refined variant of the multi-resolution algorithm was developed to address its high computational costs. This method performs MAP estimation at the coarsest level only and then refines this estimate with increasing resolution.

7.3 Future Work

The potential range of applications and benefits of a successful automated or semi-automated segmentation process has meant that image segmentation is a vibrant area of academic research. Similarly, the use of texture synthesis processes in the post-production industry has fuelled the interest in this topic. This section outlines some possible extensions to the texture synthesis and image segmentation algorithms presented here.

Example Based Synthesis with Wavelets

The accuracy and efficiency of the texture synthesis algorithm developed as part of this work is very encouraging. The synthesis process was fast and performing the modelling process across different resolutions and orientations introduced a scale independence to the algorithm.

The copy and refined searching variant of the DT-CWT TexSyn algorithm performs well for most natural textures. However, the third variant which was developed to deal with more complicated structured textures failed to accurately model the example texture and often the synthesised result had visual artifacts. This single resolution reduced neighbourhood searching process has the potential to efficiently generate sharp textures but at this stage further investigation into the implementation is needed.

The efficiency of the new texture synthesis algorithm developed here makes it suitable for generating 3D (video) textures. To synthesise video textures, the method described in chapter 4 could be easily extended to include the extra dimension of time. The GPU implementation of the single resolution method provided a good means to improve the speed of the process. As the memory available on GPUs increases, it may be possible to increase the speed of the DT-CWT TexSyn algorithm further by implementing the process on the GPU.

The symbiosis between texture synthesis and segmentation algorithms implies that any improvement in the modelling process will be beneficial to both algorithms. For the segmentation problem, the strength of the non-parametric model process was improved by including low pass and directional sub-band information. A weighting function which favoured dominant directional sub-bands was also developed and incorporated into the estimation. These extensions developed during the work on segmentation could also be applied to the texture synthesis algorithm.

Example Based Segmentation

The results presented in this thesis demonstrate the potential of the example based image segmentation technique. However, this technique is in its infancy and in order to increase the accuracy of the segmentation, the modelling process which is the foundation to its success has to be further investigated and improved. Presently, the multi-resolution analysis is used as an initialisation for the single resolution segmentation. While results shown in this thesis demonstrate that this offers an improvement over a basic single resolution segmentation, it is felt that the potential of the wavelet analysis is not being fully exploited.

Further investigation will be needed into better exploiting the multi-resolution information in the non-parametric modelling process. One possible approach is assign a feature vector to each site at the highest resolution. This feature vector should contain the intensity values, label information, sub-band wavelet coefficients at each directional orientation and resolution of the L level DT-CWT and the coarse low-pass scaling coefficient. These features vectors could then

be considered in a non-parametric modelling process. While this approach would be thorough, it is envisaged the computational burden would be very large.

A more manageable solution of improving the modelling accuracy would be to investigate different ways of exploiting the sub-band information and comparing sub-band neighbourhoods. In this implementation, the L_2 distance metric was used when comparing sub-band neighbourhoods. Other possible metrics include comparing the energy between neighbourhoods or creating and evaluating neighbourhood histograms.

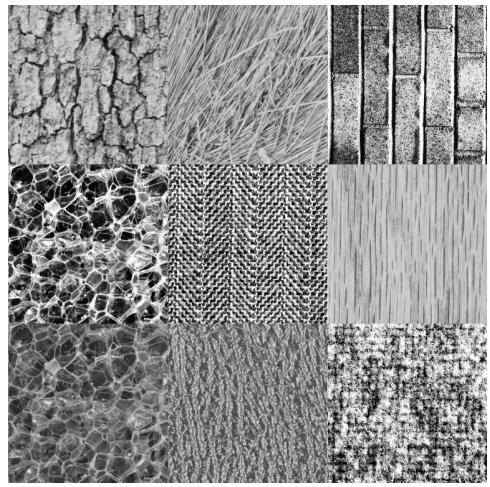
When comparing the Maximum Likelihood (ML) segmentations obtained using non-parameric modelling processes and an example parametric method, the strengths of both types of modelling were demonstrated. In terms of efficiency, the parametric approach was much faster. The speed of the non-parametric approach could be improved by implementing it on the GPU. This GPU implementation would be similar to the GPU TexSyn implementation described in appendix A.

Comparing the accuracy of the segmented result it is clear that the non-parametric model better characterises the texture while the parametric model offers a more generalised description. Evidence of this can be seen in the segmentation shown in Figure 7.1. The centre left and bottom left textures are similar and so the example parametric model cannot distinguish between them. In contrast, the non-parametric approach can differentiate between the two and so the segmentation illustrates the presence of two distinct regions. Note that the strength of the non-parametric model will be largely influenced by the correct interpretation of texture scale. A possible means to reduce this scale dependence is through multi-resolution analysis and future work should investigate this.

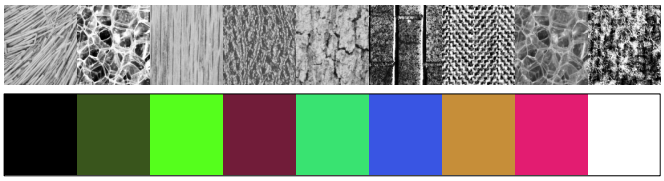
The benefit of good generalisation is illustrated by considering the brick texture located top right in the observed image. The example brick texture given as input has a darker intensity than the observed texture and so as a result the segmentation obtained using the non-parametric model fails to recognise that these textures are the same. The parametric model offers a better texture generalisation and so the brick texture is labelled more correctly in the parametric result.

Given the strength of both of these modelling approaches, the next question is can the information provided by each model be combined? If so, how can the solutions provided by each model be weighted in order to calculate which is the most suited to any given image? One of the strengths of the parametric process is that it can learn from previous estimates and so parameters can be recalculated to improve the model at each pass. Presently, the non-parametric model offers no such learning process and investigation into whether it is possible for non-parametric models to learn from previous estimates is desirable.

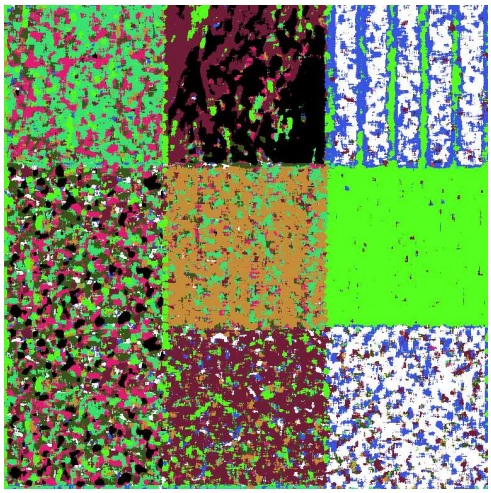
The prior imposed over the label field was based on a parametric model. While this model was simple to implement and works well at imposing a smoothness over the label field, it is far from optimum. As with all parametric modelling approaches, estimating a suitable model order and associated parameter set is an arduous task. For this work, the model order was largely chosen empirically and further investigation is needed into alternative models and parameter estimation for the label field.



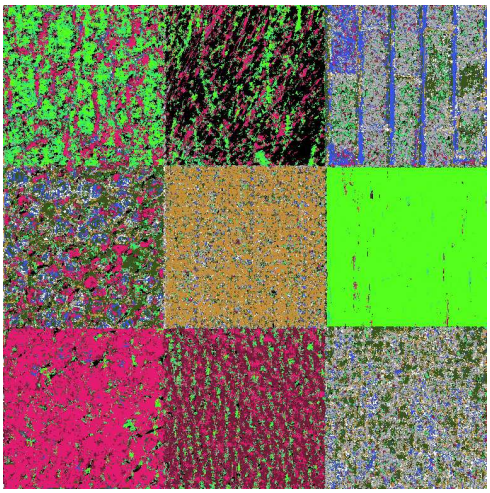
(i) Observed image



(ii) Example textures



(iii) Parametric ML Segmentation



(iv) Non-Parametric ML Segentation

Figure 7.1: *Comparing the Maximum likelihood segmentations of the 2D AR (parametric) and non-parametric energy models.*

The case of the outlier class should be further investigated. If implemented successfully, this outlier condition would allow the segmentation process to be used for object recognition. Determining a suitable value which is optimum for the detection of outliers is fundamental to the accuracy of the outlier classification. When estimating outlier thresholds, a Gaussian behaviour for distance energies between neighbourhoods was assumed over the example images. However, when these distributions were investigated it was found that in many cases they exhibited non-Gaussian behaviour. Further investigation will be needed in order to ascertain whether this Gaussian assumption is valid and if not what are the other methods to estimate the outlier class condition.

Finally, as discussed earlier the refined segmentation method offers good potential and provides a trade off between accuracy and efficiency. Initial results are promising and this approach deserves further consideration.

A

GPU Texture Synthesis

High performance $3D$ graphics systems are becoming increasingly common. Modern computers typically contain two major computational components; a central processing unit (CPU) and a graphics processing unit (GPU). The GPUs in modern graphics hardware are extremely powerful, with performance increasing rapidly year to year. Recently these highly powerful GPUs are becoming much more programmable enabling them to perform user defined graphics computations. This increased flexibility has allowed the GPU to be considered as a useful co-processor to the CPU.

A GPU in its most basic form is a very efficient vector processor. It is optimised for processing the four-component vectors used to represent position and colour information in 3D graphics environments. Since graphics computations are highly parallelisable, GPUs typically contain many pipelines working in parallel. Utilizing the vast processing power of modern graphics hardware for general purpose computing is a fast growing area of research [135, 150, 206] and is often referred to as General Purpose Computations on GPUs (GPGPU) [85]. GPU architecture and the manner in which the GPU has been used for general purpose computing will not be discussed here but a comprehensive review on these subjects can be found in [116, 135].

This appendix will describe how the GPU was used to accelerate the Efros and Leung [68] texture synthesis algorithm described at the beginning of chapter 4. Recall that the idea behind a successful texture synthesis algorithm is to use a small texture example image to create a new (typically larger) texture image by generating *new* texture which will be perceived to be perceptually similar to the original texture example image. Of the previous approaches to the texture synthesis problem, the approach adopted by Efros and Leung has proved to be

the most groundbreaking in terms of visual quality of results. However, associated with the Efros and Leung algorithm are two limitations which have hampered its use. These limitations of scale dependence and high computational expense were discussed in chapter 4. The large computational expense associated with the algorithm has meant that synthesising textures of realistic size is often very slow. To speed up the synthesis process and demonstrate the processing power of the GPU, GPU accelerated texture synthesis has been proposed. This will be described next.

A.1 GPU Accelerated Texture Synthesis[1]

A detailed description of the Efros and Leung texture synthesis algorithm can be found in chapter 4. Recall how, inherent with the Efros and Leung synthesising process, is the need to compare the neighbourhood around the pixel to be synthesised in the output texture against all possible neighbourhoods in the example texture image. These neighbourhoods are defined to be square blocks of size $w \times w$ pixels and centred on the pixel of interest. The similarity between any two blocks is defined using some 'distance' metric. The distance measure used here is the mean absolute error (MAE) between blocks. The block in the example image which has the smallest distance from the block of the pixel to be synthesised in the output image is found. The centre pixel of this block is chosen and placed in the output texture in the location of the pixel to be synthesised. This block searching and matching process is repeated for all unknown pixels in the output texture image. The output texture is initialised by placing a "seed" of the example texture in the centre of the image and pixels are *grown* from this seed using the block matching process described above. Because this block-by-block comparison is a process which is highly parallelisable, it is well suited for implementation on the GPU.

As an initial step and necessary in order to exploit the processing power of the GPU, the block search problem is rearranged. This is a key step and is necessary because the GPU is unable is perform a sliding window operation in order to search all possible blocks in the example texture image. Instead the data rearrangement allows such an operation by presenting data to the GPU which is pre-translated in a sense. In the typical CPU implementation, blocks in the example texture image are compared one at a time to the block of the current pixel to be synthesised in the output texture. This sequential type of operation is inefficient on the highly parallelisable GPU. Therefore to achieve acceleration, the operation is parallelised and the block of the pixel to be synthesised is compared to all possible seed blocks in one pass. This means that if the example texture image is $m \times n$ in size, then the rearranged data must be $(m \times w) \times (n \times w)$ in size.

This parallelisation of the sliding block operation is illustrated in Figure A.1. Two texture images are created; one large modified example image containing all possible blocks in the original example texture image, and one image containing the current block in the output

[1]Results from this section have been published in [76, 116].

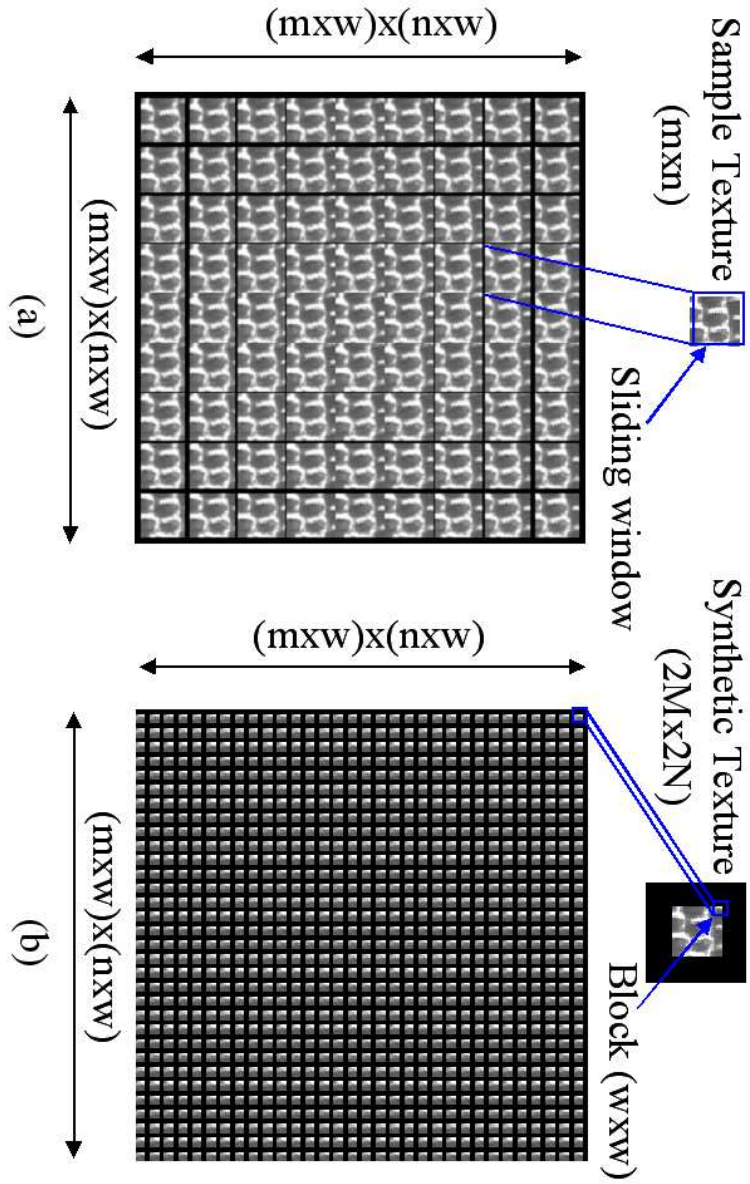


Figure A.1: *Neighbourhood searching on the GPU. (a) Large example image containing all possible neighbourhoods in the example texture. Each cell contains shifted versions of the example image corresponding to each possible position of the example block. The shifts across are $-w : 1 : w$ (Matlab notation), and similarly for each row. The shifts are not easily noticeable due to the scale of this picture. (b) Current pixel to be synthesised neighbourhood block tiled $m \times n$ times. The image in (b) is subtracted from (a) yielding in one go the difference for all possible blocks. This is then summed on a block basis to give the distance measure for each pixel.*

texture image. By tiling the current block texture $m \times n$ times and subtracting it from the large modified example image, the difference image between the current block and all possible example blocks is generated. This different image is outputted to an off-screen rendering buffer called the pixel buffer or *Pbuffer*. Pbuffers reside in GPU memory and are generated and controlled by the particular application that calls them. A second pass of the algorithm then calculates the distance measure $D(\cdot, \cdot)$ for each block based on the texture in the Pbuffer. This type of data re-arrangement is key to getting optimum performance from the GPU for general purpose computing.

Synthesising texture using the Efros method with block searching on the GPU proceeds as follows

1. A large modified example texture image is created. This image contains all possible blocks in the original example texture, laid out in a grid-like fashion. This is stored as a texture image on the GPU during initialisation and is of size $(m \times w) \times (n \times w)$, Figure A.1(a).

2. A $w \times w$ texture containing the block for the neighbourhood of the pixel to be synthesised is uploaded to the GPU. This block texture is tiled $m \times n$ times. This is shown in Figure A.1(b).

3. Using pixel *shaders* the current block texture is compared to all possible blocks in the large example texture. A *shader* is a term used to describe a user defined program that executes on the GPU. As its name suggests pixel shaders operate on pixels. The result of this operation is then rendered to a Pbuffer.

4. A distance measure for the current pixel is generated by summing up blocks of size $w \times w$ in the resulting difference image.

5. An image containing a distance measure for each possible pixel in the seed texture is returned to the CPU which chooses the output pixel.

6. Return to step 2 and repeat for all pixels to be synthesised.

The output from step 3 above is a large difference image containing the difference between the block around the pixel to be synthesised and all possible texture blocks in the example texture image. To generate a distance measure for each block a sum over each $w \times w$ block has to be performed. Two possible methods for doing this are: *automatic mipmapping* and *sum-reduce*.

In the automatic mipmapping method the image is first copied into a texture which has *automatic mipmapping* enabled. Mipmapping is a technique used in graphics programming when sampling from full-resolution textures may not be necessary, e.g. mapping the texture to an object whose size is smaller than the texture resolution. Instead it may be sufficient to sample from a lower resolution texture. Mipmaps are a multi-resolution representation of a

texture image. Each level is sub-sampled by a factor of two from the previous level. GPUs which support automatic mipmapping can generate all the mipmap levels for the texture image in hardware very efficiently. Because the mipmapping filter used is a simple 2×2 box filter, every pixel in a given mipmap level is effectively an average of 2×2 pixels in the next highest resolution mipmap. Thus each pixel at say mipmap level 3 contains an average for blocks of size 8×8 pixels at level 0, the original image resolution.

As with mipmapping, for the sum-reduce method the image is first copied into a texture. A multi-pass algorithm is then used for summing a block of pixels. At each pass a 2×2 block of pixels is summed using a pixel shader and the output image sub-sampled by a factor of 2. This sub-sampled image is then passed back through the graphics pipeline as a Pbuffer texture. This process is repeated until a block of size $w \times w$ pixels has been summed. A block size of $w \times w$ requires $\log_2 w$ passes to generate the necessary summation. The resulting image is then returned to the CPU.

While the mipmapping method is the faster of the two, it is limited to 8-bit accuracy. This is because on current GPUs automatic mipmap generation is only supported for textures with 8 bits per channel. This can lead to overflow problems while doing the summation, and only gives an approximation to the mean value for a block of pixels. The sum-reduce method can take advantage of the floating point support in Pbuffers and hence give much more accurate results. However the multi-pass approach needed means this method is slower than the mipmap method.

The limited 8-bit accuracy of the automatic mipmapping method can lead to an incorrect distance measure for a pixel. As can be seen in Figure A.2 (i), this can cause the wrong example pixel to be chosen. In contrast, the sum-reduce method can take advantage of the full 32-bit floating point capabilities of the GPU and gives results very similar to those obtained from a C++ implementation running on the CPU, Figure A.2 (ii), (iii).

A.2 GPU Texture Synthesis Performance

The performance of the GPU texture synthesis algorithm is evaluated on the NVIDIA GeForce 6800 graphics card [2]. The GeForce 6800 has a peak memory bandwidth of 35.2 GB/sec and can achieve 53 GFLOPS (giga floating point operations per second) [97]. In this section, the AGP and PCIe suffixes which will be placed at the end of the name of the graphcis card will refer to the bus over which the CPU and the graphics card will communicate. The Accelerated Graphics Port (AGP) [5] is a bus which is completely dedicated to graphics cards; the bandwidth across the AGP bus is not shared with any other components. It offers high bandwidth when transferring data *to* the graphics card but this high bandwidth is only available in one direction only. Data travelling *from* the graphics card to the host system does so at much slower rates. Alternatively, in the Peripheral Component Interconnector Express (PCIe) [3,4] bus high bandwidth is available in both directions, both *to* and *from* the device. This feature is very important for GPGPU

m × n	w × w	CPU	8-bit GPU	Speedup	32-bit GPU	Speedup
64x64	8x8	64s	18s	3.5	21s	3.0
64x64	16x16	250s	32s	7.8	50s	5
128x128	8x8	1140s	240s	4.8	287s	4.0
128x128	16x16	4442s	455s	9.8	N/As	N/A

Table A.1: *Comparing speed of CPU and GPU Efros texture synthesis with Intel P4 1.6GHz NVIDIA GeForce 6800 AGP. 8-bit GPU and 32-bit GPU refer to the automatic mipmapping and sum-reduction methods for block summation respectively. N/A = not enough memory for difference texture. The output textures were each twice the size of the input textures.*

where it is necessary to get data back from the GPU to system memory as fast as possible.

Because there is a limited amount of memory available on the GPU, typically 128 or 256MB at the time of testing, there are hardware limitations on the maximum size of textures and Pbuffers that can be created. In this case the amount of memory needed depends on the seed texture size, $m \times n$, and the block size, $w \times w$. The large seed image size is $(m \times w) \times (n \times w)$ and hence so is the difference image generated in step 3 above. These must be stored as two textures on the GPU. Due to the texture size limitations on the test graphics card the maximum example texture and block sizes for mipmapping method were 128×128 pixels and $w = 16$ respectively. Since the sum-reduce method uses floating point for each channel in the difference image, and hence needs more memory, the maximum example image and block sizes are 128×128 pels and $w = 8$ respectively. To over come these hardware limitations, the example texture could be divided into sub-images and synthesising performed on each individual sub-image. Also, the memory available on GPUs is growing all the time with 512MB not unusual today. This will be investigated as part of the future work on GPU texture synthesis.

GeForce 6800 AGP: Table A.1 compares the speed of Efros and Leung texture synthesis on both the GPU and CPU. Given a window size of $w = 8$ the GPU is on average 3.0 to 3.5 times faster than a similar CPU implementation. For $w = 16$ the GPU is on average 5.0 to 8.0 times faster. Because of the implementation used here for the distance measure evaluation, the window size in both methods is limited to $w \times w$ pixels where $w = 2^n, n > 0$. The results presented here were generated on a 1.6GHz Intel Pentium 4 with a NVIDIA GeForce 6800 AGP graphics card with 128MB on-board memory.

GeForce 6800 PCIe: The results for a machine with Intel Dual-Xeon 2.8 GHz processors and a NVIDIA GeForce 6800 PCIe GPU with 256MB on-board memory are given in Table A.2. The increased memory on this graphics card enabled the sum-reduce method to be used for the 128×128 example texture with $w = 16$. In this case the GPU implementation is 5.0 to 7.0 times faster than the CPU only implementation for floating point summation. While using the lower precision mipmapping method, the GPU is up to 17.0 times faster than the CPU. While the lower precision does mean that results are not identical to the CPU, the synthesised textures

m × n	w × w	CPU	8-bit GPU	Speedup	32-bit GPU	Speedup
64x64	8x8	42s	6s	7	8s	5.25
64x64	16x16	142s	10s	14.2	20s	7.1
128x128	8x8	663s	73s	9.1	101s	6.6
128x128	16x16	2261s	130s	17.4	308s	7.3

Table A.2: *Comparing speed of CPU and GPU Efros and Leung texture synthesis with Intel Dual-Xeon 2.8 GHz NVIDIA GeForce 6800 GT PCIe. 8-bit GPU and 32-bit GPU refer to the automatic mipmapping and sum-reduction methods for block summation respectively. The output textures were each twice the size of the input textures.*

are still quite acceptable, see Figure A.2.

A.3 Final Comments

The GPU implementation of the Efros and Leung texture synthesis algorithm has confirmed the potential of the GPU as a co-processor to the CPU in selected image processing tasks. Overall, the GPU implementation offers an ideal way to achieve faster implementations by exploiting the hardware resources available on most modern computers. It should be noted that only the Efros and Leung algorithm has been implemented on the GPU. The DT-CWT TexSyn algorithm developed as part of this work has not yet been implemented on the GPU due to the limited memory resources available on the GPU. These memory resources already placed limits in the single resolution synthesis process and with high redundancy of the DT-CWT (4 : 1 for a 2D signal), there are serious limitations on the maximum size of the texture images that can be stored on the GPU. At the time of testing the memory available on a typical off the shelf GPU was 128 to 256MB. This memory is increasing all the time and at the time of writing GPUs with a memory capacity of 512MB were not uncommon. In addition to this hardware limitation, the motivation of creating a GPU implementation was not as strong as that with the single resolution algorithm, given that the speed with which the DT-CWT TexSyn algorithm generates images is faster than the previous texture synthesis algorithms discussed in chapter 2. However, it should be noted that as part of the future work of this project it is planned to exploit the increased memory available on the today's GPU by implementing the DT-CWT TexSyn algorithm on the GPU.

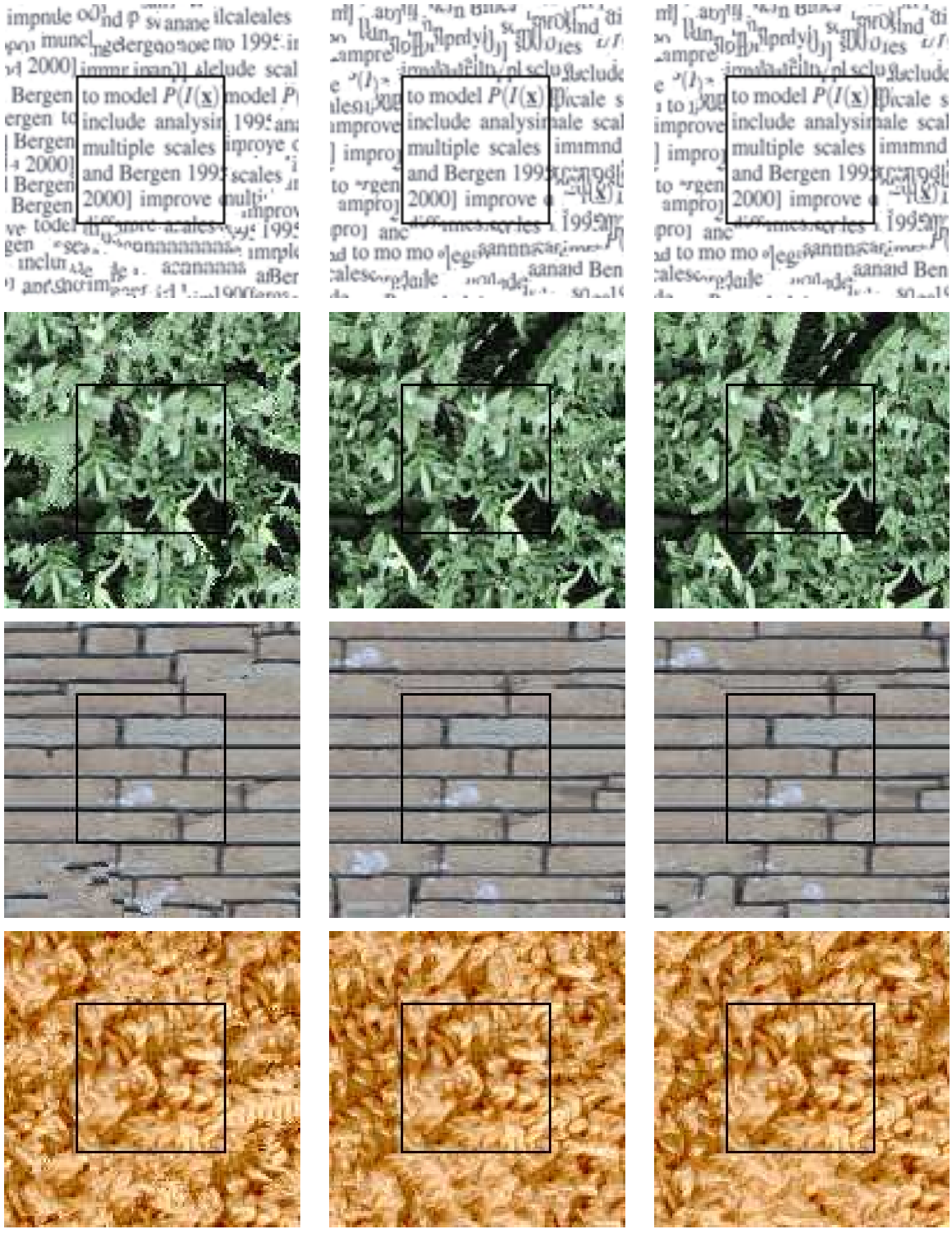


Figure A.2: *Results from the Efros and Leung texture synthesis. (i) Using 8-bit textures for distance measure. (ii) Using 32-bit floating-point textures for distance measure. (iii) CPU result. The original example texture is shown inside the black square.*

B

DT-CWT TexSyn Results

Some sample synthesised textures obtained using the copy variant of the DT-CWT TexSyn algorithm are presented in this appendix. The entire example texture was used as a seed to kick start the synthesis process and this seed is shown inside the square box in each syntheissed image. The gray-scale example textures all come from the Brodatz collection [34] and are of size 256×256. The resulting gray-scale synthesised images are of size 512×512, and were generated using the algorithm parameters of $[L = 3, w_1 = 5, \epsilon = 0.1]$ where L is the highest level of the DT-CWT considered in the analysis process, w_1 is the neighbourhood used in the searching process and ϵ is the parameter which controls the randomness of the sampling process.

For the synthesised colour images, the synthesised images are of size 1024×1024 and the original example texture used as a seed was of size 256×256. The algorithm parameters were as before for the gray-scale values.

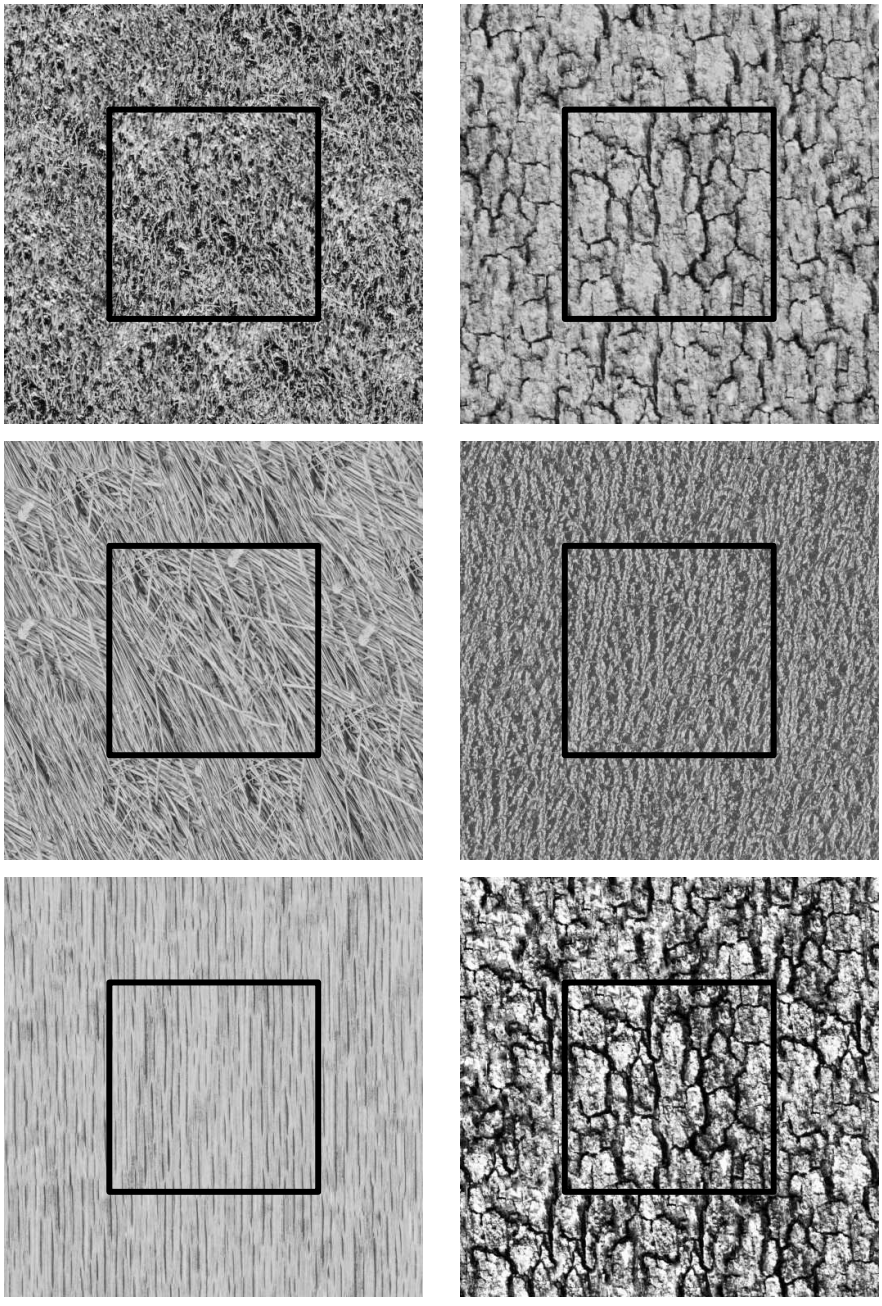


Figure B.1: *DT-CWT TexSyn Results*

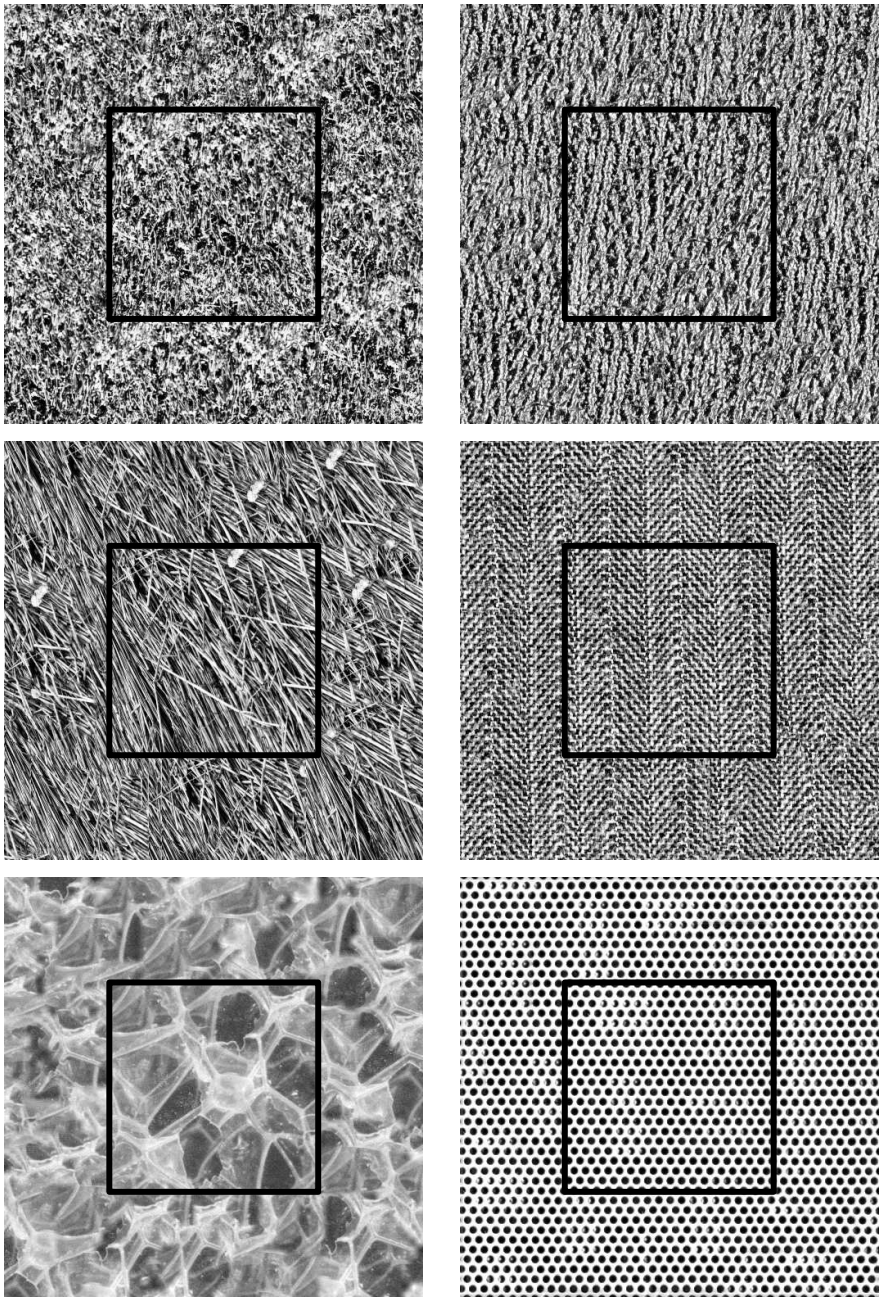


Figure B.2: *DT-CWT TexSyn Results*

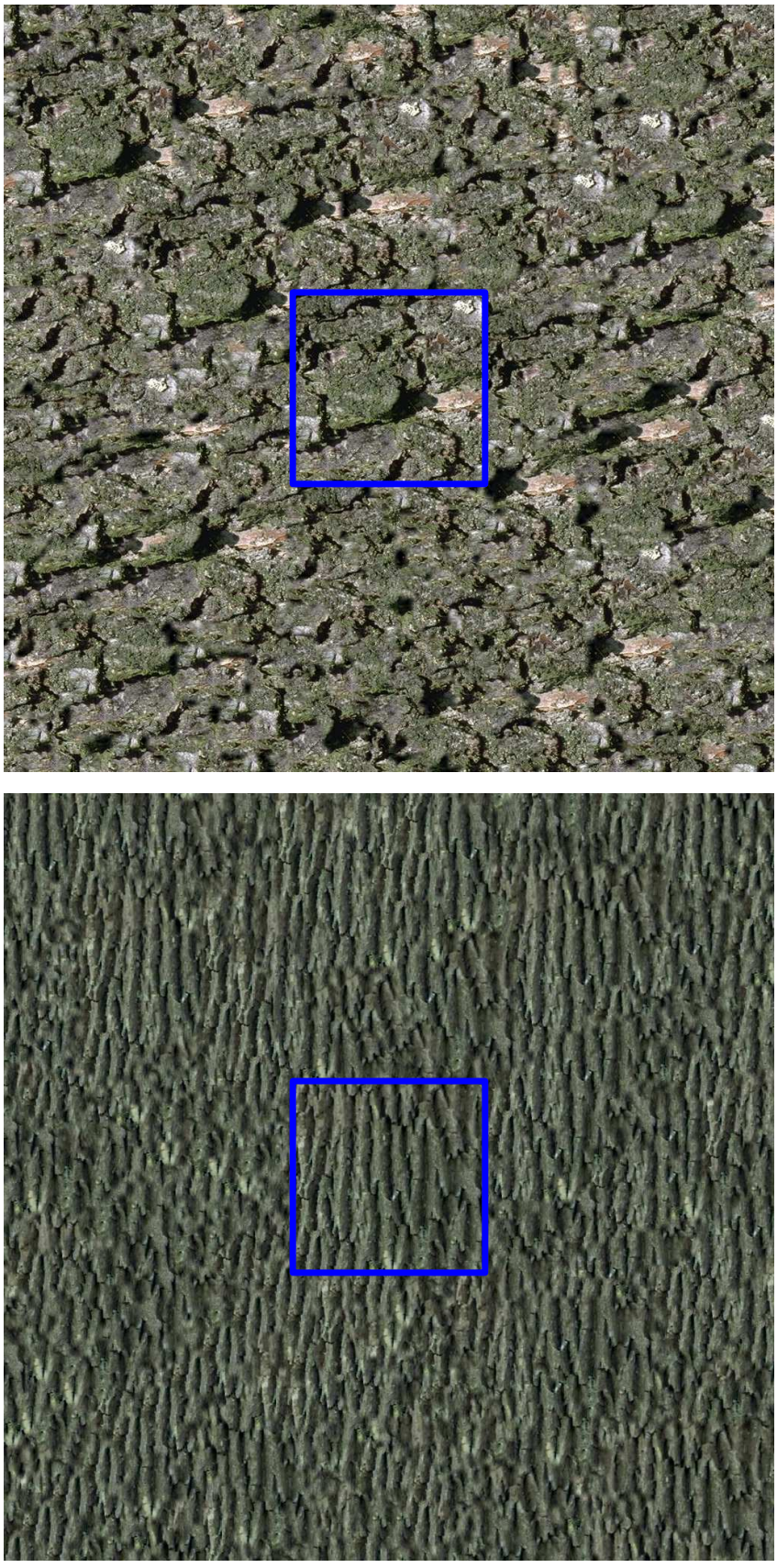


Figure B.3: *DT-CWT TexSyn Results*

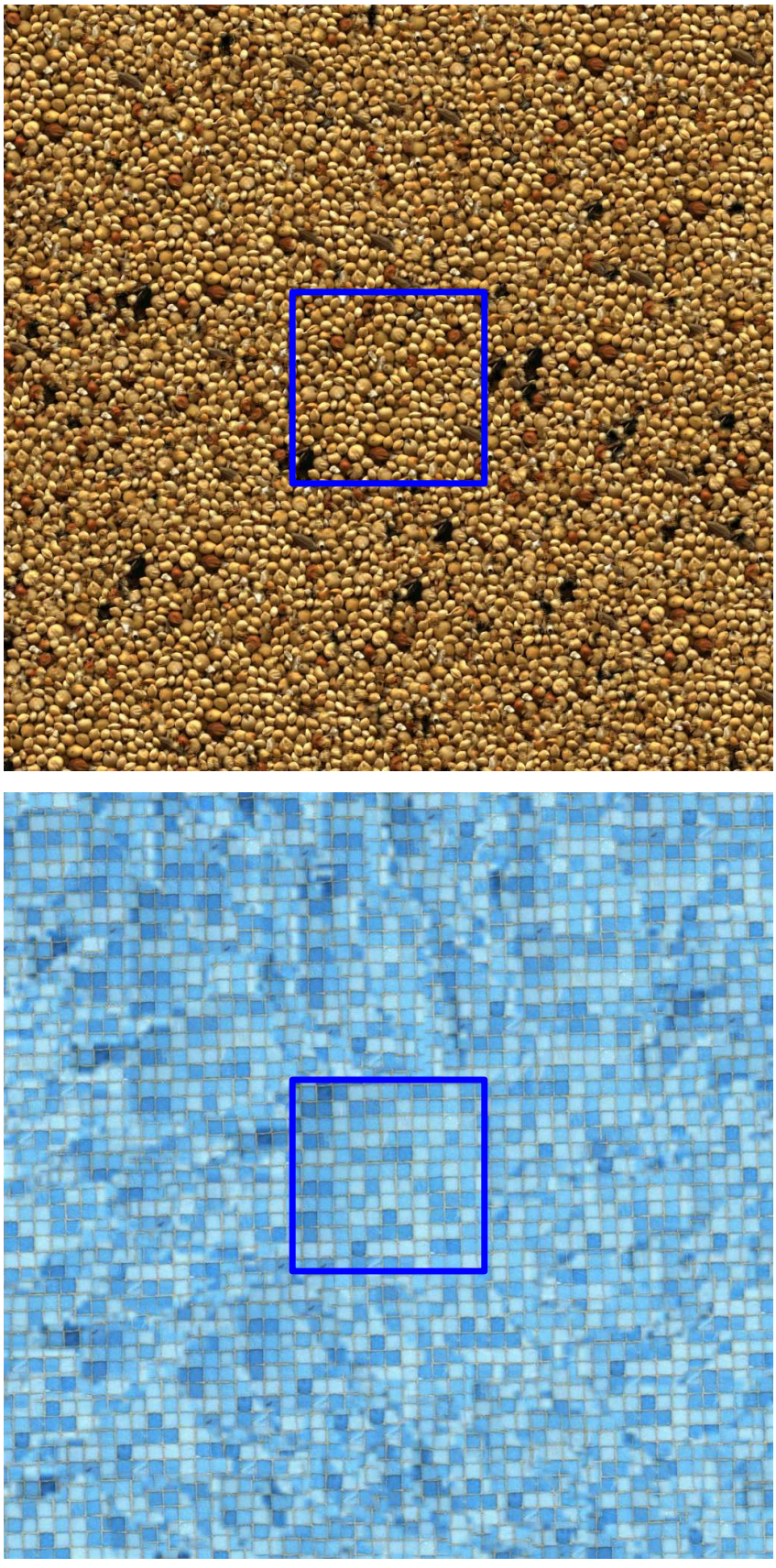


Figure B.4: *DT-CWT TexSyn Results*

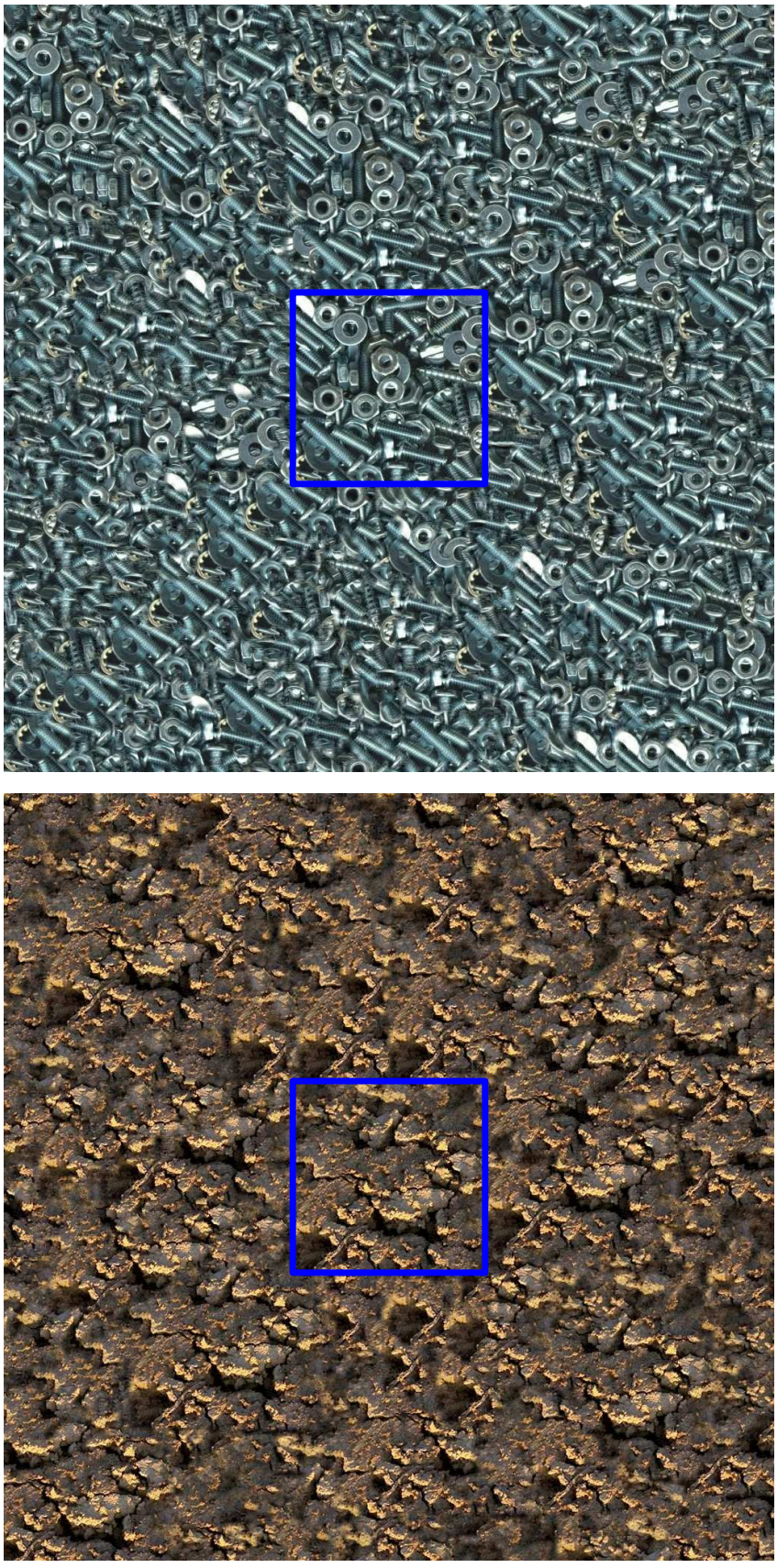


Figure B.5: *DT-CWT TexSyn Results*

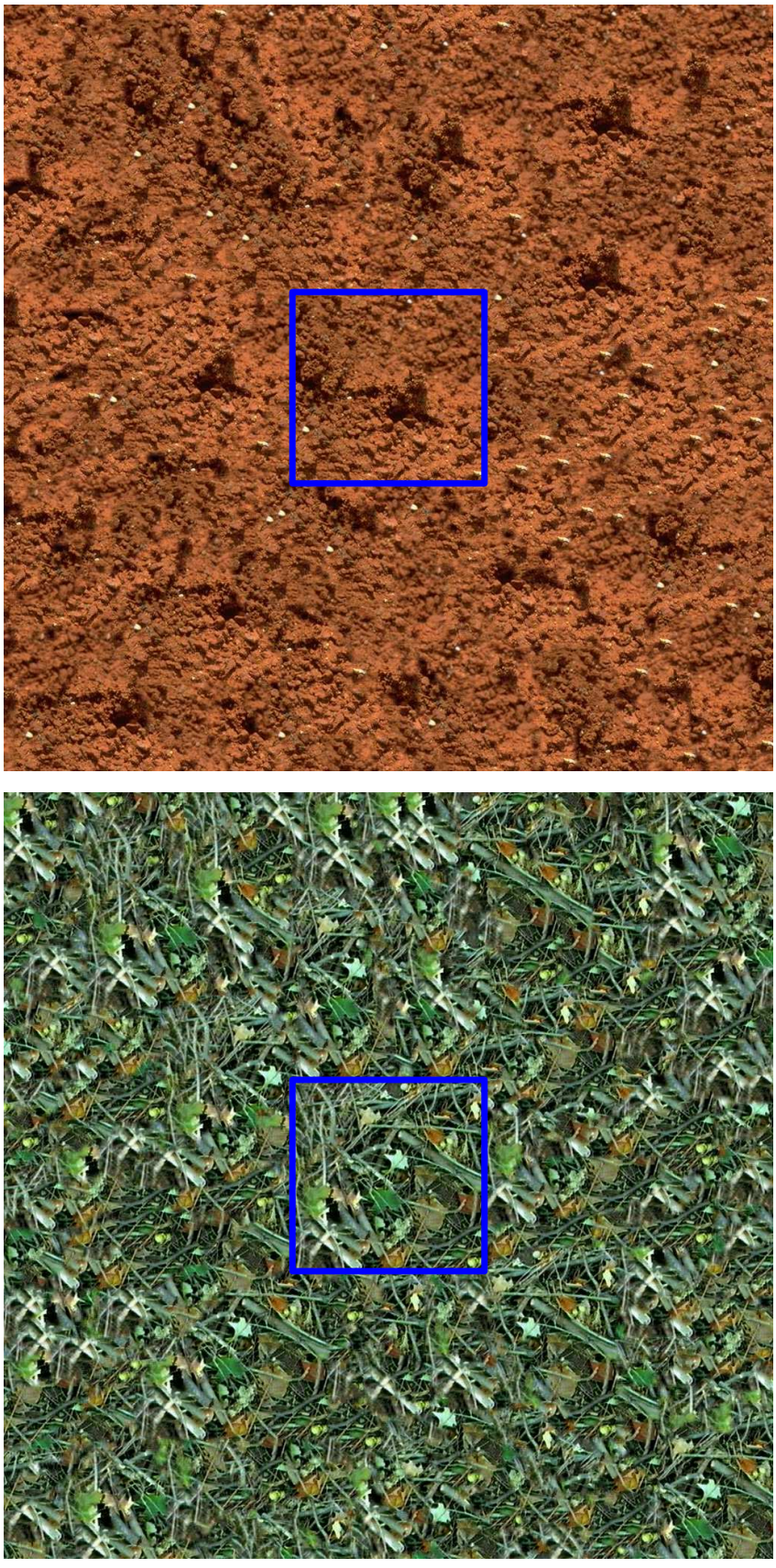


Figure B.6: *DT-CWT TexSyn Results*

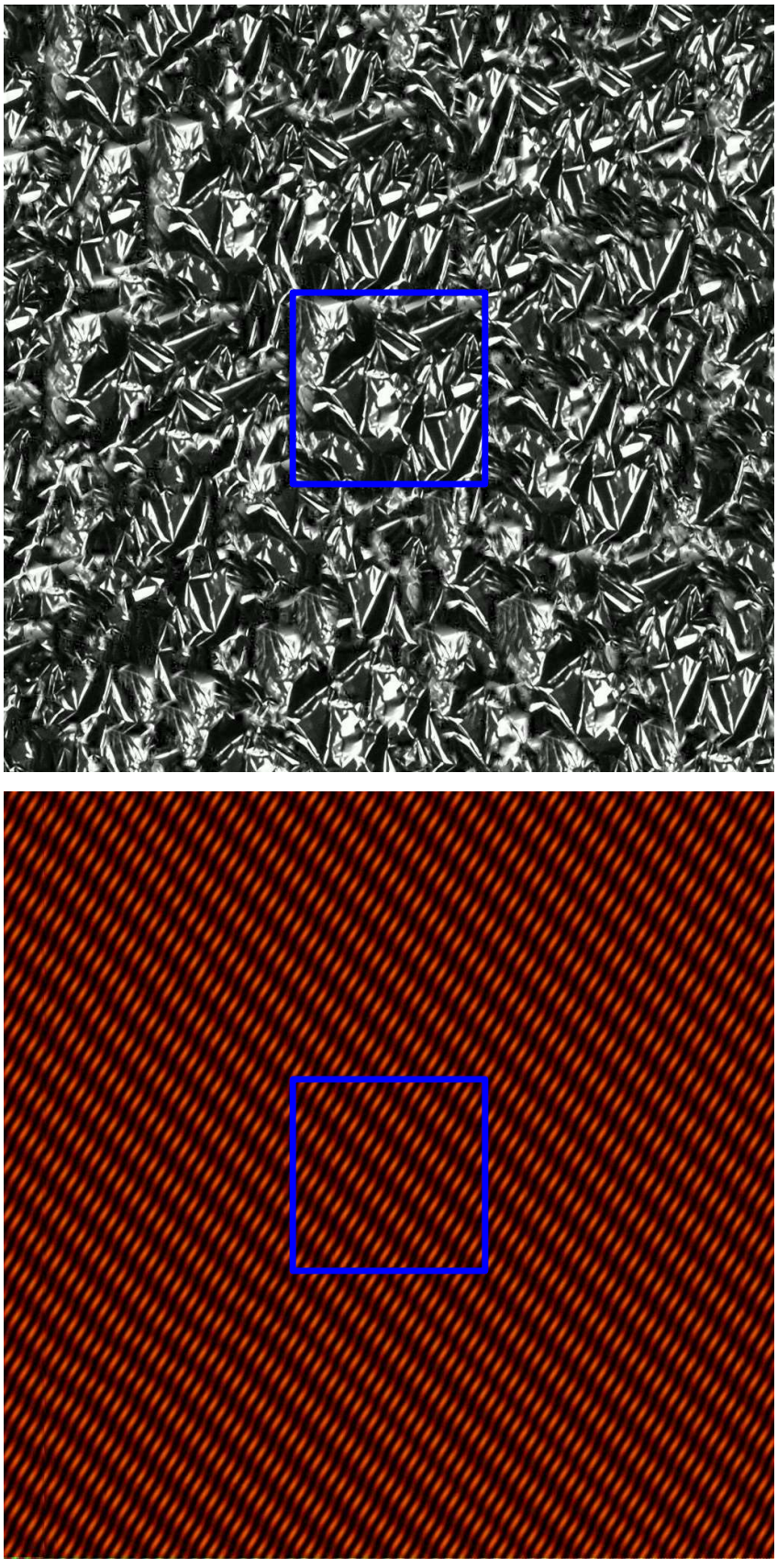


Figure B.7: *DT-CWT TexSyn Results*

C

2D Autoregressive Model

Under the 2D Autoregressive (AR) model each pixel $I(\mathbf{x})$ at the spatial position $\mathbf{x} = [x, y]^T$ in the image $\mathbf{I}$ can be modelled as,

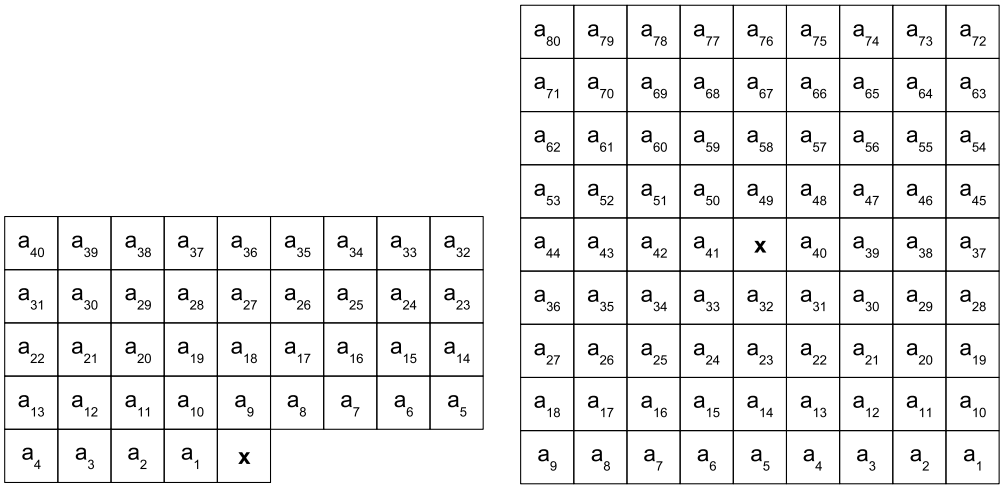
$$I(\mathbf{x}) = \sum_{k=1}^{P} a_k I(\mathbf{x} + \mathbf{q_k}) + e(\mathbf{x}) \tag{C.1}$$

where a_k for $k = 1, \cdots, P$ are the P coefficients of the model, $e(\mathbf{x})$ is an added excitation or residual error defined such that $e(\mathbf{x}) \sim \mathcal{N}(0, \sigma_e(\mathbf{x}))$ and $\mathbf{q_k}$ are the P spatial offset vectors used to index the support neighbourhood of the model. According to the 2D AR model, a particular pixel at site $\mathbf{x}$ can be predicated by a linear combination of pixels in the image $\mathbf{I}$ plus an added excitation $e(\mathbf{x})$. The model parameters are $\theta = [\mathbf{a}, \sigma_e^2]$. Figure C.1 shows the 9×9 causal and non-causal neighbourhood structures used in the implementation of the 2D AR process carried out for this work.

In order to estimate pixel values using the 2D AR modelling procsss, the first task is to estimate the model parameters, $\theta = [\mathbf{a}, \sigma_e^2]$. To estimate the model coefficients $\mathbf{a}$ the least squared estimate is used. Let E denote the sum of the squared errors at each site in the image $\mathbf{I}$. E is given as,

$$E = \sum_{\mathbf{x} \in \mathbf{X}} e^2(\mathbf{x}) \tag{C.2}$$

From equation (C.1), $e^2(\mathbf{x})$ can be written as,



(i) 9×9 Causal (ii) 9×9 Non-causal

Figure C.1: 9×9 *causal and non-causal neighbourhood structures which can be used as support in the 2D AR modelling process.*

$$E = \sum_{\mathbf{x} \in \mathbf{X}} \left(I(\mathbf{x}) - \sum_{k=1}^{P} a_k I(\mathbf{x} + \mathbf{q_k}) \right)^2 \tag{C.3}$$

At the minimum value of E, $\frac{\partial E}{\partial a_k} = 0$ for the P model coefficients a_k. By differentiating E with respect to a_k, P equations representing the set of P unknowns, i.e. a_k are obtained. These equations are

$$\begin{aligned}
\frac{\partial E}{\partial a_1} &= \frac{\partial}{\partial a_1} \left(\sum_{\mathbf{x} \in \mathbf{X}} \left(I(\mathbf{x}) - \sum_{k=1}^{P} a_k I(\mathbf{x} + \mathbf{q_k}) \right)^2 \right) = 0 \\
\frac{\partial E}{\partial a_2} &= \frac{\partial}{\partial a_2} \left(\sum_{\mathbf{x} \in \mathbf{X}} \left(I(\mathbf{x}) - \sum_{k=1}^{P} a_k I(\mathbf{x} + \mathbf{q_k}) \right)^2 \right) = 0 \\
&\vdots \\
&\vdots \\
\frac{\partial E}{\partial a_P} &= \frac{\partial}{\partial a_P} \left(\sum_{\mathbf{x} \in \mathbf{X}} \left(I(\mathbf{x}) - \sum_{k=1}^{P} a_k I(\mathbf{x} + \mathbf{q_k}) \right)^2 \right) = 0
\end{aligned} \tag{C.4}$$

Let m be any scalar used to index the $\mathbf{a}$ coefficients, then the equation for a_m can be written as,

$$0 = 2\sum_{\mathbf{x}\in\mathbf{X}}\left(I(\mathbf{x}) - \sum_{k=1}^{P} a_k I(\mathbf{x}+\mathbf{q_k})\right)\left(-I(\mathbf{x}+\mathbf{q_m})\right) \text{ for } m = 1,\cdots,P \tag{C.5}$$

Rearranging (C.5) results in the following,

$$\sum_{\mathbf{x}\in\mathbf{X}}\sum_{k=1}^{P} a_k I(\mathbf{x}+\mathbf{q_k})I(\mathbf{x}+\mathbf{q_m}) = \sum_{\mathbf{x}\in\mathbf{X}} I(\mathbf{x})I(\mathbf{x}+\mathbf{q_m}) \text{ for } m = 1,\cdots,P \tag{C.6}$$

This can be written in matrix form,

$$\mathbf{Ra} = \mathbf{r} \tag{C.7}$$

where,

$$\begin{bmatrix} \sum_{\mathbf{x}\in\mathbf{X}}\sum_{k=1}^{P} I(\mathbf{x}+\mathbf{q_k})I(\mathbf{x}+\mathbf{q_1}) \\ \sum_{\mathbf{x}\in\mathbf{X}}\sum_{k=1}^{P} I(\mathbf{x}+\mathbf{q_k})I(\mathbf{x}+\mathbf{q_2}) \\ \vdots\vdots \\ \vdots\vdots \\ \sum_{\mathbf{x}\in\mathbf{X}}\sum_{k=1}^{P} I(\mathbf{x}+\mathbf{q_k})I(\mathbf{x}+\mathbf{q_P}) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} \sum \mathbf{x}\in\mathbf{X} I(\mathbf{x})I(\mathbf{x}+\mathbf{q_1}) \\ \sum \mathbf{x}\in\mathbf{X} I(\mathbf{x})I(\mathbf{x}+\mathbf{q_2}) \\ \vdots \\ \vdots \\ \sum \mathbf{x}\in\mathbf{X} I(\mathbf{x})I(\mathbf{x}+\mathbf{q_P}) \end{bmatrix} \tag{C.8}$$

To solve for $\mathbf{a}$, the rules of matrix inversion and multiplication can be used.

$$\begin{aligned} \mathbf{R}^{-1}\mathbf{Ra} &= \mathbf{R}^{-1}\mathbf{r} \\ \mathbf{a} &= \mathbf{R}^{-1}\mathbf{r} \end{aligned} \tag{C.9}$$

Once the values of $\mathbf{a}$ have been estimated, the error at each site can be estimated from the difference in the true value and the AR predicted value. This error can then be normalised and the value of σ_e^2 is taken as the variance of this normalised error.

Bibliography

[1] The Foundry Ltd. http://www.thefoundry.co.uk/.

[2] Nvidia Corporation. Nvidia software developer site. http://developer.nvidia.com.

[3] PCI express specification. http://developer.intel.com/technology/pciexpress/devnet/desktop.htm.

[4] PCI special interest group. http://www.pcisig.com.

[5] Accelerated graphics port technology (AGP). http://developer.intel.com/technology/agp/, 1998.

[6] IEEE Transactions on Image Processing Edics. *IEEE Transactions on Image Processing*, 15(8):2470–2470, 2006.

[7] A. Achim, A. Bezerianos, and P. Tsakalides. Novel Bayesian multiscale method for speckle removal in medical ultrasound images. *IEEE Transactions on Medical Imaging*, 20(8):772–783, August 2001.

[8] M.D. Adams and F. Kossentini. Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation analysis. *IEEE Transactions on Image Processing*, 9(6):1010–1024, June 2000.

[9] M.D. Adams and R. Ward. Wavelet transforms in the JPEG-2000 standard. In *IEEE Pacific Rim Conference on Computers and Signal Processing (PACRIM), vol. 1*, pages 160–163, August 2001.

[10] R. Aditya and S. Ghosal. An integrated segmentation technique for interactive image retrieval. In *IEEE International Conference on Image Processing (ICIP)*, volume 3, pages 762–765, Vancouver, Canada, September 2000.

[11] K. Alsabti, S. Ranka, and V. Singh. An efficient k-means clustering algorithm. In *First Workshop on HIgh Performance Data Mining*, Florida, USA, March 1998.

[12] V.I. Arnold and A. Avez. *Ergodic Problems of Classical Mechanics*. Benjamin, 1968.

[13] S. Arya, D.M. Mount, N.S. Netanyahu, and R. Silverman. An approximate algorithm for nearest neighbor searching in fixed dimensions. *Journal of ACM*, 45(6):891–923, 1998.

[14] M. Ashikhmin. Synthesizing natural textures. In *ACM Symposium on Interactive 3D Graphics*, pages 217–226, Research Triangle Park, North Carolina, USA, March 2001.

[15] S.A. Barker. *Image Segmentation using Markov Random Field Models.* PhD thesis, University of Cambridge, Wolfson College, July 1998.

[16] S.A. Barker and P.J.W. Rayner. Unsupervised image segmentation using Markov random field models. *Lecture Notes in Computer Science*, 1223, 1997.

[17] S.A. Barker and P.J.W. Rayner. Unsupervised image segmentation using Markov random field models. *IEEE Pattern Recognition*, 33:587–602, 2000.

[18] R. Beichel, H. Bischof, F. Leberl, and M. Sonka. Robust active appearance models and their application to medical image analysis. *IEEE Transactions on Medical Imaging*, 24(9):1151–1169, Septmeber 2005.

[19] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889, August 2003.

[20] J. Besag. Spatial interaction and the statistical analysis of lattice. *Journal of the Royal Statistical Society, Series B*, 36:192–236, 1974.

[21] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48:259–302, 1986.

[22] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, 1981.

[23] K.S. Bhat, S.M. Seitz, J.K. Hodgins, and P.K. Khosla. Flow-based video synthesis and editing. In *ACM SIGGRAPH, vol. 23, no. 3*, pages 360–363, Los Angeles, USA, August 2004.

[24] A. Del Bimbo. *Visual Information Retrieval.* Academic Press Inc. (London) Ltd, 1999.

[25] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *International Conference on Computer Vision*, pages 489–495, 1999.

[26] J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Computer Graphics*, pages 361–368. ACM SIGGRAPH, 1997.

[27] R. Bornard. *Probabilistic Approaches for the Digital Restoration of Television Archives.* PhD thesis, Ecole Centrale Paris, 2002.

[28] C.A. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *IEEE Transactions on Image Processing*, 3(3):162–177, March 1994.

[29] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, 1990.

[30] Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *IEEE Internation Conference on Computer Vision*, volume 1, pages 105–112, 2001.

[31] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *International Workshop Enery Minimization Methods in Computer Vision*, pages 26–33, 2003.

[32] Y. Boykov, L. Veksler, and R. Zabih. Fast approximate energy minimisation via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.

[33] K. Brady. *A Probabilistic Framework for Adaptive Texture Description.* PhD thesis, Universite De Nice-Sophia Antipolis, December 2003.

[34] P. Brodatz. Texture collection. A photographic album for artists and designers, Dover, New York, 1965.

[35] S. Brooks and N. Dodgson. Self-similarity based texture editing. *ACM Transactions on Graphcis (Proceedings of ACM SIGGRAPH 2002)*, 21(3):653–656, 2002.

[36] S. Buecher and C. Lantuejoul. Use of watershed in contour detection. *In Proceedings of International Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation, Rennes, France*, pages 17–21, September 1979.

[37] C.S. Burrus, R.A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer.* Prentice Hall, 1st edition, 1997.

[38] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, April 1983.

[39] M. Cagnazzo, G. Poggi, and L. Verdoliva. The advantage of segmentation in SAR image compression. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS), volume 6*, pages 3320–3322, Toronto, Canada, June 2002.

[40] P. Campisi and G. Scarano. A multiresolution approach for texture synthesis using the circular harmonic functions. *IEEE Transactions on Image Processing*, 11(1), January 2002.

[41] R. Campisi, D. Hatzinakos, and A. Neri. A perceptually lossless, model-based , texture compression technique. *IEEE Transactions on Image Processing*, 9(8):1325–1336, 2000.

[42] J. Canny. Computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 679–698, November 1986.

[43] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, August 2002.

[44] C. Carson, M. Thomas, S. Belongie, J., Hellerstein, and J. Malik. Blobworld: a system for region-based image indexing and retrieval. Technical Report UCB/CSD-99-1041, EECS Department, University of California, Berkeley, 1999.

[45] D. Charalampidis. Texture synthesis: Textons revisited. *IEEE Transactions on Image Processing*, 15(3):777–787, 2006.

[46] R. Chellappa and S. Chatterjee. Classification of textures using Gaussian Markov random fields. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(4):959–963, August 1985.

[47] R. Chellappa and R. Kashyap. Texture synthesis using 2-d non-causal autoregressive models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(1):194–203, February 1985.

[48] J.L. Chen and A. Kundu. Unsupervised texture segmentation using multichannel decomposition and hidden Markov models. *IEEE Transactions on Image Processing*, 4(5):603–619, 1995.

[49] H.-D. Cheng and Y. Sun. A hierarchical approach to color image segmentation using homogeneity. *IEEE Transactions on Image Processing*, 9(12):2071–2082, December 2000.

[50] H. Choi and R.G. Baraniuk. Multiscale image segmentation using wavelet-domain hidden Markov models. *IEEE Transactions on Image Processing*, 10(9):1309–1321, September 2001.

[51] F. S. Cohen and D. B. Cooper. Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:195–219, 1987.

[52] M.F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. *ACM Transaction on Graphics, SIGGRAPH 2003*, July 2003.

[53] B. Collis, S. Robinson, and P. White. Wire removal. In *The IEE 1st European Conference on Visual Media Production (CVMP)*, pages 133–138, London, UK, March 2004.

[54] G.C. Cross and A.K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:25–39, 1983.

[55] M.S. Crouse, R.D. Nowak, and R.G. Baraniuk. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46(4):886–902, April 1998.

[56] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005, 1990.

[57] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Seriew B*, 39(1):1–38, 1977.

[58] H. Derin and H. Elliott. Modelling and segmentation of noisy and textured images using Gibbs random fields. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 9:39–55, 1987.

[59] P. deRivaz. *Complex Wavelet Based Image Analysis and Synthesis*. PhD thesis, University of Cambridge, UK, October 2000.

[60] P. deRivaz and N. Kingsbury. Complex wavelet features for fast texture image retrieval. In *IEEE Conference on Image Processing (ICIP)*, pages 25–28, Kobe Japan, October 1999.

[61] P. deRivaz and N. Kingsbury. Complex wavelet features for fast texture image retrival. In *IEEE Conference on Image Processing (ICIP)*, pages 25–33, Kobe Japan, October 1999.

[62] X. Descombes, R. D. Morris, J. Zerubia, and M. Berthod. Estimation of markov random field prior parameters using markov chain monte carlo maximum likelihood. *IEEE Transactions on Image Processing*, 8(7):954–963, July 1999.

[63] K. Domijan and S. Wilson. A Bayesian method for automatic landmark detection in segmented images. In *Workshop on Machine Learning Techniques for Processing Multimedia Content*, pages 70–74, Bonn, Germany, August 2005.

[64] R. Dubes, A. Jain, S. Nadabar, and C. Chen. MRF model-based algorithms for image segmentation. In *IEEE International Conference on Pattern Recognition*, pages 808–814, Atlantic City, USA, June 1990.

[65] R.M. Dufour. Statistical signal restoration with 1/f wavelet domain prior models. *Signal Processing*, 78:209–307, 1998.

[66] J.C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

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

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

[69] G. Fan and X.-G. Xia. Wavelet-based texture analysis and synthesis using hidden Markov models. *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, 50(1):106–120, 2003.

[70] H. Fang and J.C. Hart. Textureshop: Texture synthesis as a photograph editing tool. In *ACM SIGGRAPH, volume 23, number 3*, pages 354–358, Los Angeles, USA, August 2004.

[71] C. Fiorio and R. Nock. Image segmentation using a generic, fast and non-parametric approach. *Proceedings. Tenth IEEE International Conference on Tools with Artificial Intelligence*, pages 450–458, November 1998.

[72] D. Freedman, R.J. Radke, T. Zhang, Y. Jeong, D.M. Lovelock, and G.T.Y. Chen. Model-based segmentation of medical imagery by matching distributions. *IEEE Transactions on Medical Imaging*, 24(3):281–292, March 2005.

[73] B. Frey. *Graphical Models for Machine Learning and Digital Communications*. MIT Press, Cambridge, MA, 1998.

[74] J.H. Friedman. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[75] D. Gabor. Theory of communication. *Journal of the IEEE*, 93:429–457, 1946.

[76] C. Gallagher, F. Kelly, and A.C. Kokaram. Fast scale invariant texture synthesis with GPU acceleration. In *IEE European Conference on CVMP Visual Media Prodcution*, London, UK, December 2005.

[77] C. Gallagher and A.C. Kokaram. Wavelet based texture synthesis. In *The Irish Machine Vision and Image Processing Conference (IMVIP)*, pages 34–41, Dublin, Ireland, September 2004.

[78] C. Gallagher and A.C. Kokaram. Nonparametric wavelet based texture synthesis. In *IEEE International Conference on Image Processing (ICIP)*, Genova, Italy, September 2005.

[79] J. Gauch and C. Hsia. A comparison of three color image segmentation algorithms in four color spaces. In *SPIE Visual Communications on Image Processing*, volume 1818, 1992.

[80] D. Geman. Random fields and inverse problems in imaging. In *Lecture Notes in Mathematics, vol. 1427*, pages 113–193. Springer-Verlag, 1991.

[81] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):367–383, 1992.

[82] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[83] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[84] R.C. Gonzalez and R.E. Woods. *Digital Image Processing, 2nd Edition*. Addison-Wesley, 1992.

[85] GPGPU. General purpose computation on the GPU. http://www.gpgpu.org/.

[86] A. Graps. Introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2(2):50–61, 1995.

[87] D.M. Greig, B.T. Porteous, and A.H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B (Methodological*, 51(2):271–279, 1989.

[88] H. Grossauer. A combined PDE and texture synthesis approach to inpainting. In *Euopean Conference on Computer Vision (ECCV) Part II*, Prague, Czech Republic, May 2004.

[89] J. Guan and G. Qu. Interactive image segmentation using optimization with statistical priors. In *European Conference on Computer Vision (ECCV)*, May 2006.

[90] A. Gyaourova, C. Kamath, and I.K. Fodor. Undecimated wavelet transforms for image de-noising. Technical Report UCRL-ID-150931, Center for Applied Scientific Computing at the Lawrence Livermore National Laboratory, 2002.

[91] J. Ha and K.-K. Ma. Fuzzy color histogram and its use in color image retrieval. *IEEE Transactions on Image Processing*, 11(8):944–952, August 2002.

[92] J. Hadamard. Sur les problmes aux drives partielles et leur signification physique. *Princeton University Bulletin*, pages 49–52, 1902.

[93] R.M. Haralick. A texture-context feature extraction algorithm for remotely sensed imagery. In *IEEE Decision and Control Conference*, pages 650–657, December 1971.

[94] R.M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67:786–804, 1972.

[95] R.M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67:786–804, 1979.

[96] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *CVGIP*, 29(1):100–132, January 1985.

[97] M. Harris. Gpgpu: General-purpose computation on gpus. In *Game Developers Conference*, 2005.

[98] P. Harrison. A non-hierarchical procedure for re-synthesis of complex textures. In *WSCG*, pages 190–197, University of West Bohemia, Plzen, Czech Republic, 2001.

[99] M. Hassner and J. Sklansky. The use of Markov random fields as models of texture. *Computer Graphics and Image Processing*, 12:357–370, 1980.

[100] D.J. Heeger and J.R. Bergen. Pyramid based texture analysis or synthesis. In *ACM SIGGRAPH*, pages 229–238. Addison Wesley, August 1995.

[101] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin. Image analogies. In *ACM SIGGRAPH*, pages 327–340, Los Angeles, USA, August 2001.

[102] P.R. Hill, C.N. Canagarajah, and D.R. Bull. Image segmentation using a texture gradient based watershed transform. *IEEE Transactions on Image Processing*, 12(12):1618–1633, December 2003.

[103] R. Hu and M.M. Fahmy. Texture segmentation based on a hierarchical Markov random field model. *Signal Processing*, 26:285–305, 1992.

[104] T.S. Huang, S. Mehratra, and K. Ramchandran. Multimedia analysis and retrieval system (MARS) project. In *33rd Annual Clinic on Library Application of Data Processing - Digital Image Access and Retrieval*, University of Illinois at Urbana-Champaign, March 1996.

[105] S. Intajag, K. Paithoonwatanakij, and A.P. Cracknell. Iterative satellite image segmentation by fuzzy hit-or-miss and homogeneity index. *IEEE Proceedings on Vision, Image and Signal Processing*, 153(2):206–214, April 2006.

[106] ISO/IEC 15444-I ISO/IEC. Information technology - JPEG 2000 image coding system - part 1: Core coding system, 2001.

[107] G. Jacovitti, A. Neri, and G. Scarano. Texture synthesis-by-analysis with hard limited Gaussian process. *IEEE Transactions on Image Processing*, 7(11):1615–1621, 1998.

[108] A.K. Jain. *Fundamentals of digital image processing.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[109] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[110] A.K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[111] H. Jeffreys. *Theory of Probability.* Oxford University Press, 1939.

[112] S.C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967.

[113] B. Julesz, E.N. Gilbert, L.A. Shepp, and H.L Frisch. Inability of humans to discriminate between visual textures that agree in second-order statistics – revisited. *Perception 2*, pages 391–405, 1973.

[114] A.H. Kam. *A General Multiscale Scheme for Unsupervised Image Segmentation*. PhD thesis, University of Cambridge, Trinity College, 2000.

[115] A.H. Kam and W.J. Fitzgerald. Unsupervised multiscale image segmentation. In *IEEE International Conference on Image Analysis and Processing (ICIP)*, pages 316–321, September 1999.

[116] F. Kelly. *Fast Probabilistic Inference and GPU Video Processing*. PhD thesis, University of Dublin, Trinity College, June 2006.

[117] N. Kingsbury. The dual-tree complex wavelet transform: A new technique for shift invariance and directional filters. In *IEEE DSP Workshop paper no. 86*, Bryce Canyon UT, USA, 1998.

[118] N. Kingsbury. Image processing with complex wavelets. *Phil. Trans. Royal Society London A*, pages 2543–2560, September 1999.

[119] N. Kingsbury. Shift invariant properties of the dual-tree complex wavelet transform. In *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 1999.

[120] N. Kingsbury. Complex wavelets for shift invariant analysis and filtering of signals. *Journal of Applied and Computational Harmonic Analysis*, 10(3):234–253, May 2001.

[121] N. Kingsbury and J.F.A. Magarey. Wavelet transforms in image processing. In *1st European Conference on Signal Analysis and Prediction*, pages 23–34, Prague, Czech., June 1997.

[122] J.J. Koenderink. The structure of images. *Biological Cybernetics*, 50(5):363–370, 1984.

[123] A. Kokaram. Parametric texture synthesis for filling holes in pictures. In *IEEE International Conference on Image Processing (ICIP)*, pages 325–328, New York, USA, September 2002.

[124] A.C. Kokaram. A statistical framework for picture reconstruction using 2d AR models. *Image and Vision Computing*, pages 165–171, 2004.

[125] V. Kolmogorov. Primal-dual algorithm for convex Markov random fields. Technical Report MSR-TR-200-117, Microsoft Research, Cambridge, UK, September 2005.

[126] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision (ECCV)*, volume 3, pages 82–96, 2002.

[127] V. Kolmogorov and R. Zabih. What energy functions can be minimised via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, February 2004.

[128] J. Kovacevic and I. Daubechies (editors). Special issue on wavelets. *Proceedings of the IEEE*, 84(4):507–685, April 1996.

[129] S. Krishnamachari and R. Chellappa. Multiresolution Gauss-Markov random field models for texture segmentation. *IEEE Transactions on Image Processing*, 6(2):251–267, February 1997.

[130] P. Kumar, P.H. Torr, and A. Zisserman. OBJ CUT. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 18–25, San Diego, USA, 2005.

[131] Vivek Kwatra, Arno Schdl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, 22(3):277–286, July 2003.

[132] M. Lades, J.C. Vorbuggen, J. Buhmann, J. Lange, C. von der Malsburg, R.P. Wurtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, March 1993.

[133] A. Laine and J. Fan. Texture classification by wavelet packet signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1190, 1993.

[134] T.-W. Lee and M.S. Lewicki. Unsupervised image classification, segmentation, and enhancement using ICA mixture models. *IEEE Transactions on Image Processing*, 11(3):270–279, March 2002.

[135] Ming C. Lin and Dinesh Manocha. Interactive geometric computations using graphics hardware. In *SIGGRAPH'02 Tutorial Course 31*, 2002.

[136] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communication*, 28(1):84–95, 1980.

[137] P. Loo. *Digital Watermarking using Complex Wavelets*. PhD thesis, University of Cambridge, Trinity College, March 2002.

[138] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, University of Berkeley, USA, 1967. California Press.

[139] S. Mallat. *A Wavelet Tour of Signal Processing*. New York: Academic, 2 edition, 1999.

[140] S.G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[141] B.S. Manjunath, T. Simchony, and R. Chellappa. Stochastic and deterministic networks for texture segmentation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38:1039–1049, June 1990.

[142] D. Marr. *Vision*. W.H.Freeman and Company, 1982.

[143] D. Marr and E. Hildreth. Theory of edge detection. *Proc. of the Royal Society of London*, pages 187–217, 1980.

[144] J.L Marroquin, E.A. Santana, and S. Botello. Hidden Markov measure field models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1380–1387, November 2003.

[145] F. Meyer and S. Buecher. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(9):21–46, September 1990.

[146] M. Mignotte. Nonparametric multiscale energy-based model and its application in some imagery problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):184–197, February 2004.

[147] Y. Nakagawa and A. Rosenfeld. Some experiments on variable thresholding. *IEEE Pattern Recognition*, 11:191–204, 1979.

[148] R. Neher and A. Srivastava. A Bayesian MRF framework for labeling terrain using hyperspectral imaging. *IEEE Transactions on Geoscience and Remote Sensing*, 43(6):1363–1374, June 2005.

[149] R. Nevatia. Image segmentation. *Handbook of Pattern Recognition and Image Processing*, pages 215–231, 1986.

[150] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A.E. Lefohn, and T.J. Purcell. A survey of general purpose computation on graphics hardware. In *Eurographcis 2005, Start of the Art Reports*, pages 21–51, August 2005.

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

[152] N. Pal and S. Pal. A review on image segmentation techniques. *IEEE Pattern Recognition*, 26:1277–1294, 1993.

[153] A. Papoulis. *Signal Analysis*. McGrawHill, May 1977.

[154] J. Pearl. *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[155] P. Perez, M. Gangnet, and A. Blake. Poisson image editing. In *SIGGRAPH 2003*, 2003.

[156] M. Petrou, L. Shafarenko, and J. Kittler. Automatic watershed segmentation of randomly textured colour images. *IEEE Transactions on Image Processing*, 6:1530–1544, November 1997.

[157] G. Poggi, G. Scarpa, and J. Zerubia. Supervised segmentation of remote sensing images based on a tree-structure MRF model. *IEEE Transactions on Geoscience and Remote Sensing*, 43(8):1901–1911, August 2005.

[158] K. Popat and R.W. Picard. A novel cluster-based probability model for texture synthesis, classification, and compression. In *SPIE Visual Communications 1993*, pages 756–768, Massachusetts, USA, 1993.

[159] J. Portilla and E.P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–71, December 2000.

[160] J. Portilla and E.P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–71, December 2000.

[161] R.B. Potts. Some generalized order-disorder transformations. In *Cambridge Philosopical Society*, volume 48, pages 106–109, 1952.

[162] W.H Press, S.A. teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, 2nd Edition*. Cambridge University Press, 1992.

[163] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE Signal Processing Magazine*, 3(1):4–16, January 1986.

[164] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, February 1989.

[165] R.R. Rakesh, P. Chaudhuri, and C.A. Murthy. Thresholding in edge detection: A statistical approach. *IEEE Transactions on Image Processing*, 13(7):927–936, July 2004.

[166] T. Randen and J.H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

[167] K. Ratakonda and N. Ahuja. Lossless image compression with multiscale segmentation. *IEEE Transactions on Image Processing*, 11(11):1228–1237, November 2002.

[168] T.R. Reed and J.M. Hans du Buf. A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image Understanding*, 57(3):359–372, May 1993.

[169] O. Rioul and M. Vetterli. Wavelets and signal processing. *IEE Signal Processing Magazine*, 8(4), October 1991.

[170] J.B.T.M. Roerdink and A. Meijster. The watershed transform: definitions, algorithms, and parallellization strategies. *Fundamenta Informaticae*, 41:187–228, 2000.

[171] J. Romberg, H. Choi, R. Baraniuk, and N. Kingsbury. Multiscale Classification using Complex Wavelets and Hidden Markov Tree Models. In *IEEE International Conference on Image Processing*, volume 2, Vancouver, Canada, September 2000.

[172] J. Romberg, H. Choiand, and R. Baraniuk. Bayesian tree-structured image modeling using wavelet-domain hidden markov models. *IEEE Transactions on Image Processing*, 10(7):1056–1068, July 2001.

[173] J.J.K. Rooney and W.J. Fitzgerald. Bayesian model based parameter estimation and model selection in impulsive noise environments. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 141–144, Minneapolis, USA, April 1993.

[174] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH Conference, issue 3*, volume 23, pages 309–314, 2004.

[175] M. Roy, V.R. Kumar, B.D. Kulkarni, J. Sanderson, M. Rhodes, and M.V. Stappen. Simple denoising algorithm using wavelet transform. *AICHE*, 45(11):2461–2466, 1999.

[176] J.J. O Ruanaidh and W.J Fitzgerald. *Numerical Bayesian Methods Applied to Signal Processing (Statistics and Computing)*. Springer-Verlag New York Inc, 1996.

[177] D.L. Ruderman and W. Bialek. Statistics of natural images: Scaling in the woods. *Phys. Rev. Lett.*, 73(6):814–817, 1994.

[178] B.C. Russell, A.A. Efros, J. Sivic, W.T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, June 2006.

[179] T.W. Ryan, L.D. Sanders, H.D. Fisher, and A.E. Iverson. Image compression by texture modelling in the wavelet domain. *IEEE Transactions on Image Processing*, 5(1):26–36, 1996.

[180] S.I. Sadhar and A.N. Rajagopalan. Image estimation in film-grain noise. *IEEE Signal Processing Letters*, 12(3):238–241, March 2005.

[181] P.K. Saha and J.K. Udupa. Optimum image thresholding via class uncertainty and region homogeneity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):689–705, 2001.

[182] P. Saison, G. Doretto, Y. Wu, and S. Soatto. Dynamic texture recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 58–63, 2001.

[183] K. Segl and H. Kaufmann. Detection of small objects from high-resolution panchromatic satellite imagery based on supervised image segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 39(9):2080–2083, September 2001.

[184] J. Serra. Image segmentation. In *IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 345–348, September 2003.

[185] C.W. Shaffrey. *Muliscale Techniques for Image Segmentation, Classification and Retrieval*. PhD thesis, University of Cambridge, St. Edmund's College, September 2003.

[186] C.W. Shaffrey, N.G. Kingsbury, and I.H. Jermyn. Unsupervised image segmentation via Markov trees and complex wavelets. In *IEEE International Conference on Image Processing (ICIP)*, New York, USA, September 2002.

[187] C.E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27, 1948.

[188] E. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *IEEE 5th Internation Conference on Image Processing (ICIP)*, volume I, 1998.

[189] E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger. Shiftable multi-scale transforms. *IEEE Transactions on Information Theory, Special Issue on Wavelets*, 38:587–607, 1992.

[190] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *IEEE International Conference on Computer Vision (ICCV)*, Nice, France, 2003.

[191] A. Smeulders, M. Worring, S. Santini, A Guptin, and A. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1349–1380, December 2000.

[192] J. Smith. *Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression*. PhD thesis, Columbia University, 1997.

[193] J. Smith and S.-F. Chang. Transform features for texture classification and discrimination in large image databases. In *IEEE International Conference on Image Processing (ICIP)*, Texas, USA, November 1994.

[194] J. Smith and S.F. Chang. Visualseek: A fully automated content-based image query system. In *In ACM Multimedia*, Boston, USA, November 1996.

[195] J.R. Smith and S.F. Chang. Single color extraction and image query. In *International Conference on Image Processing*, volume 3, pages 528–531, October 1995.

[196] M.S. Stachowicz and D. Lemke. Image segmentation and classification using color features. In *International Symposium on Video, Image Processing and Multimedia Communications*, pages 57–64, Croatia, June 2002.

[197] J. Stam. Aperiodic texture mapping. Technical Report R046, European Research Consortium for Informatics and Mathematics (ERCIM), 1997.

[198] J. Sun, D. Gu, S. Zhang, and Y. Chen. Hidden Markov Bayesian texture segmentation using complex wavelet transform. *IEEE Proceedings of Vision, Image and Signal Processing*, 151(3):215–223, June 2004.

[199] J. Sun, H.Y. Shum, and N.N. Zheng. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, July 2003.

[200] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[201] R. Szeliski, R. Zabih, and D. Scharstein. A comparative study of energy minimisation methods for Markov random fields. In *ECCV, Part II*, pages 16–29. Springer-Verlag, Berlin Heidelberg, 2006.

[202] F. Tang, Y. Ying, J. Wang, and Q. Peng. A novel texture synthesis based algorithm for object removal in photographs. In *The 9th Asian Computing Science Conference*, Chiang Mai, Thailand, December 2004.

[203] M.A. Tanner. *Tools for Statistical Inference*. Springer-Verlag, 1996.

[204] D.S. Taubman and M.W. Marcellin. JPEG2000: standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357, August 2002.

[205] Adobe Creative Team. *Adobe Photoshop 7.0: Classroom in a Book*. Adobe Press, 2002.

[206] C.J. Thompson, S. Hahn, and M. Oskin. Using modern graphics architectures for general-purpose computing: A framework and analysis. In *International Symposium on Microarchitecture (MICRO)*, November 2002.

[207] A. Tsai, A. Yezzi Jr., W. Wells, C. Tempany, D. Tucker, A. Fan, W.E. Grimson, and A. Willsky. A shape-based approach to the segmentation of medical imagery using level sets. *IEEE Transactions on Medical Imaging*, 22(2):137–154, February 2003.

[208] M. Tuceryan and A.K Jain. *Handbook of Pattern Recognition and Computer Vision (2nd Edition) by C. H. Chen, L. F. Pau, P. S. P. Wang (eds.))*, chapter Texture Analysis, pages 207–248. World Scientific Publishing Co., 1998.

[209] L. van Gool, P. Dewaele, and A. Oosterlinck. Texture analysis: Anno 1983. *Computer Vision, Graphics and Image Processing*, 29:336–357, 1985.

[210] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro. Inpainting surface holes. In *IEEE International Conference on Image Processing (ICIP)*, September 2003.

[211] M Vetterli and J. Kovacevic. *Wavelets and Subband Coding*. Prentice Hall, January 1995.

[212] L.Y. Wei. Deterministic texture analysis and synthesis using tree structure vector quantization. In *XII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPHI)*, 1999.

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

[214] A.S. Willsky. Multiresolution Markov models for signal and image processing. *Proceedings of the IEEE*, 90(8):1396–1458, August 2002.

[215] R. Wilson and M. Spann. *Image Segmentation and Uncertainity*. Reserach Studies Press Ltd., 1988.

[216] G. Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introdcution*. Springer, 2003.

[217] L. Wiskott, J.M. Fellous, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, July 1997.

[218] Y.Q. Xu, B. Guo, and H. Shum. Chaos mosaic: Fast and memory efficient texture synthesis. Technical report msr-tr-2000-32, Microsoft Reserach, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, April 2000.

[219] C.K. Yang and W.H Tsai. Reduction of color space dimensionality by moment-preserving thresholding and its application for edge detection in color images. *Pattern Recognition Letters*, 17:481–490, 1996.

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