Automated Content-based Image Retrieval

A dissertation submitted to the University of Cambridge
for the degree of Master of Philosophy

Tian Tsong Ng, Hughes Hall

Oct 2000

Signal Processing Laboratory
Department of Engineering
University of Cambridge
Problems worth of attack,
Prove their worth by fighting back.

An old saying in mathematics
DECLARATION

The research described in this dissertation was carried out the author between October 1999 and August 2000. Except as indicated in the text, the contents are entirely original and are not the results of work done in collaboration. No part of this dissertation has been submitted to any other university. The dissertation contains no more than 15,000 words.

Tian Tsong Ng
I would like to thank my supervisor Dr. Nick Kingsbury for his dedicated guidance which has seen me through the completion of my M.Phil research. I am also grateful to Professor Peter Rayner for his insightful suggestions which have led to further improvement of my work.

I would like thank the Malaysian Public Services Department and the Association of Commonwealth Universities for granting me the essential financial support for my studies. I also like to express my heartfelt thankfulness to the British Council and Ian Marvin in particular for their concerns to my welfare in this country.

In addition, I would like to express my sincere gratitude to Alvin Kam for his selfless sharing of his knowledge and my problems and Nathan Smith for his constant support of my work on kernel methods. I am also thankful to Peter de Rivaz, Christrope Andrieu and Rafeal Rosales for their kind advice.

Last but not least, I would like to thank Alvin Kam, Wei Shi and Wee Jing Ng for their patient assistance which has made my experience of learning LaTeX much less painful and Simon Hill, Nathan Smith and Alvin Kam for their tremendous effort in improving the overall quality of this dissertation.
PREFACE

This dissertation is concerned with the task of image retrieval based on direct content of images, popularly known as content-based image retrieval (CBIR). An approach consisting of indexing and retrieving images based on texture and colour properties of homogeneous regions, achieved through image segmentation, has been adopted.

A novel image segmentation algorithm has also been developed. Experiments using this algorithm have been conducted in the context of texture segmentation. An attempt is made to relate image segmentation to machine learning by using the kernel principal component analysis, an unsupervised learning technique, to reduce feature dimensionality. The idea of the kernel method is extended to generalise the mean shift clustering algorithm. The resultant kernelised version of the algorithm demonstrates desirable adaptiveness and is capable of modelling irregular clusters, which are commonly encountered in image analysis. Finally, a classification algorithm which operates in the probability domain is introduced and its performance is demonstrated on various artificial and real images.
GLOSSARY

PCA : Principal Component Analysis
KPCA : Kernel Principal Component Analysis
GRBF : Gaussian Radial Basis Function
CBIR : Content-based Image Retrieval
K-S : Kolmogorov-Smirnov
Contents

1 Introduction ......................................................... 1
  1.1 Image Segmentation ........................................... 2
  1.2 Outline ...................................................... 3

2 Texture Segmentation and The Kernel Method ................. 4
  2.1 Unsupervised Texture Segmentation Algorithm ............... 4
  2.2 Image Segmentation and Machine Learning .................. 6
  2.3 The Kernel Method ........................................... 6
    2.3.1 Input Space and Feature Space ......................... 8

3 Feature Extraction and Dimensionality Reduction ............ 10
  3.1 Complex Wavelet Transform ................................ 10
  3.2 Data Transform ............................................ 12
  3.3 Principal Component Analysis ................................ 14
  3.4 Kernel Principal Component Analysis ....................... 18
  3.5 KPCA for Reduction of Feature Dimensionality ............. 20
  3.6 Incorporation of Spatial Information ....................... 21

4 Cluster Analysis .................................................. 25
  4.1 Kernel Density Estimation ................................... 25
  4.2 Mean Shift Clustering Algorithm ............................ 26
  4.3 The Kernel Mean Shift Clustering Algorithm ................ 28
  4.4 Properties of The Kernel Mean Shift Clustering Algorithm 33

5 Classification ....................................................... 39
  5.1 Averaging in the Feature Vector Domain .................... 39
  5.2 Averaging in the Probability Domain ........................ 40
  5.3 Classification based on Recursive Update of Posterior Distributions 42

6 Image Segmentation Experiments ................................ 45

7 Content-based Image Retrieval .................................. 53
  7.1 Unsupervised Image segmentation Algorithm ................ 53
  7.2 Colour Descriptors ........................................... 54
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3 Texture Descriptors</td>
<td>54</td>
</tr>
<tr>
<td>7.4 Query Scheme</td>
<td>56</td>
</tr>
<tr>
<td>7.5 Experiments</td>
<td>56</td>
</tr>
<tr>
<td>8 Conclusions and Future work</td>
<td>61</td>
</tr>
<tr>
<td>8.1 Future work</td>
<td>61</td>
</tr>
<tr>
<td>8.1.1 High-level features</td>
<td>63</td>
</tr>
<tr>
<td>A Principal Component Analysis</td>
<td>65</td>
</tr>
<tr>
<td>B Kernel Principal Component Analysis</td>
<td>67</td>
</tr>
<tr>
<td>C A Paper Presented in CBAIVL-2000: Content Based Image Retrieval through Object Extraction and Querying</td>
<td>72</td>
</tr>
</tbody>
</table>
List of Figures

3.1 one-dimensional wavelet filter bank ........................................... 11
3.2 Demonstration of feature extraction using complex wavelet transform ... 13
3.3 Distributions of the magnitude of the complex wavelet coefficients .... 15
3.4 PCA components for a two-dimensional toy data. .......................... 17
3.5 GRBF KPCA components with the parameter value comparable to the variance of the clusters ................................. 20
3.6 GRBF KPCA components with the parameter value relatively larger than the variance of the clusters ................................. 21
3.7 Comparing feature vector averaging and feature vector concatenation .. 24

4.1 A plot of an one-dimensional Epanechnikov kernel ........................ 26
4.2 Distance functions derived from Gaussian distributed samples .......... 34
4.3 Distance function for mean shift iterations on an anisotropic Gaussian distributed cluster with 600 data points. ......................... 37
4.4 Distance function for mean shift iterations on an anisotropic Gaussian distributed cluster with 25 data points. ......................... 38

5.1 A two-dimensional example of the region between 2 clusters ............ 40

6.1 Texture images for experiments .................................................. 45
6.2 The best segmentation results for different configurations of the algorithm 48
6.3 The error map and error percentage for the segmentation results. ....... 49
6.4 Segmentation label maps for 8 iterations of classification based on recursive update of posterior distributions. .......................... 50
6.5 Confidence maps for 8 iterations of classification based on recursive update of posterior distributions. .......................... 50
6.6 Segmentation maps for natural images ............................ 52

7.1 Sequence diagram for the semi-automated retrieval scheme ............ 57
7.2 Sequence diagram for the automated retrieval scheme .................... 57
7.3 Query regions used in the experiment for evaluating different distance measures. ......................................................... 59
7.4 Experimental results for different distance measures ........................ 60

8.1 Examples of textured and plain images .................................... 63
C.1 Decomposing an image by segmentation into classes corresponding to ‘objects’ 73
C.2 Typical segmentation maps of images ........................................... 74
C.3 Histogram of complex wavelet coefficients ................................. 76
C.4 Example of a one-class default query ........................................... 78
C.5 Example of an automated retrieval .............................................. 79
C.6 Precision vs. recall performance ................................................... 80
Chapter 1

Introduction

Traditional image retrieval systems rely on text annotation of images. There are a number of problems associated with this traditional mode of image retrieval. The main problem is failure to scale with fast-growing, ever-changing and unconstrained image databases like the Internet. Because the annotation of images requires human input this may be cost-prohibitive for ever-changing image database. Another key problem is that such annotations are subjective, also it is difficult to encode all meta-information about images which cater for image retrieval in any foreseen or unforeseen circumstances. Being subjective, the annotations may not match users’ notion of similarity between images which is dependent on the content of database, the context and the application.

Content-based image retrieval (CBIR) is another image retrieval approach. It works through the direct use of image content rather than text annotations. With this, CBIR systems may be fully automated in the indexing of the image database and free from human dependence. In general, Users make queries on CBIR systems by submitting a query image. In the background, the system evaluates the similarity between the query image and images in the database and return those with good similarity. In general, CBIR systems use features like colour and texture for image indexing and retrieval as well as shape information, spatial relationship between regions, salient points and so on.

There are a few commercial CBIR systems available in the market and the most well-known one is probably the QBIC system [4] developed by IBM. All of the major commercial CBIR systems rely mainly on global features of images although localised information may be extracted from heuristically partitioned regions in images. Global features tend to lose interesting information about localised regions which may correspond to objects. However, to extract localised information from regions with homogeneous properties, images need to be segmented using an image segmentation algorithm as a pre-processing step. CBIR systems based on localised features are more effective provided a reliable image segmentation algorithm is available [7]. Unfortunately, devising a good unsupervised image segmentation algorithm is itself a daunting task.

The research described in this dissertation has led to the development of an object-based CBIR system in collaboration with a member of the laboratory, Alvin Kam and an unsupervised image segmentation algorithm.
1.1 Image Segmentation

The goal of image segmentation is to classify image regions with distinct statistical properties into different classes. Image segmentation plays an important role in many computer vision and pattern recognition applications apart from content-base image retrieval. To segment an image, various properties or features can be employed to distinguish between regions and commonly used features are texture, colour, intensity and edges.

In this dissertation, an unsupervised image segmentation algorithm based on texture features is proposed. Being an unsupervised algorithm, the number of classes and their characteristics are not known a priori. Texture features are modelled through multi-resolution filtering using complex wavelet transform [22]. The algorithm is based on the general framework for unsupervised image segmentation proposed in [18]. Modularity of the framework has allowed innovations to be incorporated easily. Although only texture features are used in the algorithm at this stage, other features such as colour features can be incorporated without major modifications.

Image segmentation like pattern recognition [1] can be considered as a problem of machine learning. For image segmentation, the knowledge about every single texture in an image needs to be learned and described using certain representation before classification of regions is possible. Feature extraction itself is a means of unsupervised learning [38], so is cluster analysis [34] in which the concept of distinct textures are learned by partitioning the data space with reference to the clusters. Motivated by the parallel between image segmentation and machine learning, techniques familiar to the machine learning community such as kernel principal component analysis [32] and the kernel method [6] are introduced into the image segmentation algorithm.

The kernel method began to gain popularity after its introduction in the work on the Support Vector machine [32]. The kernel method allows a linear algorithm which solely relies on atomic dot-product operations to be generalised to a nonlinear algorithm by replacing the dot products with kernel functions. The beauty of the kernel method is that it is theoretically sound and the non-linearity offered comes with minimal increase of computational cost. Following the kernelisation of the Support Vector machine, kernel principal component analysis (KPCA) [32] and kernel Fisher discriminant analysis [28] were introduced and have reported impressive performance in capturing nonlinear features. In our algorithm, KPCA is used for feature extraction and dimensionality reduction.

Cluster analysis is a crucial part of the unsupervised segmentation algorithm with a purpose of estimating the number of texture classes and their characteristics. For cluster analysis, the kernel mean shift clustering algorithm which is a non-linear generalisation of the mean shift clustering algorithm [8] is introduced. The kernel mean shift clustering algorithm is a non-parametric and effectively adaptive version of its linear counterpart. Being adaptive, its representation of clusters takes the shape of clusters into account.

Classification is the final step for the segmentation algorithm where a class label is assigned to every pixel. Classification algorithms that take advantage of a collective representation in neighbourhood generally leads to better classification results. The question is how and in which domain the collective representation should be formed. For the feature
vector based image segmentation approach, forming a collective neighbourhood representation in the feature vector domain is intuitive. However, it is conjectured that forming a collective neighbourhood representation in the probability domain could be a better idea for segmentation tasks with a large number of classes. Motivated by this idea, a classification scheme functioning in the probability domain is introduced.

1.2 Outline

In Chapter 2, the unsupervised texture segmentation process is outlined and its modularity is highlighted. At the end of the chapter, the kernel method and its justification i.e. the Mercer’s condition, is reviewed to prepare for the introduction of kernel principal component analysis and kernel mean shift clustering algorithm in the later chapters.

Chapters 3 to 6 are devoted to detail discussion of the segmentation algorithm. This begins with discussing feature extraction and dimensionality reduction in Chapter 3. In this chapter, complex wavelet transform, principal component analysis and kernel principal component analysis are reviewed and the strategy for incorporating the spatial information into feature vectors is discussed.

In Chapter 4, discussion of cluster analysis is presented. It begins with reviewing the mean shift clustering algorithm and then moves on to introducing the kernel mean shift clustering algorithm before the analysis on its properties is presented. Chapter 5 also discusses the final step of the segmentation process, i.e classification. The idea of averaging in probability domain is illustrated using a simple example before the classification algorithm based on recursive update of posterior distributions is introduced. Then, the experiments and results for image segmentation are presented in Chapter 6.

The work on CBIR is presented in Chapter 7 before conclusions and future work are given in Chapter 8.
Chapter 2

Texture Segmentation and The Kernel Method

Texture can be defined [1] as a pattern composed of repetitive sub-patterns with invariant properties. The notion of a texture being a pattern makes it meaningful to be described as a region property, for example in terms of a relationship between a set of pixels. As such, Markov random fields [25], which characterise mutual influences among pixels using probabilities, can be used for texture modelling.

Texture can also be characterised based on a set of properties relating to human perception such as periodicity, directionality and randomness. Wold theory can be employed to decompose textures represented by a two-dimensional random field into three mutually orthogonal components corresponding to the above-mentioned properties [26].

Alternatively, local spatial grey-level statistics [2] can be used to model textures. Co-occurrence matrices are popular for capturing the spatial statistics from which features such as energy, entropy, contrast and so on can be extracted.

Multi-channel filtering derived from the understanding of the early processes of human vision [2], is another popular approach for texture feature extraction. Gabor filters have been widely used in this context. The complex wavelet transform began to appear in recent years and was shown to have a better directional sensitivity as compared to the standard wavelet transform [18], [10]. Furthermore, the dual-tree implementation of the complex wavelet transform has made it computational efficient [23].

The definition of texture as a region property conflicts in a sense with the task of image segmentation, in which classification is performed at the pixel level. This conflict makes texture image segmentation a challenging task.

2.1 Unsupervised Texture Segmentation Algorithm

The segmentation algorithm described in this dissertation follows the approach of statistical texture analysis based on features extracted using the complex wavelet transform. For this approach, feature vectors corresponding to a texture share common statistical properties
and form a cluster in the feature vector space.

The algorithm can be divided into 4 sequential modules:

1. **Feature extraction and dimensionality reduction**
   The goal of feature extraction is to extract features that can unambiguously distinguish the different types of textures present in an image. The complex wavelet transform extracts features that are localised in the spatial and spatial frequency domain. Eighteen subbands are obtained in the case of a three-level decomposition. Due to the correlation between the complex wavelet subbands, there is much potential for feature dimensionality reduction which results in a more compact representation. KPCA with a Gaussian radial basis function (GRBF) kernel is used for feature dimensionality reduction.

2. **Incorporation of spatial information**
   The purpose of feature vector formation is to obtain feature vectors with reinforced representation of the concept of texture. The strategy of using a collective representation for a defined neighbourhood structure is adopted. The rationale is to have a resultant representation with an improved intra-class (referring to feature vectors corresponding to a same texture) over inter-class (referring to feature vectors corresponding to different textures) discrimination. This can be viewed as a means of incorporating spatial information into the feature vectors. There are, however, drawbacks to this approach. Firstly, it risks missing out fine structures which are smaller in scale as compared to the neighbourhood structure. Secondly, representations derived from neighbourhoods that span regions with different textures could be erroneous. There is thus a trade-off between the accuracy of texture identification and the precision of their spatial localisation.

3. **Cluster Analysis**
   The task of cluster analysis is to search the feature vector space in order to estimate the number of clusters and their statistics. The kernel mean shift clustering algorithm, a non-linear generalisation of the mean shift clustering algorithm [8], is used for this purpose.

4. **Classification**
   Based on the cluster statistics obtained from cluster analysis, the classification process assigns a class label to every pixel in an image. The resultant label or segmentation map will be the output of the algorithm. Spatial information is critical in classification as it can avoid errors which would inevitably be made if a simple pixel-by-pixel classification scheme is used.
2.2 Image Segmentation and Machine Learning

Machine learning is the study of algorithms which are capable of improving their own performance through the acquisition of knowledge [24]. With this definition, image segmentation is related to machine learning in the sense that it acquires knowledge about textures from images in order to distinguish them. However, the difference is that the acquired knowledge is rarely reused for segmenting other images.

Machine learning can also be considered to be a process of generalising and transforming representations [1]. With this interpretation, a more obvious parallel can be drawn between machine learning and image segmentation, as the latter is in essence a process of *symbolisation* where every pixel in an image is mapped to a label corresponding to a texture type.

Learning generates knowledge and consequently an issue of knowledge representation arises. This is translated into an issue of cluster representation in the case of unsupervised learning based on cluster analysis. A cluster can be represented, in one end of the spectrum, by a single point corresponding to the mean or centroid or, in another end of the spectrum, by all the points belonging to the cluster. The former single-point representation loses all information about cluster distribution. On the other hand, although the latter approach captures the cluster distribution, the cluster membership for the outliers may not be certain in the first place. Furthermore, the latter approach of representing clusters using the entire data set is inefficient in terms of memory requirement.

Cluster representation has a significant effect on data classification, particularly for the ‘unseen data’ i.e. data which is not seen until the data classification stage. For the case of image segmentation, it is possible to have unseen data in the classification stage for two reasons:

1. For a multiscale classification scheme, feature vectors from a coarser scale may be used for cluster analysis, based on which those belonging to the finer scales are classified.

2. For the purpose of reducing the computational load of the clustering algorithm, probably only a subset of the total feature vectors is used; this is essentially the case for the segmentation algorithm described in this dissertation.

The ability of a classification algorithm to cope with the unseen data is an issue of *generalisation* which is commonly found in the machine learning literature.

It will be shown in Section 4.3 that the kernel mean shift clustering algorithm provides a cluster representation which captures the essential cluster distribution by using a group of points in the central region of a cluster.

2.3 The Kernel Method

An algorithm based solely on atomic dot product operations, i.e. $x \cdot y$, can be extended to capture non-linearities if the data is first mapped to a high-dimensional Hilbert space.
through a nonlinear mapping such that all the dot product operations in input space $\mathbf{x} \cdot \mathbf{y}$ are replaced by the dot product operations in feature space $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$:

$$\Phi : \mathbb{R}^N \rightarrow \mathcal{F}$$

Due to the nonlinear mapping, the original algorithm is referred as a linear algorithm, in contrast to its extension which is referred as a nonlinear algorithm. In this dissertation, the high-dimensional Hilbert space $\mathcal{F}$ is known as the feature space $^1$ and the space, in which the original data resides, is known as the input space [33]. The resultant algorithm has nonlinear properties associated with the mapping defined by $\Phi$.

If the dot product in feature space $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$ is equivalent to a nonlinear kernel function $K(\mathbf{x}, \mathbf{y})$ in input space, the mapping to feature space is not explicitly needed. Conversely, given a kernel function, which corresponds to such a nonlinear mapping, a linear dot product algorithm, can be turned into a nonlinear one by simply replacing all the dot product operations, $\mathbf{x} \cdot \mathbf{y}$, with the kernel function, even though the corresponding mapping may not be known explicitly. Examples of valid kernel functions that possess these properties are:

- Polynomial kernel
- Gaussian radial basis function kernel
- Hyperbolic tangent kernel

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^p\quad \text{Gaussian radial basis function kernel}\quad \exp(-\|\mathbf{x} - \mathbf{y}\|^2/c)\quad \text{Hyperbolic tangent kernel}\quad \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \epsilon)\quad (2.1)$$

Given a kernel function, how can one be sure that there exists a corresponding mapping? In other words, how can one be sure that when all the dot product operations $\mathbf{x} \cdot \mathbf{y}$ are substituted by a kernel function $K(\mathbf{x}, \mathbf{y})$, the algorithm is actually operating in a high-dimensional Hilbert space? The existence of the corresponding mapping $\Phi$ and feature space $\mathcal{F}$ for a kernel function is given by the Mercer’s condition [6]:

**Condition 1 Mercer’s condition**

There exists a mapping $\Phi$ and $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$ if and only if, for any $g(\mathbf{x})$ with a finite $\int g(\mathbf{x})^2 d\mathbf{x}$, the following expression is satisfied:

$$\int K(\mathbf{x}, \mathbf{y})g(\mathbf{x})g(\mathbf{y})d\mathbf{x}d\mathbf{y} \geq 0$$

The Mercer’s condition provides the existence of $\{\mathcal{F}, \Phi\}$ for a kernel $K(x, y)$ without indicating how to obtain them. However, knowing a kernel that satisfies the Mercer’s condition is sufficient to turn a linear dot product algorithm into a nonlinear one by following the procedure described above. This technique is known as the kernel method. A kernel function is valid even if the corresponding mapping ends up in an infinite dimensional Hilbert space, as in the case of the GRBF kernel. The nonlinear mapping $\Phi$ is implicitly determined by the choice of the kernel function. By using the kernel method, the operation

---

$^1$The ‘feature space’ should not be confused with the ‘feature vector space’ in which feature vectors reside.
of the algorithm remains unchanged except that it is now functioning in a high-dimensional Hilbert space.

The kernel method is a way of introducing complexities into a linear dot product algorithm. For instance, in a classification problem, a data, which is not separable for a linear Support Vector machine, may become separable when a kernelised Support Vector machine is used. Apart from this, it will be shown in Chapter 4 that the kernelised mean shift clustering algorithm displays some forms of data adaptiveness which is not found in its linear counterpart.

For a more detailed description of the kernel method, please refer to [6].

2.3.1 Input Space and Feature Space

Given a set of input data \( \{x_1, ..., x_\varrho\} \) and a nonlinear low-to-high dimensional space mapping \( \Phi \), not every point within the span of the corresponding mapped data \( \{\Phi(x_1), ..., \Phi(x_\varrho)\} \) in feature space has a ‘pre-image’ i.e. a corresponding point in input space. In order that these points do have a pre-image, there must exist a point \( z \) in input space which satisfies the following equation:

\[
\Phi(z) = \sum_{i=1}^{\varrho} \Phi(x_i) \quad \text{ where } \Phi \text{ is a nonlinear and low-to-high dimensional map} \quad (2.2)
\]

If we consider expression (2.2) as a set of simultaneous equations, then the number of equations exceeds that of the variables. Therefore, this could be an over-constrained problem and thus a solution is not guaranteed.

In this case, a solution or any intermediate terms in a kernel-based algorithm may have to be expressed using a set of points rather than just one point in input space [33]:

\[
f(y) = \sum_{i=1}^{\varrho} \alpha_i K(y, x_i) \\
= \sum_{i=1}^{\varrho} \alpha_i \Phi(y) \cdot \Phi(x_i) = \Phi(y) \cdot \left( \sum_{i=1}^{\varrho} \alpha_i \Phi(x_i) \right) \quad (2.3)
\]

where the term, \( \sum_{i=1}^{\varrho} \alpha_i \Phi(x_i) \), has no pre-image in input space and thus cannot be summarised using a simple term, \( \Phi(z) \). This is effectively equivalent to using the entire set or a subset of the input data to represent a point in feature space. The consequence of this inefficient representation is an increase in computational cost.

The Gaussian radial basis function (GRBF) is chosen for the segmentation algorithm because it was found to be better than a few other kernel functions experimented such as the polynomial kernel and biased polynomial kernel\(^2\). However, for a GRBF kernel

\(^2\)The biased polynomial kernel is given by an expression, \((x \cdot y + c)^p\), where \(c\) is a constant.
function, all points which are expressed as \( \sum_{i=1}^{\theta} \alpha_i \Phi(x_i) \) with more than one non-zero \( \alpha_i \) do not have a pre-image in input space. This is due to the fact that no single GRBF or Gaussian function in general can be expressed as a linear combination of GRBFs or Gaussian functions centred at other points [33]. This property of the GRBF kernel is responsible for the high computational cost of the GRBF kernel mean shift clustering algorithm, which will be discussed in Chapter 4.
Chapter 3

Feature Extraction and Dimensionality Reduction

It is a challenging task to devise an effective feature extraction algorithm which produces image segmentation results comparable to that of the human visual system. For an effective feature extraction algorithm, it is not just a matter of extracting sufficient features but also a matter of compact representation for the features.

In this dissertation, compact feature extraction is accomplished through two consecutive steps i.e. general feature extraction followed by feature dimensionality reduction. The complex wavelet transform [22] is used to extract features corresponding to variations in different scales and orientations while the GRBF KPCA is performed to obtain the final set of uncorrelated and compact features. As an intermediate step prior to KPCA, a nonlinear data transform is performed on the complex wavelet coefficients for data conditioning.

3.1 Complex Wavelet Transform

The wavelet transform is a multi-resolution decomposition of signals into orthogonal components which achieve dual localisation in both its original domain (e.g. time for time series and 2-D space for images) and the corresponding frequency domain (e.g. time-frequency for time series and spatial frequency for images). This decomposition is achieved by using localised orthogonal bases instead of never-ending sinusoids, as in the case of the Fourier transform.

From another perspective, the wavelet transform can also be understood as decomposing a signal through a hierarchical filter bank. In this hierarchical filter bank structure, a signal is passed through to a pair of lowpass and highpass filters. The output of the highpass filter becomes a component of the signal while the signal decomposition scheme is carried through by performing the same high-and-lowpass filtering recursively on the output of the lowpass filter. This process is shown in Figure 3.1.

As shown in Figure 3.1, there is one wavelet subband at each level for the one-dimensional wavelet transform. For two-dimensional signals like images, a high-and-
lowpass filtering is performed along the rows and columns of the signals and thus there are three wavelet subimages (i.e. two-dimensional subbands) and one lowpass subimage at each level. These subimages correspond to variations in different orientations and scales in images. Therefore, the wavelet transform can also be considered as effectively detecting sets of edges with different orientations and scales. This property makes the wavelet transform a powerful technique for texture feature extraction as textures can generally be characterised by these types of variations.

There is however a major drawback for the standard wavelet transform. It is not shift invariant in the sense that small shifts in the input signal will result in major variations in the distribution of energy between wavelet subbands at different levels. To overcome this problem, the fully undecimated wavelet transform can be used but this is not an efficient approach because it has a higher computational cost and storage requirement. The complex wavelet transform [22], which is approximately shift invariant, is another option. The term ‘complex’ highlights the fact that the coefficients are complex numbers as compared to the real coefficients of the standard wavelet transform.

Apart from being approximately shift invariant, the complex wavelet transform has the additional advantage of having better orientation selectivity. There are six complex wavelet subbands per level compared to just three per level for the standard wavelet transform. The six complex wavelet subbands are strongly orientated at angles $\pm 15^\circ$, $\pm 45^\circ$ and $\pm 75^\circ$ respectively. With this improved orientation selectivity, complex wavelet transform is capable of extracting more effective features for better texture discrimination.

Besides the wavelet transforms, the Gabor transform has also been a very popular technique for texture feature extraction. It has the advantage of having an arbitrary number of subbands with equally spaced orientation selectivity at each level but the computational cost could be considerably higher as compared to the efficient dual-tree implementation for the complex wavelet transform. On the other hand, experimental results presented in Chapter 6 show that the degree of orientation selectivity provided by the complex wavelet transform is sufficient for segmenting images with up to 16 different textures.

The output of the complex wavelet transform is 4:1 decimated down each level i.e. the size of subimages at a particular level is four times smaller than that of the previous level. To enable the formation of feature vectors, the magnitude of the complex coefficients of the subimages at the second and higher levels are interpolated to the size of the subimages at the first level. In this case, if the coefficients are upsampled rather than interpolated,
there would be a lot of discontinuities in the resultant subimages and the blocky effect would set in.

Figure 3.2 demonstrates the capability of the complex wavelet transform in extracting features corresponding to variations in different scales and orientations from a ripple-like image.

3.2 Data Transform

Prior to feature vector dimensionality reduction using KPCA, a nonlinear data transform is performed on the output of the complex wavelet transform in order to condition the data so that the resultant feature vectors lead to satisfactory segmentation results. The data transform is a series of sequential operations i.e. $(\cdot)^2$, Gaussian weighted averaging and $\log(\cdot + 1)$. This data transform was found best for unsupervised image segmentation among a set of transform experimented [41].

The Gaussian weighted averaging is given by:

$$y(m, n) = \sum_{(a,b) \in S} h(a,b)x(m-a, n-b)$$  \hspace{1cm} (3.1)

where $S$ denotes the site index within a neighbourhood. The Gaussian weight $h(a,b)$ is given by:

$$h(a, b) = ke^{-\frac{(a^2+b^2)}{2\sigma^2}}$$  \hspace{1cm} (3.2)

where the weight is normalised to ensure that the mean of the image is the same before and after the averaging operation:

$$\sum_{(a,b) \in S} h(a,b) = 1$$

The Gaussian weighted averaging in expression (3.1) can be seen as a lowpass filtering operation with a Gaussian impulse response given by expression (3.2). A Gaussian filter is chosen because it provides the optimum spatial and spatial frequency localisation. In this case, since each wavelet subband corresponds to a localisation in the spatial and spatial frequency domain, it is reasonable for the parameter $\sigma$, the standard deviation of the Gaussian filter, to be proportional to the spatial resolution of the corresponding wavelet subband at each level:

$$\sigma_s \propto \frac{1}{|\Delta f_s|} \propto \frac{1}{|\hat{f}_s|}$$

where $|\hat{f}_s|$ is the normalised magnitude of the centre frequency at level $s$ with $s=1, 2, \ldots$ and $|\hat{f}_s| \leq \frac{1}{2}$. The value $\sigma_s = \frac{1}{2\sqrt{2}\Delta f_s}$ is chosen for the algorithm. However, the performance of the algorithm was found to be rather insensitive to this parameter.
(a) A grayscale image with variation of different scales and orientations

(b) The magnitude of the complex subband coefficients for a four-level complex wavelet decomposition of the above image. The pairs of numbers represents the decomposition level and the subband index respectively.

Figure 3.2: Demonstration of feature extraction using complex wavelet transform
The Gaussian weighted averaging is a means of forming a collective representation of a neighbourhood in the feature vector domain. This operation is in line with the idea that texture is a region property. This principle can be justified using the central limit theorem. Since feature vectors within a neighbourhood are usually from the same cluster, the averaging operation reduces the variance of clusters without much modification to their mean. As a result, the clusters become more separable as outliers are removed.

There is however a drawback in the averaging operation. Without any knowledge about the location of boundaries between adjacent texture regions, the averaging operation at neighbourhoods which span the boundaries inadvertently combines feature vectors of different textures. The resultant ‘averaged’ feature vectors may be erroneous in the sense that they end up having characteristics resembling another non-constituent texture but none of the constituent textures. This is a common source of error for image segmentation. Therefore, the weighted averaging operation is essentially a trade-off between localisation of boundaries and correct identification of textures.

The knowledge about the distribution of the complex wavelet coefficients could provide an insight into the effect of the data transform. It is shown in [19] and Appendix C that the distribution of the magnitude of the complex wavelet coefficients for a texture can be modelled as a generalised Rayleigh distribution as given by the following expression:

\[ f(x) = kxe^{-\left(\frac{x}{\sigma}\right)^\beta} \]

where

- \( \beta = F^{-1}\left(\frac{m_1^2}{m_2}\right) \) with \( m_1 = \frac{1}{N} \sum_{i=1}^{N} x_i \), \( m_2 = \frac{1}{N} \sum_{i=1}^{N} x_i^2 \) and \( F(\cdot) = \frac{\Gamma\left(\frac{3}{\beta}\right)}{\Gamma\left(\frac{2}{\beta}\right)\Gamma\left(\frac{4}{\beta}\right)} \)
- \( \sigma = m_1 \frac{\Gamma\left(\frac{3}{\beta}\right)}{\Gamma\left(\frac{2}{\beta}\right)} \)
- \( k = \frac{\beta}{\sigma^2 \Gamma\left(\frac{3}{\beta}\right)} \)

An example of the distributions for a tiger texture is shown in Figure 3.3. The figure shows that a generalised Rayleigh distribution is generally long-tailed and with most of the data concentrated in a lobe. Therefore, for images with more than one texture, the distribution for each subband, being a combination of several generalised Rayleigh distributions, is possibly multimodal and certainly long-tailed. The \( (\cdot)^2 \) and the Gaussian weighted averaging operations do not have much effect on the long tail but the \( \log(\cdot + 1) \) operation reduces the long tail while maintaining an almost linear scaling for most of the data residing in the main lobes of the distributions. The result is a set of compact distributions which lead to a more confined data set in the feature vector space.

### 3.3 Principal Component Analysis

Each subband of the complex wavelet transform of an image can be considered a feature for the texture segmentation task. In this case, the standard deviation of a subband provides a
clue to its usefulness in discriminating textures. Features with higher standard deviations are more likely to be useful in discriminating textures. Conversely, features with very low standard deviations, i.e. having almost the same values for all their coefficients, are unlikely to be useful and can be omitted without much implication. This is basically the general principle for feature dimensionality reduction.

When there is room for reduction of dimensionality, this implies that the set of feature vectors has a lower intrinsic dimensionality. Since data with a lower intrinsic dimensionality can be embedded in a higher-dimensional space, it is reasonable that the feature vectors derived from the twenty four or eighteen subbands of the complex wavelet transform can be reduced in dimensionality and approximated with a more compact representation.

Due to the correlation between the complex wavelet subbands, the feature vectors should be transformed into a representation with uncorrelated components, in order to facilitate dimensionality reduction. Principal component analysis (PCA) is precisely the orthogonal transform that provides such a representation. The resultant components or transformed features are known as principal components and have the following properties:

- The first $n$ principal components with descending order of variance contain more variance than those derived from any other orthogonal transform of axis. This property is shown in Appendix A.
- The mean-squared approximation error in representing the set of data by the first $n$ principal components is the minimum.
- The principal components are uncorrelated.

Figure 3.3: Distributions of the magnitude of the complex wavelet coefficients
In this section, the concept of PCA is reviewed. More discussions can be found in [13].

Given a set of data:

\[ S = \{ x_i | x_i \in \mathbb{R}^N, i = 1, 2, ..., M \} \]

It can be linearly transformed by the following:

\[ y_i = Mx_i \]

The covariance of the data \( x \) can be expressed as:

\[ C_x = \frac{1}{M} \sum_{i=1}^{M} (x_i - \bar{x})(x_i - \bar{x})^T \quad \text{where} \quad \bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i \text{ is the mean of the data} \]

Then, the covariance of transformed data \( y \) would be:

\[
\begin{align*}
C_y &= \frac{1}{M} \sum_{i=1}^{M} (Mx_i - M\bar{x})(Mx_i - M\bar{x})^T \\
&= M \left( \frac{1}{M} \sum_{i=1}^{M} (x_i - \bar{x})(x_i - \bar{x})^T \right) M^T \\
&= MC_x M^T
\end{align*}
\]

If the linear transformation leads to a set of data with uncorrelated components, then \( C_y \) would be a diagonal matrix \(^1\).

\[
C_y = \begin{bmatrix}
\lambda_1 & 0 \\
& \ddots \\
0 & \lambda_N
\end{bmatrix}
\]

With this condition, the transformation matrix \( M \) can be determined by solving the eigenvalue problem with \( M^T \) being the eigenvector matrix where the columns are the eigenvectors. Since the covariance matrix \( C_x \) is a hermitian matrix, the eigenvector matrix \( M^T \) is orthonormal and hence satisfies the condition \( M^{-1} = M^T \). Thus:

\[
\begin{align*}
C_x M^T &= M^T C_y \\
C_x v_j &= \lambda_j v_j
\end{align*}
\]

In this case, \( v_j \) is the \( j \)th eigenvector of the covariance matrix \( C_x \) and \( \lambda_j \) is its corresponding eigenvalue. Assuming that the eigenvalues are ordered in such a way that

\(^1\)An alternative derivation of PCA through maximising the sum of the variance of the principal components is shown in Appendix A.
$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N$, the $j$th principal component can then be obtained by projecting the ‘centred version’ of the set $S$ onto the axis denoted by the $j$th eigenvector, $v_j$:

$$y_{ij} = v_j^T(x_i - \bar{x}) \quad (3.3)$$

PCA is an unsupervised algorithm in the sense that the algorithm has no prior knowledge about features that will be suitable for the requirements of a particular application. PCA assumes that important features are those with large variances. This is in contrast to supervised algorithms where the number of classes is known and the data is labelled e.g. Fisher discriminant analysis is optimised to identify the best features for discriminating data belonging to a particular number of classes.

Figure 3.4 shows the principal components for a two-dimensional data set$^2$. These principal components correspond to projection of data onto two linear axes which are orthogonal to each other. There are only two linear principal components possible for a two-dimensional data set. The axis for the first principal component is aligned in a direction where the corresponding projection has the largest collective variance while that of the second principal component corresponds to the remaining subspace.

\[
\tilde{x}_i = x_i - \bar{x}
\]

Note: The contour represents the magnitude of the PCA components.

Figure 3.4: PCA components for a two-dimensional toy data.

It should be noted that eigenvectors are actually within the span of the centred data set defined as:

$\tilde{x}_i = x_i - \bar{x}$

$^2$Compliments to B. Schölkopf for his matlab code that generates contour maps for both PCA and KPCA. The codes was obtained from http://www.kernel-machines.org/software.html
\[ \lambda_j v_j = C_F v_j \]
\[ = \left( \frac{1}{M} \sum_{i=1}^{M} (x_i - \bar{x})(x_i - \bar{x})^T \right) v_j \]
\[ = \left( \frac{1}{M} \sum_{i=1}^{M} \tilde{x}_i \tilde{x}_i^T \right) v_j \]
\[ v_j = \frac{1}{\lambda_j M} \sum_{i=1}^{M} (\tilde{x}_i^T v_j) \tilde{x}_i \quad (3.4) \]

3.4 Kernel Principal Component Analysis

In this section, a non-linear generalisation of the principal component analysis, namely the kernel principal component analysis (KPCA) [35] is reviewed.

As shown in Section 2.3, for any kernel function \( K(\cdot, \cdot) \) that fulfils the Mercer’s condition, there exists a Hilbert space \( F \) and a corresponding mapping \( \Phi \):

\[ \Phi : \mathbb{R}^N \rightarrow F \]

Through the mapping, the image of the data set in feature space is given by:

\[ X_i = \Phi(x_i) \]

and the centred data set in feature space is denoted by:

\[ \tilde{X}_i = X_i - \frac{1}{M} \sum_{k=1}^{M} X_k \]

Then, the covariance matrix for the data set in feature space can be expressed as the following:

\[ C_F = \frac{1}{M} \sum_{i=1}^{M} \tilde{X}_i \tilde{X}_i^T \]

KPCA is equivalent to performing PCA in feature space and therefore it leads to solving an eigenvalue problem in feature space which can be formulated as below:

\[ \lambda_j V_j = C_F V_j \quad (3.5) \]

Similar to the linear PCA, centred data in feature space is projected to the ordered eigenvectors to generate principal components. For instance, for a input space data point \( z \) which may not be part of the data set, kernel principal components are generated as in the following expression:
where

- $V_j$ is the $j$th eigenvector with the $j$th largest eigenvalue.
- $\tilde{Z}$ is a centred feature space data point for $z$ i.e. $\tilde{Z} = \Phi(z) - \frac{1}{M} \sum_{i=1}^{M} X_i$

The eigenvalue problem of expression (3.5) may be solved indirectly and equivalently as shown in Appendix B. As a result, expression (3.6) can be rewritten in the following form:

$$ y_j = \frac{1}{\sqrt{\gamma_j}} \tilde{K}^{proj} \alpha^j $$

where

- $\alpha^j$ and $\gamma^j$ are respectively the $j$th eigenvector and its corresponding eigenvalue for the equivalent eigenvalue problem (3.7). In this case, the eigenvalues are in descending order.

$$ \gamma^j \alpha^j = \tilde{K} \alpha^j $$ (3.7)

The matrix $\tilde{K}$ is given by the following expression:

$$ \tilde{K} = K - KM - MK + MKM $$

with

- $K$ is a square matrix of dimension $M$ with its elements $K_{ab}$ being $K(x_a, x_b)$
- $M$ is a square matrix of dimension $M$ with all its elements being $1/M$

- $\tilde{K}^{proj}$ is given by expression (3.8):

$$ \tilde{K}^{proj} = K^{proj} - K^{proj}M - M^{proj}K + M^{proj}KM $$ (3.8)

with

- $K^{proj}$ is a $1 \times M$ matrix with its elements $K^{proj}_i$ being $K(x_i, z)$
- $M^{proj}$ is a $1 \times M$ matrix with all its elements being $1/M$
3.5 KPCA for Reduction of Feature Dimensionality

As shown in the previous section, KPCA can generate up to $M$ kernel principal components where $M$ is the size of the data set. In cases where the data size is larger than the input data dimensionality, there are more kernel principal components than it is possible with PCA. This means that KPCA can extract more features than it is possible with PCA. The extra nonlinear features are essentially due to the nonlinear mapping to feature space performed implicitly in the kernel method. With a GRBF kernel, the nonlinear features correspond to the localised structure of the clusters. As shown in Figure 3.5, the GRBF KPCA could generate principal components which correspond to isolation of individual clusters.

As the feature vectors, derived from the complex wavelet subbands, could be reduced in dimensionality, the question would be “how many principal components should the algorithm retain?”.

Figure 3.5 shows the GRBF kernel principal components for a two-dimensional data set with comparable standard deviations for both the GRBF and the clusters. The first two kernel principal components are shown to correspond to features that discriminate one cluster from another. Thus, for this data set containing three clusters, the first two kernel principal components are sufficient for distinguishing all three clusters from each other. The subsequent kernel principal components correspond to the split of individual clusters. This second-type principal components are not desirable for image segmentation as it leads to over-segmentation and should be avoided. Therefore, in order to decide the number of principal components worth retaining, the algorithm needs to know the number of first-type principal components, which is closely associated with the total number of the clusters. However, being unsupervised, the algorithm could do no better than deciding heuristically the number of the principal components to retain.

![Figure 3.5: GRBF KPCA components with the parameter value comparable to the variance of the clusters](image)

Note: The parameter $c$ for the GRBF kernel assumes value 0.1.

Figure 3.5: GRBF KPCA components with the parameter value comparable to the variance of the clusters

To alleviate the problem, we choose to use a larger value for $c$ (relative to the size of the clusters). As shown in Figure 3.6, the first two kernel principal components are sufficient
for discriminating the three clusters. The following kernel principal components may still correspond to cluster splitting but the effect is relatively harmless in the sense that the clusters are all on the same contours and hence the resultant principal components are of relatively uniform values.

From the experiments in Chapter 6, it is found that three GRBF KPCA components are sufficient for segmentation of images with up to seven different textures while four is good for those with up to sixteen different textures.

Note: The parameter $c$ for the GRBF kernel assumes value 0.8.

![Figure 3.6: GRBF KPCA components with the parameter value relatively larger than the variance of the clusters](image)

**3.6 Incorporation of Spatial Information**

Forming feature vectors that are effective in representing the underlying texture is crucial for texture segmentation. With such a representation, the resultant feature vectors will have a better intra-class over inter-class discrimination. To achieve this, the derivation of feature vectors should reflect the fact that the concept of texture is region-based. With this principle in mind, the following procedure for incorporation of spatial information is proposed.

Gaussian weighted averaging on a 3x3 block neighbourhood structure is first performed on each of the principal components and then vectors within a 3x3 block neighbourhood are concatenated to form the final feature vectors. The purpose for the Gaussian weighted averaging is to reduce noise in the feature vectors while the concatenation of feature vectors is a way of incorporating spatial information.

The neighbouring original feature vectors are strongly correlated as textures usually form contiguous regions in an image and are not distributed randomly. Due to the correlation, redundancy in representation is introduced when neighbouring original feature vectors are concatenated. Although redundancy in representation results in a greater computational cost but it could lead to more separable clusters.

The idea of concatenating correlated vectors is very much similar to the principle behind a type of communication channel coding where signal bits are repeated before being
transmitted through a communication channel. This process takes advantage of the introduced redundancy in order to provide a better separation between different symbols in the symbol space so that the communication is more resilient to channel noise. This parallel can be explored further by considering communication coding theory.

For image segmentation, apart from concatenation of feature vectors, there are other ways of incorporating spatial information:

- **Using normalised x-y coordinates as extra features**: This technique is used in [7] but has a disadvantage that it may result in fragmentation of elongated objects.

- **Weighted mean of vectors in a neighbourhood**: The drawback is that it may result in severe loss of fine details and classification errors being made at the boundaries.

A very important consequence of the concatenation of feature vectors is that the resultant feature vectors become high-dimensional. The geometry of high-dimensional spaces are very different from that of low-dimensional spaces with which we are intuitively familiar. The empty space phenomenon [36], where a limited number of data points becomes sparse in a high-dimensional space, could set in. However, due to the correlation among the neighbouring feature vectors, the resultant high-dimensional feature vectors might still maintain a low intrinsic dimensionality. Therefore, the issue of high-dimensionality is not significant here.

Similar to the approach of feature vector averaging, concatenated feature vectors derived from neighbourhoods which span regions with different textures could be potential sources of errors. However, the characteristics of the erroneous vectors resulting from the two approaches are different. Figure 3.7 illustrates this difference using a simple example. For this example, a neighbourhood is defined as side by side data points. From the figure, we notice that feature vector concatenation for neighbourhoods on the boundaries produces outliers, whereas feature vector averaging produces samples which may coincide with valid clusters as for the case of the region A erroneous data in Figure 3.7(c). This may have a negative implication for the classification process in the later stage.

Despite that the resultant outliers in region A seem to be nearer to the cluster corresponding to region C, as shown in Figure 3.7(d), distance measures used for classification are not normally linear. For instance, for the Gaussian distance measure, the outliers may appear to be of an equal distance from all the clusters. In this case, if the spatial information is taken into account, the outliers may be correctly classified to the cluster corresponding to region A. This issue will be discussed further in Chapter 5.

On the other hand, Figure 3.7(d) also shows that the concatenated vectors have a lower intrinsic dimensionality as the clusters are aligned along a straight line (i.e. $y = x$ in this example). In this case, if the two-dimensional concatenated vectors were to be reduced in dimensionality by projecting it onto the axis, $y = x$, the errors incurred could be quite insignificant.
(a) An image with 3 regions composed of random samples from 3 separate Gaussian distributions. (Note that the A, B and C are not referring to the texture type but the regions)

(b)Histograms for Gaussian random samples in regions A, B and C.
(c) Distributions of data sample for regions A, B and C after feature vector averaging.

(d) Distributions of data points for regions A, B and C after feature vector concatenation.

Figure 3.7: Comparing feature vector averaging and feature vector concatenation
Cluster analysis is closely related to density estimation because clusters can be considered as the modes of the estimated density. Therefore, in theory, clustering algorithms can be derived directly from density estimation algorithms.

However, in practice, not all modes of the estimated density correspond to clusters. The non-cluster modes can be attributed to noise inherent to data, inaccurate density estimation due to sparseness of data in the low density regions, as well as the type of techniques employed for density estimation and the chosen parameters. As a result, to devise a robust clustering algorithm the issue of non-cluster modes has to be resolved.

4.1 Kernel Density Estimation

Given a set of data $S_I = \{x_i | i = 1, 2..., N \}$ in a $d$-dimensional space, the kernel density estimator is given by [36]:

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{N} K \left( \frac{x - x_i}{h} \right)$$

where $h$ is the window width of the kernel. This expression can be understood as a sum of the kernel functions centred at each of the observations, $x_i$. The kernel function $K(x)$ satisfies

$$\int_{x \in \mathbb{R}^d} K(x)dx = 1$$

and its value decreases as $|x|$ increases. Hence, $g(x) = K \left( \frac{x - x_i}{h} \right)$ can be seen as a ‘hyper-bump’ centred on $x_i$ in the $d$-dimensional space.

The multivariate Epanechnikov kernel [36] is one of the valid kernel functions:

$$K_E(y) = \begin{cases} \frac{1}{2}c_d^{-1}(d + 2)(1 - y \cdot y) & y \cdot y < 1 \\ 0 & \text{otherwise} \end{cases}$$
where \( c_d = \frac{(\pi)^{d/2}}{\Gamma(d/2)} \) is the volume of the unit radius \( d \)-dimensional hypersphere and \( \Gamma(\cdot) \) is the gamma function with \( \Gamma(\frac{1}{2}) = \sqrt{\pi} \). From the definition, \( g(x) = K_E \left( \frac{x - x_i}{h} \right) \) is non-zero only within the hypersphere of radius \( h \) and centred on \( x_i \).

Figure 4.1 shows a plot of an one-dimensional Epanechnikov kernel.

![Figure 4.1: A plot of an one-dimensional Epanechnikov kernel](image)

### 4.2 Mean Shift Clustering Algorithm

From the Epanechnikov kernel density estimator

\[
\hat{f}_E(x) = \frac{1}{nh^d} \sum_{i=1}^{N} K_E \left( \frac{x - x_i}{h} \right)
\]

we can search the modes of the estimated density \( \hat{f}_E(x) \) based on the method of gradient descent where the update of the search steps can be expressed in the following form:

\[
\Delta x \propto \nabla \hat{f}_E(x)
\]

The term, \( \nabla \hat{f}_E(x) \), the gradient of the Epanechnikov kernel density estimator is given by [36]:

\[
\nabla \hat{f}_E(x) = \left( \frac{N_{\text{inside}}(x)}{Nh^d c_d} \right) \left( \frac{d + 2}{h^2} \right) \left[ \frac{1}{N_{\text{inside}}(x)} \sum_{x_i \in S_h(x)} (x_i - x) \right]
\]

\[
= \hat{f}(x) \left( \frac{d + 2}{h^2} \right) \left[ \frac{1}{N_{\text{inside}}(x)} \sum_{x_i \in S_h(x)} (x_i - x) \right]
\]

where
\[ S_h(x) = \{ y \mid \| y - x \| < h, y \in S_f \} \] is a set of data points confined within a hypersphere of radius \( h \) and centred on \( x \). This definition gives rise to a visual picture where the set of points are captured by a mean shift kernel which is a hypersphere.

\[ N_{\text{inside}}(x) = \#S_h(x) \] is the number of elements in \( S_h(x) \).

\[ \hat{f}(x) = \frac{N_{\text{inside}}(x)}{Nh^d c_d} \] is an estimated density based on the number of data points within \( S_h(x) \).

Due to the fixed window width of the Epanechnikov kernel, the estimated kernel density is expected to be unduly rough in the low density regions. Therefore, search based on a pure gradient descent method would return many local maxima, which do not correspond to clusters, in the low density regions.

To be more robust, an adaptive gradient descent update step, the mean shift \( M_h(x) \) is defined:

\[
M_h(x) = \frac{1}{N_{\text{inside}}(x)} \sum_{x_i \in S_h(x)} (x_i - x)
= \left( \frac{d + 2}{h^{d+2} c_d} \right) \frac{\nabla \hat{f}_E(x)}{\hat{f}(x)}
\tag{4.1}
\]

The mean shift update step as in expression (4.1) is compensated by the estimated density \( \hat{f}(x) \), in such way that it becomes better in escaping the non-cluster modes in the low density regions. For mean shift iterations, the update of a search point is given by:

\[
x_{\text{new}} = x_{\text{old}} + M_h(x_{\text{old}})
= x_{\text{old}} + \frac{1}{N_{\text{inside}}(x_{\text{old}})} \sum_{x_i \in S_h(x_{\text{old}})} (x_i - x_{\text{old}})
= x_{\text{old}} + \left( \frac{1}{N_{\text{inside}}(x_{\text{old}})} \sum_{x_i \in S_h(x_{\text{old}})} x_i \right) - x_{\text{old}}
= \frac{1}{N_{\text{inside}}(x_{\text{old}})} \sum_{x_i \in S_h(x_{\text{old}})} x_i
\tag{4.2}
\]

Expression (4.2) shows [12] that the updated search point, \( x_{\text{new}} \), is essentially the mean or the centroid of a ball of points captured by a mean shift kernel centred on \( x_{\text{old}} \) as the term ‘mean shift’ implies. It also provides a mechanical view of mean shift iterations in which each mean shift iteration is essentially a sequential shift of a mean shift kernel of radius \( h \) in the data space, orientated by the location of the centroid for the ball of points captured by the mean shift kernel.

\footnote{It is called a ball of points because these points are captured by a mean shift kernel which is a hypersphere.}
As mentioned earlier, a robust clustering algorithm should address issues such as how to select initial points to begin the search and how to remove non-cluster modes. One of the algorithms proposed in [8] is outlined as below:

1. **Select a value for the radius of the mean shift kernel \( h \)**

2. **Sampling initial points:**
   Vectors from the data set \( \mathbf{x}_i \in \mathcal{S}_I \) are examined one by one and those that meet the following conditions are sampled as initial points:
   
   - \( N_{\text{inside}}(\mathbf{x}_i) > T_1 \). This is to avoid initial points in the sparse regions.
   - \( |\mathbf{x}_i - \mathbf{y}| \geq h \) for all \( \mathbf{y} \in \mathcal{S}_{\text{sample}} \) where \( \mathcal{S}_{\text{sample}} \) is the set of initial points already sampled so far.

   This step can be repeated to ensure that there are sufficient initial points for the algorithm to work reliably.

3. **Performing mean shift iterations:**
   Every \( \mathbf{x}_{si} \in \mathcal{S}_{\text{sample}} \) will climb to a stopping point \( \mathbf{y}_{i}^{\text{stop}} \) when the stopping condition \( \| M_h(\mathbf{y}_{i}^{\text{stop}}) \| < \epsilon \) is satisfied with a predefined threshold \( \epsilon \).

4. **Combining nearby stopping points to form cluster centres:**
   The distance between every pair of the stopping points, e.g. \( \mathbf{y}_{n}^{\text{stop}} \) and \( \mathbf{y}_{m}^{\text{stop}} \), is evaluated and the points are merged into \( (\mathbf{y}_{n}^{\text{stop}} + \mathbf{y}_{m}^{\text{stop}})/2 \) if \( \| \mathbf{y}_{n}^{\text{stop}} - \mathbf{y}_{m}^{\text{stop}} \| < h \)

5. **Validating cluster centres:**
   The depth of the valley between two cluster centres, e.g. \( \mathbf{C}_n \) and \( \mathbf{C}_m \), is evaluated by hopping a mean shift kernel along the line connecting \( \mathbf{C}_n \) and \( \mathbf{C}_m \). The depth of the valley is measured based on the number of data points captured by the mean shift kernel at a hop, \( N_{\text{valley}} \)
   
   - If \( N_{\text{valley}}/ \min \left[ N_{\text{inside}}(\mathbf{C}_n), N_{\text{inside}}(\mathbf{C}_m) \right] > T_2 \), then the cluster centre, either \( \mathbf{C}_n \) or \( \mathbf{C}_m \), whichever is located at the sparser region, is removed.

### 4.3 The Kernel Mean Shift Clustering Algorithm

Assume that there exists another Hilbert space which is related to input space by a mapping \( \Phi \): 

\[
\Phi : \mathbb{R}^N \rightarrow \mathcal{F}
\]

Given a data set \( \mathcal{S}_I = \{ \mathbf{x}_i|i = 1, 2...N \} \), the image of the data set in feature space is given by \( \mathcal{S}_F = \{ \Phi(\mathbf{x})|\mathbf{x} \in \mathcal{S}_I \} \). For kernel mean shift iterations in feature space, given a starting point \( \mathbf{X}^{(0)} \) with \( \mathbf{X}^{(0)} \in \mathcal{S}_F \), the search point at the \( (m+1) \)th iteration is given by:
\[ X^{(m+1)} = \frac{1}{N^{(m)}} \sum_{X_i \in S_{hF}^{(m)}} X_i \]  
\[ \text{where } X^{(m+1)} \in F \text{ and both } N^{(m)} \text{ and } S_{hF}^{(m)} \text{ are defined as the followings:} \]

\[ S_{hF}^{(m)} = \left\{ Y \mid \left\| Y - X^{(m)} \right\|^2 < h^2, Y \in S_F \right\} \]

\[ = \left\{ Y \mid \left( Y \cdot Y - 2Y \cdot X^{(m)} + X^{(m)} \cdot X^{(m)} \right) < h^2, Y \in S_F \right\} \]

\[ N^{(m)} = \#S_{hF}^{(m)} \]

From definition (4.4), \( S_{hF}^{(m)} \) is a subset of \( S_F \). As a result, from expression (4.3), it is clear that all of the search points lie in the linear span of the set \( S_F \). Since a search point may not have a pre-image, to allow tractability of the definition (4.4) in input space, we express definition (4.4) in terms of the elements in the set \( S_F \) by substituting \( X^{(m)} \) with the following expression:

\[ X^{(m)} = \frac{1}{N^{(m-1)}} \sum_{X_i \in S_{hF}^{(m-1)}} X_i \]

Then, definition (4.4) becomes:

\[ S_{hF}^{(m)} = \left\{ Y \left| \left( Y \cdot Y - \frac{2}{N^{(m-1)}} \sum_{X_i \in S_{hF}^{(m-1)}} Y \cdot X_i \right) \right. \right. \]

\[ \left. \left. + \frac{1}{[N^{(m-1)}]^2} \sum_{X_i, X_j \in S_{hF}^{(m-1)}} X_i \cdot X_j \right) < h^2, Y \in S_F \right\} \]

\[ \text{for } m \geq 1 \quad (4.6) \]

and

\[ S_{hF}^{(0)} = \left\{ Y \left| \left( Y \cdot Y - 2Y \cdot X^{(0)} + X^{(0)} \cdot X^{(0)} \right) \right. \right. \]

\[ \left. \left. < h^2, Y, X^{(0)} \in S_F \right\} \]

\[ \text{Defining the kernel function,} \]

\[ K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \quad x_i, x_j \in S_I \]

we can derive the input space counterpart of sets (4.6) and (4.7):
\[ S^{(m)}_h = \left\{ y \left| \left( K(y, y) - \frac{2}{N^{(m-1)}} \sum_{x_i \in S^{(m-1)}_h} K(x_i, y) \right) + \frac{1}{[N^{(m-1)}]^2} \sum_{x_i, x_j \in S^{(m-1)}_h} K(x_i, x_j) < h^2, y \in S_I \right\} \right\} \text{ for } m \geq 1 \quad (4.8) \]

and

\[ S^{(0)}_h = \left\{ y \left| \left( K(y, y) - 2K(x^{(0)}, y) + K(x^{(0)}, x^{(0)}) \right) < h^2; y, x^{(0)} \in S_I \right\} \right\} \quad (4.9) \]

Sets (4.8) and (4.9) are just the pre-image of sets (4.6) and (4.7) respectively. They are related through:

\[ S^{(m)}_{hF} = \{ \Phi(y) | y \in S^{(m)}_h \} \quad \text{for } m \geq 0 \quad (4.10) \]

A set \( S^{(m)}_{hF} \) contains a ball of points in feature space. Definition (4.6) shows how a ball of points \( S^{(m)}_{hF} \) is derived from its previous ball of points \( S^{(m-1)}_{hF} \) in feature space. Expression (4.8) shows the reflection of the feature space mean shift iterations onto input space. This can be understood through the mechanical view of the mean shift iterations as mentioned in the previous section.

All elements in a ball of points \( S^{(m)}_{hF} \) in feature space have corresponding pre-images in input space which are the elements of \( S^{(m)}_h \). However, the centroid of a mean shift kernel in feature space \( X^{(m)} \), as in expression (4.5), may not have a pre-image in input space and this will be the case for the GRBF kernel. Therefore, for GRBF kernel mean shift iterations, the centroid \( X^{(m)} \) has to be represented by a ball of points \( S^{(m-1)}_{hF} \) or equivalently \( S^{(m-1)}_h \).

To simplify the notation, the ball of points in input space, which represents a feature space point \( X_f \), is here on denoted by \( B(X_f) \). With this notation, the distance between two points \( X_m \) and \( X_n \) in feature space can be expressed as:

\[ \|X_m - X_n\|^2 = (X_m - X_n) \cdot (X_m - X_n) = X_m \cdot X_m - 2X_m \cdot X_n + X_n \cdot X_n \]

\[ = \frac{1}{[#B(X_m)]^2} \sum_{x_i, x_j \in B(X_m)} K(x_i, x_j) - \frac{2}{#B(X_m) #B(X_n)} \sum_{x_j \in B(X_n)} \sum_{x_i \in B(X_m)} K(x_i, x_j) \]

\[ + \frac{1}{[#B(X_n)]^2} \sum_{x_i, x_j \in B(X_n)} K(x_i, x_j) \]

The kernel mean shift clustering algorithm, which is based on the algorithm described in Section 4.2, is outlined below:
1. **Selecting a value for the radius of the mean shift kernel:**

Given a data set \( S_I = \{x_i | i = 1, 2 \ldots N\} \), select a value, \( h \), for the radius of the mean shift kernel.

2. **Sampling initial points:**

Vectors \( x_i \in S_I \) are examined one by one and those that meet the following conditions are sampled as initial points:

- \( \# \{ y | (K(y, y) - 2K(x_i, y) + K(x_i, x_i)) < h^2, y \in S_I \} > T_1 \). This is to avoid initial points in the sparse regions.
- \( [K(y, y) - 2K(x_i, y) + K(x_i, x_i)] > h^2 \) for all \( y \in S_{\text{sample}} \) where \( S_{\text{sample}} \) is the set of initial points already sampled so far. This condition ensures that initial points are not close together.

This step can be repeated to ensure that there are sufficient initial points for the algorithm to work reliably.

3. **Performing mean shift iterations in feature space:**

Let’s denote the initial points in feature space as \( X_i^{(0)} \) where \( X_i^{(0)} = \Phi(x_{si}) \) with \( x_{si} \in S_{\text{sample}} \). With mean shift iterations, every initial point will lead to a stopping point \( X_i^{\text{stop}} = X_i^{(m_i)} \) after \( m_i \) iterations. The stopping condition is given by \( \| M_h(X_i^{\text{stop}}) \| < \epsilon \), where \( M_h \) is the mean shift vector and the threshold \( \epsilon \) is predefined.

Since

\[
M_h(X_i^{\text{stop}}) = \frac{1}{N^{(m_i)}} \sum_{x_i \in S_{hF}^{(m_i)}} (X_i - X_i^{(m_i)})
\]

then

\[
\| M_h(X_i^{\text{stop}}) \| = \| X_i^{(m_i+1)} - X_i^{(m_i)} \|
\]

4. **Combining nearby stopping points to form cluster centres in feature space:**

The distance between every pair of stopping points, e.g. \( X_i^{\text{stop}} \) and \( X_j^{\text{stop}} \), is evaluated and the pair of points are merged into \( X_{\text{merge}} \) provided \( \| X_i^{\text{stop}} - X_j^{\text{stop}} \| < h \):

\[
X_{\text{merge}} = \frac{1}{2} [X_i^{\text{stop}} + X_j^{\text{stop}}]
\]

\[
= \frac{1}{2} \left[ \frac{1}{\#B(X_i^{\text{stop}})} \sum_{x_n \in B(X_i^{\text{stop}})} \Phi(x_n) + \frac{1}{\#B(X_j^{\text{stop}})} \sum_{x_m \in B(X_j^{\text{stop}})} \Phi(x_m) \right]
\]
which can be expressed in a general form:

\[ X_{\text{merge}} = \sum_{p=1}^{N} \gamma_p \Phi(x_p) \]

To simplify the representation, a mean shift step from \( X_{\text{merge}} \) is taken and this leads to a new point \( X'_{\text{merge}} \) with its representative ball of points in input space being:

\[
B(X'_{\text{merge}}) = \left\{ y \left| K(y, y) - 2 \sum_{p=1}^{N} \gamma_p K(y, x_p) + \sum_{p=1}^{N} \sum_{q=1}^{N} \gamma_p \gamma_q K(x_p, x_q) \right| < h^2, y \in S_I \right\}
\]

Assuming that \( X_{\text{merge}} \) and \( X'_{\text{merge}} \) are close to each other, \( X_{\text{merge}} \) can be represented by \( X'_{\text{merge}} \).

5. Validating the cluster centres:

The previous step of combining the stopping points derives a set of cluster centres \( C_1, C_2, ..., C_P \). To test the depth of the valley between two cluster centres \( C_n \) and \( C_m \), a mean shift kernel of radius \( h \) hops through the line connecting \( C_n \) and \( C_m \) in a number of steps. The depth of the valley is evaluated by the number of data points captured by the mean shift kernel at every hop, \( N_{\text{valley}} \).

- If \( N_{\text{valley}} / \min [\#B(C_n), \#B(C_m)] > T_2 \), then the cluster centre, either \( C_n \) or \( C_m \), whichever is located in the sparser region, is removed. The purpose is to remove false clusters which are merely little bumps on the side of larger clusters.

If there are \( k \) hops between \( C_n \) and \( C_m \), then the \( i \)th hop is given by:

\[
H_i = \frac{i}{k+1}(C_n - C_m) + C_m, \quad 1 \leq i \leq k
\]

\[
= \frac{i}{k+1}C_n + (1 - \frac{i}{k+1})C_m
\]

Similarly, \( H_i \) can be expressed in a general form:

\[
H_i = \sum_{p=1}^{N} \gamma_p \Phi(x_p)
\]

Then,

\[
N_{i_{\text{valley}}} = \# \left\{ y \left| K(y, y) - 2 \sum_{p=1}^{N} \gamma_p K(y, x_p) + \sum_{p=1}^{N} \sum_{q=1}^{N} \gamma_p \gamma_q K(x_p, x_q) \right| < h^2, y \in S_I \right\}
\]
4.4 Properties of The Kernel Mean Shift Clustering Algorithm

By using kernel functions that satisfy Mercer’s condition, performing the kernel mean shift clustering algorithm in input space is equivalent to performing the linear mean shift clustering algorithm in feature space. Since the explicit mapping may not be known, it is very difficult to probe what is happening in feature space. In the following analysis, an attempt is made to examine what happens in input space instead of probing feature space directly. From there, the properties of the kernel mean shift clustering algorithm are derived.

The mean shift clustering algorithm is a distance-based algorithm. Hence, the mean shift clustering algorithms with different distance measures behave differently. Realising this fact, the inherent distance measure of the kernel mean shift clustering algorithm is examined.

The distance measure of a point \( \Phi(y) \) from another point \( X_0 \) in feature space is given by the following expression:

\[
\begin{align*}
    d(y)^2 &= K(y, y) - \frac{2}{\#B(X_0)} \sum_{x_i \in B(X_0)} K(x_i, y) + \frac{1}{[\#B(X_0)]^2} \sum_{x_i, x_j \in B(X_0)} K(x_i, x_j) \\
    &= \left( 1 - \frac{2}{\#B(X_0)} \sum_{x_i \in B(X_0)} e^{\frac{|x_i-y|^2}{c}} \right) + \left( \frac{1}{[\#B(X_0)]^2} \sum_{x_i, x_j \in B(X_0)} e^{\frac{|x_i-x_j|^2}{c}} \right)
\end{align*}
\]  

(4.11)

where \( B(X_0) \) is a ball of points representing \( X_0 \).

With a GRBF kernel \( K(x, y) = e^{\frac{|x-y|^2}{c}} \), the distance function (4.11) can be analysed by regrouping the terms as following to facilitate explanation:

\[
\begin{align*}
    d(y)^2 &= \left( 1 - \frac{2}{\#B(X_0)} \sum_{x_i \in B(X_0)} e^{\frac{|x_i-y|^2}{c}} \right) + \left( \frac{1}{[\#B(X_0)]^2} \sum_{x_i, x_j \in B(X_0)} e^{\frac{|x_i-x_j|^2}{c}} \right) \\
    &= \left( 1 - \frac{2}{\#B(X_0)} \sum_{x_i \in B(X_0)} e^{\frac{|x_i-y|^2}{c}} \right) + \left( \frac{1}{[\#B(X_0)]^2} \sum_{x_i, x_j \in B(X_0)} e^{\frac{|x_i-x_j|^2}{c}} \right)
\end{align*}
\]  

(4.12)

The two terms in expression (4.12) will be discussed independently. As seen, the first term is expressed in terms of

\[
\frac{1}{\#B(X_0)} \sum_{x_i \in B(X_0)} e^{\frac{|x_i-y|^2}{c}}
\]  

(4.13)

which is the average of the Gaussian radial basis functions centred at the points within the set \( B(X_0) \). As a result, as shown in Figure 4.2, the shape of the first term is concave and will be herein known as a distance well.

The second term of expression (4.12) is a self-term which is a function of only the distribution of data points within the set \( B(X_0) \) and not dependent on the location where the measurement is made. Therefore, it does not affect the shape of the distance well derived from the first term. Hence, as far as the first term is concerned, the second term is considered as just a constant.
Note: The blue and red histograms represent zero-mean Gaussian distributed samples with standard deviations 0.2 and 0.4 respectively.

Figure 4.2: Distance functions derived from Gaussian distributed samples

1. **Tendency of Shifting towards Denser Regions**

   As shown in the previous section, kernel mean shift iterations are effectively sequential shifts of the centroid of a set of points captured by a GRBF mean shift kernel in feature space. Since the distance well, as shown in Figure 4.2, tends to centre on a location where the data points are denser in input space, it seems reasonable that GRBF kernel mean shift iterations, like the linear mean shift iterations, will converge towards the denser regions of input space. This would ensure that the GRBF kernel mean shift algorithm would behave as a clustering algorithm in input space.

2. **Adaptiveness of Mean Shift Kernel on Cluster Distributions**

   The region covered by the multivariate Epanechnikov kernel is a hypersphere. Therefore, the GRBF mean shift kernel, as viewed from feature space, is a hypersphere too\(^2\). However, what is of interest here is the shape of the GRBF mean shift kernel when it is viewed from input space.

   The first term in expression (4.12) shows that the shape of the distance well is dependent on the distribution of the data points in input space and hence so is the distance function \(d(y)\). Therefore, the shape of a GRBF mean shift kernel is adaptive to the distribution of data points in input space, given the fact that the region covered by a GRBF mean shift kernel is given by \(d(y) \leq h\).

   The adaptiveness of a GRBF mean shift kernel on cluster distributions is shown in Figure 4.3(a) and 4.4(a). This property of the GRBF kernel mean shift algorithm frees the GRBF mean shift kernel from capturing data points belonging to other clusters but residing close to the cluster being explored. It is particularly useful for anisotropic clusters.

\(^2\)Equivalently, this can be noticed in the linear case as shown in Figure 4.3(b) and 4.4(b).
This property is possibly because the clusters are more separable in the high-dimensional feature space. However, the properties of feature space are dependent on the kernel choice and it may be possible to find kernel functions which are more suitable for cluster analysis as compared to the GRBF kernel. For future work, we could look into kernel functions which are derived from generative probability models of data [17].

3. **Adaptiveness of Mean Shift Kernel on Data Sparseness**

Since the second term in expression (4.12) is a constant as far as the first term or $y$ is concerned, it has an effect of shifting the distance well vertically. The smaller the upward shift, the larger the region covered by a GRBF mean shift kernel in input space (which is given by $d(y) \leq h$) and vice versa.

The second term is also dependent on the distribution of the data points in such a way that its value is smaller when the data points are sparser. As a result, the GRBF mean shift kernel has a wider coverage in the sparser regions. With this property, the GRBF mean shift kernel is less likely to get stuck in the sparse regions. This may imply that the GRBF kernel mean shift algorithm is more tolerant to a sparse data set as compared to its linear counterpart.

Figure 4.2 shows the effect of a vertical shift on the distance well. The distance well corresponding to the denser distribution is more elevated as compared to that of the sparser distribution.

4. **Better Cluster Model**

In the linear mean shift clustering algorithm, the centroid of a hypersphere is represented by just one point. However, for the kernelised version, the centroid in feature space is represented by a set of points in input space. This can be considered as a non-parametric approach for representing an unseen ‘centroid’ in input space, given the fact that the centroid in feature space has no pre-image in input space and this will the case for the GRBF kernel. This leads to a better cluster representation because clusters of different shapes can be modelled better with their associated distance functions as shown in Figure.4.3 and 4.4. As such, more reliable soft cluster labels for data can be derived, as shown in the next chapter.

5. **Sensitivity to Noise due to Overlapped Clusters**

However, the GRBF kernel mean shift clustering algorithm with its complexity may have a drawback in that it is sensitive to noise due to overlapping clusters. It is expected that the algorithm may not work well on data sets with obscure clusters.

6. **Highly Computational Intensive**

For a mean shift iteration with the centroid represented by $m$ input data points, the computation load is of $O(Nmd)$ where $N$ is the size of the data set and $d$ is the dimensionality of input data. The values $m$ and $N$ are related because, in general, the
larger the size of data set, $N$, the larger number of data points, which are captured by the mean shift kernel, and hence the larger the value $m$. Due to this constraint, only a subset of the entire data set is used for the cluster analysis in the experiments shown in Chapter 6 and consequently the algorithm becomes less sensitive to the fine structures in the images.

For future work, to reduce the computational load, the reduced set method [32] can be used to approximate the $m$ data points, that represent the centroid, with fewer data points. On the other hand, a probably more effective approach which is known as the sparse greedy matrix approximation [39] may be employed. This method is trying to approximate the full kernel set, which consists of elements $K_{ij} = K(x_i, x_j)$ where $i, j = 1, ..., N$, with a smaller set.

To illustrate the properties of the kernel mean shift clustering algorithm, contour maps representing values of the distance function measured from the centroid of the mean shift kernel for a number of mean shift iterations are shown in Figure 4.3(a). The iterations are performed on a two-dimensional data set of 600 Gaussian distributed samples with standard deviation 1 and 0.25 along x and y axis respectively. The figure shows the convergence of a mean shift kernel toward the cluster centre. As a comparison, iterations with a linear mean shift kernel are shown in Figure 4.3(b). This shows the adaptiveness of the GRBF mean shift kernel in fitting an anisotropic cluster, in contrast to the isotropic distance function contour of the linear mean shift kernel. It should be noted that the kernel mean shift algorithm is a generalisation of its linear counterpart in a sense that it will return to linear if the linear kernel, instead of the GRBF kernel, is chosen.

Figure 4.3(a) also shows that the shape of the GRBF mean shift kernel slowly adapts to the shape of the cluster when approaching the cluster centre.

The same experiment is performed on a sparser set of 25 data points and the results are shown in Figure 4.4 for both the GRBF mean shift kernel and its linear counterpart. Similar observations are obtained for the sparser dataset.
Figure 4.3: Distance function for mean shift iterations on an anisotropic Gaussian distributed cluster with 600 data points.

Note: The black dots represent data points captured by the mean shift kernel.
Note: The black dots represent data points captured by the mean shift kernel.

Figure 4.4: Distance function for mean shift iterations on an anisotropic Gaussian distributed cluster with 25 data points.
Chapter 5

Classification

For an image segmentation task, spatial information is important for cluster analysis as well as classification of pixels. Spatial information is usually incorporated into a classification scheme through the usage of collective representations based on a neighbourhood structure.

For feature vector-based segmentation algorithms, it is intuitive to employ collective representations generated in the feature vector domain such as a weighted mean of the feature vectors within a neighbourhood. However, there is a drawback in this approach; erroneous representations could be introduced when the neighbourhoods span regions of different textures. These erroneous representations are increasingly likely to result in classification errors as the number of different textures in an image increases. A discussion of this type of misclassification is given in Section 5.1.

As an alternative, a classification algorithm based on a collective representation in the probability domain is proposed.

5.1 Averaging in the Feature Vector Domain

As mentioned earlier, averaging of feature vectors in a neighbourhood, spanning more than two regions with different textures, is likely to introduce erroneous feature vectors. It is because these feature vectors are ideally from different clusters corresponding to the textures. In the simplest case of two clusters, the simple average of two feature vectors, from the two clusters independently, would be a feature vector residing in the vicinity of the middle point between the two clusters. As such, if there exists any cluster nearer to the resultant feature vector than the two clusters do, then the resultant feature vector would be classified as belonging to the third cluster instead.

To assess how likely this type of classification errors could arise in the above case, let’s consider two clusters which are separated by a distance $d$ in the feature vector space and assume that the standard deviation of the clusters are relatively small as compared to $d$. Then, between these two clusters, there is a region in which if there exists any other cluster, a classification error for the simple average of the feature vectors is likely to happen. This region will be a hypersphere with volume:
\[ Vol(r, \text{dim}) = \frac{\pi^{\text{dim}/2}}{\Gamma(\text{dim}/2)} r^{\text{dim}} \] where \( \Gamma(\cdot) \) denotes the Gamma function and \( r = d/2 \).

Assume that the feature vector space is \( m \)-dimensional and the entire data set is confined in a box \([0,1]^m\). In a case where the two clusters stay near the opposite far-end corners of the box and their separation is such that the hypersphere between them fits nicely into the box. Figure 5.1 shows the hypersphere for the two-dimensional case.

![Figure 5.1: A two-dimensional example of the region between 2 clusters](image)

The ratio of the space occupied by the hypersphere to the total space in the box, \( R_1 \), and the ratio of the distance between the clusters to the distance between two farthest corners of the box, \( R_2 \) are given the following expression:

\[ R_1(\text{dim}) = \frac{\pi^{\text{dim}/2}}{\Gamma(\text{dim}/2)} 0.5^{\text{dim}} \]

\[ R_2(\text{dim}) = \frac{1}{\sqrt{\text{dim}}} \]

The ratio \( R_1 \) gives an idea of how much space is occupied by the hypersphere. The higher the ratio, the more likely it is to find others clusters in the hypersphere. On the other hand, the ratio \( R_2 \) gives a clue about how likely to have two clusters with a greater separation. The lower the ratio, the more likely it is to have such two clusters.

Table 5.1 shows the value of \( R_1 \) and \( R_2 \) for the two- to five-dimensional space. Considering the high value of \( R_1 \) and low value of \( R_2 \) as shown in Table 5.1, chances for this particular type of misclassification is non-trivial for the two- to five-dimensional space.

### 5.2 Averaging in the Probability Domain

As an alternative to averaging in the feature vector domain, the averaging or, computation of a collective representation in general, can be performed within a neighbourhood in the probability domain.

To demonstrate the difference, a hypothetical one-dimensional example is examined. Assuming that there are three clusters with an equal-variance Gaussian noise model and
<table>
<thead>
<tr>
<th>dim</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.79</td>
<td>0.71</td>
</tr>
<tr>
<td>3</td>
<td>0.79</td>
<td>0.58</td>
</tr>
<tr>
<td>4</td>
<td>0.62</td>
<td>0.50</td>
</tr>
<tr>
<td>5</td>
<td>0.41</td>
<td>0.45</td>
</tr>
</tbody>
</table>

Table 5.1: Values for R1 and R2

their centres on $\mu_1 = \mu$, $\mu_2 = 0$ and $\mu_3 = -\mu$ respectively, the samples of the clusters $X_1$, $X_2$ and $X_3$ have the following distributions:

$$X_1 \sim N(\mu_1 = \mu, \sigma) \quad X_2 \sim N(\mu_2 = 0, \sigma) \quad X_3 \sim N(\mu_3 = -\mu, \sigma)$$

We will examine a simple case where two samples $x_1 = \mu$ and $x_3 = -\mu$ are given. Define hypotheses:

$$H_i : \text{classified as belonging to } X_i \quad i = 1, 2, 3$$

A maximum a-posterior (MAP) classification with equal prior is performed. Therefore, to test two contending hypotheses, the following likelihood ratio is evaluated:

$$\frac{p(H_i)}{p(H_j)} \begin{cases} \geq 1 & i, j = 1, 2, 3 \end{cases}$$

It is obvious that $x_1$ and $x_3$ will be classified separately as belonging to classes corresponding to clusters $X_1$ and $X_3$ respectively. However, the direct average of the samples, $x_{\text{ave}} = \frac{x_1 + x_3}{2} = 0$, is classified as belonging to the class corresponding to $X_2$.

Let’s examine the case for averaging in the probability domain. The average of two likelihoods is given by:

$$p_{\text{ave}}(x_1 = \mu, x_3 = -\mu|H_i) = \frac{1}{2} \left( \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mu-\mu)^2}{2\sigma^2}} + \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(-\mu-\mu)^2}{2\sigma^2}} \right)$$

As a result, for hypotheses $H_1$ and $H_3$, the average of the two likelihoods would be:

$$p_{\text{ave}}(x_1 = \mu, x_3 = -\mu|H_1) = \frac{1}{2\sqrt{2\pi\sigma^2}} \left( 1 + e^{-\frac{-2\mu^2}{2\sigma^2}} \right) = p_{\text{ave}}(x_1 = \mu, x_3 = -\mu|H_3)$$

while for hypothesis $H_2$:

$$p_{\text{ave}}(x_1 = \mu, x_3 = -\mu|H_2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\mu^2}$$

To determine whether the average of the likelihoods is more likely to be classified as belonging to a class corresponding to cluster $X_1$ or cluster $X_2$, the following likelihood ratio is evaluated. In this case, the result would be the same if $X_1$ is replaced by $X_3$. 

41
\[ p_{\text{ave}}(x_1 = \mu, x_3 = -\mu | H_1) = 1 + e^{-\frac{2\mu^2}{\sigma^2}} \geq 1 \]

Substituting \( e^{-\frac{\mu^2}{2\sigma^2}} = k \)

\[ \frac{1 + k^4}{2k} \geq 1 \]
\[ 1 + k^4 - 2k \geq 0 \]

Hence,

\[ k \leq 0.5437 \]
\[ \sigma \leq 0.9058\mu \]

It shows that for the case of averaging in the probability domain, it is still possible for hypotheses \( H_1 \) (or \( H_3 \)) to prevail, provided \( \sigma \leq 0.9058\mu \). However, it is impossible for the approach of averaging in the data domain to achieve this. It should be noted that averaging in the probability domain can only be considered better in this example if there is an assumption that the separate classification of the individual samples is correct.

### 5.3 Classification based on Recursive Update of Posterior Distributions

In this section, a classification algorithm based on recursive update of posterior distributions will be outlined. This algorithm is functioning in the probability domain.

Given a set of cluster centres \( \{C_1, C_2, ..., C_p\} \) in feature space, the squared distance between a cluster centre \( n \) and a point \( \Phi(y) \) in feature space is given by:

\[
d(y|C_n)^2 = K(y, y) - \frac{2}{\#B(C_n)} \sum_{x_i \in B(C_n)} K(x_i, y) + \frac{1}{[\#B(C_n)]^2} \sum_{x_i, x_j \in B(C_n)} K(x_i, x_j)
\]

for \( 1 \leq n \leq p \)

Define the likelihood of a data point given a cluster centre to be a linear function of the squared distance:

\[
p(y|C_n) \propto 2 - d(y|C_n)^2 \text{ for } y \in S_n
\]

where \( S_n \) is a bound on the likelihood functions. This ensures that the integration of the likelihood function is finite and expression (5.1) has a same proportionality constant for all
On the other hand, the constant, 2, in expression (5.1) is to keep the likelihood function from being negative.

The likelihood functions defined in expression (5.1) has a property that points farther away from a cluster centre always have a smaller likelihood.

Without prior knowledge about the class labels of the pixels, an equal prior is used for class labels at all locations $s$ in an image:

$$p^0(C_n|y_s) = \frac{p(y_s|C_n)p(C_n|s)}{\sum_{n=1}^N p(y_s|C_n)p(C_n|s)}$$

for all $s$ and $n$

The derivation of the posterior probability for class labels can be considered as a step to obtain a soft cluster membership for every pixel of an image. Since the distance function derived from the kernel mean shift algorithm takes the cluster shape into account, the resultant soft cluster membership is more reliable than that derived from its linear counterpart.

For every location $s$ in an image, a neighbourhood is defined. The set of neighbourhood index pertaining to a location $s$ is denoted by $nn(s)$.

Algorithm 2 Classification based on Recursive Update of Posterior Distributions

Given classes $C_1, C_2, \ldots, C_p$ and the posterior distribution of the classes at each location in an image at iteration $i$, $p^i(C_n|y_k)$, compute the corresponding $p^{i+1}(C_n|y_k)$ by the following 2 steps:

1. **Gaussian Averaging:** For all $s$ and $n$, compute the Gaussian average of the posterior probability within a neighbourhood. $g_k$ is the Gaussian weights as given by expression (3.2).

$$p^i_s(C_n) = \sum_{k \in nn(s)} g_k p^i(C_n|y_k)$$

2. **Update of Posterior Distributions:** For all $s$ and $n$, compute the posterior probability update using the Bayes rule by taking the Gaussian average as the prior,

$$p^{i+1}(C_n|y_s) = \frac{p(y_s|C_n) p^i_s(C_n)}{\sum_{n=1}^N p(y_s|C_n)p^i_s(C_n)}$$

Stop when

$$\sum_{all s and n} |p^{i+1}(C_n|y_s) - p^i(C_n|y_s)| \leq \epsilon$$

At an iteration of the recursive updates, the Gaussian average of the posterior probability serves as the prior for the next iteration. The Gaussian averaging allows the class label of a pixel to be influenced by those of its neighbouring pixels. It is assumed that neighbouring class labels are correlated.
The Gaussian average of the posterior probability for class labels, being a collective representation for a group of neighbouring class labels, should be more reliable than an individual posterior probability (at least for those derived from neighbourhoods residing on a homogeneous texture region). As such, an improved prior is obtained and consequently this would lead to an improved posterior probability for class labels in the next iteration. The rationale of the recursive procedure hinges on the interdependence between the prior and the posterior probability for class labels, as described above.
Chapter 6

Image Segmentation Experiments

The experiments were performed on four images, which are patchwork of textures, as shown in Figure 6.1.

![Image Segmentation Experiments](image.png)

Note:
(A)(B) A patchwork of 7 different aerial imagery.
(C) A patchwork of 7 different Brodatz textures.
(D) A patchwork of 16 different Brodatz textures.

Figure 6.1: Texture images for experiments

The images \(^1\) (A), (B) and (C) in Figure 6.1 were also used for unsupervised texture segmentation experiments in [3] and [9]. These three images are composed of 7 different textures which are organised in such a way that any two textures share a common boundary at least once. The image (D) in Figure 6.1 is a patchwork of 16 different Brodatz textures. It was used for unsupervised texture segmentation experiments in [31]. All of the images are considered to be very challenging for unsupervised texture segmentation.

The segmentation algorithm proposed in this dissertation can be configured with different choices of the kernel function and different number of principal components. The procedure of the algorithm with the proposed configuration are listed below:

---

\(^1\)The images can be obtained from [http://w3.ualg.pt/dubuf/pubdat/texture/texture.html](http://w3.ualg.pt/dubuf/pubdat/texture/texture.html)
1. **Complex wavelet Transform**\(^2\):

3 level complex wavelet decomposition of a 256x256 grayscale image generates 18 subband images i.e. 6 first-level subband images of size 128x128, 6 second-level sub-band images of size 64x64 and 6 third-level subband images of size 32x32. The magnitude of the complex coefficients for all the subband images other than those of the first-level are interpolated to the size of 128x128 using bilinear interpolation. To conserve the total energy of each subband, the coefficients of each subband are scaled according to the amount of interpolation introduced i.e. subbands of the second-level are scaled by 1/4 while those of the third-level by 1/16.

2. **Performing Data transform**: A sequential operations of \((\cdot)^2\), Gaussian weighted averaging and \(\log(\cdot+1)\) are performed on all subband images.

3. **Feature dimensionality reduction using GRBF KPCA**: To reduce computational load, a subset of the full data set, which is derived from the down-sampled subband images of size 32x32, are used for solving the eigenvalue problem in KPCA. Then, a larger subset of the full data set, which is derived from the down-sampled subband images of size 64x64, are used to compute the first three or four kernel principal components of size 64x64.

4. **Formation of feature vector**: Gaussian averaging on 3x3 neighbourhood is performed at every image location in the kernel principal components. Then, feature vectors within 3x3 neighbourhoods are joined. The resultant feature vectors may be of 27 or 36 dimensions depending on whether 3 or 4 kernel principal components are used.

5. **GRBF Kernel mean shift clustering**: To reduce the computational load of the kernel mean shift clustering algorithm, a subsampled set of feature vectors with size 32x32 is used for cluster analysis. Prior to cluster analysis, feature vectors are normalised in each dimension according to the variance.

6. **Classification**: A full set of feature vectors are classified based on recursive updates of posterior distributions for class labels.

The segmentation experiment generates segmentation results for four different configurations of the algorithm including the proposed configuration:

1. With linear PCA, linear mean shift and without concatenation of feature vectors.

2. With linear PCA, GRBF kernel mean shift and without concatenation of feature vectors.

\(^2\)Acknowledgements to Dr. Nick Kingsbury for the matlab implementation of the complex wavelet transform
3. With GRBF KPCA, GRBF kernel mean shift and without concatenation of feature vectors.

4. With GRBF KPCA, GRBF kernel mean shift and concatenation of feature vectors.

The segmentation results are shown in Figure 6.2 and the corresponding misclassification maps and percentage of error are shown in Figure 6.3. All of the segmentation results are based on three principal components except for that of image (D) in Figure 6.1 which uses four principal components. Besides, a same value 1.0 is used for the parameter of the GRBF kernel for both KPCA and the kernel mean shift clustering algorithm.

It should be noted that the results shown are the best in a sense that they correspond to detection of most or all of the correct classes with least misclassification as judged visually. This criterion may not ensure that the 'best' results have the lowest percentage of error.

Some observations derived from the results are listed below:

- Comparing the results for configurations 1 and 2, it is observed that the GRBF kernel mean shift clustering algorithm produces slightly better but insignificant segmentation results, when judged visually or based on the error percentage, as compared to its linear counterpart.

- Comparing the results for configurations 2 and 3, it is observed that GRBF KPCA enables detection of more correct classes as compared to its linear counterpart. This is the reason for the better configuration 3 segmentation maps for images (B) and (D). However, GRBF KPCA may extract undesirable details that lead to misclassification in the configuration 3 segmentation results of image (C) which use GRBF kernel principal components as features.

- Comparing the results for configurations 3 and 4, it is observed that the concatenation of feature vectors within a neighbourhood enables better separation between different classes and hence better segmentation maps.

Due to the small number of images involved in this experiment, the above-mentioned observations may not be conclusive.

To provide an insight into the process of classification based on recursive update of posterior distributions, the segmentation label maps for a series of iterations are shown in Figure 6.4. The label maps are resulted from segmentation of image (D) which has 16 textures. The label map at the first iteration is the result of a simple pixel-by-pixel classification which has a lot of errors. However, most of the errors are corrected along the iterations. There are two types of errors:

1. Errors resulting from being classified to a another valid class.

2. Errors resulting from being classified to a false class which is not one of the valid classes.
Ground truth of the images

(1) Linear PCA, linear mean shift and no joined feature vectors

(2) Linear PCA, GRBF kernel mean shift and no joined feature vectors

(3) GRBF KPCA, GRBF kernel mean shift and no joined feature vectors

(4) GRBF KPCA, GRBF kernel mean shift and joined feature vectors

Figure 6.2: The best segmentation results for different configurations of the algorithm
Figure 6.3: The error map and error percentage for the segmentation results.
Figure 6.4: Segmentation label maps for 8 iterations of classification based on recursive update of posterior distributions.

Note: The intensity represents the mode value of the posterior distributions.

Figure 6.5: Confidence maps for 8 iterations of classification based on recursive update of posterior distributions.
The label maps in Figure 6.4 are made up of class labels with the highest posterior probability at every location. Figure 6.5 shows the corresponding confidence maps for the label maps in Figure 6.4. The confidence maps show the mode value of the posterior distribution for class labels at every location. Cross-referencing Figures 6.4 and 6.5, it can be seen that those recoverable misclassification are initially classified with relatively low confidence or in other words the posterior distributions for class labels on the misclassified pixels could be relatively flat as compared to those on the correctly classified pixels which are relatively peaky. The less peaky posterior distributions can be easily influenced by those of the neighbouring pixels.

A favourable property of the classification scheme is its robustness to the errors resulting from the clustering procedure, particularly to the errors due to false classes. For images with large number of classes, cluster size varies and a mean shift kernel with a small radius is preferred because a mean shift kernel with small radius is more capable of identifying small size clusters. However, a small radius mean shift kernel can be easily left stranded in the low density regions and gives rise to false classes. Therefore, robustness of a classification algorithm against false classes is important.

As a comparison, a multiscale classification could be quite sensitive to false classes because the cost associated with the misclassification at coarser scales can be quite high in a sense that the coarser scale misclassification could have a great impact on the subsequent classification at finer scales.

Figure 6.6 shows the segmentation results for eight natural images. It can be seen that the idea of homogeneous texture is ambiguous in natural images and a region with a homogeneous texture may not correspond to an object. This indicates the limitation of texture features for image segmentation. For instance, the tiger in one of the images may be well segmented in its entirety if colour features are used on top of the texture features.

The best segmentation results demonstrated earlier in Figure 6.2 are achieved by selecting the best value for the radius of the mean shift kernel. However, results in Figure 6.6 are obtained using a same value for the radius and hence they may not be the best results. In addition, the coarseness of the segmentation maps is due to the small set of data employed by the algorithm.
Figure 6.6: Segmentation maps for natural images
Chapter 7

Content-based Image Retrieval

A powerful content-based image retrieval (CBIR) system should allow users to query the image database using objects such as a horse, a building, a car and so on. The object is basically a high-level concept and cannot be described solely using low-level features extracted directly from images. For example, an algorithm can extract a green-colour region and a brown-colour region which correspond to leaves and a tree trunk respectively in an image based on low-level features. But high-level information is necessary for the algorithm to understand the concept of a tree. Furthermore, the concept of a tree is more complex than just a composite of green leaves and a brown tree trunk.

Even though, image segmentation based on low level features such as colour and texture may not be able to single out objects in its entirety from images, it facilitates object-based CBIR [7] in a way that users are allowed to query using regions which form an object. In this case, the indexing of the image database has to be region-based too.

The work presented in this chapter is a collaborative effort which led to a conference publication [19] which can be found in Appendix C. To avoid repeating the details given in the article, only a brief description of the work is presented here and emphasis is given to author’s contribution on the work which is the texture descriptor for the CBIR system. This piece of work preceded the development of the image segmentation algorithm described earlier therefore the algorithm is not integrated with the CBIR system.

7.1 Unsupervised Image segmentation Algorithm

A brief description of the unsupervised image segmentation algorithm [18] which is part of the CBIR system is given in this section. The algorithm is based on colour and texture features. The colour feature is represented in the S-CIE L*a*b* colour space [42] which is a spatial extension of the CIE L*a*b* colour space. The S-CIE L*a*b* colour space accounts for the appearance of fine-patterned colours on human visual system, for instance, a fine texture consists of black and white strips may appear to be grey in colour to humans.

The texture feature is extracted using complex wavelet transform. It is processed by a local energy function before dimensionality reduction is performed using principal
component analysis.

Mean shift procedure with a specified kernel size is used for cluster analysis while multiscale random field model-based classification is used for assigning class labels.

### 7.2 Colour Descriptors

After images are segmented into regions, the colour and texture properties of the regions are represented in a way which allows similarity or dissimilarity between two regions to be evaluated.

The colour descriptor is made up of three histograms corresponding to the three dimensions of the colour feature in the S-CIE L*a*b* colour space. To evaluate the colour dissimilarity between two regions, the Kolmogorov-Smirnov (K-S) distance between the two corresponding histograms is evaluated separately for each dimension of the S-CIE L*a*b* colour space and then they are combined through expression (7.1):

\[
d_{COL}(x^C, y^C) = \frac{1}{3} \sqrt{\left[ K-S(x^C_L, y^C_L) \right]^2 + \left[ K-S(x^C_a, y^C_a) \right]^2 + \left[ K-S(x^C_b, y^C_b) \right]^2} \tag{7.1}
\]

where K-S (·, ·) is the Kolmogorov-Smirnov distance between two histograms and is given by expression (7.2):

\[
K-S(y^1, y^2) = \max \left| F_1(k) - F_2(k) \right| \tag{7.2}
\]

\[F(\cdot)\] is the cumulative distribution function for histograms as defined by expression (7.3):

\[
F_i(k) = \frac{1}{n} \#(i : y_i^k \leq k) \tag{7.3}
\]

Since K-S distance is the magnitude of the largest difference between two cumulative distribution function, its value lies between 0 and 1. Hence, the colour similarity measure can be defined as expression (7.4):

\[
s_{COL}(x, y) = 1 - d_{COL}(x, y) \tag{7.4}
\]

### 7.3 Texture Descriptors

It is found experimentally by Mallat [27] that the histograms of the real wavelet coefficients for natural textures can be modelled by a family of exponentials as given by expression (7.5):

\[
f(x) = Ke^{-|x|/\alpha^a} \tag{7.5}
\]

where the parameters are the followings:
\[ K = \frac{\beta}{2\alpha \Gamma(1/\beta)} \] where \( \Gamma(\cdot) \) is the gamma function

\[ \alpha = m_1 \frac{\Gamma(1/\beta)}{\Gamma(2/\beta)} \] where \( m_1 \) is the mean deviation of the wavelet coefficients

\[ \beta = F^{-1} \left( \frac{m_1^2}{m_2} \right) \] where \( m_2 \) is the mean energy of the wavelet coefficients

and

\[ F(\cdot) = \frac{\Gamma^2(2/\beta)}{\Gamma(3/\beta) \Gamma(1/\beta)} \]

This model was first applied for texture characterisation in [11]. The expression (7.5) can be considered as a generalised Gaussian distribution. With this, it is logical to conjecture that the distributions of the magnitude of the complex wavelet coefficients follow a generalised Rayleigh distribution for the following reasons:

- The real and imaginary parts of the complex wavelet coefficients separately follow the generalised Gaussian distribution because the real and imaginary parts of the complex wavelet transform can be considered separately as a real wavelet transform.

- The correlation between the real and imaginary parts of the complex wavelet coefficients is negligible. It is because the real and imaginary parts of each complex wavelet basis function are orthogonal to each other. Furthermore, the phases of the coefficients are found uniformly distributed between \(-\pi\) and \(\pi\).

The generalised Rayleigh distribution is given in expression (7.6). This model is found to have a good fit to the actual distributions of the magnitude of the complex wavelet coefficients for textures. A set of the distributions for a tiger texture is shown in Figure 3.3.

\[ f(x) = K x e^{-\left(|x|/\alpha\right)^\beta} \] (7.6)

where the parameters are given by:

\[ K = \frac{\beta}{\alpha^2 \Gamma(2/\beta)} \] where \( \Gamma(\cdot) \) is the gamma function

\[ \alpha = m_1 \frac{\Gamma(2/\beta)}{\Gamma(3/\beta)} \] where \( m_1 \) is the mean magnitude of the complex wavelet coefficients

\[ \beta = F^{-1} \left( \frac{m_1^2}{m_2} \right) \] where \( m_2 \) is the mean energy of the complex wavelet coefficients

and
The texture descriptor is made up twenty-four histograms of complex wavelet coefficients for a four-level decomposition. The model relieves the storage requirement imposed by the texture descriptor by representing each histogram with only two real-number parameters.

The texture dissimilarity between two regions is the combined K-S distances between the histograms of corresponding subbands as given by expression (7.7).

\[ d_{TEX}(x_{TEX}, y_{TEX}) = \sqrt{\sum_{i=1,...,N} \left[ K-S(x_i^{TEX}, y_i^{TEX}) \right]^2} \]  

Similarly, the texture similarity is defined by:

\[ s_{TEX}(x, y) = 1 - d_{TEX}(x, y) \]  

7.4 Query Scheme

The CBIR system has two query schemes, i.e. the semi-automated and the automated scheme. The semi-automated scheme mainly caters for users who would like to retrieve images with specific objects of interest. In this case, users begin with submitting a query image and then select segmentation region(s) corresponding to object(s) from the returned segmentation map. On the other hand, the automated scheme mainly caters for query using images without objects of interest such as sunset or scenery images. For the automated scheme, users merely submit a query image to begin the retrieval process.

A sequence of events, triggered when a user is querying using two segmentation regions in the semi-automated retrieval scheme, is shown in Figure 7.1.

Figure 7.2 shows a sequence of events for an instance of the automated retrieval scheme.

7.5 Experiments

Experiments using both the automated and semi-automated retrieval schemes have been performed. In addition, a comparison between the automated retrieval scheme and a global scheme, where images are described as a whole using colour and texture descriptors, was conducted. These experimental results are given in Appendix C.

Apart from that, an experiment was performed to compare K-S distance with a few other well-known distance measures as listed below:

- **Histogram Intersection (HI):**
For all classes in an image, only that with the highest rank is retained. For example, for the semi-automated retrieval scheme, the query results are selected based on the combined score $S_{(combine)} = wS(Tex) + (1-w)S(col)$.

Figure 7.1: Sequence diagram for the semi-automated retrieval scheme

For all classes in an image, only that with the highest rank is retained. For the automated retrieval scheme, the query results are selected based on the combined score $S_{(combine)} = wS(Tex) + (1-w)S(col)$, and the lists are joined with the largest sum of top 10 entries.

Figure 7.2: Sequence diagram for the automated retrieval scheme
$D_{HI}(I,J) = \sum_{vi} [I_i - \min(I_i, J_i)]$

where $I, J$ represent histograms with $I_i, J_i$ being the bin values

- **A statistic of the Cramer/von Mises type (CvM):**

$$D_{CvM}(I,J) = \sum_{vi} [F_I(i) - F_J(i)]^2$$

where $F_I$ is the cumulative histogram for $I$

- **$\chi^2$ statistic (chi2):**

$$D_{chi2}(I,J) = \sum_{vi} \frac{I_i - mean_i}{mean_i}$$

where $mean_i = \frac{I_i + J_i}{2}$

- **The empirical Jeffrey-divergence (Jd):**

$$D_{Jd}(I,J) = \sum_{vi} \left[ I_i \log \left( \frac{I_i}{mean_i} \right) + J_i \log \left( \frac{J_i}{mean_i} \right) \right]$$

The experiment evaluates the retrieval performance for each of the distance measures. The retrieval performance is measured using the precision and recall metric. The precision of a retrieval is a ratio of the number of correct matches to the total number of retrieved images. Whereas, the recall of a retrieval is a ratio of the number of images, which are retrieved correctly, to the total number of the correct images in the database. Let’s illustrate the concept of the precision and recall metric using an example: say $N^R$ is retrieved using a query image of class A from a database with a total of $N_{total}$ images. Then, during a retrieval, the images can be categorised as in Table 7.1:

In this case,

$$precision = \frac{N^R_A}{N^R}$$

and

<table>
<thead>
<tr>
<th></th>
<th>Retrieved</th>
<th>Not Retrieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>$N^R_A$</td>
<td>$N^{NR}_A$</td>
</tr>
<tr>
<td>Non Class A</td>
<td>$N^R_{NA}$</td>
<td>$N^{NR}_{NA}$</td>
</tr>
<tr>
<td>$N^R$</td>
<td>$N^{NR}$</td>
<td>$N_{total}$</td>
</tr>
</tbody>
</table>

Table 7.1: Illustration for the idea of precision and recall
\[ \text{recall} = \frac{N^R_A}{N_A} \]

In the experiment, three queries based on the regions corresponding a tiger, a leopard and a zebra, as shown in Figure 7.3, are performed and the experimental results are shown in Figure 7.4.

![Figure 7.3: Query regions used in the experiment for evaluating different distance measures.](image)

The results show that the performance for all of the above-mentioned distance measures are comparable except that of CvM which is unsatisfactory.
Figure 7.4: Experimental results for different distance measures
Chapter 8

Conclusions and Future work

In this dissertation, a new segmentation algorithm was proposed and demonstrated using texture images. For the segmentation algorithm, complex wavelet transform and kernel principal component analysis was used for feature extraction and dimensionality reduction. Kernel principal component analysis using Gaussian radial basis function kernel was shown to be effective when capturing localised features corresponding to clusters.

The Kernel mean shift clustering procedure was introduced for cluster analysis. The Gaussian radial basis function mean shift kernel was shown to be adaptive to the shape of anisotropic clusters and it would, in theory, lead to a better representation of clusters as compared to its linear counterpart. Although the kernel mean shift clustering procedure has produced only a slight improvement in the experiment results, it is believed that with the desired properties, it could be applied to other data analysis problems.

A classification algorithm which operates in probability domain rather than feature vector domain was introduced for classification of class labels in the final step of the image segmentation algorithm. The classification algorithm was shown to be robust to false classes due to the relatively minor cost associated with the misclassification resulting from false classes.

To evaluate the performance of the segmentation algorithm proposed, a series of experiments were performed on challenging texture images and the results were analysed.

Finally, a brief description on a content-based image retrieval system, which is an application of image segmentation algorithms, was presented. Experiments were performed to evaluate the performance of the system.

8.1 Future work

The image segmentation algorithm and the content-based image retrieval system presented in this dissertation have a lot of room for improvement. Suggestions for improving the current work as listed below will form the directions for future research.

- Incorporating high-level features into the CBIR system:
Currently, only low-level features such as colour and texture are used for indexing in the CBIR system. In general, humans judge the similarity between images based on more than just these low-level features. For instance, humans sometimes consider two images with outdoor scene to be similar even though both images have very different colour and texture characteristics. Therefore, high-level features which correspond to semantic image properties which are meaningful should be incorporated in the CBIR system. The high-level features mentioned here can be a function of low-level features with a possible addition of prior information. The high-level features may be also derived from the visual cues found in images. These high-level features can be learned given examples. Recent development in statistical learning theory has led to impressive classification results using Support Vector machines. This approach of classification based on learning of high-level features can be employed for indexing as well as describing images in a database. A further discussion of these high-level features are given in Section 8.1.1.

- **Offering alternatives for query using an example image:**
  In general, it is difficult to find a suitable query image which has all the characteristics (both high-level and low-level characteristics) of the images that users intend to retrieve. Therefore, the query mode of using an example image can be limiting. As an alternative, users may be allowed to make a query by simply specifying some of the high-level characteristics of the images that they would like to retrieve. This approach of querying does not exclude querying using an example image but complements it. Although the CBIR system emphasises retrieval based on image content, text annotations may be used as an extra feature. By integration of the content and textural properties, a query interface purely reliant on text input may be feasible.

- **Intuitive browsing and relevance feedback:**
  In most cases, users have no idea of what types of images a database contains and what to expect from a query. Therefore, it is difficult for users to obtain satisfactory retrieved images in a single query. In this case, an intuitive user interface such as that of MIT’s Photobook system [30] is important for users to explore the image database, interact with the system and improve their queries progressively. On the other hand, users’ expectation can be very subjective and users that have a same set of query specifications may not have a same type of images in mind. Therefore, it is advantageous to enhance the interactivity of a CBIR system by incorporating relevance feedback which includes users into the retrieval loop so that they can refine their search to obtain satisfactory results.

- **Improving the segmentation algorithm:**
  Colour features may be incorporated into the segmentation algorithm without major modifications. Besides, reduction of the computational load for KPCA and the kernel mean shift clustering algorithm is possible through the reduced set method [32], the sparse greedy matrix approximation [39] or other techniques.
8.1.1 High-level features

There are various high-level features as listed below which can be incorporated in a CBIR system. Some of these features may not be well-defined for all types of images but they certainly match the characteristics of particular group of images.

- **Textured vs. plain**
  
  “Textured images” refers to images with a lot of edges while plain images are the opposite. Those properties which are judged purely based on human perception are considered to be high-level features. Examples of textured and plain images are shown in Figure 8.1:

  ![Textured images](image1.jpg) ![Plain images](image2.jpg)

  **Figure 8.1: Examples of textured and plain images**

  A preliminary experiment has been performed for binary classification of images using the textured and plain properties. The classification is done using a Support Vector machine with a third-degree polynomial kernel based on features consisting of histogram model parameters for the magnitude of complex wavelet coefficients. The classification used 500 images for training and 322 images for testing and yielded zero training error and 18.6% testing error. However, it was found that the incorrectly classified images are mainly ambiguous in their class membership and those images with a clear class membership are mainly correctly classified. The experiment indicates training using examples is a feasible way of indexing the image database based on high-level features such as the textured and the plain properties.

- **City scene vs. natural scene**

  Some visual cues can be used as features to discriminate between city and natural scene. In [15], information about dominant orientations of images is used to identify city scene images. It is observed that majority of the city scene images, due to the man-made structures, have dominant vertical and horizontal orientations. On the
other hand, natural scene images may have more uniformly and randomly distributed dominant orientations.

- **With human vs. without human**

  Whether an image contains humans or not may be determined through human face detection. In [29], Support Vector machines have been used to detect and locate human faces in images. Apart from human face detection, skin colour detection may be used to classify images with and without humans [14]. This particular classification of images is very helpful for narrowing down the type of images users like to retrieve from a diverse and unconstrained image database such as the Internet.

- **Indoor vs. outdoor**

  In [40], colour features, which are described in Ohta colour space, and texture features, which are characterised using the multiresolution, simultaneous autoregressive model, are used for classification of indoor and outdoor images with impressive classification results. Some of the simpler algorithms may just identify outdoor images by simply looking for blue sky at the top of the images.

- **Medium of images**

  Images may be broadly classified into photographic and synthetic images. Synthetic images include artificial images such as advertisement banners commonly found in the Internet and artwork such as painting. Numerous visual cues can be employed to classified these two broad categories of images as shown in [14]. For instance, it is observed that synthetic images commonly have sharp edges, a limited number of colours, which are usually eye-catching, and so on. Besides, artwork of different mediums such as watercolour, oil on canvas, lithography, pastel, tapestry etc have their own unique features which allow them to be classified using machine learning.

  The classifications are never fool-proof because there are always images whose class membership is ambiguous. However, they certainly help users interact with a CBIR system and narrow the scope when searching for particular images.
Appendix A
Principal Component Analysis

Principal component analysis (PCA) can be considered as computation of a set of or-
thonormal vectors $v_1, \ldots, v_k$ which leads to principal components of data set $x$ through the
following linear transformation:

$$ y_i = v_i^T (x - \bar{x}) $$
on condition that the sum of the variance of all these principal components:

$$ Var_s = \sum_{i=1}^{k} Var(y_i) $$

are maximised.

The variance of a principal component is given by:

$$ Var(y_i) = \frac{1}{N} \sum_{j=1}^{N} (y_i^{(j)})^2 $$

$$ = \frac{1}{N} \sum_{j=1}^{N} [v_i^T (x^{(j)} - \bar{x})] [ (x^{(j)} - \bar{x})^T v_i ] $$

$$ = v_i^T \left( \frac{1}{N} \sum_{j=1}^{N} (x^{(j)} - \bar{x})(x^{(j)} - \bar{x})^T \right) v_i $$

$$ = v_i^T C_x v_i \quad \text{(A.1)} $$

Maximising $Var_s$ is corresponding to solving the following set of equations:

$$ \frac{\partial (Var(y_i) + \phi_i \left[ 1 - v_i^T v_i \right])}{\partial v_i} = 0 \quad \text{where } \phi_i \text{ is a Lagrange multiplier} $$

and this leads to:
\[ C_x v_i = \phi_i v_i \]  \hspace{1cm} (A.2)

Substituting result (A.2) into expression (A.1), \( Var(y_i) = \phi_i \) is obtained. Therefore, by choosing \( v_1, \ldots, v_k \) to be the eigenvectors of \( C_x \) which have the \( k \) largest eigenvalues, the sum of the variances is maximised.
Appendix B

Kernel Principal Component Analysis

In this section, we will review a non-linear generalisation of principal component analysis namely kernel principal component analysis (KPCA) [35].

As shown in section 2.3, for a kernel function $K(\cdot, \cdot)$ that fulfils the Mercer’s condition, there exists an Hilbert space $F$ and a corresponding mapping $\Phi$:

$$\Phi : \mathbb{R}^N \to F$$

The Hilbert space $F$ will be here on known as the feature space. Given a set of data, $x_i$ with $i = 1, ..., M$, the image of the data set in feature space is given by:

$$X_i = \Phi(x_i)$$

and the centred data set in feature space is denoted by:

$$\tilde{X}_i = X_i - \frac{1}{M} \sum_{k=1}^{M} X_k$$

Then, the covariance matrix for the data set in feature space can be expressed as the following:

$$C_F = \frac{1}{M} \sum_{i=1}^{M} \tilde{X}_i \tilde{X}_i^T$$

KPCA is equivalent to performing PCA in feature space and therefore it leads to solving an eigenvalue problem in feature space which can be formulated as below:

$$\lambda V = C_F V \quad \text{(B.1)}$$

The covariance matrix is a square matrix with its dimension being the feature space dimension which could be infinite. Therefore, an equivalent equation for the eigenvalue problem, as shown in expression (B.1), is considered instead:
\[
\lambda V = \left( \frac{1}{M} \sum_{i=1}^{M} \tilde{X}_i \tilde{X}_i^T \right) V
\]

\[
= \frac{1}{M} \sum_{i=1}^{M} \tilde{X}_i \left( \tilde{X}_i^T V \right)
\]

\[
\tilde{X}_j^T (\lambda V) = \frac{\tilde{X}_j^T}{M} \sum_{i=1}^{M} \tilde{X}_i \tilde{X}_i^T V
\]  \hspace{1cm} (B.2)

As shown in expression (3.4), the eigenvectors are within the span of the centred data set, therefore they can be expressed in the following form:

\[
V = \sum_{i=1}^{M} \alpha_k \tilde{X}_i
\]  \hspace{1cm} (B.3)

Substituting (B.3) into expression (B.2):

\[
\tilde{X}_j^T \lambda \left[ \sum_{k=1}^{M} \alpha_k \tilde{X}_k \right] = \tilde{X}_j^T \left( \frac{1}{M} \sum_{i=1}^{M} \tilde{X}_i \tilde{X}_i^T \left[ \sum_{k=1}^{M} \alpha_k \tilde{X}_k \right] \right)
\]

\[
\lambda \sum_{k=1}^{M} \alpha_k \left( \tilde{X}_j^T \tilde{X}_k \right) = \frac{1}{M} \left[ \sum_{i=1}^{M} \left( \tilde{X}_j^T \tilde{X}_i \right) \sum_{k=1}^{M} \left[ \alpha_k \left( \tilde{X}_j^T \tilde{X}_k \right) \right] \right]
\]  \hspace{1cm} (B.4)

Defining a square matrix \( \tilde{K} \) of dimension \( M \) with its elements being:

\[
\tilde{K}_{ij} = \tilde{X}_i^T \tilde{X}_j = \tilde{X}_i \cdot \tilde{X}_j
\]  \hspace{1cm} (B.5)

Then, the expression (B.4) becomes:

\[
M \lambda \tilde{K} \alpha = \tilde{K}^2 \alpha \quad \text{where} \quad \alpha = [\alpha_1, \alpha_2,...,\alpha_M]^T
\]  \hspace{1cm} (B.6)

Expression (B.6) can be simplified to the following expression:

\[
M \lambda \alpha = \tilde{K} \alpha
\]  \hspace{1cm} (B.7)

By enumerating the eigenvectors, \( \alpha^1, ..., \alpha^M \), the expression (B.7) becomes:

\[
\gamma^j \alpha^j = \tilde{K} \alpha^j
\]

where \( \gamma^j = M \lambda_j \) with \( j = 1, 2, ..., M \)

\( \gamma^j \) and \( \alpha^j \) are respectively the \( j \)th eigenvalue and the \( j \)th eigenvector of \( \tilde{K} \). After obtaining \( \alpha^j \), the eigenvector of interest in expression (B.3) can be determined using the following expression:
\[
V^j = \sum_{i=1}^{M} \alpha_i^j \tilde{X}_i
\]

However, the resultant eigenvectors may not be normalised and can be normalised explicitly with a normalising constant \( r^j \):

\[
\hat{V}^j = r^j \sum_{i=1}^{M} \alpha_i^j \tilde{X}_i \tag{B.8}
\]

and the normalised eigenvectors comply the following condition:

\[
\hat{V}^j \cdot \hat{V}^j = 1 \tag{B.9}
\]

\[
\hat{V}^j \cdot \hat{V}^j = \left( r^j \sum_{k=1}^{M} \alpha_k^j \tilde{X}_k \right) \cdot \left( r^j \sum_{i=1}^{M} \alpha_i^j \tilde{X}_i \right)
\]

\[
= (r^j)^2 \sum_{k=1}^{M} \sum_{i=1}^{M} \alpha_k^j \alpha_i^j (\tilde{X}_k \cdot \tilde{X}_i)
\]

\[
= (r^j)^2 \sum_{k=1}^{M} \sum_{i=1}^{M} \alpha_k^j \alpha_i^j \tilde{K}_{ki} = (r^j)^2 \alpha^j \tilde{K} \alpha^j
\]

\[
= (r^j)^2 (\alpha^j)^T \gamma^j \alpha^j = (r^j)^2 \gamma^j \alpha^j \cdot \alpha^j
\]

\[
= (r^j)^2 \gamma^j
\]

From expression (B.9), the normalising constant is given by:

\[
r^j = \frac{1}{\sqrt{\gamma^j}} \tag{B.10}
\]

With the normalising constant given by expression (B.10), expression (B.8) becomes:

\[
\hat{V}^j = \frac{1}{\sqrt{\gamma^j}} \sum_{i=1}^{M} \alpha_i^j \tilde{X}_i
\]

With \( \gamma^j \) in descending order, i.e. \( \gamma^1 \geq \gamma^2 \geq \ldots \geq \gamma^M \) and correspondingly \( \lambda^1 \geq \lambda^2 \geq \ldots \geq \lambda^M \), the \( j \)th principal component of any data point \( z \) can be computed through projection onto the \( j \)th normalised eigenvector \( \hat{V}^j \):

\[
y_j = \hat{V}^j \cdot \left( \Phi(z) - \frac{1}{M} \sum_{i=1}^{M} X_i \right) \tag{B.11}
\]

\[
y_j = \hat{V}^j \cdot \tilde{Z}
\]

69
It should be noted that the data point \( z \) may or may not be part of the data set. The expressions so far are all in terms of the centred data in feature space. Since we may not have access to feature space, explicit computation of the centred data may not be feasible. As a result, we need to expand the expressions so that they are expressed in term of the original non-centred data in feature space.

The expression (B.5) for the elements of matrix \( \tilde{K} \) can be further expanded to a desired form which consists of only atomic dot product of vectors in feature space:

\[
\tilde{K}_{ij} = \tilde{X}_i \cdot \tilde{X}_j = \left( X_i - \frac{1}{M} \sum_{m=1}^{M} X_m \right) \cdot \left( X_j - \frac{1}{M} \sum_{n=1}^{M} X_n \right)
\]

\[
= X_i \cdot X_j - \frac{1}{M} \sum_{n=1}^{M} X_j \cdot X_n - \frac{1}{M} \sum_{m=1}^{M} X_i \cdot X_m + \frac{1}{M^2} \sum_{m=1}^{M} \sum_{n=1}^{M} X_n \cdot X_m \tag{B.12}
\]

By defining another square matrix \( K \) of dimension \( M \) with its elements being:

\[
K_{ij} = X_i \cdot X_j = \Phi(x_i) \cdot \Phi(x_j) = K(x_i, x_j)
\]

Expression (B.12) can be rewritten as:

\[
\tilde{K}_{ij} = K_{ij} - \frac{1}{M} \sum_{n=1}^{M} K_{in} - \frac{1}{M} \sum_{m=1}^{M} K_{mj} + \frac{1}{M^2} \sum_{m=1}^{M} \sum_{n=1}^{M} K_{nm}
\]

With this, \( \tilde{K} \) can be written in the following expression with matrix \( M \) being a square matrix of dimension \( M \) with all its elements as \( 1/M \):

\[
\tilde{K} = K - KM - MK + MKM
\]

Similarly, the expression (B.11) for projection kernel principal component can be expressed in a form with only atomic dot product of vectors in feature space as below:

\[
y_j = \hat{V}^j \cdot \tilde{Z}
\]

\[
= \left( \frac{1}{\sqrt{\gamma_j}} \sum_{k=1}^{M} \alpha_k^j \tilde{X}_k \right) \cdot \tilde{Z}
\]

\[
= \frac{1}{\sqrt{\gamma_j}} \sum_{k=1}^{M} \alpha_k^j (\tilde{X}_k \cdot \tilde{Z})
\]

\[
= \frac{1}{\sqrt{\gamma_j}} \sum_{k=1}^{M} \alpha_k^j \tilde{K}_{proj}^k
\]

\[
= \frac{1}{\sqrt{\gamma_j}} \tilde{K}_{proj}^j \alpha^j
\]
where $\tilde{\mathbf{K}}^{\text{proj}}$ is a $1 \times M$ matrix with its elements being:

$$
\tilde{\mathbf{K}}^{\text{proj}}_i = \tilde{\mathbf{X}}_i \cdot \tilde{\mathbf{Z}} \\
= \left( \mathbf{X}_i - \frac{1}{M} \sum_{m=1}^{M} \mathbf{X}_m \right) \cdot \left( \mathbf{Z} - \frac{1}{M} \sum_{n=1}^{M} \mathbf{X}_n \right) \\
= \mathbf{X}_i \cdot \mathbf{Z} - \frac{1}{M} \sum_{n=1}^{M} \mathbf{X}_i \cdot \mathbf{X}_n \mathbf{Z} - \frac{1}{M} \sum_{m=1}^{M} \mathbf{Z} \cdot \mathbf{X}_m + \frac{1}{M^2} \sum_{m=1}^{M} \sum_{n=1}^{M} \mathbf{X}_n \cdot \mathbf{X}_m \tag{B.13}
$$

By defining $\mathbf{K}^{\text{proj}}$ as a $1 \times M$ matrix with elements:

$$
\mathbf{K}^{\text{proj}}_i = (\mathbf{X}_i \cdot \mathbf{Z}) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) = K(\mathbf{x}_i, \mathbf{z})
$$

$\tilde{\mathbf{K}}^{\text{proj}}$ can be expressed in terms of $\mathbf{K}^{\text{proj}}$ as shown in the following:

$$
\tilde{\mathbf{K}}^{\text{proj}} = \mathbf{K}^{\text{proj}} - \mathbf{K}^{\text{proj}} \mathbf{M} - \mathbf{M}^{\text{proj}} \mathbf{K} + \mathbf{M}^{\text{proj}} \mathbf{K} \mathbf{M}
$$

where $\mathbf{M}^{\text{proj}}$ is a $1 \times M$ matrix with all its elements being $1/M$
Appendix C

A Paper Presented in CBAIVL-2000
: Content Based Image Retrieval through Object Extraction and Querying

A.H.Kam, T.T.Ng, N.G.Kingsbury and W.J.Fitzgerald
Signal Processing Laboratory
University of Cambridge Engineering Department

abstract

We propose a content based image retrieval system based on object extraction through image segmentation. A general and powerful multiscale segmentation algorithm automates the segmentation process, the output of which is assigned novel colour and texture descriptors which are both efficient and effective. Query strategies consisting of a semi-automated and a fully automated mode are developed which are shown to produce good results. We then show the superiority of our approach over the global histogram approach which proves that the ability to access images at the level of objects is essential for CBIR.

Introduction

Image retrieval has traditionally been based on manual caption insertion describing the scene which can then be searched using keywords. Caption insertion is a very subjective procedure and quickly becomes extremely tedious and time consuming, especially for large image databases which are becoming ever more common with the growing availability of digital cameras and scanners. There is thus an urgent need for effective content-based image retrieval (CBIR) systems.
We believe the key to effective CBIR performance lies in the ability to access the image at the level of objects. This is because users generally want to search for images containing particular object(s) of interest and thus the ability to represent, index and query images at the level of objects is critical [7].

In this paper, we present a framework for CBIR based on unsupervised segmentation of images into classes and querying using properties of these classes. As these segmented classes are homogeneous in some sense (in our case, colour and texture), they correlate well with the identity of objects. By decomposing images as combinations of objects in this manner, querying becomes more meaningful and intuitive than it is with global image properties. This is obviously true for images with distinct foreground objects but the rationale also holds for ‘background’ images where no interesting foreground objects are present. Images belonging to the latter category can be thought of consisting of combinations of classes with homogeneous colour and texture (for example, images of the seaside generally consist of the beach and the sea, images of sunset scenes generally consist of the reddish sky and dark silhouettes and so forth) and querying is made more effective by being based on these class combinations which characterise the scene.

In our CBIR implementation, images are firstly segmented based on joint colour and textural features using our previously developed unsupervised multiscale segmentation algorithm [20], [21]. The segmentation process is completely unsupervised and performed off-line for each image. Following this, we represent each image using effective and compact colour and textural descriptors of its classes. We then structure the descriptor database following a relational model which allows its implementation on powerful relational database engines. Class attribute queries are processed using a parallel strategy which results in significant speed-up in the retrieval process if parallel processor machines are used.

In Section 2, we will briefly describe the segmentation algorithm employed. We will then discuss the descriptors assigned to each class in Section 3. In Section 4, we present our query strategy as well as preliminary results from queries on our image database testbed consisting of various natural images.

**Unsupervised Segmentation**

Our unsupervised segmentation algorithm involves the following steps:

1. Normalised colour and texture features (three for colour and two for texture) are mapped to a multidimensional feature space. Spatial information is incorporated
into the process by including spatial features into the feature space. The colour space used is S-CIE L*a*b*, the spatial extension of the perceptual uniform CIE L*a*b*, originally developed by Zhang and Wandell [42]. This colour space takes into account the appearance of fine-patterned colours on the human visual system. Textural features meanwhile are generated using the logarithm of the energies of the 2-D complex wavelet coefficients [22]¹ and taking the top two principal components.

2. Significant features which correspond to clusters in the feature space are assumed to be representations of underlying classes, the recovery of which is achieved using the mean shift procedure [12], a robust kernel based decomposition method. The kernel size used was fixed for all images, resulting in a decomposition into an appropriate number of classes for each image.

3. By determining the number of classes and the properties of each class via step 2, a Bayesian multiscale processing approach, which models the inherent uncertainty in the joint specification of class and position spaces using the Multiscale Random Field model [5], is used for the subsequent classification process.

Typical segmentation maps of images in our database are shown in Figure C.2.

![Figure C.2: Typical segmentation maps of images](image)

**Describing the Classes**

Once an image has been segmented, we proceed to extract a description of each class with the total description of classes constituting a description of the image. A class descriptor has to embody the class characteristics (which typically translates to representing

¹For more information about the 2-D complex wavelet transform, please visit: [http://www-sigproc.eng.cam.ac.uk/~ngk](http://www-sigproc.eng.cam.ac.uk/~ngk)
a particular object) in an effective fashion to facilitate efficient indexing and accurate retrieval. Thus, designing an effective class descriptor is more difficult than designing feature extractors for segmentation and thus they should be seen as separate processes.

**Colour Descriptors**

In order to represent the colour distribution of each class, we store the colour histograms of the pixels of the class. This histogram is based on bins with width 10 in each dimension of the S-CIE L*a*b* colour space. This spacing yields 10 bins in the L* dimension and 40 bins in each of the a* and b* dimensions, for a total of 90 numbers as colour descriptors.

To evaluate the dissimilarity between the colour histograms of two classes/objects, we apply the *Kolmogorov-Smirnov* (K-S) distance, as originally proposed in [16]. The K-S distance essentially measures the difference between two probability distribution functions. If $F_1(k)$ and $F_2(k)$ are two independent sample distribution functions (i.e. histograms) defined such that:

$$F_i(k) = \frac{1}{n} \#(i : y_i \leq k) \quad (C.1)$$

where $n$ is the number of data samples, $y_i$ so that $1 \leq i \leq n$, then the K-S distance is the maximum difference between the distribution over all $k$:

$$\text{K-S } (y^1, y^2) = \max |F_1(k) - F_2(k)| \quad (C.2)$$

The overall colour dissimilarity measure between two classes with colour histograms $x^{COL}$ and $y^{COL}$ is taken to be the root mean square of the K-S distances of each of the L*, a* and b* histograms:

$$d^{COL}(x^{COL}, y^{COL}) = \frac{1}{3} \left\{ \left[\text{K-S } (x^{COL}_{L*}, y^{COL}_{L*})\right]^2 + \left[\text{K-S } (x^{COL}_{a*}, y^{COL}_{a*})\right]^2 + \left[\text{K-S } (x^{COL}_{b*}, y^{COL}_{b*})\right]^2 \right\}^{\frac{1}{2}} \quad (C.3)$$

As the range of K-S distances lie between 0 and 1, the colour similarity measure, $s^{COL}(x^{COL}, y^{COL})$ is simply taken as:

$$s^{COL}(x, y) = 1 - d^{COL}(x, y) \quad (C.4)$$

**Texture Descriptors**

For each class, texture is described by the distribution of the magnitude of its complex wavelet coefficients, $f(x^{TEX})$. Using four levels of the 2-D complex wavelet transform (which yields six complex subbands at every level) produces a total of 24 subbands, the magnitude of each is converted into a histogram and modelled as a generalised Rayleigh distribution (see figure C.3):

$$f(x_i^{TEX}) = k_i x_i^{TEX} \exp\left(-\frac{x_i^{TEX}}{\sigma_i}\right)^{\beta_i} ; \ i = 1, 2, ..., 24 \quad (C.5)$$

where, to achieve the same mean and variance as the input sample distribution:
• $k_i = \frac{\beta_i}{\sigma_i^2 \Gamma\left(\frac{2}{n_i}\right)}$ where $\Gamma(\cdot)$ denotes the gamma function

• $\sigma_i = \frac{m_1 \Gamma\left(\frac{3}{n_i}\right)}{\Gamma\left(\frac{3}{n_i}\right)}$ where $m_1 = E(x_i^{TEX})$

• $\beta_i = F^{-1}\left(\frac{m_2}{m_2}\right)$ where $m_2 = E\left[(x_i^{TEX})^2\right]$ and $F(x) = \frac{\left[\Gamma\left(\frac{3}{n_i}\right)\right]^2}{\Gamma\left(\frac{5}{n_i}\right)\Gamma\left(\frac{3}{n_i}\right)}$

![Figure C.3: Top row: Image of a typical tiger and extraction of the class corresponding to the creature. Bottom row: Histograms of the magnitude of the tiger’s complex wavelet coefficients (every row depicts a decomposition level with level 1 at the top) and the modelling performance of the generalised Rayleigh distribution (plotted in red) ](image)

Thus, for each class, the generalised Rayleigh model parameters, $\sigma_i$ and $\beta_i$ is calculated for each of the 24 histograms, for a total of 48 numbers as texture descriptors. To compute the texture dissimilarity between two classes, we:

1. Generate probability distribution functions for each of the 24 subbands of each class using the stored values of $\sigma_i$ and $\beta_i$.

2. Apply the K-S distance between histograms corresponding to the same subband.
3. The overall texture dissimilarity measure is calculated as the root mean square of all the K-S distances. The final similarity measure is given by the subtraction of the dissimilarity measure from unity.

**Image Retrieval**

The class descriptor database is structured using a relational model. This allows its implementation on powerful commercial relational database engines and for queries and retrieval to be described using SQL’s *Select* and *Join* operations [37]. For example, as the first step, descriptors of particular classes of an image can be extracted from the database using a simple *Select* operation.

**Querying Strategy**

There are two modes of operations for our image retrieval system: a semi-automated mode and a fully automated mode. In the semi-automated mode, the user composes a query by submitting an image and by seeing the segmentation map, selects the class or classes to match. There is also an option of selecting the relative importance of the classes (should there be more than one in the query composition); by default, all classes in a query are considered equally important.

All ‘compound’ queries, i.e. queries being based on more than one class, are firstly decomposed into ‘simple’ queries, i.e. queries based on a single class. The similarity match for each simple query is calculated as follows:

1. Colour and texture descriptors for the queried class are retrieved from the descriptor database
2. The similarity measures for colour and texture are computed for classes in the database whose sizes (specified as a fraction of the image) are at least 25% of the queried class
3. The overall similarity measure is taken to be the weighted combination of the colour and texture similarity measures, with the weights set by the user. By default, colour and texture similarities are weighted equally

The SQL *Join* operation on the simple queries’ match lists will obtain the set of common images, with the best match maximising the similarity measures, weighted according to their relative importance. As simple queries can be processed in parallel, significant speed-up in the retrieval process is possible with parallel processor machines.

In the fully automated mode, the user has only to submit a query image and the algorithm is designed to handle the rest. In this case, we first perform simple queries on classes of the image which constitute at least 10% of the image. In the absence of a theoretical foundation to determine the relative importance of the classes, we simply sum
up the top 10 similarity measures of the match lists of each of these classes. This step will provide us information as to which classes have relatively high matching scores and thus possess a higher probability of being an ‘object of interest’. Finally, a compound query is performed on the top two classes of the image with the highest matching scores.

Results

We have performed a variety of queries on our small image database testbed for both the semi-automated and the fully automated mode. Preliminary results are encouraging as shown in Figures C.4 and C.5. We are currently in the process of expanding our image database to include as many varied natural images as possible.

Figure C.4: Example of a one-class default query: Query image and the selected class (with green borders, top left), with the retrieved images, depicted with the matching class, arranged from highest similarity, from left to right, top to bottom

Figure C.6 depicts retrieval precision and recall performance of several image categories of our system. Results were generated using a one-class default query for zebra and tiger images while automated retrieval was utilised for sunset and autumn scene images where no interesting foreground objects are present. Retrieval performance is particularly good for sunset and zebra images while results for tiger and autumn scene images aren’t too bad either.
We also compared the performance of our method with and without the pre-segmentation stage (i.e. in the latter case, querying based on global histograms). For fair comparison, the fully automated mode is used for the approach with the pre-segmentation stage.

The table below compares the image retrieval rates between the two approaches for leopard, bear, sunset and winter scene images. The approach with the pre-segmentation stage performs better for all image categories tested although the global histogram approach produces reasonable results especially for sunset images. These results are consistent with our belief that the key to effective CBIR performance lies in the ability to access images at the level of objects.

<table>
<thead>
<tr>
<th>Image Categories</th>
<th>Precision values based on the top 15 images returned</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre-Segmented</td>
</tr>
<tr>
<td>Leopard</td>
<td>40%</td>
</tr>
<tr>
<td>Bear</td>
<td>27%</td>
</tr>
<tr>
<td>Sunset</td>
<td>93%</td>
</tr>
<tr>
<td>Winter</td>
<td>78%</td>
</tr>
</tbody>
</table>

Discussion

We have proposed a content based image retrieval system based on classes of pre-segmented images. A general and powerful multiscale segmentation algorithm automates the segmentation process, the output of which is assigned novel colour and texture descriptors which are both efficient and effective. We then discussed our query strategy, for both the semi-automated and fully automated mode and demonstrated its encouraging results.
Nevertheless, low level features like colour and texture are generally insufficient for effective retrieval of unconstrained imagery. We are currently in the process of incorporating high-level descriptors into our method by training our system using support vector machines [6]. We believe this inclusion of semantics into the query and retrieval process will be the key to successful CBIR in the future.
Bibliography


