Texture Measures for Segmentation

Stephen Haddad

Submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfilment of the requirements
for the degree of Master of Science in Engineering.

April 2007

Abstract

Texture is an important visual cue in both human and computer vision. Segmenting images into regions of constant texture is used in many applications. This work surveys a wide range of texture descriptors and segmentation methods to determine the state of the art in texture segmentation.

Two types of texture descriptors are investigated: filter bank based methods and local descriptors. Filter banks deconstruct an image into several bands, each of which emphasises areas of the image with different properties. Textons are an adaptive histogram method which describes the distribution of typical feature vectors. Local descriptors calculate features from smaller neighbourhoods than filter banks. Some local descriptors calculate a scale for their local neighbourhood to achieve scale invariance.

Both local and global segmentation methods are investigated. Local segmentation methods consider each pixel in isolation. Global segmentation methods penalise jagged borders or fragmented regions in the segmentation. Pixel labelling and border detection methods are investigated. Methods for measuring the accuracy of segmentation are discussed.

Two data sets are used to test the texture segmentation algorithms. The Brodatz Album mosaics are composed of grayscale texture images from the Brodatz Album. The Berkeley Natural Images data set has 300 colour images of natural scenes.

The tests show that, of the descriptors tested, filter bank based textons are the best texture descriptors for grayscale images. Local image patch textons are best for colour images. Graph cut segmentation is best for pixel labelling problems and edge detection with regular borders. Non-maxima suppression is best for edge detection with irregular borders. Factors affecting the performance of the algorithms are investigated.

Declaration

I hereby certify that the work embodied in this work is the result of original research and has not been submitted for another degree at any other university or institution.

Signed by candidate

Stephen Haddad
April 2007

Acknowledgements

I would like to thank the following:

- My Supervisor, Fred Nicolls, for his help and encouragement and proof reading far more than he should have.
- National Research Foundation and De Beers for their financial support.
- The members of the DIP group for their assistance and encouragement.
- Gordon Forbes for his help proof reading and guidance through the field of texture.
- My parents and brothers for their support.

Contents

List of Figures

Chapter 1

Introduction

Texture is an important visual cue in the visual systems of humans and many other animals. It is used to distinguish one object from another, to detect the orientation of an object and many other tasks. Despite this, Forsyth [13] describes texture as "a phenomenon that is widespread, easy to recognise and hard to define". Narrowly, texture is defined by the Oxford Dictionary as "the character of a textile fabric, as to its being fine, coarse, close, loose, plain, etc..". and the word texture comes from the Latin word for a weaving. Texture is generally defined as "any natural structure having an appearance or consistence as if woven". These definitions reveal the important attribute to a visual texture, that it is a repeated pattern, like a woven surface.

A second important aspect in defining texture recognised by Forsyth [13] is that whether we call an area of an image a texture depends on the scale at which we are viewing the object. In Figure 1.1, the image of a single is not considered texture, but the image of many crosses is a textured image. This is true of many natural scenes: consider a leaf and a tree or single hair and a coat of fur.

Texture is a surface phenomenon. Gibson's ecological laws of surfaces describes every surface as having a characteristic texture which results from surfaces being "speckled and rough" [15]. A surface is speckled because the substance from which it is made is seldom homogenous, and as a result a flat surface of the substance will have varying reflectivity which is viewed as a texture, for example a smoothly polished granite surface. The reflectivity is described a the Bi-directional Reflectance Distribution Function (BRDF), which parameterises the reflectance according the incoming direction of the light, the angle at which the surface is viewed and the wavelength of the light. Smooth surfaces are very rare in nature, and most objects have an approximately smooth surface with many small undulations when viewed at a much closer distance to the one where the texture is visible. These small variations change the input parameters to the BRDF, so that the reflectivity of the surface as seen by the viewer is not constant. An example of this sort of textured surface is a pile of stones or an

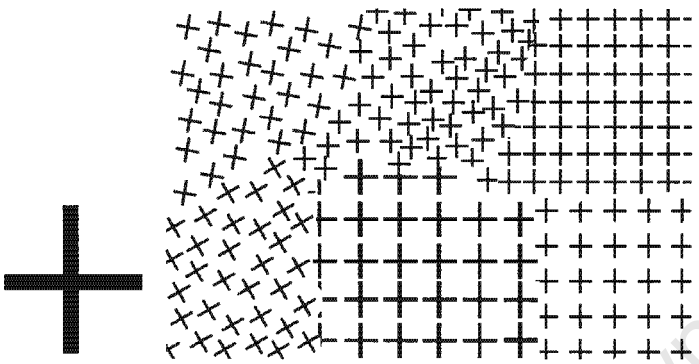


Figure 1.1: A Texture mosaic created using the same primitive throughout, a cross shown in (a). The different texture are created by rotating the primitive in the four region on the left, and by changing the distribution (structured/random) and density of the primitives on the right of the image.

orange peel.

Surface variations and changes in reflectivity describe how texture forms, but not how it is perceived by humans and other animals. Pre-attentive texture is texture that can discriminated in a very short time without conscious effort. Tests by Julesz [19] on this type of texture suggest textons, which are small elements of texture that are directly detected by the visual system. These elements are the primitives that our visual texture perceptions is built around. He found that we are able to immediately distinguish several characteristics of images of these textons, such as orientation and size, and use them to distinguish different textures. Figure 1.1 shows a textured surface consisting entirely of one primitive, a cross. Despite this we are able to sense 6 different textures in the image. The textures are being differentiated based on orientation, structure, density and size.

These features are detected by neurons which emphasise or suppress patterns based on frequency, scale and measures. We do not sense visual information in the form of a mental photograph, but rather we build a *primal sketch* [25], which collates the responses of the neurons to form a mental model which specifies important features of the image like blobs and bars which can be used directly for building a model of the environment. Marr describes how we use the texture information in the form of this primal sketch to sense information about the surfaces in the world. We can use properties of the texture, such as elongation or changes in density to infer the orientation of the surfaces towards us. We also use texture to discriminate between objects and use this for other computations such as apparent motion and stereopsis, based on the assumption that an object in one view will best match the same object in another view.

In the same way that animals (including humans) make use of texture both to process the images sensed by the eyes and then to construct a higher level mental picture of their environment, computer

vision seeks to use texture information in the images both in low-level image processing and in inferring high level information about objects in the image. Such applications include any calculations of point correspondence such as stereo reconstruction and optical flow. In these application, objects have changed position and orientation relative to the camera, and so an appropriate texture measure is more useful than a correlation measure between images. Texture also plays an important part in any object recognition system, where it plays a part both in the low-level segmentation of each image into objects and the high-level description of those objects, and to infer non-image data about objects and surfaces in the image, such as chemical composition.

1.1 The Texture Segmentation Problem

Texture in the environment is an important visual cue used to build a model of the world surrounding an organism. It is an important assumption of the human visual system in describing the world, as described by Gibson [15] and Marr [25] . We assume that the world is made up of surfaces, and that these surfaces have a distinct texture that come discriminated and used processing information about these surfaces. This thesis focuses on discriminating one texture from another in an image and use the texture description to segment the image.

This is an information processing task, taking the information in the image and producing a labelling corresponding to different textures. Marr describes three levels at which an information processing task can be understood [25]: what is the task in theoretical terms and what is the aim; how can the theory be implemented, in terms of algorithm and information representation; and how can the representation and algorithm be realised physically. The focus of this work is on the second level, how can theory be translated in a representation and algorithm to perform the task. What is to be performed is well understood. We want to segment an image based on the textural information. Our model is assuming two things. Firstly that texture, which arises from variations in BRDF or speckledness, and variation in surface normal or roughness can be modelled as a flat surface with variations only in reflectivity. The second assumption is that for natural images, which are complex scenes at different distances from the camera, have textural information that is consistent across objects despite the variation in surface orientation. The algorithms consider neither the 3D geometry of the scene in each image nor the lighting conditions affecting the image. As a result the third and fourth assumptions are that all surfaces are viewed normal to the camera and that texture in the image represents the actual texture of the surface and is not affected by lighting.

The actual implementation is described, but not the main aspect of study. Many implementations are based on the work of others. Rather comparisons between representations of texture and the algorithms that use these representations for segmentation are studied with a view to choosing a description and process that is best for segmenting texture.

The main focus of this work then is to investigate the effectiveness of texture measures and segmentation algorithms applied to different types of images. The problem can be divided into the following parts. Firstly, choosing a representation which takes the raw image data and calculates a feature vector to describe the appearance, with regard to texture and colour, of each pixel. There are many choices as to what type of representation is best, principally to dow with what changes in the appearance of the texture we want the representation to be invariant to. Figure 1.1 illustrates how different texture representations can give different segmentations. Many descriptors are invariant to a rotation or scaling of the texture. Such descriptors would find the top and bottom left and the top right and bottom middle sections of texture in the image to be the same.

The second part is mapping each point in the high-dimensional feature space to a discrete set of labels, representing objects or regions of similar appearance. This is the segmentation part which processes texture into higher level information.

State of the art methods of describing texture and segmenting an image will be compared and the best algorithm for each will be chosen.

1.2 Overview of Layout

The work in this thesis is divided into three parts. Part 1 reviews the state of the art in texture measure and segmentation. Part 2 describes the data and algorithms tested used to generate the results used to choose the best texture and segmentation methods. Part 3 presents the results and conclusions.

Part 1 consists of Chapters 2, 3 and 4. Chapter 2 introduces how texture is processed in the domain of digital images. Extracting certain information from an image using a filter bank and using textons as a simpler representation of an image to better model texture with a histogram are discussed. In Chapter 3, local texture descriptors are investigated. Selecting a local scale for these descriptors is discussed, and then some widely used descriptors are presented. Chapter 4 review both pixel labelling methods and border detection methods are discussed. Pixel labelling methods assign a class label to each pixel. Border detection methods detect whether a pixel is on the border.

Chapters 5 and 6 make up Part 2. Chapter 5 describes the data sets used to test the texture segmentation algorithms. Chapter 6 presents the details of these algorithms. Figures 7.1 and 7.2 are particularly important as these tables list all the algorithms used and which components make up which algorithms.

Chapters 7 and 8 constitute Part 3. Chapter 7 presents the results of all the algorithms and data sets, emphasising certain aspects of the results as they relate to the concepts discussed in Part 1. Figures 7.1 and 7.2 are again important as the elements of the graphs of the results are numbered according

to these tables. Chapter 8 presents the conclusions than can be drawn from the results in Chapter 7. These include which of the algorithms tested are best for texture segmentation and what work might be done in the future to improve the results further.

Chapter 2

Texture Methods

Texture is a very important part of vision systems in the natural world. It is a visual cue which allows animals to distinguish between objects, determine depth and shape, and form a high level model of their environment. The aim in computer vision is to achieve the same level of functionality in texture description as is found in the human visual system (HVS) . This means not only copying processes found in the HVS, but also applying different techniques for greater efficiency in narrow problem domains and to images beyond the realm of human sight.

In computer vision and image processing, data are recorded in the form of a digital image consisting of pixels on a rectangular grid. Texture is experienced as a pattern repeated, either deterministically or stochastically, over the grid. As result, unlike colour, one grid location cannot be said to have a texture, rather: the pixels in a local neighbourhood of that pixel will have a pattern. Depending on the period of the texture, larger or smaller neighbourhoods are needed to accurately measure the statistics of the texture.

The neighbourhood of the pixel, called an image patch, is effectively a point in a high-dimensional feature space, and has dimension equal to the number of pixels in the patch. This space is called the *image space*. We could, in practice, use this as the descriptor for our texture, but it has several undesirable properties. Firstly, in order to capture a texture with a large period or at high resolution, the vector would have to be large (20x20 pixels gives a 400-dimension feature space). Secondly, it is not invariant to changes in the appearance of a texture in an image, such as changes in intensity or orientation, which do not affect a human's perception of the image. Consequently, such changes should not significantly affect the descriptor, as is the case when using the intensity or colour neighbourhood values.

Although the image space for either a whole image or an image patch is enormous, interesting images with high-level coherence, like images of the natural world, occupy only a very small part of this space. The set of all images of a particular texture will also be found on some manifold of the feature space. A manifold is a subset of an n-dimensional space that can be represented in an space with less than n dimensions. The task in texture characterisation is to, either explicitly or implicitly, learn the manifold that best describes the texture. The points in the high-dimensional image space are then projected on to the manifold which is parameterised by fewer parameters.

The focus for this application is on texture segmentation, with several classes of texture to be represented in each image. This changes the optimal manifold. The best descriptor is not just one that most accurately describes all texture present, but the one that best differentiates different classes of texture so that an accurate segmentation or classification can be performed.

An obvious starting point in a survey of texture descriptors would be to approach the problem from a purely pattern recognition perspective. We have a multi-class data set with high dimensionality descriptors. A dimensionality reduction algorithm can be used to reduce the number features in the descriptor, and also produce features which capture the variability of the texture. Dimensionality reduction has been an area of active research for a long time so there are many algorithms available for this purpose [6]. Popular algorithms include Principal Component Analysis (PCA) and Multi-Dimensional Scaling (MDS). The problem is that the raw image patches contain features which meaningfully describe the texture as well as lot of data which does not differentiate textures in an image, which should not be presented for use by a segmentation algorithm.

There are many varieties of texture features and descriptors. For segmentation, only measures that can be calculated on a small image patch are worth considering. The discussion of texture is structured as follows. Section 2.1 looks at types of filter banks commonly used. Section 2.2 looks at using histograms to describe texture, including textons.

2.1 Filter Banks

Filter Banks are a widely used first step for many problems in computer vision. A filter bank is an array of filter kernels which are convolved with an image in order to emphasise or detect certain patterns in an image and to suppress others, so that only the relevant information is present. A linear, shift-invariant filter calculates a new value for pixel in the image based on the pixels its local neighbourhood [14]. The pixels in the neighbourhood are weighted by an array of values called the *filter kernel* and summed. The pairing of pixels and weights is determined by the position of a pixel in the neighbourhood relative to the central pixel. The relative position is the same for all pixels, which gives the filter the property of shift-invariance, and the weighted sum is a linear combination of the

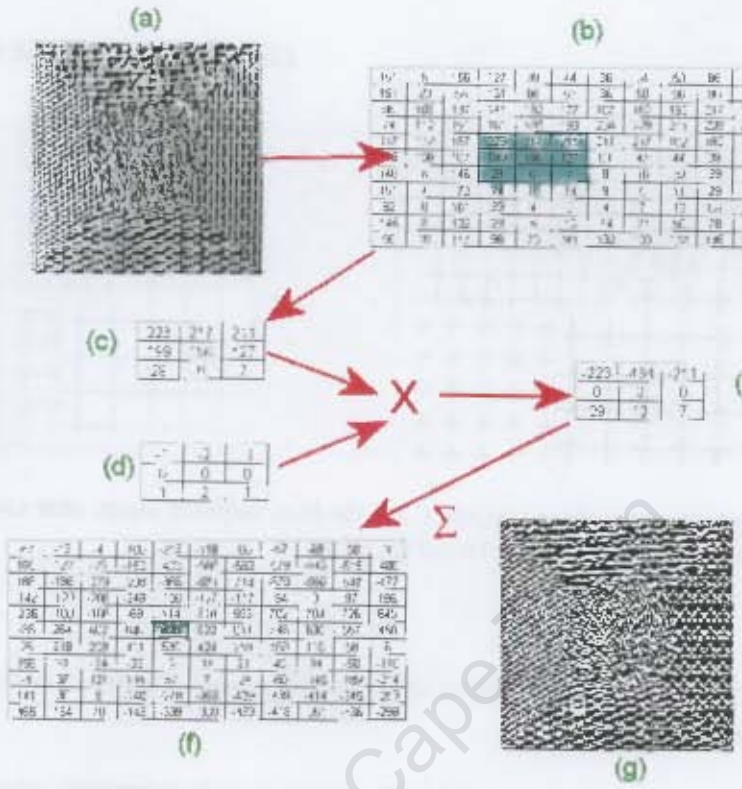


Figure 2.1: The image (a) is filtered using the filter kernel (d) to produce the filter response image (g). A small patch of the input is shown as a matrix in (b), with the neighbourhood of a pixel highlighted. The pixel values in (c) are weighted by the the kernel values, shown in (d), to produce (e), which is summed. The result is highlighted in (f), which is a patch of the filter response image (g).

pixel values, making it a linear filter. This weighted sum is calculated for each pixel in the image and is called the filter response. The filter response image is calculated by applying the filter kernel array to the image using the convolution operator.

This is linear, shift-invariant filtering. The linear combination of weighted pixels means that it is a linear filter, and the fact that the same weights are used throughout the image gives it the property of shift-invariance. The function of the filter is determined by the weights that make up the kernel. Typical kernels include smoothing kernels (also called a low pass filter), and edge detection kernels. Figure 2.1 demonstrates the filtering process using a horizontal edge detector. Some filter kernels correspond to image patches depicting recognisable shapes like circles or bars, called *templates*, while others are constructed using Fourier Analysis or other filter design techniques. The first type is used for texture comprising deterministic patterns like bricks or circles. Figure 2.2 shows such a filter being applied to the textured image made up of cross primitives used in Chapter 1. The strongest response is in the black areas. The second type is far broader, and the design of these filter kernels has been

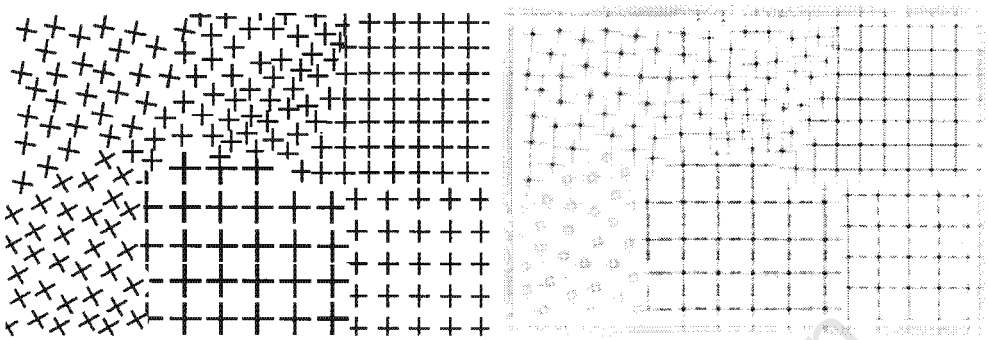


Figure 2.2: The textured image from Chapter 1 and the filter response image after calculating the cross-correlation between the cross primitive and the image.

widely researched. Filters used in this work are of the second type as the texture in most images does not contain repeated templates.

The size of a filter kernel, from here on called a filter, is important in determining what pattern will be detected. A filter that is too small will result in large scale texture information being lost. Also, if one considers the intensity at a pixel to be random variable generated by an underlying distribution which models the texture, we need to sample enough values to be able to know which distribution, and thus which texture, the pixel is from. Conversely, a large neighbourhood sacrifices accurate boundary localisation. This is because at the boundary, the window will be a mixture of classes, and the larger the window, the larger the area of uncertainty becomes.

Most images have many features that must be detected. Each filter will respond to one type of feature in the image, so to detect many features, several kernels must be applied to the image. Using Figure 2.2 as an example, in order to get strong responses from the areas where the cross is rotated, a different kernel will have to be used. Using responses from several filter kernels will more accurately describe the texture. This array of filter kernels is called a filter bank. Applying a filter bank to an image produces a stack of filter response images. The responses for each pixel are grouped together to form a feature vector, which is used as the descriptor for that pixel.

Designing a filter bank and deciding which features to emphasis and which to suppress is an important task. Filter kernels for texture segmentation should contain filters to detect many features so that textures can be easily differentiated. A trade off in designing such filters is the descriptive power to differentiate many textures versus invariance to changes in the texture we want the filter bank to ignore. These may include a rescaling or rotation of the texture. In that case we would choose a filter bank that is invariant to a change in orientation, though now less information about the texture is available for segmentation. An example of this is shown in Figure 2.3, where the filter response is

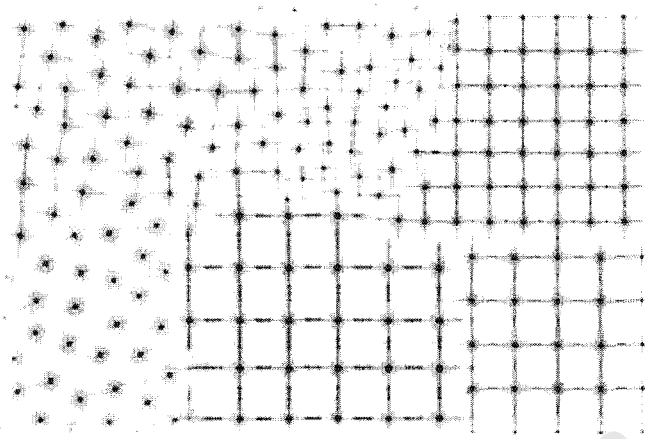


Figure 2.3: A filter response image that combines the output of several variations of the cross primitive kernel to obtain a response invariant to changes in rotation and scale of the primitive. This information is now more difficult to use for segmentation as the filter does not respond differently to the different areas.

invariant to changes in rotation and scaling of the primitive. A comprehensive review of filter banks for texture was done by Randen and Husøy [30]. After reviewing a number of the filters presented there and in other works, a few filters have been chosen for their simplicity and descriptive power. Four filter banks will now be discussed.

2.1.1 Ring/Wedge Filter Bank

The first filter bank is called the Fourier Ring/Wedge filter bank and it consists of several ring and wedge filters shown in Figure 2.4 [1]. Each filter is the spatial representation of a ring or a wedge in the spatial frequency domain. The rings are standard band pass filters, while the wedge filters extract features at particular orientations. The intersection of a particular ring and wedge will respond to texture of a certain scale and orientation. Separate filters can be constructed from just the rings or just the wedges. The former will respond to varying scale texture and the latter to differently oriented texture.

The band pass filters are similar to the pyramid or multi-resolution approaches to texture modelling, where the image is low pass filtered repeatedly. This produces a coarser version of the image, which is down-sampled to produce a smaller image. By repeating this step on the output of the previous step, a pyramid is produced. Repeated smoothing using a low pass filter is widely used [12].

If only the residuals (the original image minus the filtered image) are used, the representation is less redundant, with each layer of the pyramid representing texture energy at a certain scale. Scaled up in size to same size as the original, each layer would be similar to a the output of a band pass filter.

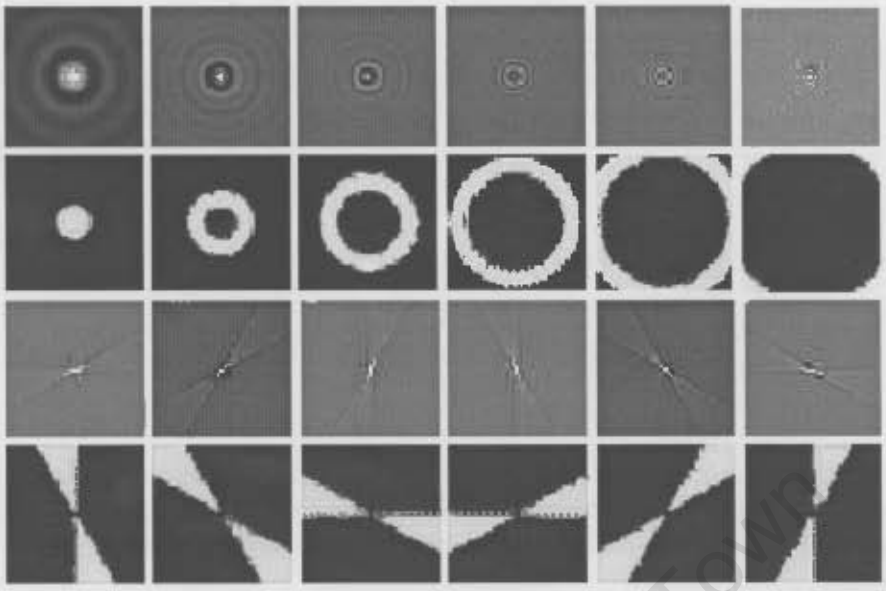


Figure 2.4: The Rings/Wedges Filter Bank. The first and third rows are the filter kernels, and the second and fourth rows are the Fourier Transforms of the kernels, showing the rings and wedges in frequency space.

Depending on the requirements, the filter can be rotation sensitive or invariant. Invariance can be achieved by ordering the output from the rotation sensors by the magnitude of response.

2.1.2 Gabor Filter Bank

The second is the Gabor filter bank, shown in Figure 2.5, which also partitions the frequency space, but many prefer it due to its links with the human visual system (HVS) [9]. In the spatial-frequency domain, the power spectrum is a Gaussian centered on a particular frequency and orientation modulated by a cosine. Often instead of spacing filters evenly throughout the frequency domain, the position of each filter will be determined by the content of the image. The placement can be determined either for each image, or learned for a class of images. A Gabor filter bank can be designed for a specific problem [9].Each filter in a Gabor filter bank is a Gaussian modified sinusoid, with a filter kernel in the spatial domain given by Equation 2.1 [38].

$$h(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left(\frac{-1}{2}\left[\frac{(x\cos\theta + y\sin\theta)^2}{{\sigma_x}^2} + \frac{(-x\sin\theta + y\cos\theta)^2}{{\sigma_y}^2}\right]\right)\cos 2\pi(x\cos\theta - y\sin\theta) \tag{2.1}$$

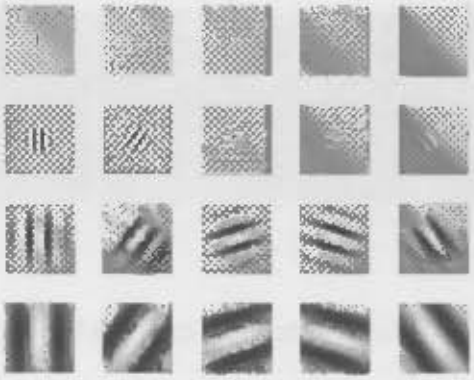


Figure 2.5: The Gabor Filter Bank.

where σ_x and σ_y control the support of the filter in the x and y directions, λ specifies the frequency of the cosine that modulates the Gaussian and θ determines the orientation of the filter. In Figure 2.5, σ_x and σ_y and λ increase downwards and θ varies from left to right.

2.1.3 Maximum Response Filters

Varma and Zisserman conclude that the maximum response (MR) filter bank gives the best accuracy for texture classification [34] and use this filter bank in work on textons [24, 36]. This filter bank is also used in , and is designed both to be rotation invariant and to have a small number of features. It consists of an even symmetric edge filter and an odd symmetric bar filter at several orientations and scales, as well as a Gaussian and a Laplacian of Gaussian, as shown in Figure 2.6. There are three scales and six orientations for each edge and bar filter, which when combined with the two Gaussian filters, gives a total of 36 rotation sensitive filters. The image is filtered with all of these filters, but for each scale, only one of the six responses from the differently oriented filters is selected. For each pixel, the orientation which gives the maximum response represents the edge or bar at that scale. This gives 6 features for each pixel, to which the response to the two Gaussian filters is appended. Thus there are 8 features from the maximum response filter bank.

2.1.4 Berkeley Quadrature Pair Filter Bank

Odd and even symmetric filters at multiple scales and orientations is also the basis for the filter bank which will be referred to as the Berkeley Quadrature [11] shown in Figure 2.7. Here, the even filter is a second derivative Gaussian and the odd symmetric filter is the Hilbert transform of the even filter.

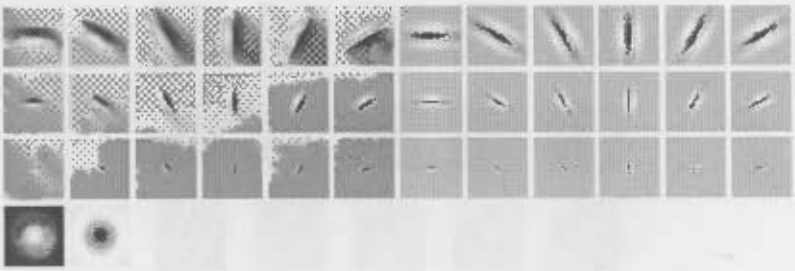


Figure 2.6: The Maximum Response Filter Bank.

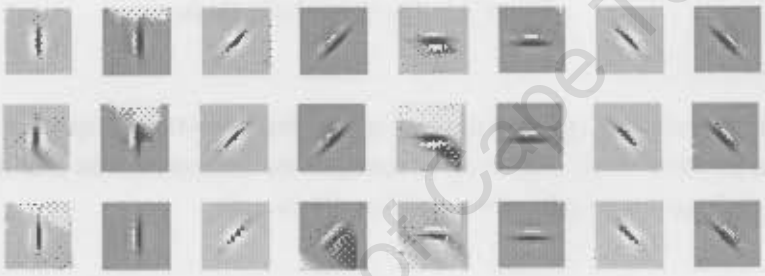


Figure 2.7: The Berkeley Quadrature Pair Filter Bank.

Many variations of these filter banks which have been explored by researchers. Filters with particular mathematical or signal processing properties are often described and implemented by researchers with a background in that particular area. Fundamentally, the aim of all filter banks used in describing texture is to respond only to the energy if the frequency domain that corresponds to discriminating texture information. This requires enough filters so that the response discriminates between different textures, but few enough number that the resulting description generalises well and members of the same class have similar filter response vectors.

2.2 Textons

Textons are local texture features in an image. Textons were originally introduced in the context of texture discrimination in the human visual system (HVS) [19], where Julesz introduced textons as local features that are used by the HVS in pre-attentive texture discrimination. He found that humans cannot pre attentively discriminate textures that have identical first and second order statistics (mean and variance of intensities) unless there are locally distinguishable features. He suggested that

humans must have neurons which detect local texture features, and called these patterns textons. This focused more on the perception of texture, and did not define how the concept might be extended so that it would be useful to researchers in computer vision. An algorithm for using textons in texture discrimination was developed later.

A texton is a special case of a signature and are sometimes called "generic descriptors" [32,34,39]. In many classification tasks, a histogram of points in a feature space is desirable. In a high dimensional space this histogram can be very sparse, causing unpredictable results in some comparison methods. Also, bins should be smaller in areas where the data is more dense, to give more information about the distribution. Rather than bin the space in a uniform manner as is typical in histograms, signatures can be used. These signatures represent typical values for data in the space. They will be dense where most of the data is located, and absent where data is sparse and uninformative. A histogram can be calculated using the signatures rather than the raw data, and used in consequent classification or segmentation.

No spatial information is explicitly used at this stage. This is important because with textons adjacent pixels in areas of homogenous texture are no more likely than any other pixels to have the same filter response. The use of a filter bank imposes some smoothness, but neighbouring pixels often have different responses and will be better represented by different textons.

There is a dual relationship between the texton approach and the histogram approach [37]. It is shown that the use of textons is in fact equivalent to a histogram where the bins are Voronoi polygons (or regions in general) surrounding the textons in feature space. Thus signatures amount to adaptive binning histograms. A normal histogram can also be implemented in the texton framework by having a texton at the centre of each bin. This is illustrated in Figure 2.8.

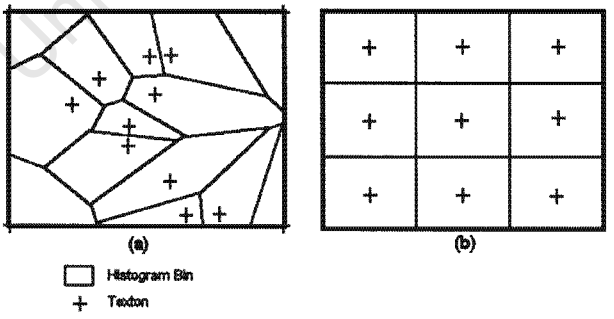


Figure 2.8: Comparison between histograms and textons in feature space [37]. In (a), the Voronoi polygon enclosing each texton represents the histogram bin corresponding to that texton. In (b), a regular grid of histogram bins is shown, with the dot in the middle of each representing that bin's equivalent texton.

2.2.1 Texton Dictionary

A texton is a signature representing texture information in a feature space. It characterises a dominant filter response to a region in an image. A clustering algorithm, typically k-means or one of its derivatives, is used to calculate the textons. All the textons used to describe the images are called a texton dictionary.

The result of applying a clustering algorithm to a set of filter responses is that each pixel will belong to a cluster, called a texton channel. The array of cluster centres is called a texton dictionary. The textons are the cluster centres for each channel, but each channel can be represented graphically. This is achieved by inverting the filter coefficient matrix and applying it to the filter response vector. Figure 2.9 shows the representation of a texton dictionary next to the image.

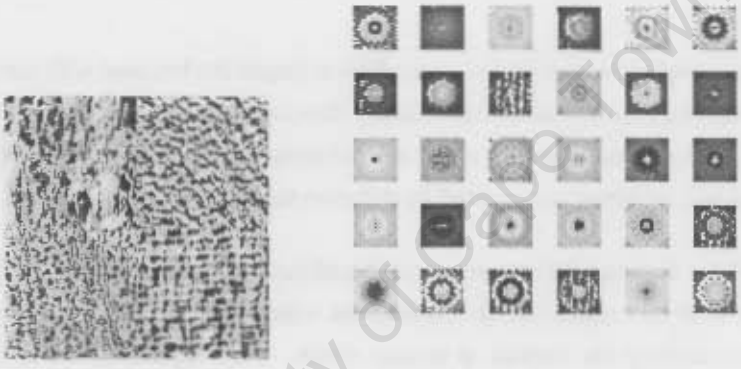


Figure 2.9: A texture mosaic and its texton dictionary.

Texton dictionaries may be specific or universal. A dictionary can be calculated using each new image to be segmented as the input, but this is not desirable as it is both inefficient and gives inconsistent texton assignment. It is best to build a texton dictionary based on a large sample of images that are similar to those to be segmented or classified. These textons are applicable across that class of images. The correct number of textons is context dependant, as in any clustering problem, and trial and error or some statistical analysis may be needed to find a suitable number. Testing by Martin et.al. [11] suggests two guidelines for choosing the number of textons: a universal texton dictionary requires approximately twice as many textons as an image-specific dictionary, and the number of textons increases linearly with the size of the neighbourhood used in calculating texton histograms.

2.2.2 Texton Descriptor

The texton descriptor is a histogram of the textons, either in the whole image, for classification, or in a pixel's neighbourhood, for segmentation. A histogram comparison method is used to measure

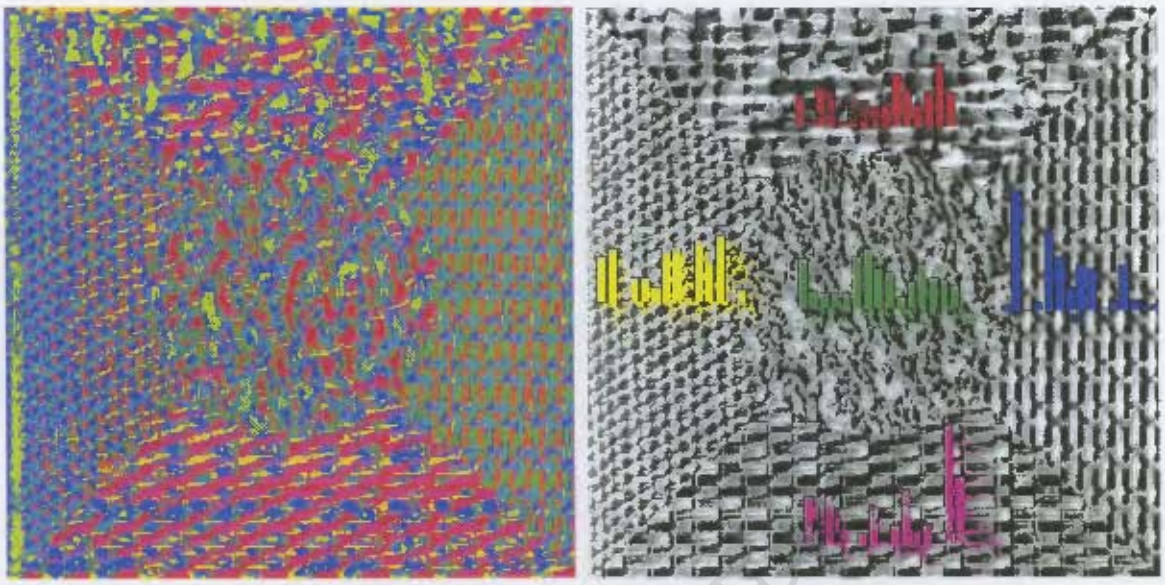


Figure 2.10: A texton labelled image, and the texton histograms of each class overlaid on the original image.

which class is closest. Figure 2.10 shows a map of the texton labels applied to a mosaic of images from the Brodatz album. Next to it is the original image with the texton histograms for five points in the image, roughly the point in the middle of each histogram, superimposed over the image to show the differences between typical histograms for each class.

The histogram may be calculated in many ways. In image classification algorithms [37], the histogram is calculated for the whole image. In segmentation it is more appropriate to calculate a histogram for each local neighbourhood. Characteristic histograms for each class by clustering the histograms for the pixels in that class [32].

The histogram of textons is similar in some ways to Grayscale Co-occurrence Matrices (GSCOM) [1]. GSCOMs measure the frequency with which pairs of graylevels occur adjacent to one another. The texton image is like a grayscale image with as many gray levels as textons. The texton histogram only measure what textons occur in the local neighbourhood, while GSCOM also describes the relative positions of the textons in the image. For segmentation, the GSCOM will be very sparse due to the relatively small number of pixel in a local neighbourhood. This reduces the effectiveness of the GSCOM.

2.2.3 Textons for Segmentation

The general framework for segmentation using textons, can be described as follows. The input to the algorithm is a feature vector for each pixel. This usually obtained as the output of a filter bank, though any series of representations of the image can be used to form feature vectors. The first training step is to cluster the training data, in the form of feature vectors. The cluster centres form the texton dictionary , which is the first part of the texture model. Each pixel in the training images is labelled according to which cluster centre it is closest to. For each class, a histogram or histograms of textons is calculated. The distribution of textons can be calculated globally for all pixels in the training set (referred to as One Histogram per Class) or several histograms or "spatial frequency clusters" can be used [32] (referred to as Multiple Histograms per Class). This involves computing a texton histogram for each pixel and clustering, instead of global computation. These then form a training set which can be clustered to find the dominant texton distributions for that class. The histograms found this way are the second part of the texture model. The textons and the histograms for each class together are the complete description of the texture in the training data.

When presented with a new image to segment, the feature vectors are first extracted, and then each pixel is labelled according to the closest texton in the texton dictionary. A histogram is calculated for each point and compared to the histograms for each class. The class with the best match is the label assigned to that pixel.

2.2.4 3D Textons

Much research has been done recently in the area 3D textons [21, 34]. 3D texture techniques aim to classify the texture of a surface independent of viewpoint and lighting conditions. This is done by including images of the texture from many viewpoints and under many lighting conditions. There is a model (textons and histograms) for each set of conditions. The assumption in this work is that the viewpoint and lighting are fixed and that 3D geometry is not considered.

Chapter 3

Local Texture Features

A fundamental property of texture is scale. A good example is a tree. At one level, there are leaves and bark, then zooming out successively, a branch, a tree, a forest and so on. At each stage the texture of the image is different. The field of sunflowers in Figure 3.1 demonstrates this. We can see that 'field of flowers' texture is the same at the top and the bottom of the image, but at a different scale. Marr [25] describes how such changes in scale are used in the HVS to detect surfaces in an image, either real or perceived, and to aid in depth perception.

Figure 3.1: This field of sunflowers demonstrates how change in scale is used by humans to detect surface orientation. The scale of the sunflower texture is smaller at the top of the image compared to the bottom, suggesting a surface whose normal is pointed up and towards the viewer.

To compare textures accurately we need to have some measure of the scale of texture in the image, and calculate texture descriptors at that scale. As texture is only measurable over a region of an image rather than at each pixel, there is a scale parameter in every texture algorithm. In the texton algorithm, scale is represented in two ways: by the size of the filters in the filter bank and in the responses to filters that respond to different spatial frequencies. Other non-filter approaches also need a sense of scale in order for the measures to best describe the texture.

To accurately match texture descriptors in all cases, including where the texture is at different scales, the scale of the texture currently being analysed needs to be measured. The image is represented at many scales in scale space. Scale space is a 3D representation of the image, with the third dimension being the scale. Each band is a derivative based measure which varies with scale calculated for each point in the image. The scale is varied by integrating (smoothing in practice) over groups of pixel to calculate pixel based measures. The scale as which extrema in the derivative based measure across scale occur in scale space representation is the scale of the image at that point. Scale detection is discussed in Section 3.1. The local descriptor is then calculated at that scale, the scale determining area of integration, in effect the radius of the smoothing kernel. Local descriptors are investigated in Section 3.2.

3.1 Scale Space

Scale Space is an image representation which enables the estimation of a scale measure for each point in an image. The aim is to always calculate texture descriptors at the correct scale which makes our descriptor invariant to changes in scale.

3.1.1 SIFT Scale Space

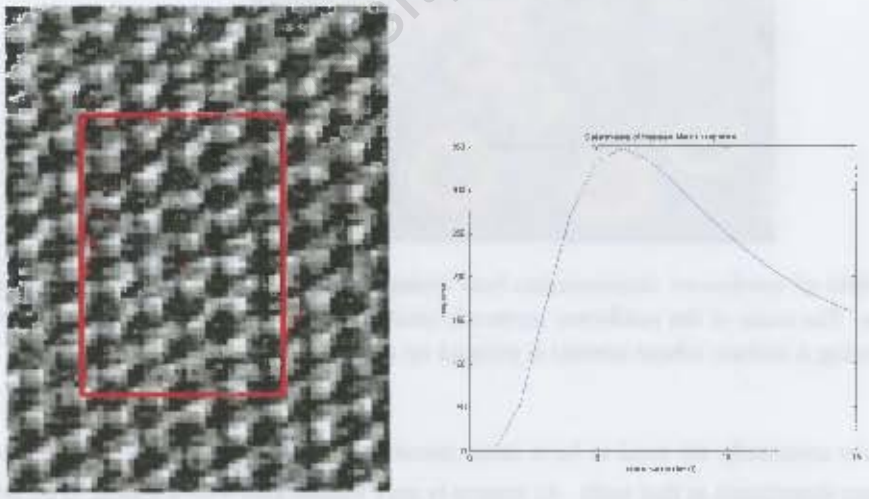


Figure 3.2: A patch of texture. The dot shows a pixel whose Hessian Matrix has been calculated for a range of scales. The graph shows the determinant of the Hessian Matrix across scales.

Lindeberg [22] uses a scale space image representation to estimate the scale of the image. Every location in a scale space representation contains information not only related to the corresponding image location but also to a surrounding region, the size of which is specified by the scale parameter. Effectively, the scale representation is the output of a bank of low pass filters. Equation 3.1 is the formal definition of a scale space representation:

$$S(x, y, t) = g(x, y, t) * im(x, y) \tag{3.1}$$

where t is the scale parameter, $g(x, y, t)$ is the bank of Gaussian filters and $im(x, y)$ is the input image. The use of Gaussian filters is essential in the formulation of scale space. These correspond to specific cells in the Mammalian vision system which are equivalent to Gaussian derivative filters [22].

Once we have formed a scale space representation, we can use it to select a scale at each pixel in the image. A pixel scale, $S(x, y, t)$, represents all the values in a t-sized region around the (x, y) pixel location. Lindberg [22], calculates derivative-based statistics at each location. The scale at a point in the image is the value of the scale parameter t at the maximum of these statistics.

A typical feature is the trace or determinant of the Hessian Matrix. The Hessian Matrix is a 2×2 matrix in this case. The top left and bottom right elements are the squares of the first difference in the x and y directions, while the other two elements are the product of the differences. The exact number of the response at each pixel is not important: we only want the t corresponding to the maximum response. In Figure 3.2, we can see that the maximum occurs at approximately $t = 5$.

The problem is that response curves with this form only occur at some points in the image: responses for other pixels show a maximum either at the smallest possible scale, or increase with scale up to the largest scale tested. This suggests that reliable scale measurement is only possible at certain locations in the texture pattern.

3.1.2 Blobworld Scale Selection

The Blobworld representation employs another scale selection method [7]. This method tries to determine the period of the texture by detecting when the pattern repeats itself. The method starts with a scale space representation, then calculates a feature called Polarity at each scale. This feature measures what percentage of surrounding pixels have gradients in agreement with the central pixel. The difference across scales is used to determine the scale parameter. As the window size increases, more pixels will have conflicting gradients, so the polarity decreases. When the window size exceeds the period of the pattern, no new gradients are included, and the polarity measure stabilises. The scale

where the difference across scale is less than some threshold is the chosen scale for that pixel. Polarity as a feature is discussed further in the Section 3.2.

Scale Space methods give a scale for each pixel in the image. The next step is to calculate a descriptor for the texture at each point at the scale calculated. As mentioned, a problem encountered in testing these methods is that a consistent scale measure is hard to achieve. For an area of consistent texture, the scale measure should be constant. Variations in the pattern prevent this. As a result, only certain pixels will give a meaningful result. Thus many algorithms select points where the scale measure is stable and use only these points [22, 23]. This is useful in applications such as stereo or optical flow where point correspondences are sought, but in segmentation, every point must be considered.

3.2 Local Descriptors

In the texton approach to texture analysis, the starting point for any algorithm is a bank of filters applied to the input image. A local descriptor algorithm starts with an operation on the 4-connected or 8-conected neighbourhood around a pixel and then integrates the results across a window of a certain size. The size of the window is calculated by a scale selection algorithm, and thus the two are usually linked. Mikolajczyk and Schmid [26] analyse the performance of a number of local descriptors and a few of these will be discussed in more detail.

3.2.1 Blobworld Descriptor

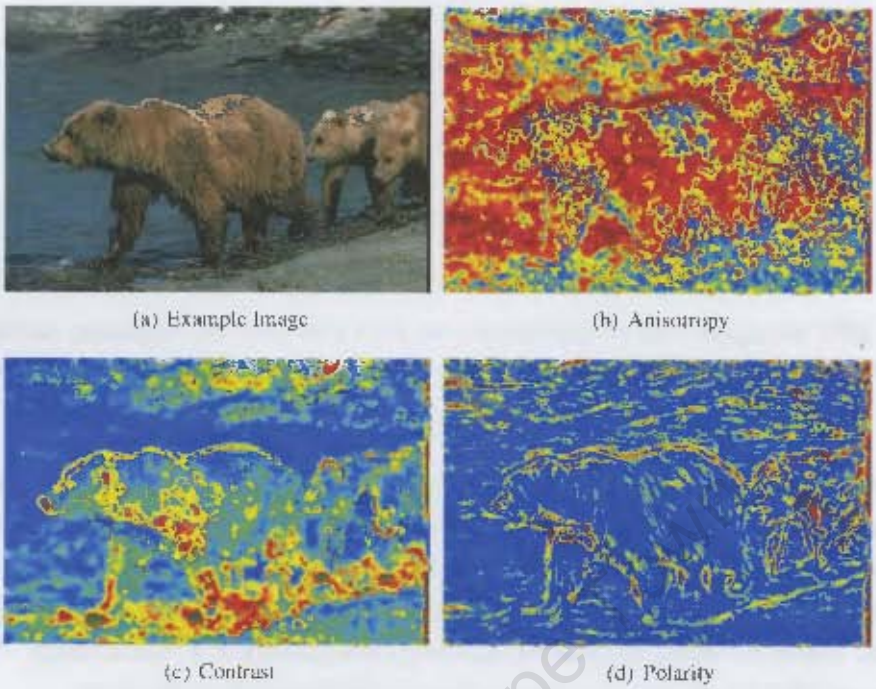
The Blobworld descriptor is part of the algorithm from for the Blobworld image retrieval system [7]. Section 3.1.2 introduced polarity, used to determine scale, which is elaborated upon in this section. The Blobworld features are based on the eigenvalues of the second moment matrix (SMM) for each pixel. The SMM is similar to the Hessian Matrix in this case. The first feature is polarity, which is used to measure the scale of the texture in an image. The polarity is given by:

$$p_\sigma = \frac{[E_+ - E_-]}{[E_+ + E_-]} \tag{3.2}$$

$$E_+ = \sum_{x,y} G_\sigma(x,y)[\nabla I \cdot \hat{n}]_+ \tag{3.3}$$

$$E_- = \sum_{x,y} G_\sigma(x,y)[\nabla I \cdot \hat{n}]_- \tag{3.4}$$

where $[\cdot]_+$ is the positive rectified part of the argument, $\hat{n}$ is a vector perpendicular to argument of the dominant eigenvector and ∇I is the image gradient at that point. The quantity $G_\sigma(x,y)$ is the Gaussian filter which integrates the gradients over scale σ. Varying σ (which is equivalent to varying



(a) Example Image (b) Anisotropy

(c) Contrast (d) Polarity

Figure 3.3: Examples of Blobworld features calculated from example image.

the scale parameter t) gives a response curve for each pixel. The first value of σ for which the difference between successive scales is less than some constant is the value used in calculating all the features. The polarity measure itself is a measure of the homogeneity of gradient directions in a window around a pixel, and will respond strongly to an area of directional textures such as stripes and edges, as shown in Figure 3.3.

The other features, contrast and anisotropy, are calculated on the eigenvectors of the SMM, and the resulting image is passed through a Gaussian filter with standard deviation σ. If the eigenvalues of the SMM are λ_1 and λ_2 then the contrast c_σ and anisotropy a_σ are calculated:

$$c_\sigma = 2\sqrt{\lambda_1 + \lambda_2} \tag{3.5}$$

$$a_\sigma = 1 - \frac{\lambda_2}{\lambda_1}. \tag{3.6}$$

Figure 3.3 shows each of these features, which will collectively be referred to as the Blobworld descriptor, calculated for the example image shown. The two eigenvalues give a measure of edge strength in the major and minor edge directions, so the higher the values, the greater the contrast. Anisotropy

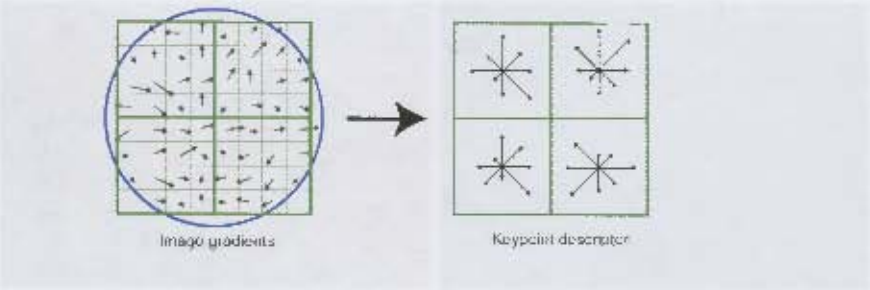


Figure 3.4: The SIFT descriptor. The window around the pixel is divided into segments, and then a histogram of the gradient directions for each is calculated, as represented by the arrows of varying lengths.

measures the directionality at a point. If λ_2 is small compared to λ_1, then there is a strong gradient direction and there is strong directionality. Comparable values indicate an edge or no direction information, so the ratio of the value leads to the anisotropy feature. The contrast measure increases with the sum of the eigenvalues, and so corresponds to the amount of change in intensity in the neighbourhood of the pixel. Figure 3.3 shows the contrast, anisotropy and polarity of a typical image. These three descriptors, calculated at a scale σ, are used to characterise the texture in the image.

3.2.2 SIFT Descriptor

Lowe uses a descriptor called SIFT (Scale Invariant Feature Transform) to characterise texture in an image [23]. The scale parameter is calculated from the scale space representation and determines the size of the window of operation around each pixel. The descriptor itself is a histogram of the gradients around the pixel. The window is divided into a number of segments, and a histogram of the gradient direction in each segment is calculated. Thus the size of the descriptor is $s \times d$, where s is the number of segments and d is the number of directional bins per histogram.

Many variations of SIFT have been proposed. The SIFT descriptor tends to have high dimensionality, and so is a candidate for Principal Component Analysis (PCA) and related techniques [20]). Instead of first calculating a histogram, the gradients for each pixel n in the window are concatenated to form a very high dimension feature vector. These feature vectors from a number of sites are put into a data matrix, and PCA is performed on the covariance matrix of the data. The top n eigenvectors are then used to project the data into an n-dimensional subspace.

Gradient Location and Orientation Histogram (GLOH) is a SIFT variant that uses a log-polar system to divide the window into segments, rather than a cartesian grid [26]. A histogram is calculated for each segment, and the histograms concatenated to form the feature vector. PCA is done on the resulting data to reduce dimensionality.

A technique for describing shapes can also be applied to texture [2]. *Shape Context* includes only edges in the histogram, rather than gradients of all points in the neighbourhood window. Edges are extracted using the Canny edge detector. Edges are binned according to orientation and location in the window, on a log-polar grid. Each edge's contribution to the histogram is weighted according to the magnitude of its gradient. This was used for matching shapes of silhouettes, but can also be used as a texture feature. Most SIFT type features use image gradient as the base feature for the descriptors. *Spin Images* use image intensities in the same way that SIFT uses gradient orientation. The neighbourhood window is segmented on a grid, and a histogram of intensities for each segment is calculated.

Mikolajczyk and Schmid tested local descriptors performance for point correspondence [26]. An image was transformed in some way, for example rotated and rescaled, and points in the original were matched to points in the transformed image. In this testing, the SIFT and GLOH features were found to perform best.

3.2.3 Local Binary Patterns

An alternative local descriptor is the local binary patterns (LBPs) [33]. LBPs describe texture in terms of the intensity of surrounding pixels relative to the centre pixel. The 8-connected pixels around each pixel in the image are labelled 1 or 0 depending if they are greater or less than the pixel they are connected to. As only relative grayscale values are part of the descriptor, grayscale invariance is achieved. The LBP concept is similar to texton in the way it models texture as a distribution of standard patterns. The patterns LBPs use are on a much smaller scale, using only the directly connected neighbourhood of each pixel, and only binary information for each pixel. As a result there is finite number of patterns. The eight binary values form a binary number, so that each pixel in the image is labelled with a number from 0 to 255, which is the label of that binary pattern.

The texture of an image or a region in the image is then specified by the distribution of these patterns in an image, or a window in the image in the case of segmentation. The distribution is represented by a histogram. For segmentation, 255 bins is often too many and the resulting histogram will be quite sparse.

The problem of sparse histograms is solved by using only those pattern that are not a rotation of another in the set. This reduces the number of patterns to 36, and makes the descriptor rotation invariant. The binary patterns in the rotation invariant set are illustrated in 3.5. The invariance of LBPs to changes in illuminance and image orientation is a distinct advantage.

Figure 3.5: Set of rotation invariant local binary patterns. White represents neighbouring pixels that are less than the centre in pixel, in grey, and the black pixels are the neighbours of greater intensity.

3.2.4 Local Image Patches

Filter banks, textons and local descriptors are all different representations of an image, taking the raw pixel values and mapping them to higher dimensional feature space. Pixels from different textures will have feature vectors that are better separated in a suitable feature space, which will give better results when the image is segmented. The assumption is that the image patches are not sufficiently separable. Having investigated filter bank methods thoroughly and developed filter bank based texton algorithms for various classification tasks. Varma and Zisserman, having investigated filter bank methods thoroughly and developed filter bank based texton algorithms for various classification tasks, question the need for large scale filter banks in texture classification [35]. They suggest that instead of processing a large neighbourhood, which will typically have at least 1000 pixels in it, only the pixel intensities in a much smaller neighbourhood are actually needed. They compare the effectiveness of a traditional filter bank and texton based classifier to a Markov Random Field based classifier.

Raw intensity value are taken from a small neighbourhood around the pixel, from 3×3 to 7×7 pixels, and used as a feature vector. This is used as the input to the texton algorithm. They obtain good results in classification tests, and so it is worthwhile examine whether similar results can be obtained in segmentation. The ability of a distribution of LBPs, which use even less information than an image patch, to be used in segmenting images suggests that there is sufficient descriptive information to differentiate between different classes of texture. The descriptor can be made rotation

invariant by using Principal Component Analysis. This also allows us to use larger neighbourhoods that would otherwise be infeasible due to computational constraints.

Chapter 4

Segmentation

Filter banks, textons and local descriptors take an image and transform the raw data into a feature space which separates pixels with different texture. This representation is used to segment the image into regions of different texture. Segmentation is a special case of classification, in that each pixel is assigned to a class. In classification, each data point is separate. It is not the case in segmentation: even if one is using the raw image data, there is a degree of correlation between the descriptors, caused by the spatial consistency of most images. The texture in an image is not a point property and is only detectable for a group of pixels, which means that nearby data points will have descriptors calculated from overlapping areas of the image. A penalty can be imposed on solutions which are not spatially consistent as this is most likely incorrect. This same property is a problem in areas of the image where there is a sudden change in appearance, such as the border of two objects in an image. The descriptors on the border will be a mixture of the two, describing neither texture accurately.

There are two tasks that are being performed by a segmentation algorithm, namely labelling areas of the image as belonging to a particular object or class, and finding the location of the boundary. They are complimentary activities: labelling the pixels will implicitly place a boundary along some pixels, while choosing the boundary location will implicitly divide the image up into a number of regions. Segmentation algorithms can be divided into those that label pixels and those label boundaries. Since each method is implicitly doing the other task as well, the main difference is how the algorithm is constrained. Algorithms that label pixels will specify how many classes there should be or how coherent members of a class should be. Algorithms that label boundaries will specify boundary length and will quantify how strong the edge must be to be considered a boundary. The challenge is to find a way to combine the border and region information. Freixenet et. al. surveyed a wide range of segmentation methods [17].

Section 4.1 looks at pixel labelling methods and Section 4.2 discusses border detection methods. Graph cut segmentation is investigated in Section 4.3. Section 4.4 looks at how to measure the accuracy of a segmentation.

4.1 Pixel Labelling

The simplest way to label the pixels is to have an example descriptor of each class, and label the pixel according to which class has the closest example, which is the nearest neighbour segmentation. If there is no training data available from which to calculate the example descriptor, an supervised segmentation can be calculated using a clustering algorithm like k-means. This approach is simply applying a classification to segmentation. This usually provides a good first effort at labelling pixels, which can be used by other algorithms to refine the labelling or boundaries.

A more advanced clustering algorithm is the Gaussian Mixture Model (GMM) [3]. A GMM can be trained either on all the data, for unsupervised segmentation, or a separate GMM on each class of data, for supervised segmentation. In the unsupervised case, each cluster is now represented by a mean vector and covariance matrix, rather than just a mean vector, as is the case for a k-means cluster. In the supervised case, a GMM can represent more complex data distributions by training a more detailed probability density function for each class.

4.1.1 Histogram Calculation

Central to the algorithm, and to most segmentation algorithms, is the distance calculation. The descriptors produced by the texture methods of Chapter 2 and 3 produce a histogram as the feature vector for each pixel. These do not occupy a typical Euclidian space, so the distance between histograms must be calculated differently. Rubner et. al. [39] and Puzicha et. al. [18] both give an overview of and comparison between different dissimilarity measures.

The first prominently used histogram comparison measure is the χ^2 distance. For histograms, this measures the difference between each corresponding pair of bins as a proportion of the sum of the two bins, and is given by:

$$D(h_1, h_2) = \frac{1}{2} \sum_i \frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)} \tag{4.1}$$

The χ^2 distance considers each corresponding pair of histogram bins, but not interactions between bins. A more complete solution is the Earth Movers Distance (EMD), which is a solution to the transportation problem, and has a linear programming interpretation. The EMD view each histogram as a pile of sand (or bricks in the case of a histogram), and the distance between the histogram is the amount of work needed to move the bricks so that distribution 1 is the same as distribution 2. The amount of work is the number of bricks that have to be moved, weighted by the distance they have

to be moved. The distance they have to be moved is the ground distance between histogram bins, in the form of a cross-bin similarity matrix. The EMD algorithm finds the minimum cost rearrangement of the bins and uses this as the distance between histograms. The two main advantages of the EMD is that there can be different numbers of bins and that includes cross-bin information. The problem for segmentation is the complexity of the problem, which requires an optimisation for each pair of histograms to calculate the distance. If calculating distances for a nearest neighbour classifier, on an image that is $n \times n$ pixels in size and there are m classes, $n^2 \times m$ comparisons need to be done. With typical values of $n = 500$ and $m = 5$, there are over a million comparisons. This results in very long computation times. An alternative is the Quadratic Form distance, given by:

$$D(h_1, h_2) = \sqrt{(h_1 - h_2)^T A (h_1 - h_2)} \tag{4.2}$$

This retains the use of a ground distance matrix, but is a closed-form solution, and requires no optimisation, making it more efficient and suitable for use in segmentation.

4.1.2 Descriptors on Class Boundaries

Segmentation by labelling pixels is essentially classification of pixels. This breaks down at the border between two classes, where the descriptor for the pixels are dependant on pixels from 2 or more classes. Thus the descriptor will be a combination of the classes. The result is a simple linear combination for histograms, but for other features, like filters responses, the result can be dissimilar to both classes and a linear combination. The texton algorithm uses a histogram as the descriptor for each point, so the histogram for border pixels can be modelled as a linear combination of two classes. To use this in segmentation, combinations of all pairs of class histograms can be calculated, and used as new example histograms. The point would be labelled according to which class makes up the largest part of the combination. For example, if a point's closest match is a combination of 70% class A and 30% class B, then the point is labelled as belonging to class A. Figure 4.1 shows how the ratio between the number of pixels from each class varies as neighbourhood moves.

After labelling pixels, a postprocessing step is often added to finesse the segmentation so that it has more desirable properties. This often includes steps to smooth the boundaries. This may take the form of a morphological operation, which will eliminate small "tendrils" sticking out of a region. The Blobworld system's post processing step consists of two operations [7]. The first separates region in the image with the same label into connected components. This is especially important in unsupervised segmentation, where the algorithm may group two spatially separate but visually similar objects together and give them the same label. Each label is separated into contiguous regions, each with its own label. The second step is to use the original image colour data for each pixel, rather than the

to be moved. The distance they have to be moved is the ground distance between histogram bins, in the form of a cross-bin similarity matrix. The EMD algorithm finds the minimum cost rearrangement of the bins and uses this as the distance between histograms. The two main advantages of the EMD is that there can be different numbers of bins and that includes cross-bin information. The problem for segmentation is the complexity of the problem, which requires an optimisation for each pair of histograms to calculate the distance. If calculating distances for a nearest neighbour classifier, on an image that is $n \times n$ pixels in size and there are m classes, $n^2 \times m$ comparisons need to be done. With typical values of $n = 500$ and $m = 5$, there are over a million comparisons. This results in very long computation times. An alternative is the Quadratic Form distance, given by:

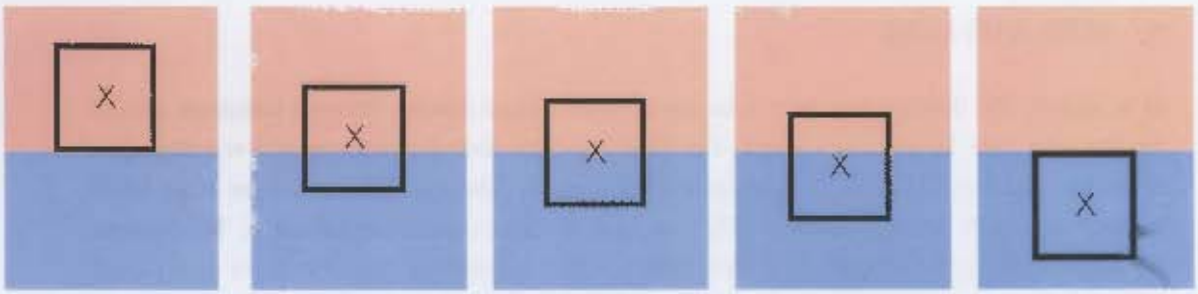
$$D(h_1, h_2) = \sqrt{(h_1 - h_2)^T A (h_1 - h_2)} \tag{4.2}$$

This retains the use of a ground distance matrix, but is a closed-form solution, and requires no optimisation, making it more efficient and suitable for use in segmentation.

4.1.2 Descriptors on Class Boundaries

Segmentation by labelling pixels is essentially classification of pixels. This breaks down at the border between two classes, where the descriptor for the pixels are dependant on pixels from 2 or more classes. Thus the descriptor will be a combination of the classes. The result is a simple linear combination for histograms, but for other features, like filters responses, the result can be dissimilar to both classes and a linear combination. The texton algorithm uses a histogram as the descriptor for each point, so the histogram for border pixels can be modelled as a linear combination of two classes. To use this in segmentation, combinations of all pairs of class histograms can be calculated, and used as new example histograms. The point would be labelled according to which class makes up the largest part of the combination. For example, if a point's closest match is a combination of 70% class A and 30% class B, then the point is labelled as belonging to class A. Figure 4.1 shows how the ratio between the number of pixels from each class varies as neighbourhood moves.

After labelling pixels, a postprocessing step is often added to finesse the segmentation so that it has more desirable properties. This often includes steps to smooth the boundaries. This may take the form of a morphological operation, which will eliminate small "tendrils" sticking out of a region. The Blobworld system's post processing step consists of two operations [7]. The first separates region in the image with the same label into connected components. This is especially important in unsupervised segmentation, where the algorithm may group two spatially separate but visually similar objects together and give them the same label. Each label is separated into contiguous regions, each with its own label. The second step is to use the original image colour data for each pixel, rather than the



(a) 100:0 (b) 70:30 (c) 50:50 (d) 30:70 (e) 0:100

Figure 4.1: The black square in each image shows the local neighbourhood of the pixel marked by the X near the boundary between class A and B, shown in red and blue respectively. As the neighbourhood moves across the border between the classes, the ratio of the number of pixels belonging to each class varies, shown below each image as class A:class B.

calculated features which have a neighbourhood, to refine the boundary position. Alternatively, the region labelling may be used as a seed to a boundary detection algorithm of the sort to be discussed in Section 4.2.

All these algorithms label each pixel individually, with no consideration given to the labels of other pixel in the image. Smoothness is imposed on the labelling by the correlation between descriptors of neighbouring pixels, which is caused by the similarity of the neighbourhoods of adjacent pixels. Nevertheless, if the neighbourhoods differ sufficiently, or perhaps are outliers of a particular class, the resultant segmentation can become jagged and fragmented. A solution to this problem is to explicitly impose a cost on adjacent pixels having different labels. A method of doing this is discussed in Section 4.3.

4.2 Boundary Detection

The alternative to labelling each pixels is to find where the boundary between classes should be. This can be used when there is no information about how many classes there should be. Often, the aim is not to segment into classes, but into contiguous regions so that further descriptive statistics can be calculated. The basic operation in this paradigm is to find a gradient at each pixel. Any feature or descriptor can be used to calculate the gradient. Martin et. al calculate histograms of the texton labels of each pixel in each half of a circular neighbourhood and take the difference as a the texture gradient at a pixel [11]. Figure 4.2 demonstrates the difference between texton histograms across edges, where there is clear dissimilarity between the histograms. The gradient can also be caulculated from the Gabor filter responses [29]. The key is that the gradient encodes the probability that a pixel is on a boundary, and not just variation of the image within an object or class.

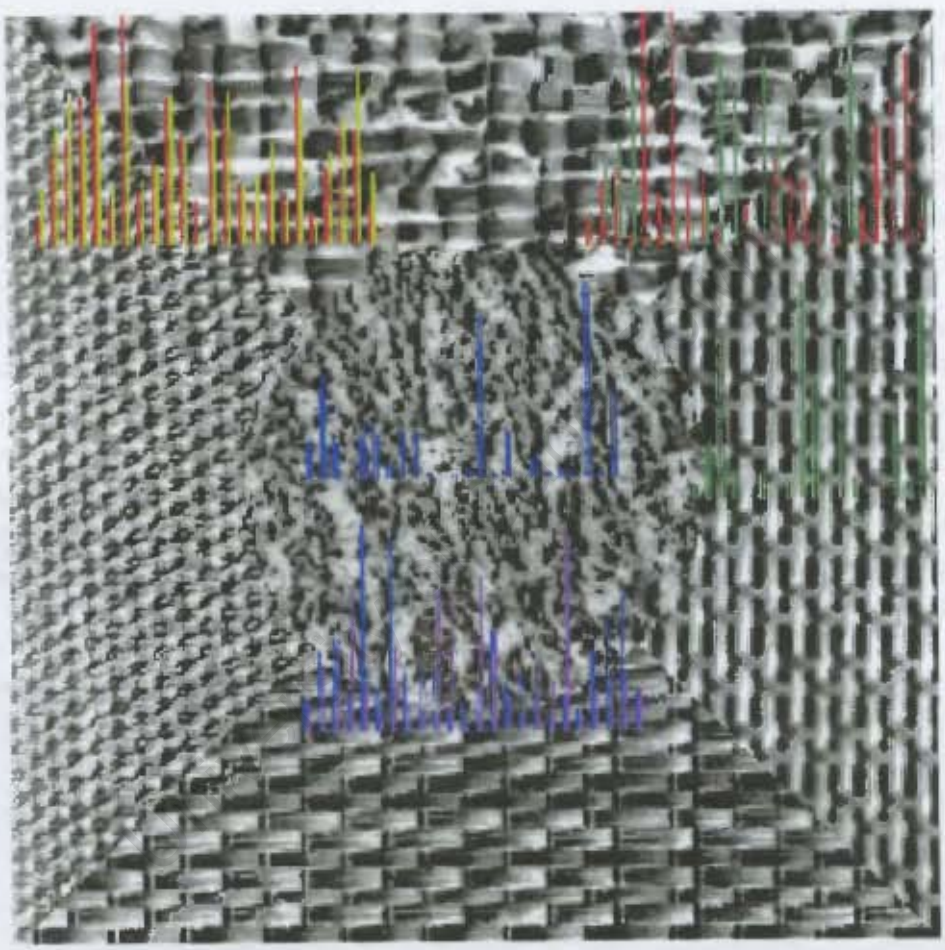


Figure 4.2: A Brodatz mosaic with texton histograms for each half the neighbourhood, showing the differences in histogram across the edges.

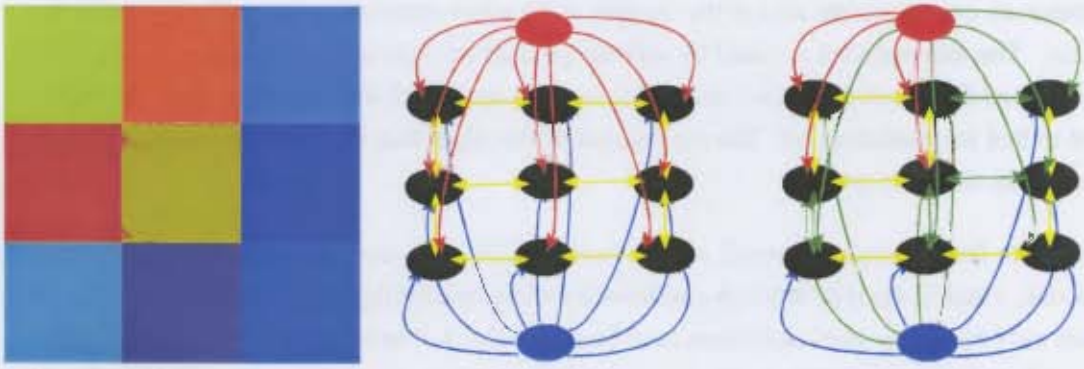
Once the gradient is calculated, an obvious method for finding edges is to threshold the gradient map, choosing edges with a gradient magnitude greater than some threshold. There are a number of problems with this. The first is the localisation of the boundary. The actual boundary is a line (or curve) along the edge of two classes or regions wholly or partly bounded by this border. Most edge detectors, particularly texture filters, use a neighbourhood around the pixel to calculate the gradient features at a point. As a result, pixels on either side of the true edge will respond to the edge detector, and have a value in the gradient map near that of the true edge. Instead of being a line of pixels, the edge will be found as a strip of pixels. Related to this problem is the problem of double detection. The pixels at the border often have different feature vectors to the pixels away from the border. The pixels at the border often form a group of their own. Thus a peak in the boundary often occurs between object 1 and the border area, and again between the border area and the border area and object 2.

There are many approaches to solving this problem. The first to thin out the edge in some way. This can be done morphologically, through dilation and erosion after thresholding. To solve the double detection problem, a lower threshold can be used, and the strip of pixels eroded. This does not work well as the lower threshold produces false edges and the morphological operation may remove some edge pixels. A better solution is to model the problem directly by attempting to fit a function to the gradient map, and emphasise the peak of the function and suppressing the surrounding pixels [11]. A parabola is fitted in the direction of the gradient to remove multiple edge detections, then non-maxima suppression is used to mask out the pixels adjacent to the peak in gradient intensity.

The second method is to assume there is one unknown boundary and that the position of that boundary is unknown. This can be described in several ways, including curves, polygons or cuts. Many algorithms take the curve / polygon approach, and calculate the segmentation by optimising over the parameters of the curve. This approach is often known as the active contour or active region approach [31, 42]. Alternatively, the boundary can be found by representing the image as graph, and cutting it such that two groups of nodes are formed with no interconnection between the groups. This approach is discussed further in the Section 4.3.

4.3 Graph Cuts

The problem of segmentation is closely related to the problem of classification. We are classifying each pixel, but the difference is that the data points are ordered and have a set of neighbour relations to other data points. Because of this, we are not classifying each pixel in isolation, but as part of the image, which has other properties besides the relation of each pixel to each class. Usually this means that a smooth segmentation is desirable, rather than a fragmented segmentation or regions with jagged, ill-defined edges. To enforce this smoothness property, a penalty is incurred by label configurations which have these properties. This is done by constructing a graph, with each pixel in the image



(a) Image (b) Graph (c) Graph with cut

Figure 4.3: An example 2-class labelling problem, based on a similar graph by Boykov and Kolmogorov [4]. The 3×3 pixel image is red and blue with added gaussian noise. (b) shows the graph representing this problem, and (c) shows the cut representing the solution.

represented by a node in the graph and relations between pixels by edges in the graph. Additional nodes represent each label in the desired segmentation. The minimum cut which divides the graph such that each node is connected to only one class node gives us the minimum cost segmentation.Thus it minimises the cost of segmentation globally, not just for each pixel individually.

4.3.1 The 2-class Graph Cut Problem

Graph cuts for the image labelling problem was first introduced by Greig et. al. [16], where they are used to solve the two class segmentation problem. An example problem is shown in Figure 4.3. The 3×3 image is to be segmented into red and blue pixels. The original image has been corrupted by Gaussian noise. A graph representation is constructed as shown. The red and blue nodes are the source and sink respectively. They are called terminal nodes and represent the 2 classes in the segmentation problem. The black nodes are called data nodes, and each one represents a pixel in the image.

The edge weights in a graph represent the terms in the energy function that is to be minimised. The red and blue lines from the terminal nodes to the black nodes are called terminal links. The weights of these edges are proportional to the cost associated with assigning a pixel to particular class. For reasons explained later, the cost of assigning a pixel to the blue class is the weight of the edge connecting that pixel to the red terminal node, and vica versa. The yellow links between data nodes are called neighbour links. The weights of these edges are proportional to the cost of assigning neighbouring pixels to different classes. The correct segmentation is found by separating the nodes into two groups,

by removing edges such that each data node is connected to only the source or the sink, and there is no path from the source to the sink. This is called a cut. The cut which corresponds to the solution is the minimum cut, where the sum of the weights of the edges removed is the smallest possible for a valid cut. The minimum cut is found by solving the dual problem of maximum flow through the graph. Ford and Fulkerson proved the two problems to be equivalent, and maximum flow algorithms are used to find the minimum cut. The cut consists of the edges that are saturated when maximum flow is reached, shown in green.

The maximum flow through a network can be found quickly and exactly when there is one source and one sink, which is the type of graph constructed for binary labelling. Several implementations of minimum cut / maximum flow algorithms have been developed. The standard algorithm for finding maximum flow finds paths from source to sink, and pushes flow along the path until one edge in the path is saturated. This is done on all paths until there are no unsaturated paths. The saturated edges correspond to the edges that are cut in the minimum cut. Currently used algorithms speed up this process by reducing the number of visits required to each node by pushing more flow than the smallest edge on a path can take or by maintaining trees of the possible paths [4]. This implementation has been used in tests done for this thesis.

4.3.2 Graph Cuts for Segmentation

The Ford / Fulkerson theorem which proves the minimum cut and maximum flow problems to be equivalent deals only with the type of graphs that solve a binary labelling. A multi-class labelling problem, which describes a segmentation problem, requires a source or sink node for every label. A cut which separates the nodes of such a graph into disconnected groups is not solvable in the same way. In fact, an n-way cut, for $k > 3$, is an NP-hard problem [8], and the binary graph cut solution has to be extended to n-classes in other ways.

One method is to embed the binary cut into the multi-class problem [40]. Their method is called α-expansion, which recasts the n-cut problem into a series of relabelling problems, each of which is solved with a binary graph cut. The algorithm iterates through the labels in any order, and for each label α, decides for each pixel whether it should keep its current label or be relabelled α. At each step, only the size of label α can increase. The terminal nodes in the graph are α and the current label. The solution given by the α-expansion algorithm is a local minimum that is guaranteed to be less than a known factor multiplied by the global minimum. This factor is based on the weights of the n-links, and is proportional to the ratio to between the maximum and minimum n-links in the graph. This means there is a trade-off between using very descriptive edge weights and being guaranteed a solution within a small factor of the optimum solution for that construction.

There are several efficient algorithms for cutting a graph. The challenge for the user of graph cuts is to construct a graph that accurately represents the problem, and whose minimum coincides with the solution desired. The edge weights define the function to be minimised in an optimisation context. Boykov et. al. [40] show how graph cuts are used in α-expansion. The graph encodes whether a pixel should be relabelled α. The t-links are the cost of a pixel being α and of being its current label. The n-links are more complex, with auxiliary nodes being used to represent the various combinations of labels.

The numerical value of the weights for the graph depend on the application, which in this thesis is segmentation. Graph cuts for segmentation are widely used segmentation, with a good introduction given by Boykov and Jolly [5]. The weights of the t-links will be the cost assigning a particular label to a pixel. In the texton algorithm of Chapter 2, this is the dissimilarity between the texton histogram of the point and that of the class. In cases where region seeding is used and certain pixels are known to have a certain label, the t-links can be set to ∞ so only the neighbour links matter. The neighbour links can use the POTTS model, where a fixed cost is incurred for pixel p having label α and pixel q, its neighbour, having label β. Alternatively, gradient information can be used and the edge weight will be inversely proportional to the strength of the gradient between the pixels.

Typically, labelling problems will use the cost of assigning a label to a class as the weight of a t-link with the POTTS model for n-links. Border detection graph cut algorithms will use seed pixels or some other form of initialisation to attach some nodes to the terminals via t-links of weight ∞. The n-link weights are proportional to a gradient between the pixels, for example a texture or colour gradient.

4.3.3 Unsupervised Graph Cuts

The processing of unlabelled data presents an additional problem. Here, we do not have training images which can be used to calculate the energy / cost of a assigning a pixel to a class. The parameters of the class descriptors θ, which may be elements of class histograms or the means and variances of each mixture in a GMM, and the parameters of the segmentation L (the label of each pixel) have to estimated. We are looking for the configuration $f(\theta, L)$ which has the minimum cost. Algorithms like k-means and expectation-maximisation (EM) divide the task into two steps. The first calculates L based on the current θ, and the second updates the model parameters θ using the current label configuration L. Starting from an initial segmentation (which for a good algorithm should not affect on the final result) the algorithm iterates by going between the first step (expectation) and the second (maximisation).

A problem with the algorithms for unsupervised segmentation is that they cluster only in the feature space. There is no consideration given to the spatial coherence of the clusters in the image. Zaboh

(a) Segmentation 1 (b) Segmentation 2 (c) Segmentation 3

Figure 4.4: Three example segmentations. Regions have been merged and split in both directions between segmentations 1 and 2, and in one direction between segmentations 1 and 3.

and Kolmogorov propose a graph cut segmentation algorithm [41]. Again, the graph cut is used as part of the α-expansion minimisation algorithm. Each expansion move forms the expectation part of the algorithm, with the graph cut used to find the best possible α-expansion for the current model θ. The maximisation part of the algorithm is the same as before, with the label configuration L being used to update the model parameters θ. This is done after each successful expansion move step. The use of graph cuts to calculate the optimal L penalises label configurations which are fragmented, thus introducing spatial coherence and smoothing the boundaries.

4.4 Segmentation Comparison

A segmentation algorithm can output one of two labellings: a labelling of each pixel in the image as belonging to a class and a labelling of each pixel as being on a boundary. The former labelling will be from a set of 1 to k, where as the latter is a binary labelling with most pixels likely to labelled 0 (not an edge). Consequently, the accuracy of each sort of algorithm must be measured differently. Section 4.4.1 describes accuracy measures for pixel label segmentations and Section 4.4.2 describes border detection accuracy measures.

4.4.1 Pixel Labelling Accuracy

Well-structured class labelling problems like the segmentation of the Brodatz mosaics (see Chapter 5) are easy to benchmark. There is unambiguous ground truth available and the accuracy can be

Figure 4.5: A Venn diagram of the disagreement between regions of segmentations at a pixel.If A is the set of pixels with same label as p in S_1 and B is the set of pixels with the same label as p in S_2, then the red area corresponds to $d(p, S_1, S_2)$, and the blue area to $d(p, S_2, S_1)$.

measured in terms of the percentage of pixels that are correctly classified, which is the same way the accuracy of a classifier is measured. However, segmentation is different from classification, particularly for pixels on the edge of two or more image regions. Thus, we use three accuracy measurements: the accuracy for the whole image; the accuracy for pixels whose neighbourhood is all in one region; and the accuracy for pixels that are near the boundary and whose neighbourhood consists of pixels from more than one class.

The accuracy of solutions to general labelling problems is more of a challenge, especially where there is no precise ground truth. The accuracy measurement falls apart when there are different numbers of regions in the segmentations being compared. Regions in one segmentation may split and become multiple regions or multiple regions may merge and become one region. The splitting and merging process is called *refinement*. Figure 4.4 shows an example of three segmentations where this has happened. For many images all segmentations are equivalent. A comparison measure is needed which will find that these two segmentations agree.

Martin et. al. developed a solution to this problem [10]. Their measure examines the disagreement between the region that a pixel is a member of in one segmentation and the region it is a member of in the other segmentation. Typically this will be the result of a segmentation and the supplied ground truth. The disagreement at pixel p and the global error are given by:

$$d(p, S_1, S_2) = \frac{|L(S_1, p) \backslash L(S_2, p)}{L(S_1, p)|} \tag{4.3}$$

$$GCE(S_1, S_2) = \min \sum_p d(p, S_1, S_2), \sum_p d(p, S_2, S_1) \tag{4.4}$$

$$LCE(S_1, S_2) = \sum_p \min d(p, S_1, S_2), d(p, S_2, S_1) \tag{4.5}$$

where $L(S, p)$ is the set pixels in segmentation S with the same label as pixel p, $A \backslash B$ is the set

difference operation between sets A and B and $|A|$ is the number of members of set A. Equation 4.3 gives the disparity at a pixel. The position of S_1 and S_2 are not commutative. Figure 4.5 shows why this is so. Because of this, there are two ways to calculate an total error for comparison. The Global Comparison Error (GCE) is given by Equation 4.4, while the Local Comparison Error (LCE) is given by 4.5. The difference between the two is that the GCE sums the area of the red and blue regions from Figure 4.5 for all pixels and find the minimum between the two sums (the global minimum), while the LCE finds the smaller of the red and blue areas for each pixel, and finds the sum of these local minimums for the whole image. What this means in terms of segmentations is that the GCE allows splitting of regions in only one direction between segmentations and merging in the other, while the LCE allows splitting and merging in both directions. The GCE would find segmentations 1 and 2 different but 1 and 3 the same, while the LCE would find all three equivalent.

The GCE and LCE express the disagreement between the two segmentation. Any two labellings can be compared with this method. The answer given by either calculation is only as reliable as the ground truth that the segmentation is being compared to. Many problems have no objective ground truth. In such cases a combination of the errors for each ground truth should be used, such as the mean or median.

4.4.2 Border Detection Accuracy

Border detection accuracy is a measure of how well the boundaries are detected and localised. This includes how much of the correct boundary is detected, how much of the detected boundary is correct and how close to the true boundary is it. This fits in with standard detection measures including Receiver Operator Characteristic (ROC) and precision/recall curves. Both methods examine how detection is affected by changing the weighting between detection and rejection, in this case the threshold at which a border pixel is accepted. In this application, approximately 1% of pixels will be border points, and so the precision/recall is preferable to ROC, because the false positives measure will be meaningless as the number of false border detections will be overwhelmed by the large number of correct non-border pixels. The precision/recall curve is only concerned with the correspondence between the detections in the result and the ground truth.

Martin et. al. use Precicision/recall to benchmark the performance of various border detection algorithms in finding the borders of objects for their data set of natural images [11]. Precision/recall was originally used in the context of information retrieval, for example retrieving images from a database [32]. Precision is the probability that a piece of information that has been retrieved is relevant to what was requested. Recall is the probability that a piece of relevant information has been retrieved as part of the request. The trade-off is between retrieving all the relevant information and a lot of irrelevant information, and retrieving only relevant information, but retrieving only a part of the

relevant information. In the case of a region border, precision is the probability that a detected border pixel is on a true border, and recall is probability that a border pixel has been detected as such by the algorithm. The measure of the algorithm is the F-measure. This is given by

$$\text{F-measure} = \frac{PR}{\alpha P + (1-\alpha)R} \tag{4.6}$$

where α is a parameter which weights the cost of retrieving irrelevant information against missing some relevant information. As the threshold for accepting a pixel as a border pixel is varied, the values of P and R will change. The F-measure which characterises the performance of the algorithm is the maximum value of all values of this parameter.

There is a hard and a soft way to implement the P/R calculation in the context of border detection. The hard method looks only at detected pixels in the result and ground truth and uses the percentage of matches as the values for P and R. This is not a good measure of what is acceptable for a border, as a border which is out by one pixel will be heavily penalised, while a human would find such a border entirely acceptable. Martin et. al. use a soft measure [11], which looks for the closest border point in the ground truth for each border point in the result and vice versa, and calculates the probability that the point are from the same border.

Chapter 5

Texture Databases

A variety of texture measures have been discussed thus far, chosen to represent as many categories of texture as possible. Even so, these are but a fraction of the many variations of texture measures that have been researched and implemented. It is important to note that each researcher does not approach the problem looking for a general solution which will adequately describe all texture. Each texture measure is designed for a specific purpose and will function best in that problem domain.

The first texture data set used is a series of mosaics based on the Brodatz album. Each mosaic contains grayscale images of several natural textures. Each mosaic has training images for each class of texture and ground truth. The second data set is a series of natural images [10]. For this data set, there are no training images, and the aim is to use the texture and colour information to segment the image into objects or parts of objects as a human would. Human judgement is subjective, so multiple human segmentations are compared to the results of the various algorithms.

5.1 Brodatz Album Mosaics

The Brodatz Album is a collection of textures published in 1966 as an essay on the photography of natural textured surfaces. It includes a number of sample images intended to be used as stock imagery by photographers and artists. It has subsequently come to be used by image processing researchers as a standard set of images on which to test any texture based segmentation or classification algorithm.

The Mosaics used were created by the Machine Vision group of Infotech Oulu at the University of Oulu [28]. The data set consists of 12 mosaics created from images of texture in the Brodatz album. Each mosaic consists of several texture classes. Each class has an associated training image. An example is shown in Figure 5.1. The images in this database are all grayscale, so no colour information is available to combine with the texture data.

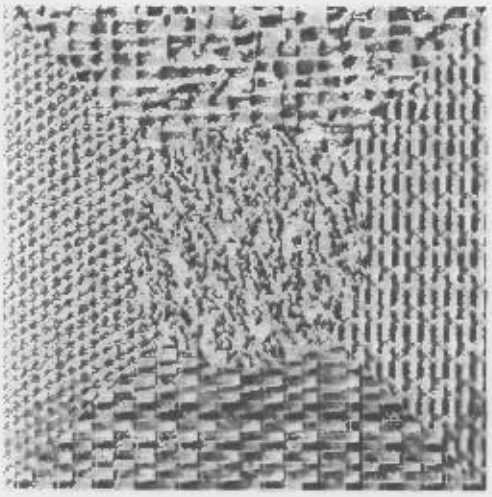


Figure 5.1: An example of the Brodatz texture mosaic.

In order to create more test images, all the training images were grouped together and four selected at random to create a new mosaic. Each of the four texture images chosen was divided into quarters. One quarter from each image was combined to form the mosaic, and the other three quarters was used as the training image for each class. Cross validation was used so that each quarter from each image was used as the test mosaic.

5.2 Berkeley Natural Images

While each image in the Brodatz album is of a natural texture, the mosaics used to test the segmentation algorithms are synthetic, with clearly defined borders. Natural images present a more challenging environment where the correct segmentation is not clearly defined. The aim is to segment the images as a human would. Segmentation of natural images is subjective, so segmentations done by different people will lead to different, and often contradictory, segmentation maps. The notion of ground truth is obviously less applicable in this test. Rather, the segmentation result is compared to each human segmentation of the image and the average accuracy over all the human segmentations for that image is used.

The data set used in these tests is a subset of those used by Martin et. al. [10]. That data set was taken from the Corel image database, which is a database of 40000 images that was widely used in computer vision at the time. It consists of many different natural images. The database used in the paper selected images with at least one large object that could be segmented out, examples of which are shown in Figure 5.2. These images are more challenging than the mosaics of the previous database. The scale of

Figure 5.2: Two examples of the Berkeley Natural Image data set.

the texture varies across objects that are of uniform texture because the surface orientation varies, and there is often not a clear dividing line between the textures. The images are in colour, which provides extra information that can be used on its own or combined with texture information to segment the images.

There are two parts to the data set, 200 images to be used for the training of the segmentation algorithms and 100 images to test the accuracy of the algorithm. Segmentations done by humans were obtained for the 300 images chosen to make up the data set. Due to the subjective nature of segmentations, there are multiple human segmentations of each image. An online program was set up so user could submit segmentations of images in the database. They were not given any guidelines on how they should segment the images, so there is little consistency in the different segmentations. The main difference is the number of regions in each segmentation. While some users segmented an object as one region (e.g. a house), others divide that object into its constituent parts. The Local Comparison Error measurement is used with each of the segmentations in turn and the average LCE for all human segmentations is used as the accuracy for that image.

Chapter 6

Texture Algorithms

Based on the texture measures reviewed, several methods were used to test the performance of each type of texture measure on the chosen data sets. The different image sets used require the texture measures to be used in different ways (for example supervised versus unsupervised segmentation). The implementation of each of these algorithms is now discussed, along with the choices and compromises that were necessary to implement the system.

6.1 Texton Labels

The largest category is the filter based texton algorithm. This is a broad category as the various choices made can lead to quite different algorithms. The first major choice is the filter bank to use. Based on their prevalence in the literature, the following filter banks were used:

1. Ring/Wedge (R/W)
2. Gabor
3. Berkeley Quadrature Pair
4. Maximum Response (MR8)

The Berkeley Quadrature Pair filter software is part of a set of texture related functions associated with the Berkeley Natural Images data set [1].

[1] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

There are at least two free parameters when using these filter banks: the size of the filters and the number of different scales and orientations. Filter size is dependent on the period of the texture in the image and on the image size.

There is a compromise between the stability of the descriptor and the area of ambiguity at the boundary between two classes. A range of parameters were tried to observe the effect of these settings on accuracy. While each of these filters has the same support, each one detects energy in the Fourier domain at different frequencies, at thus detect features in the image at different scales and orientations. The numbers of different scales and orientations are two additional parameters. Here, the parameters used in the literature have been used, and for algorithms where these parameters are not specified, values similar to those used by other filter banks have been used to ensure the results are comparable.

In the case of the Berkeley Natural Images data set , there is colour information as well. This information, after smoothing to reduce noise, forms a feature like a filter response, and this has been used as another input to the texton algorithm. Following standard practice for natural images, the CIELAB colour space is used, where distances between points in the colour space is based on dissimilarity in colour as perceived by humans. This is appropriate for natural images where the aim is to model the performance of a human segmenting the image. The three colour values can also be appended to a filter response vector to combine the colour and texture information before calculating the textons. The filter response vectors are fed into the texton algorithm. Having chosen the input, there are still several other unknowns in the texton algorithm, including

1. The number of textons to use in modelling the texture.
2. The size of the neighbourhood for calculating Texton Histograms.
3. The number of representative histograms per class.
4. The histogram comparison method.

A range of values for each parameter was tried to ascertain the effect of the variation on segmentation performance. The textons are modelled by a Gaussian Mixture Model and the parameters are calculated using an expectation-maximisation [3]. The Netlab toolbox implementation of the algorithm in the book was used for these tests [27].

The data set determined the way the textons were trained. The Brodatz mosaic had labelled training data, and so image-specific textons were calculated for each test mosaic. The texton histogram for each class was calculated from texton maps of that classes training images. The distance from the local histogram for each point in the image to each classes' histogram or histograms is calculated, and the nearest neighbour result is found by labelling each point using the nearest neighbour principle. The

Quadratic Form Distance is used, with the distance between each pair of textons used as the ground distance matrix between bins. The distance to each class's histogram was also used for a graph cut based segmentation using the α-expansion algorithm. The graph cut code uses the implementation of Boykov and Kolmogorov [4] [2]. The distance is used as the weight of the t-link in the graph cut. The POTTS model is used for the n-links. The correct weighting for the n-links in a POTTS model is subjective. A higher value will result in more smoothing, and a lower value gives a segmentation closer to the nearest neighbour value. A value of the same order of magnitude as the average t-link weight leads to a good balance between the data and neighbour links.

Three methods are used to calculate the histograms for each class. The first is a histogram of the whole training image (One Histogram per Class). The second calculates local histograms for each point in the training image and clusters them (Multiple Histograms per class), so that there are multiple histograms for each class. This allows for multi-modal texture classes to be accurately described and also has the ability to model textures that repeat over larger areas than the current support. The Berkeley Natural Images are an supervised segmentation problem, and each histogram found will be a class. Instead multiple histograms for each class, classes are split into one class for each histogram. The third method deals with the misclassification of boundary points. At the border, the neighbourhood will be a combination of two or more classes, and the point is often assigned to a third class which is similar to a combination of the two classes. This method uses the average histograms for each class and forms linear combinations of all pairs Border Modelling Histograms). This models the histograms found near the edge of a class, where the window will contain part of class A and part of class B. A histogram could be calculated for each stage of the progression, but initial testing showed that two histograms for each combination of classes, with a mixing ratio of 70:30 in each direction, is enough to substantially improve border localisation.

The Berkeley Natural Images data set has no training images for each class in the image, since the notion of a class in natural images is entirely subjective. The aim is to segment the image so that it best agrees with how a human would split the image into objects for the purpose of describing its contents. This means that segmenting these images are an unsupervised classification task, with both the classes and the labels to be estimated. The texton-based algorithms use k-means and graph cuts methods to segment the image. After the segmentation is done, there is a post processing step applied. After k-means, this step includes running a connected components algorithm and border refinement, the same post-processing as used by Carson et. al. [7]. After the graph cut is processed, only the connected component algorithm is run.

[2] http://www.cs.cornell.edu/ rdz/graphcuts.html

6.2 Texton Border Detection

The border detection algorithm also uses textons as a texture measure. The texton calculation and assignment phases are the same as for the labelling algorithms, consisting of filtering the image and then clustering the response vectors to form textons. Once each pixel is assigned a texton label, two histograms are now calculated for each pixel by the border detection algorithm. The neighbourhood is divided into two halves perpendicular to the gradient direction. The texton gradient for that direction is the difference between the histograms for the halves . The difference is calculated for halves with several orientations. Colour images also have a colour gradient. This is calculated in two ways, either using a regular bin histogram or an adaptive bin histogram like that used in textons. Again, the difference between the histograms for the two halves of a disc around the pixel is used to calculate edge energy. The texture and colour information has also been combined by appending the colour information to filter responses and using this feature vector for each pixel to calculate textons.

The exact boundary is then calculated using the algorithm described by Martin et. al. [11], using their software [3]. First the data is put through a logistic model to convert it to a probability, the a parabola is fitted to the data at each pixel for each set of gradient energy (one for each gradient orientation) at the same orientation as the gradient. This removes double peaks. Non-maxima suppression is used to locate the boundary a long a pixel wide line or curve. This method is called "Non-maxima suppression Edge Detection". The binary boundary label image is calculated by choosing a probability threshold at which to accept a pixel as being an edge point. Varying this parameter gives different results, and precision/recall curves for these algorithms can be obtained.

The boundary can also be detected by the graph cut boundary detection algorithm. The gradient between two pixels is used to calculate the cost of placing a border between the two pixels. The labelling of pixels produced by the graph cut labelling algorithm is used to seed the regions of the image. All pixels with a neighbourhood wholly in one class but adjacent to pixels with neighbourhoods that are not in this class are used as seed pixels. The set of seed pixels for each class is chosen in turn as the source, with the set of seed pixels from all the other classes as the sink. The boundary for that class is found when the pixels inside the border are claimed by that class. After iterating through all classes, some pixels may be unclaimed. The new labellings are used to produce new seeds, and the algorithm is run again until all pixels are claimed.

[3] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

6.3 Gaussian Mixture Modelling

An alternative to the two-stage texton modelling is direct modelling of the filter responses using Gaussian Mixture Models (GMM). The filter responses for each texture is represented by a mixture model in an n-dimensional space, where n is the number of filter responses. Each class has a GMM which is trained using an expectation-maximisation algorithm. The filter response of each pixel is then given a probability of coming from the GMM of each class. The average probability of the pixels in the neighbourhood is calculated, and this is used as the probability that the pixel belongs to a particular class. The pixel is given the label of the class with highest probability.

An advantage of this approach is that there are fewer free parameters to choose. Only the number of Gaussians the distribution of the mixtures model and the type of covariance matrix need to be chosen. This approach to modelling will work when the filter responses or other features obtained from the image are consistent across fairly large areas and clearly differentiate between classes.

6.4 Local Descriptor Methods

Four local descriptor methods have been employed: local binary patterns, Blobworld descriptors, image patches and a SIFT-like descriptor.

The *local binary pattern* algorithm calculates the features (the LBPs) directly from the image and there is no clustering needed, as with textons. The implementation used was that of Ojala et. al. [33] [4]. For each pixel, a number from 1 to 36 is generated, which is the label of the rotation invariant LBP at that pixel. The distribution of these patterns is the description used for segmentation. The image of binary pattern labels is processed in exactly the same way as the image of texton labels. All of the variants on the basic texton algorithm have also been applied to LBPs, including multiple histograms, border modelling and graph cuts for labelling, and non-maxima suppression and graph cut based border detection algorithms.

The *Blobworld descriptor* aims to be a parameter-free texture measure, though there are some implicit parameters. For instance, the exact value of the scales used by the scale selection algorithm affect what scales can be detected. Once the Blobworld descriptor is calculated for each point, the features are modelled using a GMM. Colour data is appended to the 3 texture values for the natural images, as well as the x and y position of the pixel in the image. There is one GMM for each class for the Brodatz mosaics where there are training images for each class, and one for all classes for the Natural Images. The implementation of the Blobworld descriptor used comes from the Blobworld Image Retrieval project [5].

[4] http://www.ee.oulu.fi/research/imag/texture/lbp/lbp.php
[5] http://elib.cs.berkeley.edu/blobworld/

The *SIFT descriptors* as used for point correspondence choose a scale based on a Scale Space algorithm. Most texture segmentation algorithms use multiple scales to represent the texture. The SIFT descriptor has been used as a basis for a similar gradient based feature. Three scales of gradient filter are used. The second moment matrix is formed, and the eigenvalues are used to calculate the direction and magnitude at each pixel for each scale. A histogram of directions is calculated, each point weighted by the magnitude of the gradient at that point. This is used as the descriptor for each pixel.

The last local descriptor algorithm uses the actual intensity values at each pixel and its neighbours in in a small patch around each pixel, called a *local image patch.* This is then processed in two ways. The first models the typical image patch for each class with a GMM. This produces a probability for each class, which is summed as described in Section 6.3. The second uses the image patch as an input to the texton algorithm, and proceeds in the normal manner.

Image patches are not invariant to rotation. The size of the descriptor is proportional to the square of the radius of the path and contains much redundant information. For example, an image patch of size 25×25 pixels would have a descriptor with 625 elements. A filter bank of the same size represents that patch with a descriptor that has far fewer elements, only 10 in the case of a R/W filter bank with 5 scales and orientations. Principal Component Analysis (PCA) is thus applied to the image patch to reduce the size of descriptor and make it invariant to rotation. PCA is applied to a 7x7 image patch to reduce it to 9 components, the same as a 3x3 patch.

Chapter 7

Testing and Results

The texture and colour based descriptors and segmentation techniques described in this work have been applied the Brodatz Album Mosaics and Berkeley Natural Images data set. Algorithms that span a range of texture paradigms have been included.

The tables in Figure 7.1 and Figure 7.2 give a complete description of all the algorithms that have been been tested on the complete set of data. Each algorithm is represented as a column in the table. The Labels at the top of each column are used in the text to refer to that algorithm. The codes starting with L refer to the labelling algorithms, those that assign a class label to each pixel. The codes starting with E are algorithms that assign a binary label to each pixel to specify whether that point is one a boundary. Labels from Figure 7.1 are used in Section 7.1 and labels from Figure 7.2 are used in Section 7.2. The labels at the bottom of the tables refer to the position of that algorithm in the graphs of results. Each graph that uses these number will refer back to the table. For example, Figure 7.4 refers to the second from bottom line of Figure 7.1. Each bar in the graph represents the LCE for the corresponding algorithm in the table.

Each X in the column represents an attribute of the algorithm. The attributes are divided into three categories. Primary Descriptors are derived directly from the image. Secondary descriptors are derived from Primary Descriptors. All the secondary descriptors are texton-based, so if there is a cross under Secondary Descriptors, it is implied that the Primary Descriptors are clustered to form textons. The Secondary Descriptor section describes how the texton histogram is calculated. The Segmentation section describes how the descriptor is used in segmentation. Each algorithm has one X in each section. The exception is the "Rotation Invariant" label which is not a Primary Descriptor, but is an attribute of some Primary Descriptors. For example of how to read the table, look at algorithm L1 in Figure 7.1. L1 has an X next to "Ring/Wedge Filters" "One Histogram per class" and "Nearest Neighbour". This means algorithm L1 proceeds as follows:

1. Apply the Ring/Wedges filter bank.
2. Cluster descriptors to form textons.
3. Calculate one texton histogram for each class of texture.
4. Compare histogram for each point with histogram for each class and label point according to which is closest.

There are a large number of possible algorithms that can be created by implementing every combination of options. It is assumed, however, that the three sections of the algorithm are independent. This means that if one descriptor performs better than another with a certain segmentation algorithm, it is assumed that the relative performance will not change using a different segmentation method. Algorithm development started by testing each of descriptors with the nearest neighbour segmentation method. The best descriptors were tested with more advanced segmentation techniques.

7.1 Brodatz Album Mosaics

Figure 7.3 and 7.4 show the accuracy of the algorithms tested on the Brodatz Mosaic Data set. The first benchmark is the accuracy of the segmentation in terms of what percentage of pixels is correctly labelled. This is measured for three groups of pixels. The first is the whole image. The second is only the pixels whose neighbourhood falls entirely within one class, and whose whole neighbourhood falls within the image. This excludes pixels on the edges of classes and the image. The third group consists of those pixels where no one class makes up more than 90% of the neighbourhood. This tests the accuracy of segmentation in the vicinity of the boundary.

The second benchmark also tests the accuracy of the labelling: thios is the Local Comparison Error (LCE), which measures the percentage of pixels with same label as a pixel in image 1 that have different labels in image 2. The third benchmark, the F-measure, measures the accuracy of the boundary location using the precision/recall metric. This has been applied to both the boundary detection methods and to the labelling methods, to compare the accuracy of the borders detected.

The best performing algorithm by the labelling accuracy measure is the local binary pattern algorithm using α-expansion with edge detection (L24). The best algorithm by the LCE measure is the texton algorithm with Ring/Wedge filter bank as input, using the graph cut based segmentation (L24). Both textons and LBPs model the texture as a distribution of low levels patterns. The other filter banks used as input to the texton algorithm did not perform as well. The Image Patch descriptor achieved results slightly worse than the best, but results agree with the findings of Varma and Zisserman [35], which question the need for filter banks. The worst performing descriptor was the SIFT-like gradient

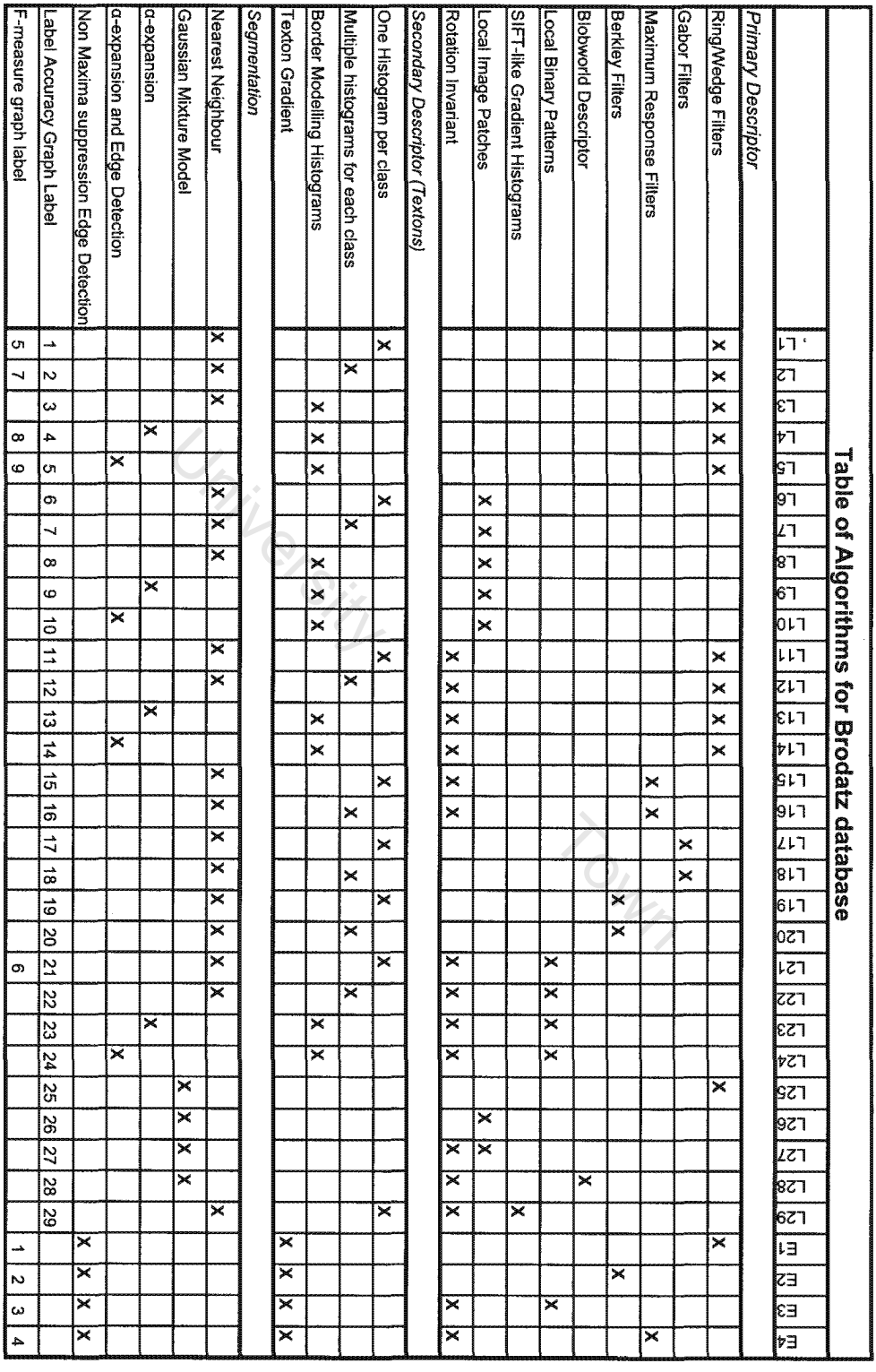
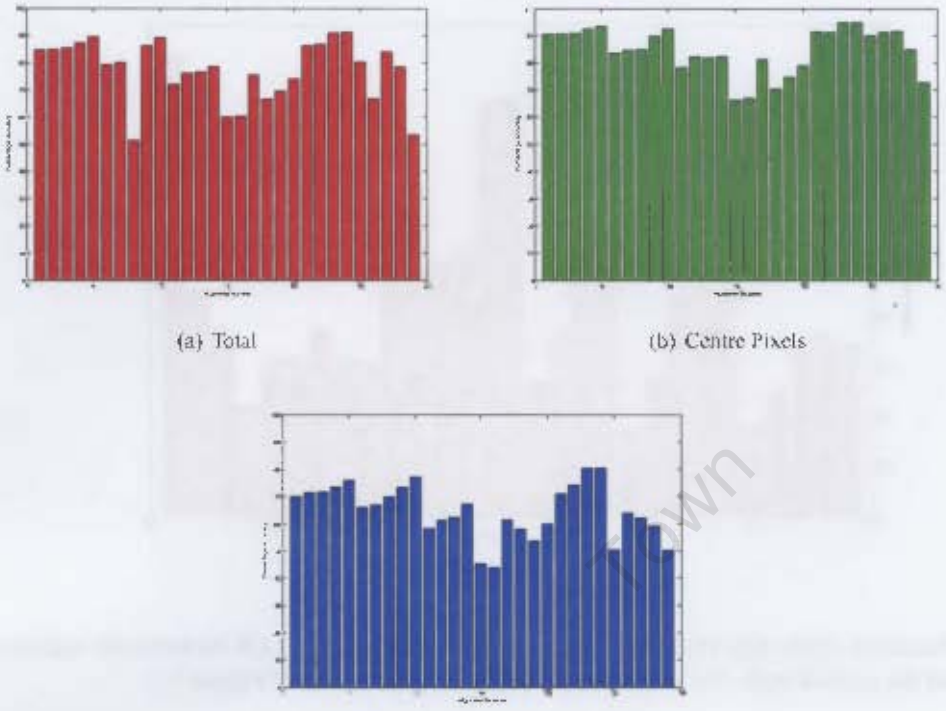


Figure 7.1: A summary of the algorithms used on the Brodatz Album Mosaics data set.

Table of Algorithms for Natural Images																					
	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	L16	E1	E2	E3	E4	E5
Primary Descriptor																					
Ring/Wedge Filters							X	X	X	X		X	X	X	X			X		X	
Berkley Filters																					X
Smoothed Colour			X	X	X	X						X	X	X	X		X			X	X
Blobworld Descriptor																X					
Local Binary Patterns											X								X		
Local Image Patches	X	X																			
Rotation Invariant		X									X					X					
Secondary Descriptor (Textons)																					
One Histogram per class	X		X				X					X									
Border Modelling Histograms				X	X	X		X	X	X			X	X	X						
Texton Gradient																	X	X	X	X	X
Segmentation																					
K-means			X	X			X	X			X	X	X								
K-means with Post-processing					X				X					X							
EM with Post-processing																X					
Unsupervised α-expansion	X	X				X				X					X						
Non Maxima suppression Edge Detection																	X	X	X	X	X
Label Accuracy Graph Label	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17					
F-measure graph label			3			4	6			7		10			11	12	2	5	8	9	13

Figure 7.2: A summary of the algorithms used on the Berkeley Natural Images data set.



(a) Total

(b) Centre Pixels

(c) Border Pixels

Figure 7.3: Percentage accuracy of texture algorithms on the Brodatz Album Mosaics data set. For labels see second from bottom row of Figure 7.1.

descriptor. The Blobworld descriptor (L28) ranks near the middle of the algorithms. Analysis of the performance of individual algorithms and comparisons between different approaches follows.

Rotation invariance is desirable for a texture descriptor. The tests do show that there can be a decrease in overall accuracy, though it still gives good results. The best comparison is between the standard R/W, sorted R/W and maximum response 8 (MR8) filters. These three filter banks are quite similar in that they filter over multiple scale and orientations. The MR8 takes the information at each scale of the orientation with maximum response, where the Sorted R/W filter bank keeps all the information, only sorting the orientation information from maximum response to minimum response. The MR filters throw away a lot of information and thus are handicapped in their description of the texture, resulting in poorer performance. The Sorted R/W filters (L11-L14) also do not perform as well as the unsorted R/W filters, but the performance difference is much smaller, as less descriptive power is lost by sorting the information compared with discarding it.

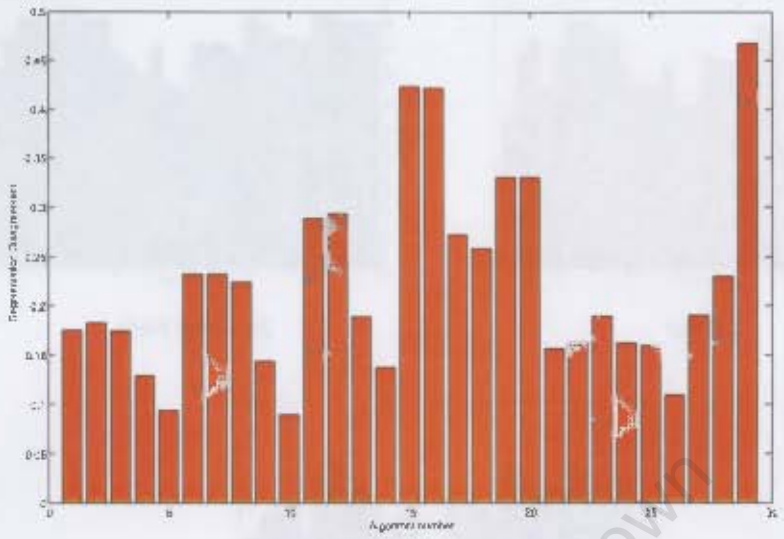


Figure 7.4: Accuracy of the algorithms used as measured by the average LCE between the segmentation result and the ground truth. For labels see second from bottom row of Figure 7.1.

On the other hand, the performance of the image patch based algorithms improves by discarding information.The LBP method (L21-L24) in fact uses only binary information for each pixel, and does better than algorithms which use the full intensity values at each pixel. It is also rotation invariant, choosing a subset of the 256 possible patterns such that no pattern in the set is a rotation of another. This is supported by the results using full image patches. The algorithm that uses the Principal Components of the image patches (L27) gives better accuracy than that which uses the raw image patches (L26).

In all texture algorithms, there is a parameter which in some way determines the scale of the texture detected in the image. The effect on accuracy of changing the scale parameter is interesting. As the scale increases, more pixels are included in pixel neighbourhood. As a result the sample of the texture at that point will have a histogram that is a better representation of the true histogram for that texture, which gives better segmentation. This is true for both homogenous neighbourhood and border pixels. There is a trade off though: the accuracy of the border pixels is always substantially lower, and as the scale increases more pixels are border pixels. At some point, the benefit of increasing the accuracy of some pixels is offset by increasing the number of pixels for which the histogram is ambiguous. In the tests, this occurred for a neighbourhood size of between 25 and 30 pixels, as shown in Figure 7.5. Figure 7.6 shows an example of the segmentation at scales of 10, 25 and 45 pixels.

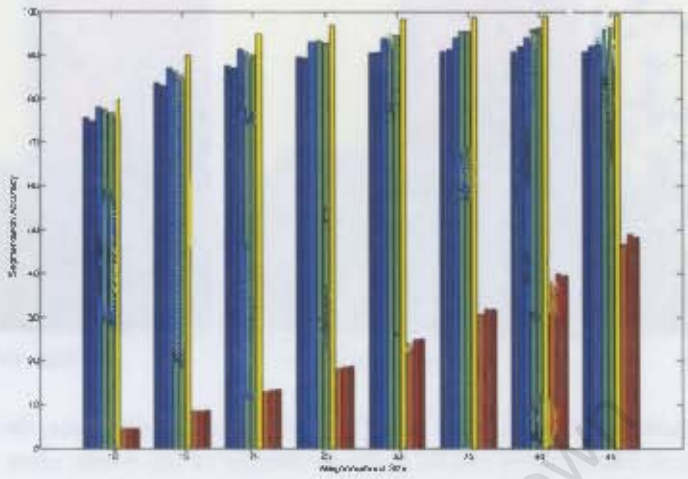


Figure 7.5: The effect of varying scale on accuracy. Each group of bars corresponds to a value used for size of the neighbourhood in calculating local histograms. In each group, the first three bar are accuracy for the whole image, the second three for the homogenous regions and the third lot of three for border region.

Two labelling approaches were tried using multiple histograms per class to improve accuracy at class borders. The first uses multiple representative histograms for each class, which should capture the appearance of the class at the edges. This also helps solve the problem of local histograms that are class outliers, by modelling local histograms for the class rather than the histogram of all points in the test data. The second algorithm attempts to model the feature vectors of pixels on class boundaries by creating linear combinations of each pair of classes and using these as additional classes. There is generally a small improvement in accuracy for both techniques, though rather small overall, they do make a difference in images which are not well captured using only one histogram per class. The multi-modal approach increases accuracy mainly for homogenous pixels, particularly when the texture being described is varying in appearance. For the second method, the accuracy for border pixels in increased. However accuracy in the homogenous regions can suffer when using this modelling technique, as outlying descriptors for a class can be more similar to a mix of the correct class and another class than to the descriptor from the correct class.

The use of global optimisation is the form of graph cuts shows gives an improvement in the segmentation. This is achieved by attaching a cost to pairs of labels along with a cost of assigning a label to each. Overall the improvement in accuracy is about 5 – 10%. This is achieved largely by reducing errors in small areas in the image which do not match any class well, where the likelihoods of each class may be similar. This can lead to erroneous labelling, but by attaching a cost to small

(a) Scale 10 pixels (b) Scale 25 pixels (c) Scale 45 pixels

Figure 7.6: Examples of Mosaic Segmentations at different scales. At small scales, the segmentation is fragmented as each local histogram is not an accurate sample of the whole class and there are many outlier regions. As the scale parameter increases, the segmentation becomes smoother as each neighbourhood better matches the class model histogram.

region, the cheapest cut is to smooth out the segmentation and include them in the larger regions. The second graph cut algorithm also includes edge information, which helps to refine the labelling at the class boundaries and improve accuracy further. The improvement can be dramatic in some images, as demonstrated by the results shown in Figure 7.7, where the nearest neighbour labelling is poor, with many small regions, while the graph cut labelling correctly assigns almost all the points.

The border accuracy of the labelling algorithms is compared to the accuracy of the border detection algorithms in Figure 7.8. This gives the F-measure for the precision/recall curves of the various methods. The labelling algorithms do not have a threshold parameter to vary. Instead, the neighbourhood size is varied to create a P/R curve for the labelling algorithm. Each labelled image is converted to a boundary map, using the boundaries between classes as edges, which is then used in the precision/recall calculations.

The algorithm that most accurately detects edges or borders according to the average F-measure metric is the α-expansion algorithm with graph cut edge refinement (L5). The nearest neighbour labelling algorithms (L1,L21) perform as well as the non-maxima suppression edge detection algorithms (E1-E4), with the R/W Textons labelling doing best and the LBP edge detection giving the best results of that set. The other two filter banks tried did relatively poorly.

The improved performance of the graph cut based algorithm is due to the constraints placed upon the location of the edge by the graph cut framework. The non-maxima suppression approach only chooses whether a particular pixel is an edge or not. It does not take into account the edge as a whole, or ensure that the edge is continuous. With the labelling approach, the edges divide regions, and so

(a) Nearest Neighbour (b) α-expansion (c) α-expansion and Edge Detection

Figure 7.7: Improvement in Segmentation through the use of Graph Cut based algorithms.

are continuous, but no edge information is used explicitly and again only the cost of assigning a label to each pixel is considered.

The graph cut algorithms are imposing a cost on the edge, represented by the cut. The cost of the cut is higher for jagged borders, as more graph edges will have to be included in the cut. This creates smooth regions in labelling and edge detection, which generally leads to better results. The first graph cut algorithm is only using the nearest neighbour labelling data in the cut. The second performs much better because it is using the labelling data learned from the training images and the edge data from the image. The boundaries detected by the non-maxima suppression and the graph cut methods are shown in Figure 7.9, which demonstrates the improved performance of the graph cut methods.

The overall results for the Brodatz Mosaics show that distributions of low-level texture features, be they textons or LBPs, worked best for these images. Features like SIFT and Blobworld describe the texture in such a way that is not sufficiently differentiating between classes or does not show enough similarity between members of the same class for successful segmentation, which leads to poorer performance of these algorithms. Another reason may be that varying the scale at which the descriptor is calculated does not increase the accuracy of the classifier, but rather gives different descriptors for members of the same class because different scales are selected, again to the detriment of overall accuracy. Also, scale space in SIFT is designed to be used at "feature points" in the image, rather than at all points, which means that the scale value at most points in the image is not reliable. The direct modelling approach of a Gaussian Mixture Model is also not as successful as the indirect approach of first transforming the picture into a labelled image with each label representing a particular low-level pattern, and then finding distributions of these patterns in the labelled image to use for segmentation. The choice of low-level pattern is not as significant, with image patch textons, LBPs and R/W filter bank textons all producing good results. The most important part of the algorithm is whether the

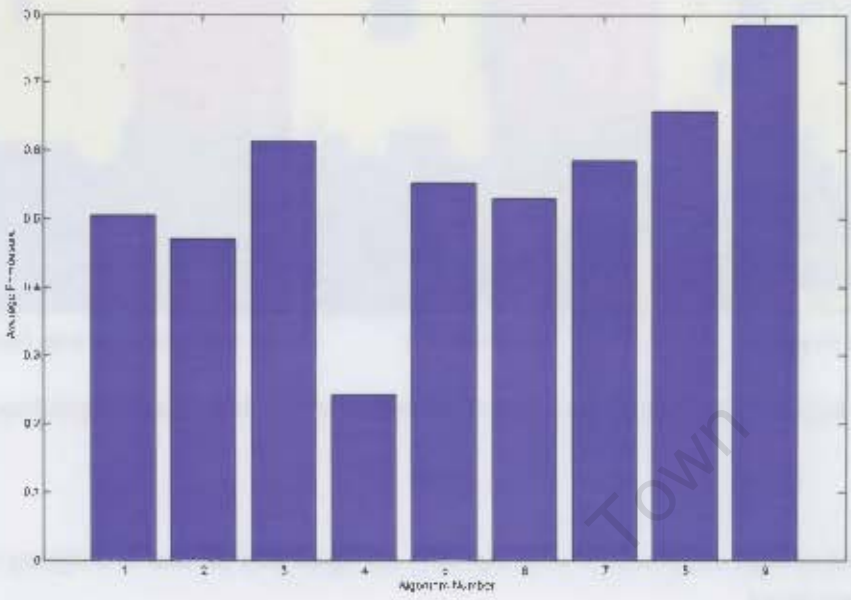


Figure 7.8: The average F-measure for boundary detection for the Brodatz Album Mosaics data set. For labels see bottom row of Figure 7.1.

segmentation is optimal only for each pixel, or globally as with graph cuts. Using both edge and region information has also proved to make a significant difference, and this gives the best performance of all.

7.2 Berkeley Natural Images

The best performing of the algorithms from the previous tests were applied to the Berkeley Natural Images data set. This data consists of 200 Training Images and 100 Test Images. Training of the algorithms is done on the first set, and accuracy results are generated by applying the trained classifiers to the second set. The classifier should be as universal as possible and be applicable to a wide range of natural images. The list of algorithms tried is shown in Figure 7.2.

The main difference between algorithms in this data set and those in the Brodatz data set is that these images are colour images, and because they are of natural scenes, each region of the image does not necessarily have significantly differing texture. Many areas can best be described as flat colour with noise, rather than colour texture. This means that colour based measures are important, and were used

(a) Non-Maxima Suppression (b) α-expansion (c) α-expansion and Edge Detection

Figure 7.9: Edge Detection results for different algorithms on Brodatz Album Mosaic.

as well as (grayscale) texture methods. A combination of texture and colour was also used, such as textons with the colour values at a pixel appended to the filter responses, or the Blobworld descriptor which appends the colour information to its three texture measures.

The validity of the accuracy measurements of the various algorithms is based on the validity of the ground truth. The Brodatz images are mosaics of real texture images, and thus there is well defined ground truth. The Berkeley Natural images are not constructed, so the ground truth is subjective. Multiple ground truth segmentations, done by users drawing edge, were transformed into a region labelling, and used instead of objective data. What constitutes an edge, or conversely what groups are homogenous enough to be considered a region, is different for each person. The agreement of the results with each ground truth labelling is compared with the agreement of the labellings with each other. The desired output of an algorithm operating on this data set, rather than 100% accuracy, is an algorithm which produces results with the same level of agreement with the human labellings as the human labellings have with each other.

Figure 7.10 shows the results of running the various algorithms on the images in this data set measured by the LCE metric compared to the LCE measurement between the human segmentations. The texton examples have several results. The first is the nearest neighbour result, which clusters the texton distributions using k-means. The second includes border modelling using linear combinations of histograms. The third applies the postprocessing step from the Blobworld framework, which consists of a connected-components algorithm and a border refinement algorithm. The fourth result uses the graph cuts based unsupervised α-expansion algorithm to segment the image, which also has a connected-component step.

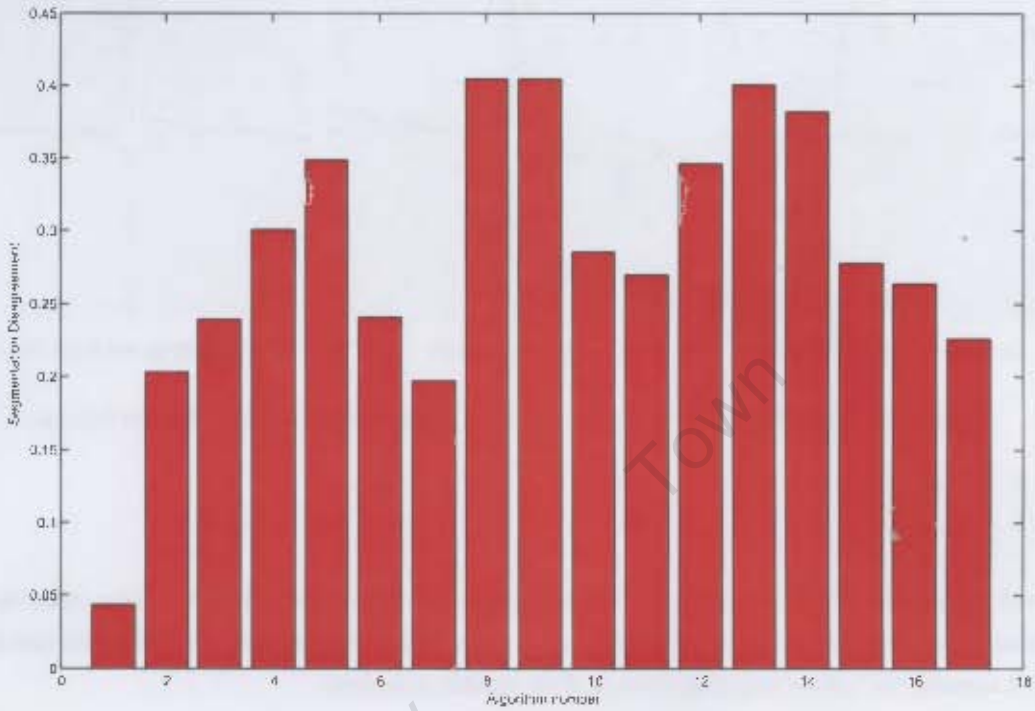


Figure 7.10: The average Local Comparison Error between the results and the ground truth for natural images. The first bar is the benchmark (LCE between human segmentations). For labels see second from bottom row of Figure 7.2.

The edge detection results are shown in Figure 7.11. Again, the average F-measure calculated from the precision/recall measurements between human segmentations of the same image is used as the benchmark and is the first result from the left in the graph. As with the Brodatz results, the edge detection algorithms are compared with the edge detected by the labelling algorithms, including the graph cuts algorithms.

The LCE results for the labelling algorithms measured show that the nearest neighbour results are the worst for each descriptor. The connected-component algorithm clearly makes a large difference to the LCE measure. This gives a segmentation which may not seem correct to all human observers, as objects of the same type (like a group of bears, for instance) will be given different labels if they are separated in the image. As long as the regions output by the connected-component algorithm are large enough to represent a real object, and not so small that the LCE measure becomes meaningless, it is safe to say that the improvement in this measure represents an actual improvement in performance.

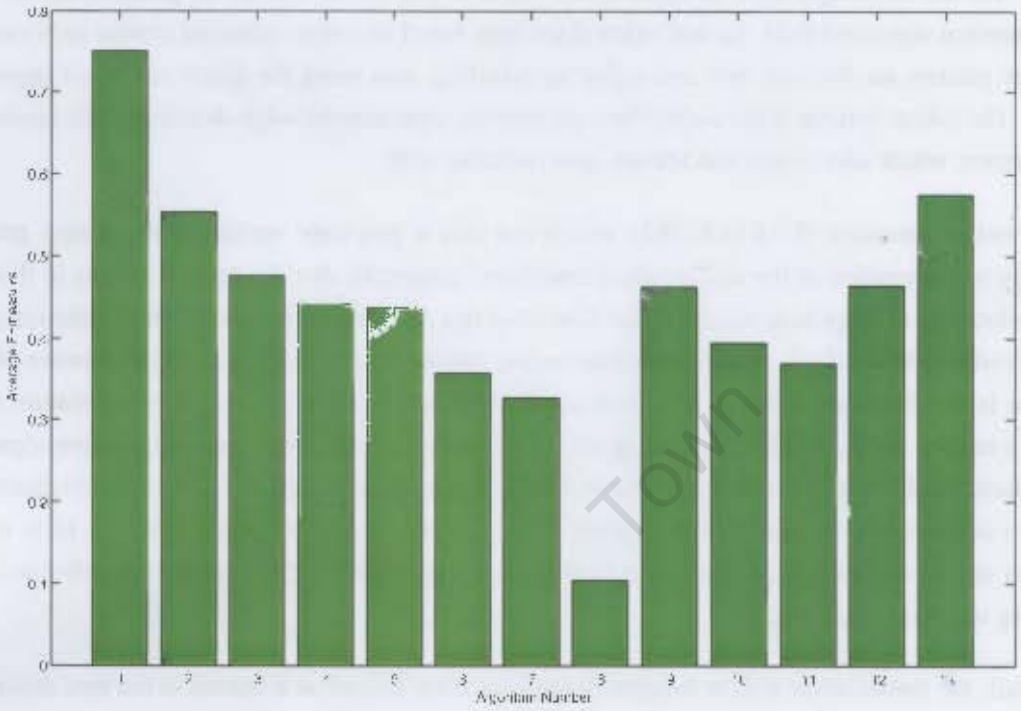


Figure 7.11: The average F-measure for natural images. The first bar is the benchmark (F-measure between human segmentations. For labels see second from bottom row of Figure 7.2.

The edge detection results here show that the non-maxima suppression edge detection achieves better edge location than the graph cut approach, which is the opposite of the results achieved in the case of the Brodatz data set. This is most likely due to the added complexity of borders in natural images, which are often irregular and can be at any angle. The borders obtained from graph cuts tend be regular, due to the added penalty associated with cutting multiple neighbour links which is necessary when borders are curved. This is because borders which are not at an angle divisible by 45 degrees will have to cut more neighbour links per pixel. This penalises irregular borders, which can hamper the performance of graph cuts in segmenting natural images. The non-maxima suppression algorithm aims to suppress pixels near the maximum only in the direction of the edge and can handle any edge direction. The graph cut algorithm in fact performs worse than nearest neighbour algorithms in edge detection.

In terms of descriptors, the best performing algorithms are those that use colour, or colour with texture. Overall, the lowest LCE is achieved by the colour texton with graph cut based segmentation (L6), and the best edges were detected by the combination colour and texton gradient non-maxima suppression algorithm (E5). Several other algorithms based on colour achieved similar performance. Image patches are the next best descriptor for labelling, also using the graph cut based segmentation. The colour textons is the second best performing algorithm for edge detection. The Blobworld descriptor, which uses colour and texture, also performs well.

The texture measures (L7-L11,E2-E3), which use only a grayscale version of the image, perform poorly in comparison to the colour based descriptor, suggesting that the natural images in this data set, which are of large scale scenes rather than close ups of distinctly textured surfaces like images in the Brodatz album, do not contain sufficient texture information to easily distinguish between objects in the image based on texture. In fact, it would seem that it is a hindrance to segmentation rather than a helpful feature as the combination of texture and colour features used by the texton algorithm produced results worse than those produced using just colour information. The Blobworld descriptor, which combines high level texture measures like contrast with raw colour, produces label results which are no more accurate than the colour textons, suggesting that the texture information is not adding much new information.

Overall, the results show that in images of natural scenes, colour as a feature is the best descriptor. It is important to note that the colour as used by these algorithms is not just the colour at each pixel, but rather describes the distribution of colours in a neighbourhood like the standard filter bank texton algorithm does with filter responses. This means that patterns in the colour, or colour texture, is really what is being described by colour textons. Thus texture information is still be used, but using colour rather than only grayscale texture, which does not have sufficient descriptive and differentiating power in this context.

Chapter 8

Conclusions

The aim of this work is to investigate methods for modelling and describing texture in images for the purpose of segmentation. Several paradigms of texture measures were reviewed and implemented. Several segmentation algorithms were tried so as to best use the description of the texture. The algorithms were applied to two data sets. The first is a set of mosaics of images from the Brodatz photographic album of textured surfaces. The second consists of images on natural and human scenes with several types of objects and backgrounds in each image which can be differentiated by texture and colour.

The results suggest that texture is best modelled by a two stage descriptor like textons. The descriptor first detects low level patterns in the image and produces a new image which has label for each pixel representing the pattern present in the neighbourhood of that point. The description is then based on the distribution of these labels in the neighbourhood. The alternative is to model the texture based on the raw image data or the responses to a bank of filters.

The best primary descriptor for grayscale images is the Ring/Wedge filter bank, while for colour image smoothed pixel colour or local image patches are best. The best method is best for calculating the textons. The best method for calculating the texton histogram is to calculate the average histogram for the whole class and model the interaction between different classes at the borders.

The effect of the two stage process is to transform the original image, with many possible gray-levels or colour value, into an image with fewer possible values for each pixel and which can be described by a simple distribution. In terms of the original description, the distribution of labels is adaptive histogram in the original feature space. This method improves the performance of the underlying descriptor because the same low-level patterns can occur in both classes, as long as the distribution of these patterns is different.

The underlying descriptor is less important, and several that were implemented have similar results in many respects. Of particular interest are local binary patterns and image patches, which use only the immediate neighbours of a pixel as a descriptor. These suggest that very small scale patterns are responsible for low-level texture patterns, and that the large support of typical filter banks are unnecessary. This is demonstrated well by the colour textons, which use only the colour at each pixel as the input to the texton algorithm, and accurately describes the textural appearance of an object.

A problem with current texture methods has been the distribution of textons or similar low-level patterns near the boundary of two regions. The results show that accuracy can be improved by using multiple histograms to represent each class. Some of these histograms will then explicitly model those parts of a class that overlap with other classes. Equally important in obtaining accurate segmentation results is the use of a segmentation algorithm that penalises fractured labellings by associating a cost with neighbouring pixels that are labelled differently. Region and edge texture information are both important, and they should be used together obtain the best texture segmentation results. The best supervised segmentation methods are the graph cut α-expansion methods, especially those that include edge refinement. For unsupervised segmentation, unsupervised α-expansion is best for pixel labelling, and Non-maxima suppression is best for edge detection.

8.1 Future Work

3D textons were mentioned in Section 2.2.4. In multi-view applications, and images like those in the Berkeley Natural Images data set, the assumption that the surface normal is facing the camera and constant will not always hold. Further work in texture measures should investigate what measures are best at describing texture when the surface normal changes.

In a more general work, work needs to be done on using 3D geometrical information with texture measures by rectifying the texture so that it appears as it would if viewed face on.

The individual pixel intensity values in a textured image do not follow a deterministic pattern, and the randomness appears to be more than Gaussian noise. Further work should investigate models like Markov Random Fields for the probability of the intensity of a pixel taking on a certain value.

The graph cut methods proved successful, but more work is needed to properly integrate region and edge information into the graph cut framework so as to achieve smooth regions and precisely located edges.

Bibliography

[1] Ballard and Brown. *Computer Vision*, chapter 6, pages 166–194. Prentice Hall, 1982.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[3] Christopher M. Bishop. *Neural Networks for Pattern Recognition*, chapter 6. Clarendon Press, 1995.

[4] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimisation in vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(9), 2004.

[5] Y.Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *International Conference on Computer Vision*, pages 105–112, 2001.

[6] Miguel Carreira-Perpian. A review of dimension reduction techniques. Technical Report CS–96–09, Dept. of Computer Science, University of Sheffield, January 1997.

[7] Hayit Greenspan Chad Carson, Serge Belongie and Jitendra Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, August 2002.

[8] Papadimitrou Seymour Dalhaus, Johnson and Yannakakis. The complexity of multiway cuts. *ACM Symposium on the Theory of Computing*, 1992.

[9] M Jernigan David Clausi. Designing gabor filters for optimal texture separability. *Pattern Recognition*, 33(11):1835–1849, November 2000.

[10] Doron Tal David Martin, Charless Fowlkes and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, pages 416–425, 2001.

[11] Charless C. Fowlkes David R. Martin and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(1), JANUARY 2004.

[12] Michael Grossberg Efstathios Hadjidemetriou and Shree K. Nayar. Multiresolution histograms and their use for texture classification. *Proceedings of the 3rd international workshop on texture analysis and synthesis, in conjunction with ICCV 2003*, 2003.

[13] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*, chapter 11, pages 287–326. Prentice Hall, 2003.

[14] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*, chapter 8, pages 206–235. Prentice Hall, 2003.

[15] J.J. Gibson. *The Ecological Approach to Visual Perception.* Houghton Mifflin Company, 1979.

[16] Porteous Grieg and Seheult. Exact maximum a postereori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2), 1989.

[17] D. Raba J. Marti J. Freixenet, X. Munoz and X. Cufi. Yet another survey on image segmentation: Region and boundary information integration. *European Conference on Computer Vision*, 2002.

[18] Carlo Tomasi Jan Puzicha, Yossi Rubner and Joachim M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. In *International Conference on Computer Vision*, pages 1165–1172, 1999.

[19] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, March 1981.

[20] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 02:506–513, 2004.

[21] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance ofmaterials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[22] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2), 1998.

[23] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[24] Leung Malik, Belongie and Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1), 2001.

[25] David Marr. *Vision*. W.H. Freeman and Company, 1982.

[26] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[27] Ian T Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer, 2002.

[28] Timo Ojala and Matti Pietikainen. Unsupervised texture segmentation using feature distributions. *Pattern Recognition*, 32(3):477–486, March 1999.

[29] C.N. Canagarajah P.R. Hill and D.R. Bull. Texture gradienft based watershed segmentation. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 4:3381–3384, 2002.

[30] Trygve Randen and John H Husoy. Filtering for texture classification: A comparative study. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

[31] J. Buhmann S. Will, L. Hermes and J. Puzicha. On learning texture edge detectors. In *International Conference on Image Processing*, volume III, pages 877–880, Vancouver, Canada, September 2000.

[32] C. Schmid. Constructing models for content-based image retrieval. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[33] Matti Pietikinen Timo Ojala and Topi Menp. Multi-resolution gray scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7), July 2002.

[34] M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *European Conference on Computer Vision*, volume 3, pages 255–271, May 2002.

[35] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 691–698, June 2003.

[36] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1–2):61–81, April 2005.

[37] M. Varma and A. Zisserman. Unifying statistical texture classification frameworks. *Image and Vision Computing*, 22(14):1175–1183, 2005.

[38] Changsong Liu Xuewen Wang, Xiaoqing Ding. Gabor filters based feature extraction for character recognition. *Pattern Recognition*, 38:369–379, 2005.

[39] Carlo Tomasi Yossi Rubner and Leonidas J. Guibas. The earth movers distance as a metric for image retrieval. *International Jounal of Computer Vision*, 40(2):99–121, 2000.

[40] Olga Veksler Yuri Boykov and Ramin Zabih. Fast approximate energy minimisation via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11), 2001.

[41] R. Zabih and V. Kolmogorov. Spatially coherent clustering using graph cuts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 437–444, 2004.

[42] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing and bayes/mdl for multiband image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(9):884–900, September 1996.