University of London
Imperial College of Science, Technology and Medicine
Department of Computing

Discovering images: features, similarities and subspaces

Peter D Howarth

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of the University of London and
the Diploma of Imperial College, June 2007

Abstract

This thesis investigates three of the core components of content-based image retrieval: visual features, similarity functions and indexing methods. In the content-based paradigm images are searched in a purely visual domain, where they are represented by high-dimensional features. Exhaustively searching this feature space can take a prohibitively long time. I argue that by the use of judicious approximations we can search large collections interactively, while keeping a good level of retrieval performance. I first carry out a thorough investigation of texture features. This results in novel feature modifications which give significant improvement in retrieval performance. The adaptations are based on considering the representation of texture within whole, real images. Secondly, I present the first application of fractional dissimilarity functions to image retrieval. These functions emphasise points close to the query in each dimension, while reducing the noise from others. Through a thorough evaluation we show that these functions give a consistent improvement in retrieval performance across a wide range of collections and features. Following this theme, I generalise a class of local similarity functions that exhibit similar behaviour. These functions decompose features into dimensions and index these independently. By selecting only the points local to the query in each dimension, we can ignore a large proportion of data and maintain retrieval performance. Experiments with five different image collections show that the optimal proportion of each dimension, to maximise retrieval performance, decreases from 10% to 0.1% — with a corresponding collapse of computing cycles — as the collection size increases from ten thousand to one million images. The culmination of this work brings together the separate strands of research to implement an image search system. We demonstrate that, using normal hardware, this system was able to search four million images in just over one second, thus satisfying the goal of effective real-time searching of large image collections.

Acknowledgements

There are many people who have helped me during the, considerable, ups and downs of my time as a PhD student. I would like to thank them all for the positive influence they have had.

- Firstly, I would like to thank my supervisor, Stefan Rüger, for his guidance and encouragement over the last three years.
- My examiners Arjen De Vries and Frederic Fol Leymarie gave me a challenging, stimulating and enjoyable viva, that provided a fitting end to the work.
- Also, I must thank the EPSRC for partially funding my research.
- My colleagues Alexei, João, Simon, Ed, Daniel, Marcus and Shyamala, in the Multimedia and Information Systems research group at Imperial, have provided invaluable help and inspiration that has been vital to my progress.
- The other occupants of the PhD lab, Paul, Uri, Dave, Alok, Georgia and Mohammad, have helped me keep my sanity. They, together with my colleagues, generated many lively discussions in the lab and tea bar. These times are some of my fondest memories of the last three years.
- Although she knew nothing about it, my daughter Georgina has been the nicest possible distraction.
- Finally, and most importantly, thanks to my wife Rachel for her everlasting support and constant motivation. Without her nothing would be possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent years digital imaging technology has improved in performance and become more accessible. This, coupled with a reduction in storage and network costs, has led to significant changes in the way that both individuals and organisations create and share pictures.

For personal photography the advent of digital cameras and video camcorders has resulted in the virtual elimination of film, processing and paper. This has had the double effect of both lowering the cost and allowing easy distribution of images. Correspondingly, peoples' behaviour has changed; they are taking many more pictures and sharing these widely. These images can be in personal picture collections, shared on-line with applications such as Flickr[1], or just added to the vast repository of servers connected to the Internet. Recent phenomena include the use of eye-witness photographs and video to report news events as they happen.

Many organisations are creating vast libraries of digital images. For example, libraries and museums are digitising their existing photograph collections and making these available, both within their buildings and over the Internet, to a much wider audience. Images of artifacts and documents that were once almost entirely the preserve of a privileged few are now available to anyone with a computer and access to the world wide web.

This explosion of digital imaging has created the need for new ways to search picture collections. In many circumstances there may simply be too many for individuals to look at every one. People used to be able to browse through a shoebox of prints; now that shoebox is digital and may contain tens of thousands pictures. On shared Internet applications this number can easily increase to tens of millions.

Some collections may have textual metadata associated with images. These tags can range from the

[1] http://www.flickr.com

detailed, accurate cataloguing of a museum library, to a few keywords that a user thinks describe an image they have just uploaded to a web application. This metadata can be very useful for finding images. However, it has limitations. To generate the comprehensive and precise tagging needed by a library each image needs to be assessed by an expert, thus, making the production of the metadata time consuming and expensive. In contrast, the keywords entered by a person who took an image are less accurate, but easy to create. The quality of metadata has a direct impact on the accuracy of any search. Moreover, a given set of tags locks any subsequent search into the concepts determined by the original tagger. If the information need of the user does not fit these concepts then satisfying their query may be difficult.

These problems highlight the need for retrieval based on the *content* of an object. For a digital image this content comes entirely from information within the pixel representation: visually this may correspond to colours, textures and shapes. Basing search on visual content obviates the need for metadata, thereby opening up unannotated collections for users to search. In addition, where collections are already tagged it provides an ability to search and browse in the visual dimension, lifting the constraints of a purely textual representation.

1.2 Objectives

The aim of the work presented in this thesis is to research methods that will allow CBIR to be used effectively with large image collections (greater than one million images). From this goal we get the more detailed objective of researching the properties of three core components of content-based image retrieval (CBIR). These are, the visual features used to extract content from a picture, the functions used to quantify the similarity between features and the indexing methods used to facilitate a search. Two key performance areas are addressed, retrieval accuracy and speed of search. Combined together these are fundamental for building a practical CBIR system that can return usable results in a realistic timescale.

Both the speed and accuracy of the CBIR method can be compromised by the high-dimensional nature of the feature vectors representing visual content. The effects of high-dimensionality can reduce the meaningfulness of a nearest neighbour search and the effectiveness of indexing techniques (Chapter 5 Section 5.2). However, the task of image search is, by its nature, inexact. The approach in this thesis is to consider where accuracy is needed and where it is superfluous to performance. By relaxing the need for accuracy where it has no benefit we can find ways to address the scalability problem and still maintain the effectiveness of CBIR as an image search and browsing tool.

1.3 Outline of this thesis

This thesis presents a body of work researching CBIR by focusing on the two goals of retrieval effectiveness and efficiency (speed of search). The end result is a tool-set for CBIR comprising of visual features, similarity functions and an indexing method. Together these enable rapid search and browsing of large generic image collections. The task of CBIR is subtly different to that of many other vision applications. Focusing on this specific task is a common thread through the work and has led to several novel adaptations. The strengths and limitations of the work are highlighted throughout the thesis. The thesis is structured as follows:

Chapter 2 first discusses the task of image retrieval. CBIR is often criticised for its lack of performance when compared to metadata-based searches. These problems are said to stem from the semantic gap between a user need an the content-based representation. To counter these criticisms several scenarios are discussed to highlight the strengths and weaknesses of the various image search techniques. From this we see that CBIR has its rightful place in the panoply of methods. A general CBIR system is then described followed by a discussion of each component in turn.

Chapter 3 describes the experimental methods employed throughout this work. The bulk of the chapter is taken up with a discussion on image collections and evaluation approaches. Understanding the effect of these is important as it impacts the applicability of any conclusions. A range of four very different collections have been used. This gives some assurance that findings are not just a feature of the collection. In addition, to evaluate the performance of indexing methods a large image collection sourced from the web was created. This had no ground truth and, hence, necessitated a new evaluation approach. The options for this are discussed. The commonly used information retrieval evaluation approaches using precision and recall are described, together with the use of statistical testing of results. Finally some additional methods and features, used in the experiments, are described.

Chapter 4 looks in detail at the first core component of CBIR, features. Specifically the focus is on features that represent texture. The concept of texture is described followed by a taxonomy of texture extraction methods used in computer vision. The experimental work modifies and tunes existing features to show how it is important to consider their use in the context of the image retrieval domain. In this case the features are representing whole, real images, rather than the more usual task of classifying homogeneous texture tiles. From this some novel modifications to the features were developed together with general guidelines for building texture features for CBIR.

Chapter 5 investigates localised similarity functions: classes of functions that emphasise the points local to a query whilst reducing the impact of noise from distant points. This property has the effect of emphasising clusters within the feature space. High dimensionality has a pervasive effect on CBIR; its impact on nearest neighbour search is set out. The commonly used distance measures are summarised. This leads into an argument for localisation of similarity functions.

Two classes of functions are investigated: fractional dissimilarity functions and local similarity functions. Detailed experimental results show how these functions behave with a real IR task. Further experiments show how it is possible to find representative local subspaces by considering which elements of the query are atypical. These are used to further localise the search. This chapter also contains a topological interlude discussing the non-metric properties of fractional functions and a proof that local similarity functions can be made metric if desired.

Chapter 6 takes the local similarity function investigated in the previous chapter and implements it as a practical indexing method for CBIR. These functions have the exciting property that a large proportion of the data is discarded for each search. This reduces the amount of data loaded from disk, which is the major speed bottleneck with image search. Existing approaches to indexing high-dimensional features are discussed to set the work in context. Alternative implementations for the local similarity index are put forward. This is followed by an evaluation of both speed and average precision of search as the size of a collection increases.

Chapter 7 draws some conclusions from the work and ideas in this thesis, highlighting both successes and limitations. Recommendations for future research to extend the usefulness of the local search methods are set out. These focus on the use of query sensitive searches to prune the search space whilst maintaining retrieval performance.

1.4 Contributions

This thesis presents new work across several elements of the CBIR tool chain. The points below are novel ideas and results published in this thesis and the papers listed in the following section:

1. Tuning and adaptation of texture features for CBIR: specific novel modifications included introducing logarithmic scaling for the Tamura coarseness feature and an entropy measure for Tamura's directionality feature (Section 4.4.5);

2. Identified concise and general principles for constructing texture features for CBIR, briefly these are: looking for micro-textures in images as informative textures occur at small scales; keeping orientation information as it is an important discriminator; isolating smaller texture regions using image tiling improves performance; and, using less detailed features as this may produce more robust performance (Section 4.5);

3. The first application of fractional dissimilarities to CBIR: A thorough evaluation across a wide range of data sets and features showed that these functions consistently outperformed all other L_p-norm distances; found that a parameter of $p = 0.5$ produced robust performance in nearly all circumstances (Section 5.5.2);

4. Introduced the use of local similarity functions for CBIR and showed experimentally that their performance is comparable to L_p distance measures whilst using only 5–10% of the feature space (Section 5.5.3);

5. Showed that selection of atypical dimensions when using local similarity improves retrieval performance; this enables the amount of feature space searched to be reduced further (Section 5.5.3);

6. When using localised similarity functions for CBIR, demonstrated that it is the membership of the local neighbourhood that is important rather than the details of the distance function; this led to the introduction of localised similarity using a simple voting function as the basis for a fast search method (Section 5.5.3);

7. Proved the metric properties of a set of local distance functions, including when using a voting function (Section 5.4.4);

8. Showed empirically that the optimal neighbourhood size, for a local similarity index, reduces as the size of the collection increased; correspondingly, the index gave meaningful retrieval results when searching over one million images using a neighbourhood of 0.1% of each dimension (Section 6.4.1);

9. Demonstrated that a test implementation of the local index could search four million images in just over one second (Section 6.4.2).

1.5 Publications

Many of the results presented in this thesis have already been published. These group naturally into three categories, given below:

Texture features. Work on modifying and evaluating texture features for content-based image retrieval was presented in Dublin, at the Conference on Image and Video Retrieval (Howarth and Rüger, 2004). This work was extended with additional experiments and published as a journal paper in the IEE Proceedings on Vision, Image and Signal Processing (Howarth and Rüger, 2005c).

Similarity. The use of fractional dissimilarity functions for image search was presented at the European Conference on Information Retrieval (Howarth and Rüger, 2005a). This was followed by work on localised similarity, which was published as part of the Conference on Image and Video Retrieval in 2005 (Howarth and Rüger, 2005b). These two pieces of work were unified and enhanced with experiments on the implementation and performance of a practical index, searching over one million images. This was written up as a journal paper, which has been submitted for review (Howarth and Rüger, 2007).

Evaluation exercises. The features and similarity functions developed in this work have been evaluated extensively. Some of this was within the framework of formal evaluation exercises. The first of these was as a submission to the image track of the Cross-Language Evaluation Forum 2004. The details of the submission were initially published as a notebook paper (Howarth et al., 2004) and then following additional investigation in formal proceedings (Howarth et al., 2005). The texture features have been used for submissions to several TRECVID submissions including in 2004 (Heesch et al., 2004).

1.6 Statement of originality

The work presented in this thesis is my own, except where otherwise stated.

Chapter 2

Image search

This chapter considers the task of image search. It starts with a comparison of content and metadata based methods. Their strengths and weaknesses are highlighted by considering different image search scenarios. This leads to a clear view of the application areas suited to CBIR. Following this, a generic CBIR system is described with each component considered in turn. This chapter provides an overview of the CBIR landscape; each of the later chapters contains background specific to the work they describe.

2.1 Image search

The rapid growth in the quantity and availability of digital images provides a motivation for research into ways of automating image search. The underlying techniques of image search can be divided into two broad camps, those based on metadata and those that use the content of the image.

Metadata approaches can be further subdivided into how images are annotated, manually or using automated methods. In general, the generation of accurate metadata is expensive as it takes the time of an expert cataloguer. The alternative of automated annotation, using statistical learning or object recognition methods, can make annotation much cheaper. These methods are a hybrid of the two image search approaches as they use the content of the image to generate keywords, and then search using this metadata (van Gemert et al., 2006; Yavlinsky and Rüger, 2007). If automated annotation techniques could reliably identify a wide range of objects and scenes then they could form the basis of a system that may solve the image search problem. However, at present these methods can only identify a relatively small vocabulary of concepts with limited accuracy.

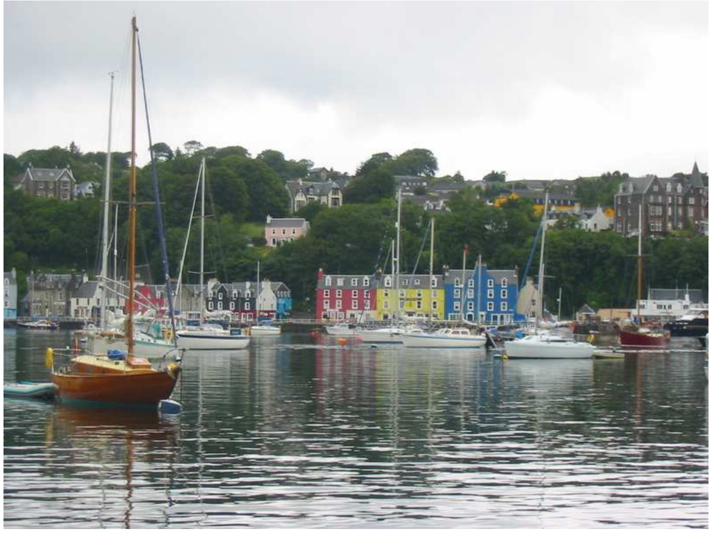


Figure 2.1: A polysemous image

2.1.1 The semantic gap

The semantic gap is the difference between the *real* meaning of an object and its representation. It naturally occurs wherever computers are used to model complex objects. It can lead to limitations in all information retrieval applications. For example, consider a text based search; it is easy to find documents containing the word sorrow, but much harder to find prose that expresses sorrow. The documents in a retrieval system are likely to be indexed by the words they contain rather than their emotional meaning. So there is a gap between the document's representation as symbols (words) and its meaning to a reader.

With text retrieval a user can quickly understand how the system works and to some extent format their queries to work around the problem. With image retrieval the manifestation of the semantic gap is much more fundamental. To a computer an image is just a sequence of bytes, like any other data. The sequence represents a two dimensional arrangement of pixels, each with a colour and intensity value. This data is all the information that an image retrieval system has to work with.

But what does a picture mean to the human user? Even for a seemingly simple image this can be many things. Understanding can be at the detailed level of objects within the image: it can represent the whole scene, a time of day, an emotion or a more abstract concept. For example the image in Figure 2.1 may mean, houses, water, boats, sky; village, harbour, Scotland, colourful; or calm, peaceful, reflection, holiday. The meaning of a picture can vary for individual viewers depending on its context, their prior experience, knowledge or cultural influences.

Even when a person views an image and annotates it with a textual meaning the keywords they use depend on their own skill with language, subject knowledge and circumstances. So metadata can suffer from the semantic gap. However, when an image is represented purely by its content the gap is more fundamental and much larger.

These varying concepts and meanings of a picture are very difficult to capture or to represent in a query. The major criticism of content-based approaches is that they they do not meet the real need of the user. Study of the information need of users showed that the majority of queries contained a semantic component that required a deep understanding of the image (Armitage and Enser, 1997). In a content-based approach this semantic meaning cannot be easily captured, either in an example image used as a query, or the images in the collection. So it appears that using the visual content in the image alone is not sufficient to enable a search on the semantic meaning of the image. The final conclusion of this argument is that the only way to enable a fulfilling image search is using accurate metadata. Enser and Sandom (2002) used this same argument to come to their pessimistic conclusion that retrieval of video footage required human intellectual input and that CBIR offered little in this respect.

(Smeulders et al., 2000) presented a rounded discussion on the scope of content-based methods. They split the search task into three types: target search, category search and search by association. Target search is based on pattern matching using visual features. This is the method tackled in this thesis. Category search includes classification and object recognition using supervised learning techniques. The final type of search is where the user associates a semantic meaning with an image. This case is where the gap between the visual representation in a computer and the understanding of the user are greatest.

2.1.2 Image search scenarios

It would seem from these arguments that the usefulness of CBIR is limited. However, the visual nature of content-based methods gives them their strength. A good way to investigate the properties of each method is to consider different image search problems and how these can be solved.

Historic image library. An image library is likely to have high quality metadata created by librarians experienced in cataloguing objects. The keywords they associate with each image will encapsulate the meaning of the image. Users of the library are likely to be searching for very specific images. For instance the user may want to find an image of the first black football player to play for England[1]. This search can only be satisfied using accurate and detailed cataloguing.

Web image search. Another scenario is that the user wants to find pictures of a specific object from the web. An example could be that they are looking for pictures of an angelfish. They would like to use a single word query as this adequately describes their information need. The image collection on the web is very large and evolving over time so it is not practical for an expert to annotate the images. However, there is a lot of contextual information from the web page itself. Keywords can be extracted from the filename, the URL and the text associated with the image.

[1] The Nottingham Forest defender, Viv Anderson, on the 29 November, 1978, against Czechoslovakia at Wembley Stadium

This is the general idea used by web based image search, such as Google[2]. In general, searches using this method return highly relevant images at the top of their results list, but miss a large proportion of the total available relevant images: high precision and low recall (Section 3.3.1).

Digitising an existing photograph library. It is common that a museum that is digitising a collection of photographs of artifacts may only have the resources to create metadata for a small proportion of the images. A text search using this metadata will be accurate, but only give access to a limited selection of images. To make the bulk of the collection available the museum could provide a content-based search. This would enable a user to search and browse the otherwise inaccessible unannotated images. A user searching for images could use an initial text search and then choose a image from the results as a query for the content-based search.

Personal image collection. An individual can quickly build up a large collection of pictures. Before the advent of digital photography a few of these would be displayed in frames or sorted in albums, but the bulk would remain stored in a shoebox or similar. The digital age has not changed this, except now the shoebox is likely to be a directory on a hard disk. Unless a person is very methodical their images are likely to become unannotated and unorganised with nondescript filenames. Here content-based search is the only option other than browsing thumbnails. The user can pick an image as a query, view the results, refine the query and re-search. In this way CBIR provides a method of searching and browsing an otherwise inaccessible collection.

Amateur video. Video shot by non-professionals presents a different search problem. Unless accurate records are kept the subject matter on miniDV tapes and in video files will quickly be forgotten. To search these manually will be a time-consuming process as the video must be viewed. Content-based search presents an alternative indexing method for video. Shot boundaries can automatically be detected and keyframes for each shot extracted from the video. These can then be searched using their content in the same way as any other image collection.

From this discussion we can see that each method has its strengths and weaknesses. The potential of content-based image retrieval lies in its ability to give access to large unannotated collections of images. It can be a powerful tool to employ on its own or in combination with metadata. This can be in a situation where the content-based search piggybacks off an initial text search or where text is used to filter the results of a content-based search. On its own CBIR offers the user an ability to search in the visual domain. This is orthogonal to a text search. Although using a visual search tool may not be as familiar as searching using text it can provide the user with access to parts of the collection that would otherwise be hidden to them. The results can often be better than a user expects. They may attribute

[2] http://images.google.com/

this to serendipity, however, this is really a characteristic of the of the behaviour of CBIR, in that it retrieves images where the visual similarity is real, but not necessarily semantically obvious.

2.2 The content-based approach to image retrieval

Vision is the most highly developed human sense and the one we rely on most to get an appreciation of the external world. Except in total darkness, our eyes are constantly feeding a vast amount of data into our visual system. The processing of this data by the brain enables us recognise objects, see motion, reach for things and build an internal map of our surroundings. At a higher level we can detect emotions in others and appreciate visual art. We can do this with amazing speed and accuracy. A glimpse of just a fraction of a second is enough to let us see, interpret and understand a scene. To do this a large proportion of the brain is devoted to vision. One part of the system, the visual cortex, occupies a large proportion of the cerebral cortex. Vision scientists and psychologists have researched many aspects of how vision works, but we are still a long way from having a definitive understanding (Bruce et al., 1996).

One model of human vision splits the processing into two phases. The first of these is named the preattentive phase. Treisman (1986) described preattentive processing of visual information as being performed automatically on the entire visual field detecting basic features of objects in view. It is done quickly without any need for focused thought. After this phase the system enters an attentive mode, recognising objects and generating a semantic meaning.

In this model of vision we can see that the initial stage of human vision is based on the content of what we see. Content-based systems can be thought of as trying to mimic the preattentive phase of vision. With CBIR the basic features of colour, texture and shape are extracted from a set of images and these compared to each other to give a measure of similarity between the original images. Given the adaptability, power and speed of the human visual system and our gaps in knowledge of how it works, it would be ambitious to claim that computer systems are capable of performing all the same tasks. However, CBIR has the potential of being able to visually compare and sort large collections of images very quickly. In this way it can be a useful adjunct to human vision in that it can enable the visual comparison of a large number of images that a person would otherwise be unable to view.

2.2.1 System overview

The components of a typical CBIR system are shown in Figure 2.2. This diagram is an adaptation of the generic retrieval model presented by van Rijsbergen (1979). At the centre of a CBIR system are features and similarity. These two components are in fact key to the performance of a system as they are the parts that contain the visual information and similarity judgements. In the following sections each of the building blocks of CBIR is discussed.

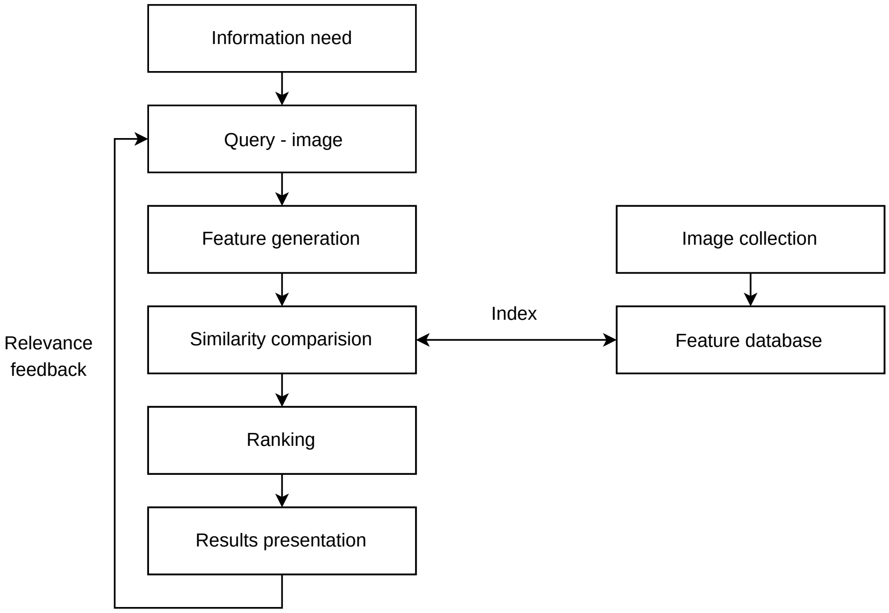


Figure 2.2: A generic image search system

2.2.2 The information need and query

The first problem to consider with image retrieval is, given an information need, how to specify a query. For CBIR the query must be translatable into a content representation. For this reason queries are generally specified visually. This can be with an example image or perhaps a sketch. The obvious shortcoming of using an example image is where to get it from. Sketching out a query can overcome this problem. A good example of this in action is the *retrievr* application[3] based on work done by Jacobs et al. (1995). Although fun to use, in reality sketch-based methods are not always feasible for a general search application: it takes both time and skill to draw a query. So, as a practical choice, we are left with the "query by example image" paradigm. This is widely used for CBIR. It can naturally develop into a browsing approach, where the user selects further suitable query images from the initial results, thus refining the search.

The advantage with using an image as a query is that it directly converts the query into a pixel representation. From this we can get the query in the form of visual content – features. This approach circumvents many of the issues in specifying a query. However, the user is now working in the purely visual domain. There can be a disparity between the semantics of their real information need and what is available to them in this visual representation – the semantic gap. Moreover there can be images with the same semantic meaning that are very visually dissimilar. A way to circumvent this problem is to pose multiple queries for each visually different sub-query, or to combine them into a single multiple

[3] http://labs.systemone.at/retrievr

image query. Ultimately, the link with the semantics of a query is left up to the user. They become an implicit part of the system, interpreting the visual characteristics of the images in relation to their search need, and thus adapting to the multiple meanings that they attribute to an image.

2.2.3 Features

The aim of a CBIR system is to produce a ranked list of images that are relevant to the query. The system starts with the pixel representation of the image. Although this can be used to find exact matches it is of very limited use for comparing other visual properties of an image. Therefore, to rank images we must first extract a quantitative representation of what we want to compare. These are the visual features that are at the core of CBIR systems.

Features are an attempt to distill out some aspect of the image data that will be useful for comparing pictures. In essence they are a compact representation of a visual facet of the image. They can represent many things such as colour, texture, shape or structure. Practically, they are usually a vector of real numbers. These vectors commonly can have from one to several thousand elements.

A multitude of different features have been developed for CBIR, computer vision and classification tasks. Section 3.5 gives an overview of colour features for CBIR while Section 4.2 gives some detailed background on texture features. One thing all features have in common is that, although they are calculated precisely, they do not have a mapping to any clear semantic meaning. They are always just visual features. With text there is a close link between user understanding and the representation, but with visual features this link is less obvious.

So when designing features we have to work in an empirical framework. We can use sophisticated techniques to build a feature that represents a specific characteristic, but only when it is tested do we find how effective it is for a real retrieval task. This is not to say the choice of features is arbitrary. In fact the features chosen will have one of the most significant impacts, of any component, on the performance of a visual retrieval system. This can be evidenced by looking at the variation of retrieval performance when using different features with a single collection (Howarth et al., 2005).

2.2.4 Similarity

Once the system has a representation of the image collection as features, it then needs to compare these features. To humans, similarity is a judgement based on an interpretation of the image and the context in which it is viewed. For CBIR we need to be able to take a query and the feature space and produce a ranked order of pictures that reflect the user need.

In computer-based vision systems, similarity measurements are routinely carried out using metric distance functions such as the Euclidean or Manhattan distance. Distances are calculated in the feature space from the query image to all the others in the collection. These are then sorted. Obviously the distance function will have a significant impact on the ordering. The combination of feature and similarity function provides the raw retrieval power of any CBIR system.

There is a clear gap between the cognitive nature of a human similarity assessment and the deterministic similarity function. Indeed it was argued by Tversky (1977) that a geometric interpretation of similarity, such as a metric distance, bears little relation to perceptual similarity. Human perception frequently violates the metric properties of symmetry and triangular inequality. However, these geometric functions have worked effectively in CBIR systems for some time. Section 5.3 provides some detailed background on distance and similarity functions used for image retrieval.

If a person were to try and visually rank images in the same order as a simple feature and similarity function combination, it would prove very difficult. Consider the case of a colour feature, we may be able to match a few similar images with the same overall hue, but beyond this any ordering would not be clear. It seems that the correspondence between a person's similarity judgement and a similarity function, is useful only for the most similar images.

As a final thought, if we consider the task of image retrieval it becomes clear that what is important is the top few nearest neighbours that are retrieved. For someone searching a collection of one million images they are much more interested in the ranking of the top hundred than the ordering at ten thousand. So we should be able to concentrate on the behaviour in the region local to the query.

2.2.5 Indexing

Efficient indexing of the feature database is an important task as it can possibly enable CBIR to offer something that the human visual system cannot: searching a vast number of images in real time. If CBIR can be scaled up it will be able to add value as a search tool for large unannotated image collections.

The straightforward approach to a content-based search is to compare the features relating to each image with that of the query image, and hence to find its nearest neighbours. If we consider this sequential scan of the entire database, then the complexity of the task is linear, $O(n)$, with respect to the collection size. In this respect CBIR is scalable. However, the potentially large size of feature vectors and complex similarity computations make the constants within this linear complexity large. In practice this means that the search time for a relatively small collection can be too large to achieve real-time searches. To increase the number of images that can be searched a faster indexing method is required. This may come from either reducing the size of the constants within the existing linear task, or developing an indexing method with a lower complexity class.

Unfortunately, there are several things hindering the application of indexing methods to CBIR. The problems stem from the very features and similarity functions at the core of the system. The features being used are high dimensional and the consequence of this, often referred to collectively as the curse of dimensionality (discussed in Section 5.2), makes traditional indexing approaches ineffective. This is a significant problem: the area where CBIR appears to offer most is hampered by the technology at its heart. Many high-dimensional indexing approaches have been developed. These are summarised in Section 6.2. However, none of these have yet solved the indexing problem for the dimensionalities common with CBIR features.

2.2.6 Ranking, results presentation and relevance feedback

These three components form an important part of the CBIR framework. A large number of researchers work in these areas, however, they are not the focus of this work. They are often combined, with user interface controls and relevance feedback allowing the user to influence the ranking function.

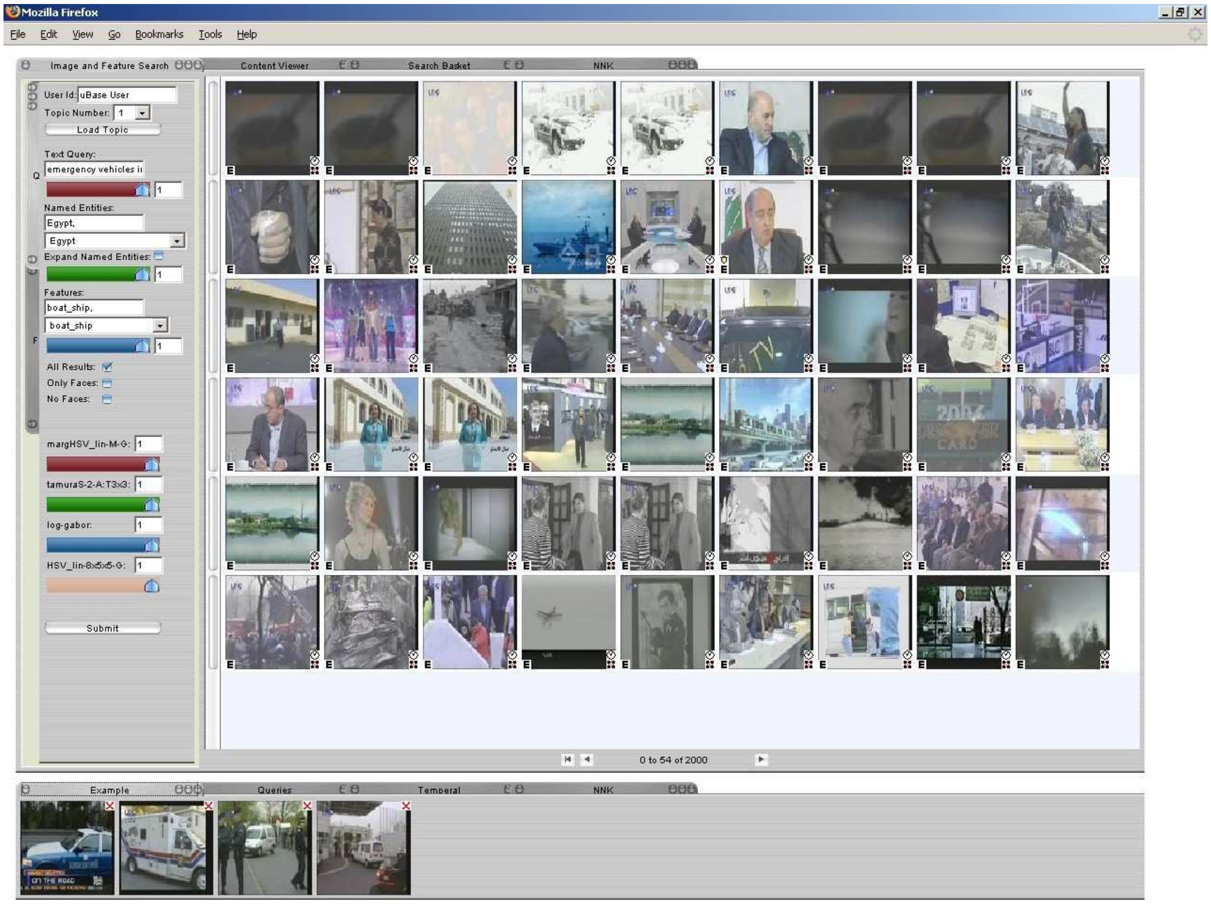
In its simplest form the ranking of results is a linear combination of distances computed using several features. For multiple image queries there are several options for combining distances. These include using a vector approach and the k-nn method, both described in Section 3.4, which are used in this work.

Relevance feedback methods have been widely studied. From an initial set of results the user specifies images that are close to their requirements and, in some systems, images that are counter to their needs. A common approach is to generate a single query point and move it so that it is closer to the positive images and farther from the negative (Rui et al., 1997). An alternative method is query expansion. This uses the same, user chosen, sets of positive and negative images, but supposes that the relevant images fall into multiple clusters in the feature space. The example images are used to generate multiple query points that should be close to these clusters (Porkaew et al., 1999). Relevance feedback methods have been shown to be effective providing that the user is willing to be proactive in their search.

The most straightforward approach to results presentation is to show the user the top ranked images and allow them to page through them. An alternative is to graphically indicate the degree of similarity between images by varying their size or placement on the screen. An example of this can be seen in the way that the NN^k network results are displayed in a spiral to represent the underlying similarity graph (Heesch and Rüger, 2004).

2.2.7 Current research

Content-based image retrieval has a rich research community. Each of the areas described earlier is being very actively investigated. The separate components are brought together for the major evaluation exercises, such as the TREC Video Retrieval Evaluation (TRECVID) (Kraaij et al., 2006) and the image retrieval track of the Cross Language Evaluation Forum (ImageCLEF) (Müller et al., 2006). An example of an image search system is shown in Figure 2.3. This shows a prototype interface for the image search engine built by our group. The interface is designed to be dynamically generated and has an XML based API. This flexibility means that it can connect to any search engine that supports the API, which makes it ideal for integrating new research ideas produced by the group (Howarth et al., 2007).



User Id: uBase User
Topic Number: 1
Text Query: emergency vehicles i
1
Named Entities: Egypt,
Egypt
Expand Named Entities: □
1
Features: boat_ship,
boat_ship
1
All Results: ☑
Only Faces: □
No Faces: □
margHSV_lin-M-G: 1
tamuraS-2-A:T3x3: 1
log-gabor: 1
HSV_lin-8x5x5-G: 1

Figure 2.3: SIRIL - prototype search engine

CBIR also plays its part in some large collaborative projects, for example, the European Union funded Knowledge Space of Semantic Inference for Automatic Annotation and Retrieval of Multimedia Content (K-Space)[4].

Currently the most active areas of image retrieval research appear to be the detection of features and topics within images using automated annotation methods (Bosch et al., 2006; Wilkins et al., 2006; van Gemert et al., 2006; Yavlinsky and Rüger, 2007). These approaches are seen as the most promising way of trying to narrow the semantic gap between visual content the user's information need.

2.3 Conclusion

This chapter has given an overview of content-based image retrieval. More detailed background material is in later chapters, close to the relevant work: Chapter 3 gives details of collections, query types and

[4] http://k-space.eu

performance measures; Chapter 4 reviews texture features; Chapter 5 focuses on distance and similarity functions; and lastly Chapter 6 covers high-dimensional indexing approaches. A thorough review of the CBIR landscape was given by Smeulders et al. (2000).

This chapter has argued that CBIR is ideally suited to searching unannotated collections of images. In addition it can provide a visual search for annotated collections. In this way it complements searches using metadata.

Although the image retrieval task uses many of the same methods as other computer vision applications its aim is quite different. With object recognition or classification the success of the method is very clear; an object is in the image or not; a texture patch is a specific type or not. With CBIR, success cannot always be as clearly defined. The user may have an idea of what they are looking for, or they may only know when they see it. This stems from the multiple meanings that can be attributed to a single image, that are not always obvious from its visual appearance. This “vagueness” in the problem definition of image retrieval means that approaches from other applications can often be too detailed or rely on theoretical models where the assumptions do not stand. In any work on CBIR it is important the the characteristics of the task are taken into account. Relaxing the need for accuracy and using simple methods may be fruitful lines to take in the quest to apply CBIR to large collections as a useful image search and browsing tool.

Chapter 3

Collections and evaluation

3.1 Introduction

In this chapter we lay the foundation for the experimental work carried out in later chapters. CBIR is largely an empirical discipline; although theoretical ideas can be employed they are only useful if they can be verified in a practical context. It is therefore essential to establish an experimental methodology and baseline against which new techniques can be judged.

The two main components of an evaluation framework are the collection of images and evaluation measures. These have been a contentious issue in CBIR as, despite attempts to build a benchmarking framework (Müller et al., 2003), there is no routinely used single standard collection and measure. It is therefore usual for each researcher to define their own evaluation environment. We have gone down the route of using several of the commonly available collections to allow our results to be judged widely. Much of this work is about relative changes in performance with different methods. Hence, we have defined a standard evaluation environment that has been deployed consistently throughout this work.

For testing purposes a collection comprises not only of a number of images, but also includes a set of queries and relevance judgements. The design of the collection will affect the results of any experiment run with it (Müller et al., 2002). In this Chapter, we discuss the choices available for building a collection, the impact of these choices and the actual collections used.

Similarly, the choice of performance measures will affect interpretation of results. Here, as with the collection and query design, it is important that the measured characteristics mirror real world user tasks. We discuss the strengths and weaknesses of mainstream IR evaluation techniques. This is followed by a description of statistical significance tests that are commonly used with information retrieval evaluation. These tests go some way to ensuring that differences in performance are not just down to random variations. Hence, they form an essential part of the evaluation framework.

When evaluating scalability of a CBIR system we have to measure the search speed whilst still maintaining retrieval performance. With indexing approaches that rely on approximations there is often a trade-off between the two. This is discussed together with the problem of building a large (greater than one million images) test collection.

The final sections cover some common methods and features that have been employed for this work. Specifically, how to deal with multiple image queries, using either the the k-nearest neighbour approach or the vector space model. I also give an overview of some of our visual features, and explain their characteristics. These details will be useful when discussing results. Note that the description of texture features is left until Chapter 4.

3.2 Image collections

It is widely recognised with image retrieval that the data set can have a large influence on the results from any experiments and the resultant conclusions. Some collections, such as Corel, are regarded as being "easy" whilst others are "hard". However, these terms are not easily quantifiable. Indeed, although commonly used by image retrieval practitioners they can be misleading. What is important, when we are trying to evaluate the effectiveness of CBIR methods, is how the characteristics of an image collection relate to a real image retrieval task.

3.2.1 Design considerations

In Chapters 1 and 2 we discussed the nature of retrieval in CBIR systems. Essentially, these systems only use visual information: it is left to the user to infer semantic meaning to queries and results. The combination of features and distances defines a topological space. From a simplistic viewpoint an ideal query is one where the query image's representation in this space cluster with those of the relevant images. The later chapters of this thesis look at altering and using the characteristics of the feature space. However, before doing this we consider the influence the image collection itself.

Types of images

Image collections have several obvious characteristics. The first of these we consider is the domain. Collections can be heterogeneous such as a random collection of images from the web. Alternatively they can have increasing degrees of specificity. Examples of these could be a collection of key-frames from a regular news program, pictures of objects in a museum collection, medical images, pictures of trademarks, satellite images or a personal album. Differing degrees of homogeneity can be given by the subjects of the images, the conditions that the pictures were taken under or the equipment used. Other relevant characteristics of images are the quality, size and compression. These are likely to go hand in hand with the domain.

If we consider the visual nature of the images then it is clear that a greater degree of similarity will occur with a closed domain. In these circumstances we may be looking for small differences between images. If we were to ask a user to group images visually it is likely that there will be a number of relatively tight clusters. In a heterogeneous collection the situation may be different in that there will be a greater number of looser clusters, possibly with fewer members in each. It is likely that the nature of this clustering will be mirrored in the feature space. If this is the case then this will affect the type of images retrieved by a CBIR system. For the closed domain the results will be dominated by similar images and qualitatively look better than those of the more general collection, where the results of a search may quickly degenerate into seemingly random images. This may affect the features and similarity methods that are preferred for each collection.

Types of queries and relevance judgements

To carry out evaluation experiments we need a set of queries and relevance judgements together with an image collection. In the query by example approach, queries can be single or multiple images. These can also be combined with text. For still images this can be in the form of annotations or keywords; for video, subtitles or speech recognition data may be available.

Given a query we need to know the relevant images in the collection. As discussed in the introduction the notion of relevance is subjective. It will vary from user to user and also depend on the context. This variation cannot be replicated convincingly.

There are three types of relevance judgements that are regularly employed:

- **Groupings.** This is where the collection is divided into groups of similar images. Examples of categories from the Corel collection (described later) are *penguins, reflective effects and places of worship*. The simplest way to define a query and ground truth is to choose a set of query images from the group and have the remainder as relevant images.

- **Keywords.** Often images will be annotated with multiple keywords. For instance a beach scene may have the keywords, sand, sea, sky, beach, people, holiday. The same keyword may appear in many visually diverse pictures. There will also be common associations, such as a cow normally occurring in a field of grass. Again it is simple to choose queries and ground truth by splitting those with the same keyword.

- **User judgements.** This type of query and relevance judgement aims to be as close to realistic real-world queries as possible. A topic is created with one or more images together with an associated statement of information need; relevant images are then manually identified. The expertise of the people making the relevance judgements (for example, general users or domain experts) will have an impact on what they identify as relevant. With an evaluation conference, such as TRECVID, the effort of manually reviewing a large collection can be reduced by using a pooling method (Harman, 1992). At such a conference many teams will submit retrieval results from their own systems. The top ranked images from each run are pooled together. This reduced set is then manually reviewed to identify relevant images. There is a risk that some relevant images in the collection are missed as the entire collection has not been examined. Normally all those images not in the pool will be marked non-relevant. The generation of incomplete ground truth in this way will introduce a bias against any new system that is judged using it. The new system may find relevant images that were not in the original pool, but will not be rewarded for this.

3.2.2 Discussion

The problem with any collection is that by slightly changing the images or queries we can change the results significantly. Müller et al. (2002) demonstrated how it easy it is to get differing performance with identical CBIR systems. They used a subset of the Corel collection and manipulated it by removing bad images or groups, or by choosing the best image query. They demonstrated the difficulty of comparing systems unless identical collections, queries and relevance judgements are used. Results from systems with a low number of queries are inherently unstable. If one or two queries give high average precisions changes in these can swamp differences in lower performing queries. So in general there is a problem with stability of results.

The purpose of of a CBIR system is to retrieve images for a user. Correspondingly it would seem that the most realistic relevance judgements would be those created by people. Other than the variability of user judgements the main drawback of these is that they are time consuming to produce. For this reason they are only usually created for large evaluation exercises. Often, only a small number of queries are created. In certain circumstances this can make the results sensitive to the performance of a single query.

The inherent problem with groupings and keywords is that often they do not correspond to the visual similarity selections that a human user might choose. The concepts they represent may not have any visual similarity, making retrieval an impossible task. Their major advantage is that a large number of queries and related ground truth can be created quickly. Easiness of a group depends also on the others present. For example in the ImageCLEF collection, described later, there are a few colour images amongst the predominantly black and white repository. This makes the colour images extremely easy to find.

If a single collection is used then there is a danger that systems and methods will become over-fitted to the characteristics of that specific collection. This criticism can be levelled at TRECVID, which is the major video retrieval evaluation exercise. The nature of the collection and search task has remained the same for several years and a huge amount of research has been focused on this. There is concern that this could bias research and therefore resulting systems. However, the big advantage of using collections from evaluation exercises is that they are standard and known by other researchers.

The main conclusion from this discussion is that no single collection adequately represents the complexity and variability of the CBIR task. To perform comparative evaluations of methods we need to employ a wide range of collections in a consistent evaluation environment. We can then vary a single component and see if improvements occur with most collections. If this is the case then we can be confident that the new method will robustly outperform the original. This is exactly the methodology taken in this thesis. The actual collections used are described in the next subsection.

3.2.3 The collections used

Corel

The Corel Gallery 390,000 product contains 31,997 high-quality images, divided into subject groupings. Subsets of the collection are widely used by image retrieval practitioners. We used a selection of Corel that was created by Pickering and Rüger (2003) to evaluate visual features. Their goal was to create a collection that would allow a fair evaluation, i.e., one with significant intra-category similarity and inter-category difference. Confusing categories, such as those that were very visually similar to another category or those that were highly abstract, were removed. This left a total of 6,192 images spread across 63 categories. This collection was then split into 25% training or query data and 75% test data.

The queries and ground truth were generated using the Corel categories. Single image queries were made using each of the 1,548 images in the test set. Multiple image queries of 2 to 6 images were also created. In each category and for each number of images, ten queries were randomly generated. This resulted in 630 n-image queries, a total of 3,150 multiple image queries. Relevance judgements were based simply on whether the test image belonged to the same category as the query.

This collection has been used throughout the work for initial investigations. Its size and large number of queries meant that it was ideal for indicating how methods were likely to affect retrieval performance.

Getty

This collection was built by Yavlinsky et al. (2005) to address the criticism that using the Corel data set is an unrealistic retrieval task. It is a collection of 24,396 medium-resolution images downloaded from the Getty website (Getty Image Archive). The pictures are varied in nature, many of them are fairly abstract and “artistic”. Attached to each image are multiple subject keywords that represent the content.

The collection was split in half to form a query and test set. The keywords were used to generate 100 single image query topics and associated ground truth. This retrieval task is therefore different to that of Corel, being based on a soft overlapping classification rather than hard categories. A proportion of images associated with a subject keyword can be visually different making the task a tough challenge for CBIR. Indeed, these “rogue” images within a keyword set may give retrieval results worse than random as the relevant images may be retrieved last. This data set provides a more realistic and challenging task.

TRECVID 2003

This collection was created for the search task of the TRECVID evaluation workshop (Smeaton et al., 2003) and is widely used. The task is that of video retrieval, and the collection comprises of 32,318 key-frames taken from ABC and CNN news broadcasts and C-SPAN programming. Associated with each shot is text from automatic speech recognition and closed caption based transcript.

The images appear fairly noisy because they have been extracted from a video stream. The news programmes are quite formulaic; this means that there are a large number of key frames with news readers in studio followed by outside reports. There are also adverts that occur repeatedly. This means that the collection contains a large proportion of very similar images. This is typical of production video of this genre.

The search task specified for TRECVID consists of 25 topics. For each topic a few example images are given as a query together with some text. The aim of the queries was to get a realistic search task that a real user might come up with. The relevance judgements were created by expert users, often intelligence analysts, using a pooling method.

ImageCLEF 2004

This is a medical image collection that was created for the image track of the Cross Language Evaluation Forum (Clough et al., 2005). It comprises of 8,725 images together with associated notes in English and French. The data set is quite different to others in that the images are mainly X-rays, CT-scans and medical photographs. The majority of these are monochrome and are carefully posed. The result of this is that that the collection contains groups of very similar images.

There are 24 single image queries. Ground truth was created by medical experts who reviewed the entire collection. The task associated with this collection is in a very specialist domain. It therefore provides an interesting contrast to the retrieval tasks set by the other collections.

Web - A collection for testing scalability

There are specific problems that arise when we want to build a collection for evaluating scalability of CBIR systems. The collection needs to be large. In addition we need to be able to vary the size of the collection whilst preserving its characteristics. We decided on a maximum collection size of over one million images for no reason other than this was more than one order of magnitude larger than any other commonly used test collection.

One problem is where to source a large number of images. When this work was started there were no suitable copyright-free image collections available at low cost. We decided to use the world wide web as a source of pictures. These are not copyright free. However, it is generally acceptable to use them for research purposes as long as they are not made widely available. The collection was originally built by Yavlinsky (2005) by downloading 1.3 million images from .edu domain sites. At the same time the alt tag details were saved and keywords extracted from the URLs. After filtering duplicates and navigation images we were left with 1.1 million images.

To measure retrieval performance we still need queries and relevance judgements. We could generate our own ground truth by exhaustive judgements, but this would be prohibitively time consuming. A second option is to mix in an existing collection and use its queries and relevance judgements. A problem with this is that the ground truth will not include any relevant images that exist in the large collection. This could mean that precision results are understated. This artificial situation can also make retrieval easier. The inserted images may be very homogeneous in terms of style elements such as lighting, composition and colour. This means that they will cluster well in the feature space and thus be easier to retrieve than if they were from the larger collection.

This, combination of collections, approach does give a practical method of obtaining queries and ground truth for a large collection. As long as the inherent limitations are kept in mind when viewing results it will be a reasonable collection for large scale retrieval experiments. It is the method I have adopted for the experiments in Chapter 6.

3.2.4 Summary

My approach in this work is to apply the same methods consistently to the full range of collections as described above. These are diverse, not only in the type of images, but also in the actual retrieval task derived from the query and relevance judgements. If improvements are consistent across the range of collections then one can argue that they are independent of the collection and generally applicable.

The Corel collection was used for exploratory work. Its design makes it ideal for development as it highlights changes in performance in the visual domain. Westerveld and de Vries (2003), who used the Corel collection in a similar way, stated that its characteristics make it an ideal vehicle for testing new ideas and models. Experiments with this collection often reveal statistically significant differences between methods, and can aid in giving an understanding of underlying behaviour. I have followed up any initial findings by systematically varying the collections and repeating experiments to determine whether the results are indeed robust.

3.3 Measuring performance

This section discusses how to measure the performance of an image retrieval system. The text retrieval community has been dealing with this issue for many years and has well established methodologies. These have been discussed extensively by van Rijsbergen (1979) and Salton (1992). In most situations the image retrieval world has either used or adapted these methods. This section does not aim to provide any further insight but to explain the well known methods, their strengths and weaknesses and to justify their use in the context of this work.

In the early Cranfield studies, Cleverdon et al. (1966) came up with six areas to quantify the performance of document search systems. These have subsequently formed the basis of evaluation measures. These were: coverage of the collection, presentation of results, user effort, time lag between query and answer, precision and recall.

In this work we are primarily interested in the last three. The next subsections discuss how to evaluate the quality of search results using precision and recall and how to measure the time lag and, associated with this, the efficiency of a system. It is essential to measure both of these. For a usable system there is no point improving precision or recall if the results are not obtained in a realistic time. As image retrieval systems are scaled up, there is usually a trade-off between speed and retrieval performance.

3.3.1 Precision and recall

First it is worth revisiting the task of image retrieval to set the context of what evaluation is attempting to quantify. A user supplies a query image or images and gets back a page full of images that the system marks as relevant. Depending on the system the user may then browse the next and subsequent pages of relevant images or possibly select positive examples and search again.

From this simple description we can see that an important factor is how many images are on a page and how many pages a user is likely to browse. For example if there are twenty images per page and the user looks at a maximum of five pages then we should be interested in retrieval performance for the top one hundred items. As the first page is particularly important, we could justify measuring retrieval performance for the top twenty.

There are further complications for real systems and searches. The number of relevant images in a collection will affect any measure. In a controlled collection this will be known. In an open collection this will be unknown. Sometimes a user may be searching for a specific image they have seen before, only one image is relevant. Perhaps there will be no relevant images, or thousands.

We need to find measures that give a good indication of visual search effectiveness. They should enable us both to compare the algorithms that are being investigated and how the results fit a user task. The ultimate aim is to find out how well does image retrieval help users when faced with a task of searching an image collection and finding what they want.

Definitions

Precision and *recall* are the cornerstones of information retrieval evaluation. Together with *fallout* they make up the standard measures, which are defined below. There are many references for these definitions, however, we often use the trec-eval software for evaluation and therefore point the reader to proceedings of the TREC workshops (Voorhees and Harman, 1999). These give a detailed description of the calculation of measures and construction of precision–recall graphs, which we summarise in the next section.

$$\text{Recall} = \frac{\text{number of relevant items retrieved}}{\text{total number of relevant items}} \tag{3.1}$$

$$\text{Precision} = \frac{\text{number of relevant items retrieved}}{\text{total number of items retrieved}} \tag{3.2}$$

$$\text{Fallout} = \frac{\text{number of non-relevant items retrieved}}{\text{total number of items retrieved}} \tag{3.3}$$

Precision – recall graphs

If we calculate precision and recall for a list of retrieved documents the values will have no dependency on the ordering of the list: they are set-based measures. Usually we are interested in retrieving relevant documents first. To get this information we need to calculate the measures as the depth of the result list is varied. The result of this is often displayed using a precision versus recall graph. Researchers are familiar with these graphs and thus able to interpret them easily. They give a visualisation of the characteristics of the search engine and therefore underlying algorithms being used. As an example of this we can consider the graph shown in Figure 3.1. The two curves shown have approximately the same area under them, and hence, have about the same average precision of 37%. However, the characteristics of the curves, and therefore the search engine that produced them, are quite different. Curve A has high precision at low recall. This is suited to a web search engine, where the user wants as many positive images as possible in the first page of results, but is not concerned about finding everything that is relevant. Curve B has a higher precision at high recall. This response is ideal for a user that wants to ensure that they retrieve all the relevant documents, i.e. intelligence analysts.

To construct a precision recall graph, precision is plotted against recall as we step through the retrieved document list. Take an example where the collection has 10 objects and the query has 3 relevant items that are 1st, 3rd and 6th in the ranked results list. This situation is shown in Table 3.1,

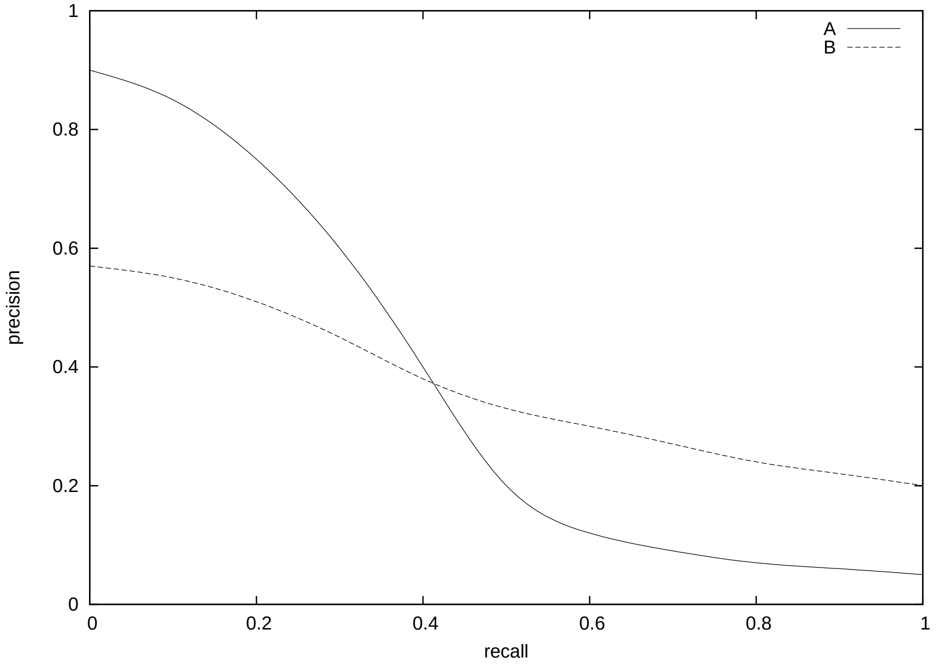


Figure 3.1: Example precision–recall graphs

	Depth of retrieval list										
	-	**1**	2	**3**	4	5	**6**	7	8	9	10
Recall	-	0.33	0.33	0.67	0.67	0.67	1.0	1.0	1.0	1.0	1.0
Precision	-	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.33	0.30

	Standardised Recall										
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Interpolated Precision	1.0	1.0	1.0	1.0	0.67	0.67	0.67	0.5	0.5	0.5	0.5

Table 3.1: Example calculation of recall, precision and interpolated precision

with the relevant items in bold. We start with just the first item in the result list; this is relevant so precision will be 1.0 and recall 0.33. The next item is non-relevant, precision drops to 0.5 and recall stays at 0.33. The third item is relevant and precision is now 0.67 with recall 0.67. The next two items are non-relevant, recall will stay the same, but precision will drop to 0.5 and then 0.4. When the final relevant item is retrieved recall will be 1.0 and precision 0.5.

The trec-eval software calculates interpolated precision for a standard set of eleven recall levels, increments of 0.1 between 0 and 1. The interpolated precision for level x is the maximum precision for any recall level equal or greater than x, as shown in Table 3.1. For a set of queries the precision at these standard levels is averaged over the set and the graph drawn with these values.

Average precision

It can be easier to compare performance with a single number that represents some combination of precision and recall. There are several alternatives for this. The first we consider is average precision. This is obtained by averaging the precision values as each relevant item is retrieved.

$$\text{Average precision} = \frac{\sum_{r=1}^{N} \text{Precision}(r)}{N}, \tag{3.4}$$

where Precision(r) is the precision when each of the N relevant items is retrieved. For the earlier example this will be 0.72 ($\frac{1.0+0.67+0.5}{3}$). Average precision rewards systems that return more relevant items earlier. This can be said to equate to the user who wants to see relevant items on the first few pages rather than later.

If we repeat this over a set of queries and take the mean this becomes the mean average precision (m.a.p.). M.a.p is a good single performance indicator, but as with all averaging processes, it may hide interesting anomalies in the results for each query. If we want to look deeper into the results, then a histogram or bar graph showing each query is a useful tool for more detailed analysis.

Precision at n documents

This is simply the precision measured when a specific number of objects have been retrieved. So we may use precision at 5, 20, 100 etc.. The choice of what level to use can be made by considering the task and type of user interface. As described in the earlier example we may choose values of n relating to the number of images displayed on a page. So measures of precision at twenty and one hundred would be suitable.

One of the limitations of this measure is that it is sensitive to the number of relevant images in the collection. For instance a query with only one relevant item could only achieve a maximum of 1% precision at 100, and consequently it would give very different results to a query with one hundred relevant items.

R-precision

R-precision is useful to get around the effect on precision at n documents caused by variable numbers of relevant images. It is defined as the precision at the R-th document retrieved, where R is the number of relevant items for that query. It is a useful measure for comparing the performance of different features or algorithms for a specific query. R-precision can be averaged over a set of queries with the usual caveat about masking potentially important behaviour.

F measure

The F-measure is the harmonic mean of precision and recall and is computed as:

$$F(i) = \frac{2}{1/\text{Recall}(i) + 1/\text{Precision}(i)}, \tag{3.5}$$

where Recall(i) and Precision(i) are, respectively, the recall and precision for the i^{th} item retrieved. The F-measure combines precision and recall into a single measure, which has a high value when both of them are high.

As a generalisation the E measure was proposed by van Rijsbergen (1979). This is similar to the F-measure, but with a relative weight for precision and recall. This variable allows the user to specify a parameter that weights their own degree of interest in precision or recall.

Alternatives

Many alternatives to the standard measures described above have been suggested. For example, Müller et al. (2001) proposed normalised average rank. This normalises a rank measure by the collection size and number of relevant images, thus making it easier to compare results from different queries. Unfortunately, despite some advantages none of these new methods have been generally adopted.

An area where new measures and methods have been taken up is evaluation of performance for collections with incomplete relevance judgements. This is particularly important to the very active TREC evaluation community. TREC uses a depth pool approach, as discussed in Section 3.2.1, where the union of the top results from each run is judged to find the relevant documents. A depth of 100 objects has been empirically found to give a good balance between effort and quality of evaluation. However, this size of pool still needs considerable work producing the ground truth. In addition, Smith (1998) noted that the effect of incomplete relevance judgements on precision-recall plots is that they may become inaccurate after around 40% recall.

Buckley and Voorhees (2004) proposed a new measure, *bpref*, which is computed by considering the relative ranks of documents that have been judged relevant or irrelevant. Documents not in the pool are not considered in the measure. They showed that this measure was effective for ranking systems. However, at low sampling rates it is not correlated with any of the usual measures. It was adopted for the TREC 2005 terabyte track.

Another new measure that has been taken up by the TREC community is inferred precision. This was put forward by Aslam et al. (2006) again to evaluate performance for collections with incomplete relevance judgements. Their aim was to drastically reduce the workload of generating relevance judgements while still producing an unbiased estimator. They showed that highly accurate estimates of average precision, R-precision and Precision(n) could be obtained using as little as 4% of the typical TREC pool.

Their method is based on random sampling. They show that their estimates are unbiased and then derive a sampling distribution that gives low variance estimates. The distribution gives high sampling probabilities to documents near the top of the ranked lists of results. Thus these are more likely to be judged.

3.3.2 Discussion

Precision and recall measure different aspects of the performance of a search engine. Although they are the basis of most widely used measures, precision and recall are open to several criticisms. The most obvious of these is that to calculate recall correctly it is necessary to know all the relevant items in the collection. This is not always practical. Several enhancements have been proposed that improve evaluation of precision for collections with incomplete relevance judgements.

Certain measurements using precision and recall are also reliant on an ordering. Some systems have a weak ordering and may return many items with the same similarity value. To cope with these we have to randomly order results that match to break the ties. Queries with few relevant images may be very sensitive to the the exact order results are returned. For instance where there is a single relevant image that is returned in the top ten results average precision can vary from 10% to 100%. In reality, if the image is included on the first page of results viewed by the user then the practical significance of the ordering is much less.

Possibly the most important limitation is their relation to the real image retrieval task. These measures are applied in a batch mode using a set of standard queries. In a real system the user will perform repeated searches, refining the query and browsing through the collection. The effectiveness of this is not captured by applying these evaluation methods with fixed queries and relevance judgements. Of course these methods can be used to evaluate the results of manual runs where a real user queries a system and collects their own ranked list of results. However, when we step into this arena there are several other things to be considered. The first of these is as discussed earlier, each user will have a different judgement of relevance. When they are using the system this will implicitly affect the results as they make choices what to include. There are user-oriented measures that aim to capture the performance in this situation (Kekäläinen and Järvelin, 2003; Korfhage, 1997).

In addition, real user relevance judgements are not binary. They may judge several images to be relevant to their search need but within this set have some that are more desirable than others. Similarly although a high recall may appear desirable, it is not needed if the user needs to find only one high quality result.

However, this work is focused on algorithms for features, similarities and indexing. The variability introduced by a user would make it impossible to judge these in isolation. For this reason we choose to stay with the "traditional" evaluation methods. Generally in this thesis we use mean average precision and interpolated average precision versus recall graphs to measure performance over a set of queries. When looking in detail at individual queries we use a histogram of average precision. Over a wide range of queries these generally give a good measure of performance for an image retrieval task, trading off between precision and recall. They are also widely understood and adopted by other researchers, hence, our results can easily be assessed. We therefore feel justified in the use of these measures.

3.3.3 Time lag and efficiency

The speed of a retrieval system can be quantified in several ways depending on the viewpoint taken. To the user, the response time of a system is the most important. This equates to the time lag referred to at the start of this section. The time lag of a search system is the average interval between the time the search request is made and the time an answer is given. This can be easily measured as a wall-clock time. Although this will include many variables it will give a realistic performance measure if experiments are run repeatedly, in the same circumstances, and the times averaged.

If we are considering the underlying efficiency of the retrieval engine we can look at individual components that make up the total response time. This is discussed below, but first we consider the effect of response times on user perception.

Responsiveness

The general advice about response times for interactive systems has been around since the Sixties (Miller, 1968). These were discussed by Nielsen (1994) in the context of web usability. In reality the guidelines remain the same for all applications whether they are web-based or not. They are driven by what users expect and feel comfortable with. The three response times of interest are explained below.

0.1 second User interface components, such as menus, navigation and rollovers, should react within this timescale. This speed makes the user feel that the system is responding instantly to their actions.

1.0 second The user will notice a delay of this length but their flow of thought will not be interrupted. They now have the feeling the the computer is working on the data rather than them working on it directly.

10 seconds This is the limit for keeping the users attention focused on the task. With these sort of times the user should be provided with some feedback, such as a progress bar. If the response time is greater than this it is likely that the user will start doing something else in the gaps.

Given these timescales what is an acceptable search time? A user would expect an image search to take some time as they would accept that the computer has to do something. Therefore an ideal response time would be one second, with a maximum of ten seconds. With a web-based application there would also be all the overhead of loading images, multiple connections and so forth. If the user starts to see the first images load then their interest will be held and this point can be taken as the search being complete.

A search engine deployed as a web service may have to deal with concurrent requests. Whilst this is an important factor it has not been considered or measured explicitly within this work.

Although this is all a little vague we can come up with a target time for a single search of 1 second. This would be acceptable for a local system and also for a web-based version where the extra communication time would be expected. We have therefore used this time as a target for the time lag between query and response.

Efficiency of specific components

One way to break down the total time would be into the components identified on the system diagram in Figure 2.2. We can then consider the time taken for each part: query submission, feature generation, loading features, similarity comparison, ranking of results and result presentation. A straightforward approach is to measure these separately and use the times to compare the efficiency of algorithms for each component.

When searching a large collection the time to search the feature database — loading the required features and comparing them — dominates the time taken. As a collection grows there will be a size where it is no longer possible for the feature data to be stored in memory and it will then need to be stored on and loaded from disk. At this point the disk access time will become the bottleneck in the system as it is orders of magnitude greater than memory access (Hennessy and Patterson, 2006).

When considering indexing approaches a common alternative to recording actual time is to measure the amount of data accessed. Indexing methods aim to reduce the amount of data that is searched so a naive approach is to measure the number of bytes retrieved. This has many flaws as it does not take into account the location of data on disk and the page structure. Reading a single byte will necessitate loading a complete disk page, so if two bytes are on different pages these will both need to be loaded. The size of a page can be varied when the file system is generated. A common size is 4kB.

So counting the number of disk page accesses will give a better measure of indexing efficiency. However, in reality a lot of hardware optimisations and caching occurs in a the I/O system, which are difficult to account for. Without tight control of this the actual search times may be quite different to that expected from the number of disk page accesses. For example, there are significant efficiency gains to be made by reading data as a contiguous multi-page block rather than many single pages (Comer, 1979). The implications of these issues are discussed in Chapter 6 as part of the design and implementation work.

For the purposes of this thesis the primary speed measure used is wall-clock time. As long as care is taken with the experiments this provides a useful measure. Indeed, as it is directly the concern of any user it fits with the overarching principle of building practical retrieval systems focused on the user need. In addition where appropriate we have included the quantity of data loaded and the number of disk pages accessed as an indication of the relative efficiency of indexing algorithms.

Baseline

As a baseline measurement for time lag we have chosen to use a sequential scan of the feature database stored contiguously on disk. With this approach the disk and I/O system will be working at maximum efficiency. In these circumstances the average time to retrieve a page from disk can be ten times faster than normal. For this reason, although this simple approach looks at the entire database, it is hard to beat when searching high-dimensional features unless an indexing system is truly effective.

3.3.4 Statistical significance

The results of any experiment need to be interpreted. We have a dual purpose when carrying out these experiments. Firstly, we are assessing the relative performance of different retrieval algorithms. Secondly, we want to decide if these differences in results are meaningful for a user. The latter point is the primary aim of an image retrieval system.

As we are taking a methodological approach it is important that we determine the statistical significance of our results. That is, are the differences in mean average precision between two methods significant or are they likely to be due to random errors. Having a valid statistical methodology for this is an important part of the experimental method.

Of course having performance differences that are statistically significant does not necessarily mean that these differences will appear meaningful to the user. Keen (1992) calls this *practical significance* as opposed to statistical significance. As with all user driven measures this is much harder to quantify. We have not attempted to do this systematically in this thesis. However, where any qualitative or anecdotal evidence has appeared this has been presented in context.

Getting back to statistical significance testing a good summary of their practical use within retrieval experiments was presented by Hull (1993). We are concerned with testing if the difference between mean average precisions for two methods is statistically significant. The problem needs to be formulated in terms of hypotheses. The alternative hypotheses, H_1, will be that the higher result is in fact different from the other. This is a one-sided test. Of course the null hypothesis, H_0, is that there is no difference between the results.

The retrieval results are obtained for a set of queries, so for each query we have a pair of results. From these we to calculate a test statistic, T. In this situation there are three tests that are commonly used: the sign test, the paired Wilcoxon signed-rank test and the paired Student's t-test. The first two are non-parametric whilst the last is a model-based test.

Given the test statistic and its probability distribution we then choose a critical region, that is the range of values that point to H_1 being true by rejecting H_0. The size of the critical region corresponds to the risk of rejecting H_0 incorrectly; this is called a Type 1 error. In this work we have chosen a significance level of $\alpha = 5\%$. Of course there is also a risk of a Type 2 error, that is not rejecting H_0 when it is false. The probability of this is usually denoted by β and is linked to the power of the test. Each of the three tests has different assumptions and power. Correspondingly, their suitability will vary depending on the circumstances. These are discussed below, starting with a description of the tests.

The starting point for all of the tests is the difference between the paired measurements. Suppose we have n queries, each having a pair of average precision measurements. For query i we have observations x_i and y_i, and difference $w_i = x_i - y_i$. The tests all use different aspects of this random variable W.

Sign test

The sign test is the simplest test with no assumptions about the distribution of W. It is non-parametric and uses only the sign of the differences, ignoring any information in the magnitudes of the differences.

The null hypothesis, H_0, is that the probability of observing a positive or negative value for w_i is the same, i.e. $P(W > 0) = 1/2$ and that the median value of w_i for $i = 1 \dots n$ is zero.

The procedure is to count the number of positive differences, n^+, and negative differences, n^-. Zero differences are ignored so we reduce the sample size to $n = n^+ + n^-$. We therefore have n Bernoulli trials. Under the null hypothesis n^+ and n^- are binomially distributed, $N \sim B(n, 1/2)$. We consider the case where it appears that $X > Y$, so the test statistic is $t = n^-$. We can now calculate the probability of the observed result and determine if it falls in the chosen critical region. The critical values for the test can be found from tables, or with small values of n the probability of $N \leq t$ can be calculated directly. For larger values of n we can use a normal approximation for the Binomial distribution, remembering to apply a continuity correction. The mean and variance are $\mu = n/2$ and $\sigma^2 = \sqrt{n}/2$, respectively. The critical value is then $\lfloor \mu - z\sigma - 1/2 \rfloor$, where z is the required percentage point of the standard normal distribution.

Paired Wilcoxon test

The Wilcoxon signed-rank test (Wilcoxon, 1945) takes account of the absolute rank of the differences as well as their signs. This means that it is more powerful than the sign test, but it also has more assumptions. These are that the differences are assumed to be independent, each coming from a continuous distribution that is symmetric about a common median θ. The null hypothesis H_0 tested is that $\theta = 0$.

The absolute values of the differences $|W|$ are ranked. Where ties occur the average is used. Each rank is then given the sign of the corresponding difference and the positive and negative ranks summed separately. For a two-sided test the smaller of these is taken as the test statistic T. As before we are normally performing a one-sided test and take the value that the alternative hypothesis suggests should be smaller. This can be compared with tabulated critical values to determine if the null hypothesis holds.

For large enough values of n (> 10), T has an approximately normal distribution. Its mean and variance can be calculated as $\mu = n/4(n+1)$ and $\sigma^2 = n/24(n+1)(2n+1)$. As before we can then find the required critical value using the standard normal distribution.

Paired Student's t-test

The paired t-test is a parametric test. It assumes that W is distributed normally and the differences are random samples form this. The null hypothesis supposes that the mean of the differences is zero. If the populations are not normal then the test can be regarded as approximate.

To calculate the test statistic we find the sample mean and variance of the differences, $\bar{w} = (w_1 + \cdots + w_n)/n$ and $\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(w_i - \bar{w})^2$. We then have the test statistic:

$$t = \frac{\bar{w}}{\sigma/\sqrt{n}} \tag{3.6}$$

Under the assumptions this will be a Student's t-distribution with $n-1$ degrees of freedom. We can use confidence intervals derived from the Student's t-distribution to determine if the null hypothesis is supported. If it is not then the difference between the measurements is statistically significant with the chosen confidence level.

Discussion of tests

When choosing which test to use the first thing to consider is, does it test the hypotheses we have formulated? The three tests considered do this. Secondly, are the assumptions of the test satisfied?

The non-parametric tests make the fewest assumptions about the distribution of the measurements. Their drawback is that they are less powerful than the t-test and thus more likely to result in a Type 2 error. Of the two non-parametric tests the Wilcoxon test is generally more powerful (Daniel, 1978) and, hence, is preferred.

The value of the t-test for information retrieval was savagely criticised by van Rijsbergen (1979) because it is based on a continuous distribution whereas precision and recall are discrete. He concluded that only the sign test could be considered valid. A counter argument is that with sufficient data, bounded discrete measures are often well approximated by continuous distributions. The t-test remains powerful and a useful tool in evaluation as long as this is kept in mind. We can perform tests of normality along with the test itself to determine its validity. Hull (1993) suggested the use of quantile plots to judge if the distribution has either heavy skewness or large outliers. These are the most likely properties to affect validity of the test.

In general, we have found that the assumptions of the t-test are routinely satisfied for experiments where there are a large number of queries. Correspondingly, we have used this test where possible but moved to the paired Wilcoxon signed-rank test where the underlying assumptions are not met.

One final point to note is that it is useful to consider where the test statistic lies in the critical region. The further it is from the boundary the stronger is the evidence that the null hypotheses should be rejected. Hence, examining the actual significance level of T may provide additional information.

3.4 Approaches to multiple image queries

With a single image query the distance or similarity measure automatically produces a ranked list of results. When the query consists of several images we need to use a method to aggregate them to produce a ranked list. The two methods used throughout this thesis are described below. Although it was not one of the aims of the experiments described in subsequent chapters, it is interesting to note that k-nearest neighbours consistently outperforms the vector based approach for image retrieval. This is consistent with the findings of Pickering and Rüger (2003).

3.4.1 Vector based combination

This approach uses a simple sum of distances. Consider the case where we have n query images, with feature vectors, $q_1, \ldots, q_n$, and a test image with feature vector t. Using a distance function d, the total distance of the multiple image query to the test image is the sum of the distances from each query feature to the test feature, $\sum_i d(q_i, t)$. This method has the advantages of being efficient and simple to implement.

3.4.2 K-nearest neighbours

This method is based on the idea that, given positive and negative example images, test images can be classified according to their proximity to these examples. A version of the distance weighted k-nn approach described by Mitchell (1997) was used (Pickering and Rüger, 2003). Positive examples are supplied as the query and negative examples randomly selected from the collection. To rank a test image t in the collection we identify the k positive or negative examples that are nearest to t. If P and N are the sets of these positive and negative examples respectively, then $|P| + |N| = k$. Using these neighbours we determine the overall dissimilarity as

$$D(t) = \frac{\sum\limits_{n \in N} (\text{diss}(t,n) + \varepsilon)^{-1}}{\sum\limits_{p \in P} (\text{diss}(t,p) + \varepsilon)^{-1} + \varepsilon} \quad . \tag{3.7}$$

Varying the value of k can alter retrieval effectiveness across different queries and collections. However, it was determined experimentally that a setting of $k = 40$ gave robust results. This value was used consistently in our experiments. In the implementation a small positive constant value ε is added to the denominators to prevent division by zero exceptions.

It is interesting to consider why k-nn outperforms the vector approach for image retrieval in a high-dimensional feature space. Take a case where the query and relevant images are clustered together in several distinct regions of the feature space. The formulation of the k-nearest neighbour approach means that it will favour images in that are in one or more of the clusters, whilst the vector approach will give all images equal weight. Hence, k-nn should outperform the vector approach where separate clusters exist. This extra effectiveness comes at a price as the method requires additional operations to compute the relevance $D(t)$.

3.5 Visual features

The experiments throughout this thesis employed a range of high dimensional visual features based on colour, texture and structure. Chapter 4 describes detailed work on modifying texture features for CBIR and Chapter 5 looks at some specific characteristics of different feature types. It is left to this section to give an overview of the non-texture features used in this work. Full details of the implementation of these was described by Pickering and Rüger (2003).

3.5.1 Colour features

Retrieval of images using colour features was one of the earliest CBIR approaches used (Swain and Ballard, 1991). The commonest method is to build either marginal or joint histograms based on a colour space. Naturally, the joint distributions contain more information and are usually preferable. We can choose the number of bins in the histogram through the degree of quantisation.

There are many colour spaces in general use (Travis, 1991). The majority of digital images are represented in RGB space. The colour of each pixel contains a different proportion of red, green and blue light. The representation often has a direct mapping to a physical device after a gamma correction. This colour space will be a cube, often of dimensions 256 × 256 × 256.

HSV is a cylindrical colour space. Its representation appears to be more intuitive to humans than the hardware representation of RGB. The hue coordinate H is angular and represents the colour, the saturation S represents the pureness of the color and is the radial distance, finally the brightness V is the vertical distance. There are a few intricacies when quantising this space. Hue is singular at the achromatic axis where it represents greys, therefore the hue bins can be merged.

The HMMD (hue, min, max, diff) colour space is relatively new. It is a combination of the RGB and HSV colour spaces. The hue component is straight from the HSV value. The min and max values are the minimum and maximum values of the RGB values. They give a measure of whiteness and darkness respectively. Finally the diff value is the difference between min and max, this measures the saturation of the perceived colour. Three dimensions, hue, sum and diff, are sufficient to describe the colour space (sum is defined as (max+min)/2). HMMD is supported in the MPEG-7 standard. The core experiments for the standard showed that it was very effective.

The actual features used are described below:

RGB is a joint colour histogram defined in RGB colour space. The usual configuration subdivides each axis into 8 sections leading to 8 x 8 x 8 = 512 bins.

HSV is a joint colour histogram defined in the cylindrical colour-space. The arrangement of bins normally used was 8 x 5 x 5. This results in 165 bins rather 200 because of the merged hue bins along the axis.

HDS aims to capture the local colour structure of an image. Manjunath et al. (2001) give details of its construction. It uses the HMMD colour space, replacing the min and max components by their average. The quantisation used results in 184 non uniformly quantised bins. The local image structure is captured by sliding an 8 x 8 structuring window over the image. The size of the window may vary with the image size but the number of samples remains constant. Each of the bins counts the number of window positions for which there is at least one sample pixel falling into the corresponding bin.

3.5.2 Other features

Several other features have been used in experiments, including the texture features described in the next chapter. It is important that features with widely differing characteristics are employed. This allows us to determine if methods are efficacious across the full range or limited to specific circumstances. Two features are described below:

Thumbnail is one of the simplest features. It is created from the pixel intensity values of a scaled down image. We used a size of 40 by 30 resulting in a dense vector of length 1200. This feature is a good discriminator for near identical images.

Convolution was designed to discriminate between low level structures in an image. It uses the method of Tieu and Viola (2000). It is created by filtering the grey scale image with 25 low level filters designed to detect primitive structures. The resulting feature maps are then re-filtered with each of the filters giving a 625 dimensional feature. This can be used or the filtering process can be repeated to give 15,625 elements. This can also be applied to each of the three RGB channels resulting in up to 46,875 elements. The underlying idea of this feature is that by creating so many elements each will only have a small proportion of images where they have a high value. Thus each element is very selective in its own right. We found as part of our work that there was a duplication in the original filters meaning that there are only 24 unique filters. This reduces the size of the feature correspondingly.

3.6 Conclusions

We have aimed to build a good experimental framework, which is essential for evaluating the outcome of image retrieval research. When the research focuses on the retrieval of generic images then it is important to overcome any bias that the image collections may create. By doing this we hope to ensure that the methods we prefer are robust, and that we understand their limitations.

The first part of the framework is the collection. To ensure that results were not just a feature of a data set used we ran experiments using up to four different collections. The primary experiments were done with a collection taken from the Corel image library. These were then followed up with further experiments using Getty, TRECVID 2003 and ImageCLEF 2004 collections. This enabled us to validate our results and draw conclusions about the general applicability across the different collections. These collections vary in their content, the nature of the queries and relevance judgements. In effect each reflects a different user task.

For our evaluation method we stuck with the tried and tested measures based on precision and recall. Despite their limitations we believe that these remain the most realistic approach to use when evaluating different retrieval algorithms. They are widely understood in the retrieval community. We coupled these with statistical significance tests, the paired t-test and Wilcoxon signed-rank test, using a significance level of $\alpha = 5\%$.

For scalability testing we used a collection of one million images gathered from the web. The Corel collection was merged with this to supply queries and associated ground truth. This combined collection could then be varied in size, and both the search speed and retrieval performance measured.

Additional elements of the experimental system were the features and methods, such as k-nearest neighbours. These were kept constant, meaning we could change just one thing and measure the impact. Comparisons made in this way showed relative changes in performance.

In a similar vein we set up a baseline system against which we could measure absolute performance. This used methods known to be robust and effective. These were, the Manhattan distance measure, k-nearest neighbours and linear sequential scan of the database. This gives us both a retrieval performance and time-lag benchmark to beat.

Chapter 4

Texture features for content-based image retrieval

4.1 Introduction

This chapter focuses on the first of the core components of CBIR highlighted in Chapters 1 and 2, visual features. Image retrieval practitioners place varying emphasis on the features used in their research and systems. The majority of features have been developed and tuned for application areas different from CBIR, such as segmentation or classification. If features are used without any thought an opportunity is being missed to improve retrieval performance of systems by ameliorating the fundamental building blocks.

Features are so important because, in the CBIR approach, they contain the only information that is available to differentiate between images. If the visual property that the user is interested in is not captured within the feature then no system, no matter how sophisticated, will be able to deliver what the user requires. This work concentrates on the construction and use of texture features for generic image retrieval.

Texture is a key component of human visual perception. Like colour, this makes it an essential feature to consider when querying image databases. Everyone can recognise texture, but it is more difficult to define. Unlike colour, texture occurs over a region rather than at a point. It is normally perceived by intensity levels and as such is orthogonal to colour. Texture has qualities such as periodicity and scale; it can be described in terms of direction, coarseness, contrast and so on (Tamura et al., 1978). It is this ambiguous nature that makes texture a particularly interesting visual facet of pictures and results in a plethora of ways of extracting texture features. To explore a wide range of these methods three very different approaches to computing texture features were studied in detail: The first takes a statistical approach in the form of co-occurrence matrices, next the psychological view of Tamura's features and finally signal processing with Gabor wavelets.

This work in this chapter has been published in several papers. The studies of Howarth and Rüger (2004, 2005c) were the first to focus on evaluation of texture features on heterogeneous everyday images, and the first to tailor features for optimum retrieval performance in this context. Howarth et al. (2005) describes the subsequent use of these texture features for a submission to the ImageCLEF evaluation exercise.

The majority of original papers devising or evaluating texture features used classification or segmentation tasks to measure performance. Segmentation and classification tasks are significantly different to the problems faced in image retrieval, where one looks at generic queries for an entire picture. Real pictures are made up of a patchwork of differing textures rather than the uniform texture images often used in studies, such as the ones taken from Brodatz's photo book (Brodatz, 1966).

The remainder of this chapter is organised as follows: The next section surveys the range of methods used for creating texture features and ends with a summary of related work. In Section 4.3, we discuss the general properties of visual features and what makes a good feature for CBIR. Texture is difficult to formally define. As a substitute for this we try to give some intuition into what is perceived as texture by the viewer of an image. With this done the next step is to look at ways this information can be extracted into a visual feature thus enabling the searching of an image collection. The discussion concentrates on the details of the features chosen for this exercise.

Section 4.4 presents the experimental work. Initial experiments were with a training set of Corel images. The results from this work led to the parameters and feature modifications that were found to give the best retrieval performance. This study is then complemented by a larger performance comparison on the TRECVID 2003 data set and a set of medical images from the ImageCLEF 2004 evaluation. This work shows that the principles and modifications found with Corel are generally applicable across several collections. Finally the conclusion to the chapter highlights the general principles discovered for modifying features for CBIR and makes some concrete recommendations for the specific features used.

4.2 Background and related work

4.2.1 A survey of texture features

A vast amount of work in the field of computer vision has been done on texture features. This ranges from devising new features for texture classification to very specific applications such as identifying abnormalities in medical images. The performance of most new features has been measured and evaluated in the context of segmentation and classification tasks.

There are many surveys of texture features. Two of the more recent and comprehensive surveys are by Tuceryan and Jain (1998) and Sebe and Lew (2001). They largely present the same material but organise it using different taxonomies.

Tuceryan and Jain (1998) discuss the motivation for and applications of texture analysis. They then present their taxonomy of texture models in which they subdivide features into statistical, geometrical, model based and signal processing methods. In Chapter 3 of their widely used book, Principles of Visual Information Retrieval, Sebe and Lew (2001) use a taxonomy of stochastic texture models to organise their discussion. There are three major groups: probability density function, gross shape and partial models. Probability density function methods are subdivided into parametric and non-parametric; gross shape into harmonic and primitive; and partial into line and fractal. A summary of the principal methods of texture feature extraction is given in the list below. There are hundreds of papers on this subject, this summary references some of the earliest work on each feature. Although later work may refine and deploy the ideas to new applications the underlying principles remain unchanged. The grouping of methods below is not definitive and many other could be used. It does, however, give one way to organise similar methods.

Statistical: A thorough exposition of statistical and structural approaches to texture was given by Haralick (1979). His paper covered auto-correlation, optical transforms, digital transforms, textural edgeness, structural elements, grey tone co-occurrence, run lengths and autoregressive models.

First order statistics such as the mean or variance of pixel intensity values are the simplest of features. They give a global measure of the spread of intensity values. Various combinations can be used to measure characteristics such as the smoothness or uniformity of an image. Combining these statistical approaches with general feature methods, such as windowing or tiling, can extend their usefulness and power.

Grey level co-occurrence matrices (GLCM) are amongst the most widely used texture features. They were originally proposed by Haralick (1979). They are based on second order statistics, that is the distribution of grey levels of pairs of pixels, where the pairs are defined by the end points of a vector. A co-occurrence matrix of these pair values is built and various features created from this. The GLCM method was chosen for this study, full details are given in Section 4.3.2.

Grey level difference methods are similar to GLCMs in that they consider pairs of pixels defined by a vector. However, they use the absolute difference between the intensity values so are a first order statistical measure. Usually a histogram is built using the differences. The histogram can either be used directly as the feature or properties may be computed from it and these used instead.

Autocorrelation features capture information about the repetitive nature of a texture, such as regularity and coarseness. Autocorrelation treats the image as a two dimensional discrete process and the autocorrelation function is simply the cross-correlation of the image with itself. The rate of drop off of the function indicates how large the textural elements are; slower drop off equates to larger elements. If the elements are spatially periodic then the function will itself be periodic with repeating peaks and valleys. The autocorrelation function is the Fourier transform of the power spectral function (and vice versa).

Model based: This subdivision contains features based on parametric statistical models together with other significant methods.

Autoregression models can be used to synthesise texture as well as generate a feature representation. For a one dimensional signal they are a method of time series analysis where the signal is predicted from past values. If p samples are used then this is a Markov-p process. For an image, time is exchanged for two dimensional space thus the value of each pixel depends on its neighbourhood plus random noise values. The autoregressive moving average model is often employed. The image can be treated as a two dimensional autoregressive process or a seasonal one dimensional model where the rows (seasons) are concatenated. This is fitted using least squares. McCormick and Jayaramamurthy (1974) used this to estimate the parameters of a given texture and then synthesise it. An autoregressive distance measure for classifying textures was developed and tested by de Souza (1982). Mao and Jain (1992) successfully used a multi-resolution simultaneous autoregressive model for texture segmentation and classification. Their approach overcame one of the difficulties of autoregressive models, that is the selection of appropriate neighbourhood and window sizes, by enabling dynamic selection rather than empirically determined fixed sizes as used by other methods.

Markov Random Fields (MRF) are closely related to two-dimensional autoregressive models. MRF's were first used to model texture by (Hassner and Slansky, 1980). An image is represented as a lattice of pixel locations each with an intensity value. The assumption is that probability density function of the intensity of a pixel depends only on the intensity of its neighbours. The neighbourhoods can be defined in may different ways, such as the four first order, eight second order or cliques of neighbours. Unfortunately, to model real textures accurately a large number of parameters are needed making the models computationally infeasible. Hassner and Slansky (1980) recommended Gauss MRF as a practical alternative. These use a Gaussian probability density function to model pixel intensity thus simplifying the model and computation (Chellappa and Chatterjee, 1985).

Wold: Picard and Liu (1994) suggested a new model to measure perceptual similarity based on the two dimensional Wold decomposition. Wold (1938) originally applied his ideas to economic time series with Kolmogorov subsequently completing the work on prediction. The decomposition is a sum of three orthogonal components, a harmonic, a generalised evanescent and a purely indeterministic field. These correspond approximately to perceptual qualities of periodicity, directionality and randomness (Rao and Lohse, 1993). They were shown experimentally to provide a good texture discriminator.

Fractal models can be used to model the properties of self similarity across different scales. Correspondingly, the fractal dimension of a region in an image can be used to measure the roughness and scale of a texture. Keller et al. (1989) used the box dimension to estimate the fractal dimension. They also showed that this alone was insufficient to discriminate between all textures. A further measure, lacunarity, was proposed to address this. It uses the difference between the actual and expected mass of the fractal set to determine the coarseness. Dubuc et al. (1989) showed that the both the box dimension and *variation method* can be good choices for evaluating the fractal dimension of functions (images). In (Dubuc and Dubuc, 1996) they presented numerical approximations of the variation and derived error bounds that showed that this method should be more accurate than the the box dimension.

Tamura features are interesting because they were designed to mirror the results of psychophysical experiments. Tamura et al. (1978) identified six facets that humans use to differentiate between textures and then built features to quantify these. Three of these features proved to have the strongest correlation with human perception of texture: coarseness, contrast and directionality. These were chosen for the study and are described Section 4.3.4.

Geometrical methods view a texture as being composed of elements and analyse the geometric properties of these. These methods, whilst appealing, have not been widely adopted.

Voronoi tessellation of an image to extract texture elements was proposed by Tuceryan and Jain (1990). Tokens are extracted by first applying a difference of Gaussians filter and then selecting pixels that lie on a local maximum. This creates a binary image from which connected components are identified which define texture tokens. From these the Voronoi tessellation can be constructed and a feature generated using the moments of area of the polygon. They evaluated their method using a texture segmentation task. The test images contained patches of Brodatz textures, which the method successfully separated. The artificial test images gave a very coarse, well defined segmentation problem. The effectiveness of this method for image retrieval has not been tested.

Structural models assume that a texture is generated by rule based placement of texture elements. Most methods use blobs, i.e. regions of uniform intensity in the image, as the texture element.

Generally, they appear to be only effective when analysing a very regular texture (Tuceryan and Jain, 1990).

Signal processing techniques have been fruitfully applied to texture classification and segmentation tasks for many years. The majority of these approaches filter the image in some way and then calculate features from the energy of the transformed image. The filters are usually based on either a family of edge detectors, second order derivatives, or frequency and orientation filters such as Gabor or wavelets.

Spatial domain filtering, the simplest of the signal processing approaches, uses edge detection masks such as Roberts or Laplacian operators. These give a transformed image where the response over an area indicates the "edgeness" of the original. Other types of convolution filters have been used with success. Malik and Perona (1990) used a bank of even-symmetric filters (differences of offset Gaussians) followed by half wave rectification and inhibition of spurious responses. Their method introduces a non-linearity which enables discrimination between textures with identical first and second order statistics.

Gabor filters are the most popular signal processing approaches to feature generation. This is probably due to their robust performance across a range of tasks. The window Fourier transform filters a signal in both the frequency and spatial domains. The width of the window determines the frequency and spatial resolution. Unfortunately, as one improves the other degrades. The Gabor transform is obtained by using a Gaussian window function. This has the advantage that it achieves the best possible frequency decompositions. A two-dimensional Gabor function can be used to construct a bank of filters that are selective to a range of frequencies and orientations. The characteristics of the response to these filters can then be used as a feature. Complete details are given in the Section 4.3.3. The first use of Gabor filters for texture analysis was done by Turner (1986).

Wavelets are another method of spatial and frequency filtering. They do not suffer from the same trade off between frequency and spatial resolution as the window Fourier transform. They can give good spatial resolution for high frequency events and vice versa. Kundu and Chen (1992) first used a quadrature mirror filter (QMF) bank for texture classification. A QMF splits the input signal into two bands which are then recursively subsampled. Orthogonal wavelets such as the Haar and related Daubechies wavelets can be scaled to satisfy the QMF relationship. The image is filtered by the sub-bands at each decomposition level. The feature vector is then constructed from the mean and variance of the energy distribution of each transformed image.

4.2.2 Related work

Work on evaluating the performance of texture features varies both in the features covered and the application area used. To my knowledge the only quantitative comparison of feature performance for CBIR was carried out by Deselaers et al. (2004). They evaluated the performance of a range of colour and texture feature on the WANG and IRMA databases. The former is a hand-picked subset of 1000 images from the Corel collection whilst the latter is a database of 1,617 medical radiographs. Their work, predictably, showed that different features performed better depending on the task. They did not focus on texture or look at modifying the features for CBIR.

Other evaluations have not used the CBIR task but do provide insight into feature performance. One of the earliest was by Wezska et al. (1976). They compared Fourier power spectrum with co-occurrence, grey level run length and second order grey level statistics, with the co-occurrence features performing the best. Several studies have compared filtering methods against co-occurrence methods. The results have varied presumably because of different tasks, setups, collections and implementations. For example, Strand and Taxt (1994) found in favour of co-occurrence methods where as Pietikäinen et al. (1983) conclude that the filtering approach was better. A detailed paper by Randen and Husøy (1999) compared many filtering methods with co-occurrence and autoregressive features using a classification task. Their conclusions stated that there was no single approach that performed close to the best for all images. They would not give a general recommendation but suggested testing Gabor filter banks, autoregressive and co-occurrence features with the specific data set. Ohanian and Dubes (1992) compared fractal, co-occurrence, Markov random field and Gabor features. They found that co-occurrence generally performed the best. Other comparison studies include those by Gotlieb and Kreyszig (1990) and Jain and Farrokhnia (1991).

All of these studies used specific texture image collections rather than generic pictures. The VisTex Database has also been used for texture feature evaluation work. It does contains both reference textures and texture scenes. Although its images are real-world pictures they are subsequently divided into homogeneous patches and used in a classification context rather than for an image retrieval task.

Summary

What general principles can be drawn from this review of the literature? The first is that no one texture feature is the best in all circumstances. Performance depends on both the collection and the task. It should be remembered that implicit within each task is a specific performance measure. Very little work has been done with an image search task that produces a relevance ranking. This study homes in on what makes a good feature for CBIR. In addition it attempts to identify characteristics of images that should be captured in relation to the search task. By doing this I aim to crystallise some broad principles for texture feature design.

4.3 Texture and how to capture it in a feature

In this section we look at the details of three texture features. Before this we discuss one of the main theories of human visual perception and relate this to the representation of texture within images. This links into the theme, developed in the introductory chapter, of CBIR as an application to assist visual perception.

For this study we have chosen three features originating from very different viewpoints: Tamura, co-occurrence and Gabor features. These cover three of the major texture feature methodologies: psychological, statistical and, signal processing. Within their respective categories, each feature performs amongst the best for both classification and segmentation tasks. They have not been refined too far from the original idea, hence, are good representatives of their respective classes. Therefore, they should give an insight into what characteristics of features perform well on a CBIR task, as well as allowing the tuning of each specific feature. For these reasons we feel justified in the choice of these features for our experiments.

4.3.1 Texture

This section aims to show the close link between how we choose to define texture and the subsequent construction of features. To understand what makes a good feature we first discuss a model of visual perception that proposes that texture is recognised at a primitive level. Then we link this to how texture is represented in an image. This allows us to determine what are characteristics are likely to be the most discriminatory.

Visual perception

The theory put forward by Marr (1982) treats vision as an information processing task. The input for visual processing is the retinal image formed by light reflected from the structures a person is viewing. This can be thought of as an array of intensity values. The first stage of visual processing aims to interpret these varying intensities as a *primal sketch*. The primal sketch captures information about edges, blobs and regions withing the image. After this comes the recognition of objects and events.

This fits in with the suggestion that texture is recognised in the pre-attentive phase of vision processing. That is, it is one of the very low level properties recognised rapidly and accurately by the lowest level of visual processing. It is possible to capture some essence of a scene with a very quick glance, which equates to the primal sketch. After this stage comes the focused attentive mode where detailed recognition and understanding occurs. If this model of vision is accepted then texture is seen using the simple, primitive properties of an image. All texture features work on this basis, by assuming that the characteristics of a texture can be extracted from an image using some low level property of the image.

Texture in an image

Synonyms for the noun texture include, structure, organisation, pattern and plan (Roget's Thesaurus, Kirkpatrick, 1987). These include the two main ways that texture can manifest itself on surfaces, as pattern or structure. Patterns are variations of tone or colour within the material while a structure implies that there is a physical variation in the surface. Often these can be felt as well as seen, such as on a rough, smooth or hairy surface. To view the underlying structure or pattern that makes up the texture we need light to illuminate the surface. The quality of light, for example focused or diffuse, will affect how the texture appears by creating different shadows and relief. So it is the combination of surface and light that creates a visible texture.

When a two dimensional image is used as a representation of a scene, all texture is reduced to pattern. For a digital image patterns are just changes in pixel intensity. At some scales the pattern may become homogeneous over a region in the image. Thus texture may appear at varying scales in the same region of an image. Consider a brick wall; at the finest detail there is the roughness of the brick itself, moving up a level there are repeating bricks and mortar and at a larger scale the walls themselves may form a pattern, as in a row of houses.

The obvious characteristic from this description is the multi-scale nature of texture withing an image. The scale that a texture appears at will depend on how the subject was photographed, for example, if any magnification was used. So the same texture may appear at different scales in two pictures. Matching the same surface at differing scales is a difficult task. Indeed, for a search task it is likely that the user would regard the images as being quite different. The resolution of the imaging process will act as a limitation on the scale at which texture becomes apparent. The finer levels of texture from a scene may well not be visible in the image. In addition there may be texture artifacts introduced by the image making process itself. With halftone printing, used for posters, the dots used to make the image form a texture. Similarly, with film photography there is the grain of the film or the pixels for digital media. An additional texture may be introduced by noise, for example when high gain is used in low-light conditions or with compression techniques. These are not part of the original scene but have been introduced by the image making process.

So there is a bound on the fine scale of a texture in a picture. The upper bound is more a characteristic of the photographers composition. Very large textures by their nature are only likely to occur in more abstract, close-up pictures, where the texture occupies most of the image. The region a texture occupies is another significant characteristic. If we consider that all real surface have textures, then images that are pictures of real scenes will always consist of a patchwork of textures.

Another characteristic appearing in the brickwork example is orientation. In the case of the mortar between the bricks the orientation is vertical and horizontal. However, the texture of the brick surface itself has no distinct orientation. Finally there is the variation between intensity levels in the texture pattern. This will manifest itself as the contrast or sharpness of the texture.

Visual features for CBIR operate at the primitive level of colours, textures and shapes. Features alone cannot hope to disambiguate the polysemous nature of a picture. What they can do is try and differentiate between the visual facets that a user wants to search by. For textures we have identified *scale, region, orientation* and *intensity variation* as some of the main visual properties. It is these that we wish to capture within our features.

Next we look at some of the models for texture that researchers have developed and how these naturally lead to feature extraction methods.

4.3.2 Second order statistics – co-occurrence matrices

The statistical properties of grey levels were some of the earliest measures used to classify textures. One of the first low level properties put forward to describe texture was simple statistics of intensity distribution. In his early work, Julesz (1962) and Julesz et al. (1973), originally suggested that first and second order statistics were sufficient to describe all the textures that a person could discriminate between. First order statistics include the average and variance of pixel intensity values. Second order statistics consider the distribution of intensities of pairs of pixels at either end of a vector. The length and direction of the vector can be varied to capture direction and scale information. Julesz carried out much work in this area, initially finding that textures with the same first and second order statistics appeared the same to viewers. Later he decided that this original finding was incorrect in some cases. This led to him developing the idea of textons (Julesz, 1981). He proposed that textons, elongated blobs with properties of colour, orientation, dimensions and endpoints, were the basic building blocks of texture.

A huge number of texture features have been based on the statistics of pixel intensities in an image. Many of the later approaches have used parametric models such as those described earlier. However, the widely used co-occurrence matrix is almost certainly the most important feature derived from this statistical viewpoint. Haralick (1979) suggested the use of grey level co-occurrence matrices (GLCM) to extract second order statistics from an image. GLCMs have been used very successfully for texture classification, often outperforming all others in evaluations (Ohanian and Dubes, 1992).

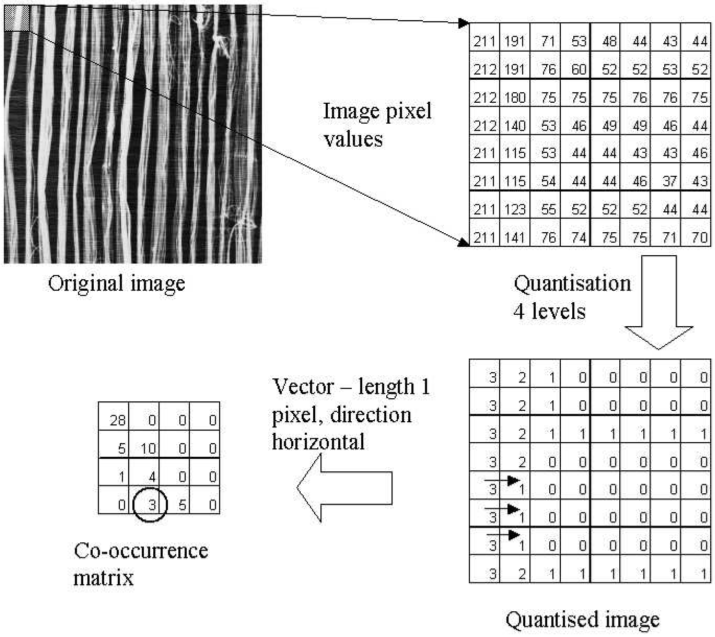
Implementation

Haralick defined the GLCM as a matrix of frequencies at which two pixels, separated by a certain vector, occur in the image. The distribution in the matrix will depend on the angular and distance relationship between pixels. Varying the separation vector allows the capture of different texture characteristics.

Figure 4.1 shows a simplified example of the construction of a GLCM. The rows of the co-occurrence matrix represent the value at the start of the separation vector and the columns the value at the end (where $(0, 0)$ is the top left corner). The circled entry in the GLCM therefore represents the number of

Feature	Formula
Energy	$\sum_a \sum_b P^2(a,b)$
Entropy	$\sum_a \sum_b P(a,b) \log P(a,b)$
Contrast	$\sum_a \sum_b (a-b)^2 P(a,b)$
Homogeneity	$\sum_a \sum_b \frac{P(a,b)}{1+\lvert a-b \rvert}$

Table 4.1: Features calculated from the normalised co-occurrence matrix $P(a,b)$



211	191	71	53	48	44	43	44
212	191	76	60	52	52	53	52
212	180	75	75	75	76	76	75
212	140	53	46	49	49	46	44
211	115	53	44	44	43	43	46
211	115	54	44	44	46	37	43
211	123	55	52	52	52	44	44
211	141	76	74	75	75	71	70

3	2	1	0	0	0	0	0
3	2	1	0	0	0	0	0
3	2	1	1	1	1	1	1
3	2	0	0	0	0	0	0
3	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
3	2	1	1	1	1	1	1

28	0	0	0
5	10	0	0
1	4	0	0
0	3	5	0

Figure 4.1: Construction of a grey level co-occurrence matrix

occurrences of the separation vector [1,0] that have a start value of 3 and an end value of 1. The three occurrences of these in the quantised image are shown by the arrows representing the vectors.

Once the GLCM has been created, various features can be computed from it. These have been classified into four groups: visual texture characteristics, statistics, information theory and information measures of correlation (Haralick, 1979; Gotlieb and Kreyszig, 1990). Four features which have been successfully used for texture classification were chosen for this evaluation. These are listed in Table 4.1. Note that $P(a,b)$ is the frequency that quantised grey level a co-occurs with quantised grey level b, separated by a vector v.

4.3.3 Multiscale filtering – Gabor filters

Moving away from the statistical viewpoint, many psychophysical researchers, including Campbell and Robson (1968) and de Valois et al. (1982), carried out experiments on the response of cortical cells to compound gratings with differing orientations and spatial frequencies. Their results led them to conclude that the vision processing system operates in multiple channels, each sensitive to a different band of frequencies. Later research showed that at high contrast the channels do interact and so are not totally independent. However, all the research still supports the existence of multiple spatial filters in the early stages of visual processing. Graham (1989) presents a thorough exposition of this area.

This multiple filtering theory of vision led to the idea of representing visual responses using signal processing methods. Daugman (1980) proposed that the receptive fields in the visual cortex could be modelled using two-dimensional Gabor filters. Turner (1986) then suggested that the same filtering approach could be used for texture analysis. This ties in with the earlier discussion that texture is a primitive property, identified in the early, pre-attentive, phase of vision.

Many spatial filtering methods for texture analysis have sprung from these ideas. However, the Gabor filter remains the most popular of the signal processing based approaches for texture feature extraction. A bank of filters at different scales and orientations allows multichannel filtering of an image to extract frequency and orientation information. The responses of these filters can then be used to decompose the image into texture features.

Implementation

My implementation is based on that of Manjunath and Ma (1996), later expanded in (Manjunath et al., 2000). The feature is computed by first filtering the image with a bank of orientation and scale sensitive filters and then computing the mean and standard deviation of the output in the frequency domain.

In the spatial domain a Gabor function is a sinusoid modulated by a Gaussian with an aspect ratio of σ_x/σ_y. A two dimensional Gabor function can therefore be expressed as

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right) + 2\pi jFx\right). \tag{4.1}$$

We can generate a bank of self-similar filters by appropriate dilations and translations of the parent wavelet $g(x,y)$. For this we follow Manjunath's method using the generation function:

$$g_{mn}(x,y) = a^{-m}g(x',y') \quad \text{with} \quad a > 1 \quad \text{and} \quad m,n \quad \text{both integers,} \tag{4.2}$$

$$x' = a^{-m}(x\cos\theta_n + y\sin\theta_n) \text{ and } \quad y' = a^{-m}(-x\sin\theta_n + y\cos\theta_n), \tag{4.3}$$

where $\theta_n = n\pi/K$ and K is the total number of orientations. The scale factor a^{-m} is meant to ensure that the energy is independent of m.

We can now generate our bank of filters following a design strategy that ensures that the half-peak

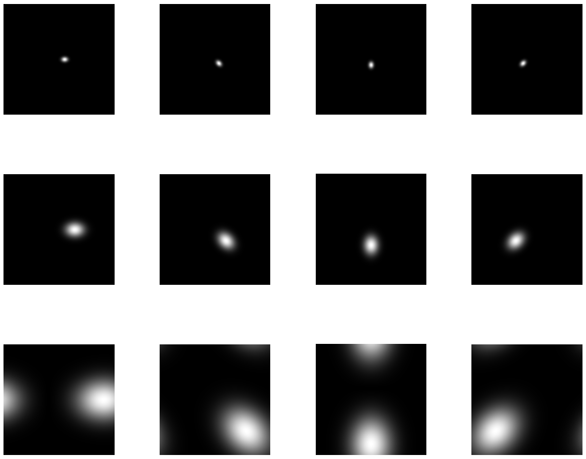


Figure 4.2: Filter responses of a Gabor filter bank with 3 scales and 4 orientations

magnitude supports of the filter responses in the frequency domain touch each other (Manjunath and Ma, 1996). An example of a filter bank generated in this way (using Matlab) is shown in Figure 4.2.

Filtering an image I with Gabor filters g_{mn} results in its Gabor wavelet transform

$$W_{mn}(x,y) = \int I(x_1,y_1) g^*_{mn}(x-x_1, y-y_1) dx_1 dy_1. \tag{4.4}$$

The mean and standard deviation of the magnitude $|W_{mn}|$ are used for the feature vector. The outputs of filters at different scales may be over differing ranges. For this reason each element of the feature vector is normalised using the standard deviation of that element across the entire database.

4.3.4 Psychophysical experiments – Tamura features

Tamura et al. (1978) took an alternative approach of devising texture features that correspond to human visual perception of texture. Through experiments with people they identified six characteristics (coarseness, contrast, directionality, line-likeness, regularity and roughness) that are used to discriminate textures. Features were designed to quantify these characteristics and the results compared with psychological measurements using the same people. The first three were the most closely matched to human perception and proved to be the best at classifying textures. Hence, these three features are used in this study, both separately and as joint values.

Implementation

Coarseness has a direct relationship to scale and repetition rates and was seen by Tamura *et al.* as the most fundamental texture feature. An image will contain textures at several scales; coarseness aims to identify the largest size at which a texture exists, even where a smaller micro texture exists. Computationally one first takes averages at every point over neighbourhoods the linear size of which are powers of 2. The average over the neighbourhood of size $2^k \times 2^k$ at the point (x, y) is

$$A_k(x,y) = \sum_{i=x-2^{k-1}}^{x+2^{k-1}-1} \sum_{j=y-2^{k-1}}^{y+2^{k-1}-1} I(i,j)/2^{2k}, \tag{4.5}$$

where $I(i, j)$ is the grey level at the image pixel coordinates (i, j). Then at each point one takes differences between pairs of averages corresponding to non-overlapping neighbourhoods on opposite sides of the point in both horizontal and vertical orientations. In the horizontal case this is

$$E_{k,h}(x,y) = |A_k(x + 2^{k-1}, y) - A_k(x - 2^{k-1}, y)|. \tag{4.6}$$

At each point, one then picks the size of k that maximises E in either the horizontal or vertical direction; this is k_{opt}. The coarseness measure is then the average of $S_{\text{opt}}(x, y) := 2^{k_{\text{opt}}}$ over the picture.

Contrast aims to capture the dynamic range of grey levels in an image, together with the polarisation of the distribution of black and white. The first is measured using the standard deviation of grey levels and the second the kurtosis α_4. The contrast measure is defined as

$$F_{\text{con}} = \sigma/(\alpha_4)^n. \tag{4.7}$$

The kurtosis is calculated by dividing the fourth moment about the mean, μ_4, by the variance squared,

$$\alpha_4 = \mu_4/\sigma^4 = \frac{\sum_i \sum_j (I(i,j) - \mu)^4}{N\sigma^4}, \tag{4.8}$$

where μ is the mean intensity and $N =$ the number of samples.

Experimentally, Tamura found $n = 1/4$ to give the closest agreement to human measurements. Initial retrieval experiments confirmed this choice, and this is the value used in the following experiments.

Directionality is a global property over a region. The feature described does not aim to differentiate between different orientations or patterns, but measures how directional an image is. Two convolution masks are applied to the image to detect edges. The masks used are horizontal and vertical Prewitt operators. These estimate the rate of change of intensity over each 3×3 region of the image. The results of these are combined to approximate the angle and and magnitude of intensity change (edge) at each pixel. A histogram, H_d, of edge probabilities is then built up by counting all points with magnitude

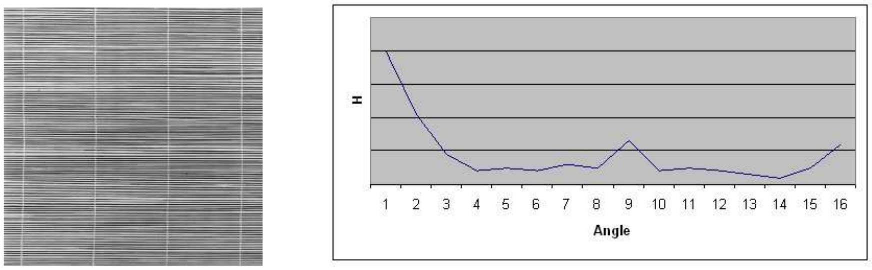


Figure 4.3: An example image and its direction histogram

greater than a threshold and quantising by the edge angle. The peaks in the histogram will reflect the degree of directionality. To extract a measure from H_d the sharpness of the peaks are computed from their second moments. An example image and its direction histogram is shown in Figure 4.3. This shows a main peak in the first bin which represents the horizontal direction in the image and a smaller peak in the ninth bin representing the vertical component.

Tamura Image is a notion where I calculate a value for the above three features at each pixel and treat these as a spatial joint Coarseness-coNtrast-Directionality (CND) distribution, in the same way as images can be viewed as spatial joint RGB distributions. The regional nature of texture means that the values at each pixel are computed over a window. Colour histogram style features, both marginal and 3D, can then be created using the Tamura CND image. A similar 3D histogram feature was used by MARS (Ortega et al., 1997).

4.4 Experimenting with Texture Features

The aim of this experimental work is twofold. Firstly, to evaluate and modify the specific features for the task of CBIR; secondly, to pull together some general principles for capturing texture in features. In this way I provide both concrete recommendations and more general guidelines and intuition that can be applied elsewhere. The work focuses on the representation of texture identified earlier, region, scale, orientation and variation. These guide the pre-experimental questions and development work.

The experimental approach is explained in the next section. Following this, some general properties of features are discussed, including tiling and windowing. These are particularly applicable to texture because of its regional properties. The evaluation and feature modification work is described. This starts with an investigation into the effect of tiling on feature performance. Each of the three features is then discussed in detail with modifications evaluated using the Corel collection. Following this the best performing features are re-evaluated using both the TRECVID 2003 and ImageCLEF 2004 collections. This includes combining texture and colour features and looking at the effects.

4.4.1 Experimental set up

Full details of the collections and methods used are in Chapter 3. A three-stage approach was used for this work. Initial evaluation and modifications to the features were tested using a subset of the Corel image library. Further tests were then run on the TRECVID 2003 (Smeaton et al., 2003) data to confirm the findings on a large collection with a different search task. This was followed by an external evaluation using the best texture features. This was a submission to the image track of the Cross Language Evaluation Forum (ImageCLEF) (Clough et al., 2005; Howarth et al., 2005). It demonstrated the effectiveness of the features on another, very different, search task.

Mean average precision was used throughout the work as a performance measure. Where necessary results were tested for statistical significance using the paired Student's t-test with $\alpha = 5\%$. For TREC multiple image queries a version of the distance weighted k-nn approach, with $k = 40$, was used to combine distances from each query image, refer to Section 3.4.2. Single image queries were used for the Corel and ImageCLEF experiments.

4.4.2 General properties of features

In this section some general properties of features are discussed. These can have a big impact on the information captured and subsequent search performance. We have already noted that texture is a regional property of an image rather than a point property like colour. A single pixel does not have a texture. Therefore we need to choose an area over which to calculate the feature. The most obvious choice for this is the entire image. However, this gives a single value and does not take into account that an image is really a patchwork of texture regions. To capture the regional information a technique such as tiling or windowing can be used. With these methods it may be possible to isolate an area where the texture is more homogeneous.

Tiling is where the image is split into tiles or sub-images and the features calculated for each tile. This will capture the regional variation across the image. The features from each tile can be concatenated together or considered independently. The former will capture some of the structure of the image, such as sky occurring in the top tiles. The latter will allow matching of textures that occur at different locations.

Windowing is where a fixed size window is slid across the image and the feature calculated at each position. The size and movement of the window can be varied but essentially the feature is capturing a texture value for areas centred on certain points in the image. An example of this, where the window is moved one pixel at a time, gives the Tamura Image feature as described earlier. Often the values for each window position are collected into a histogram. If each point or window has more than one value there is a choice of constructing a marginal or joint distribution histogram.

Finally there is the issue of normalisation of features. One option is not to normalise the raw feature. Another choice is to normalise the entire feature to a value. An example is where the area of a histogram is normalised to one. This makes each feature a vector of the same length. Normalisation of this type can be done with respect to the Euclidean or Manhattan distance (or other functions as we see in the next chapter). Lastly each component of a feature can be normalised across the collection. A method used by the Gabor feature is to make the distribution of each element have the same mean or variance and thus give them the same weighting. Robust estimators, such as the median, can be used to reduce sensitivity to outliers. Each of these techniques affects the weighting of complete features or elements within the feature. By normalising we are discarding some information from the feature, in each case we need to decide if this is the correct action to take.

4.4.3 Image tiling

As part of the evaluation I carried out experiments to determine the effect of applying the features to image tiles. The method was to split the image into non-overlapping tiles and calculate the feature for each tile. The resulting features for each tile are then concatenated into a single vector. Tilings up to 11×11 were evaluated. The effect of tiling is to add location information to the feature and isolate regions of the image. It is more likely that a homogeneous texture will occur over these smaller regions. The distance between images will be the sum of the distances between corresponding tiles.

The effect of tiling was investigated with all three texture features. It was found that retrieval performance consistently improved with tiling. The graph in Figure 4.4 shows the average retrieval performance for different tilings with the Tamura features (Section 4.4.5). The detailed results can be seen in Table 4.4. The biggest jump is going from no tiling to 3×3, with returns diminishing as the number of tiles is increased. It was found that performance peaked at 7×7 or 9×9 depending on the feature.

Beyond the optimum level of tiling there was no significant gain. In fact performance began to degrade with tilings above 9×9. As the image tiles decrease in area the features have smaller regions to work on, thus reducing their effectiveness. At the extreme limit the feature will approximate to comparing at images pixel level. It appears that high orders of tiling begin to add noise to feature rather than further information. This was for images of a typical linear size of 300 pixels and was a general finding, common across all the features investigated.

4.4.4 Co-occurrence

There are many details and parameters to consider when constructing co-occurrence matrices. These issues are presented below together with a discussion of their impact on capturing texture. The results shown in Tables 4.2 and 4.3 are for features generated with 7×7 tiling, as described previously.

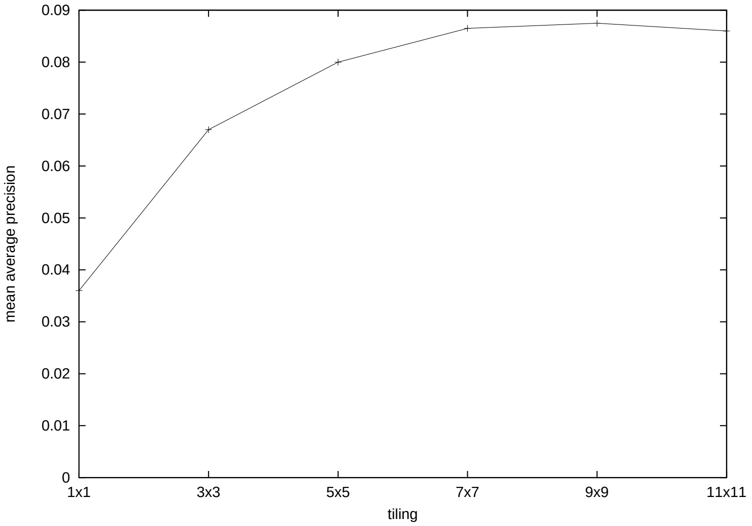


Figure 4.4: The effect of tiling on mean average precision using Tamura features

Vector

The vector used to choose pixel pairs defines both the scale and orientation that the feature can detect. Normally a range of vector angles and distances are used to construct a set of GLCMs that will capture the scale and direction of texture in the image. Results of some initial experiments are shown in Table 4.2. These showed that four vector angles (0, 45, 90, 135 degrees) and four distances of between 1 and 4 pixels, consistently gave the best discrimination. This combination can be used to calculate up to sixteen GLCMs. A feature is calculated from each matrix and the resultant values concatenated together. The results in the table are using the energy feature. The next two items explain further variations relating to how the vector can be used.

Symmetry

Each instance of a vector has a direction and starting point associated with it. By including the vector in the reverse direction the GLCM can be made symmetrical. This approach has been widely used. However, it was found experimentally to give no significant improvement over the asymmetric version. As the symmetrical version takes extra effort to compute it was not adopted.

Rotational invariance

Another approach described in the literature is aimed at making the feature invariant to the orientation of a texture. This is done by combining the pairs for all the angles at a distance into a single GLCM. This will create a smaller feature. The results in Table 4.3 show that across a range of features the concatenated features outperform the rotationally invariant version.

No. of Orientations	Pixel distances			
	1,2	1,2,4	1,2,4,8	1,2,3,4
2	7.8	8.8	8.8	9.2
3	7.9	8.9	9.6	9.8
4	8.2	9.8	9.8	10.4
6	8.4	9.8	10.0	10.2

Table 4.2: Mean average precision (%) with varying vector combinations

	Quantisation in bits					
Feature	2	3	4	5	6	7
Energy: cat	7.6	8.1	9.3	9.9	9.5	9.3
Energy: sum	7.0	7.8	8.9	9.2	9.0	8.8
Entropy: cat	8.1	9.2	10.4	11.1	11.4	11.3
Entropy: sum	7.5	8.8	9.8	10.4	10.7	10.6
Contrast: cat	8.5	8.5	8.4	8.3	8.3	8.2
Contrast: sum	7.8	7.9	7.7	7.6	7.6	7.5
Homogeneity: cat	9.2	10.2	11.2	11.8	12.2	12.0
Homogeneity: sum	8.5	9.5	10.4	10.9	11.3	11.2

cat = 16 concatenated matrices
sum = 4 rotationally invariant summed matrices

Table 4.3: Mean average precision (%) with varying co-occurrence features and quantisation

Quantisation

The number of quantisation levels used dictates the size and density of the matrix. More is not necessarily better, as with small images or tiles the matrix may become very sparse. Quantisations between 2 and 7 bits were tried. From Table 4.3 we can see that using 5 or 6 bits usually gives the best performance. The exact optimal level for this collection varies with the feature.

Feature

A final and obvious choice, was which feature to calculate from the GLCMs. The four evaluated cover the major families. From Table 4.3 we can see that the largest variation is for 6 bit quantisation from homogeneity with 12.2% to contrast 8.3%. At lower quantisation levels the variation is much less. Indeed the contrast feature outperforms energy and entropy with 2 bit quantisation. It is dangerous to draw too strong a general conclusion from these results. However, for the Corel collection it is clear the the homogeneity feature outperforms the others.

From these results we can draw conclusions for this feature and collection. It is interesting that increasing the scales of textures considered (vector length) does not continually improve performance.

It appears that for image search the important textures occur at relatively small scales. This is a theme that reoccurs with the later features. Similarly, increasing the number of orientations or quantisation bits gives diminishing returns and ultimately reduces performance. There is a point beyond which these add no further information.

Some specific conclusions for this feature are:

There was no significant difference between generating matrices symmetrically or asymmetrically.

Increasing quantisation up to 5-6 bits improves performance. The optimum level depends on the feature.

Concatenated matrices outperformed the rotationally invariant summed matrices.

With this collection the homogeneity feature performed best with a m.a.p. of 12.2%.

4.4.5 Tamura

Each of the individual Tamura features is calculated for a region of the image. Two methods to define areas, tiling and windowing, were tested. The results for each method are shown in Tables 4.4 and 4.5, respectively. For tiling, the features are calculated for each individual tile. The resulting feature is referred to as the standard feature. The size of the image or tile places a limitation of the largest value of k that can be used for the coarseness calculation. Where this is exceeded a dash indicates this in the table.

With the windowing method a window is slid across the image and the feature value calculated at each position. This is then used to build a histogram. Coarseness is calculated for an area centred on a pixel. The coarseness feature is calculated at each pixel within the window but the borders may extend outside the window if required. Both the directionality and contrast features operate over a region. A large window may smear the features and lose resolution; conversely a small window may invalidate the feature, for example if the directionality histogram is too sparsely populated. To evaluate this the features were run over several window sizes, creating a histogram for each feature, see Table 4.5. The histograms were all L_1 normalised. This means that the components of the vector are scaled so that they sum to one (see Section 5.5.2 for experiments with normalisation).

Contrast

The contrast feature is the simplest of the three. I experimented with various values of n from Equation 4.7 and was in agreement with Tamura's finding that $n = 1/4$ gave the best results. The feature performed slightly better with tiling rather than windowing, 8.1% versus 7.0%.

Coarseness

Tamura highlighted coarseness as the most important of his six features. It identifies the scale at which the largest intensity variation occurs rather than any micro-texture that may be present. A major parameter that can be altered is k, the maximum scale that is considered.

A little surprisingly, initial results showed that increasing the maximum k value considerably reduced the performance of the coarseness feature. The optimum value was 2 for both the tiled and histogram feature. This may be due to the large borders necessary for higher values of k. However, it is more likely caused by the nature of textures in images and the way the algorithm averages the 2^k values. In normal images textures occur at much smaller scales than in Brodatz style texture images. Correspondingly, there are unlikely to be textures with a coarseness of 64 or 32 pixels in images with a typical diagonal of 450 pixels. The algorithm may still detect noise at this dimension, biasing the average value of the feature and masking any finer texture present.

A change to the algorithm was made so the measure took the values of k rather than 2^k. The value of k_{opt} was found as before, but now the coarseness measure is an average of $S_{\text{opt}}(x, y) := k_{\text{opt}}$ over the image. This effectively introduces a logarithmic scaling of the coarseness value and gives less influence to the larger scales. This change gave a significant increase in performance for the windowed feature, from 6.1% to 10.1%, but no improvement when applied to the standard feature.

Directionality

Performance of the original directionality feature was poor, 5.6% for the window and 6.6% for the standard feature. A detailed look at the operation of the algorithm showed that this was largely due to the sparse population of the direction histograms and subsequent difficulty in calculating a valid variance of their peaks. The feature value was sensitive to slight changes. Several options for improvement were tried including calculating global variance of the direction histogram and using entropy. The latter gave a substantial improvement, from 6.6% to 9.7%, for the standard feature but negligible effect on the windowed feature. The entropy of the normalised histogram H_d was computed using $\sum_{i=0}^{d} H_i \log H_i$.

Combined

The three individual features were combined into a single feature using two methods. The first was simply to compute the standard features for each tile and concatenate the values together. For a 7 × 7 tiling this gave mean average precision of 14.3%.

The second approach was to generate marginal and 3D histograms. This was done by calculating the three values, coarseness, contrast and directionality (CND) for points across the image using a window size of 8. This could be done using every pixel as a point, or speeded up by moving in larger steps. In this way a TamuraImage was created using the CND triples. All the techniques usually employed with

	Tiling				
Feature	1x1	3x3	5x5	7x7	9x9
Contrast	3.2	6.1	7.2	8.1	8.0
Directionality: peak finding	2.9	4.2	5.0	5.8	6.6
Directionality: entropy	2.7	5.4	7.5	8.9	9.7
Coarseness 2^k: max $k = 2$	4.4	8.3	9.5	9.9	9.9
Coarseness 2^k: max $k = 3$	3.5	7.6	8.8	9.2	9.0
Coarseness 2^k: max $k = 4$	3.5	7.2	7.7	7.0	—
Coarseness 2^k: max $k = 5$	3.3	5.7	—	—	—
Coarseness 2^k: max $k = 6$	2.9	—	—	—	—
Coarseness k: max $k = 2$	4.4	8.0	9.3	9.6	9.6
Coarseness k: max $k = 3$	3.9	7.5	8.9	9.1	8.9
Coarseness k: max $k = 4$	3.4	7.0	7.7	7.2	—

Table 4.4: Mean average precision (%) for standard Tamura features

	Window size			
Feature	2	4	8	16
Contrast	6.0	6.7	7.0	6.9
Directionality: peak finding	5.4	5.6	5.6	4.9
Directionality: entropy	4.9	4.4	5.2	5.4
Coarseness 2^k: max $k = 2$	6.9	6.0	6.1	6.0
Coarseness 2^k: max $k = 3$	6.5	5.9	6.0	5.8
Coarseness 2^k: max $k = 4$	6.1	5.7	5.6	5.4
Coarseness k: max $k = 2$	6.4	10.0	9.8	8.2
Coarseness k: max $k = 3$	5.7	10.1	9.2	7.9
Coarseness k: max $k = 4$	8.8	9.3	8.1	7.7

Table 4.5: Mean average precision (%) for windowed Tamura features

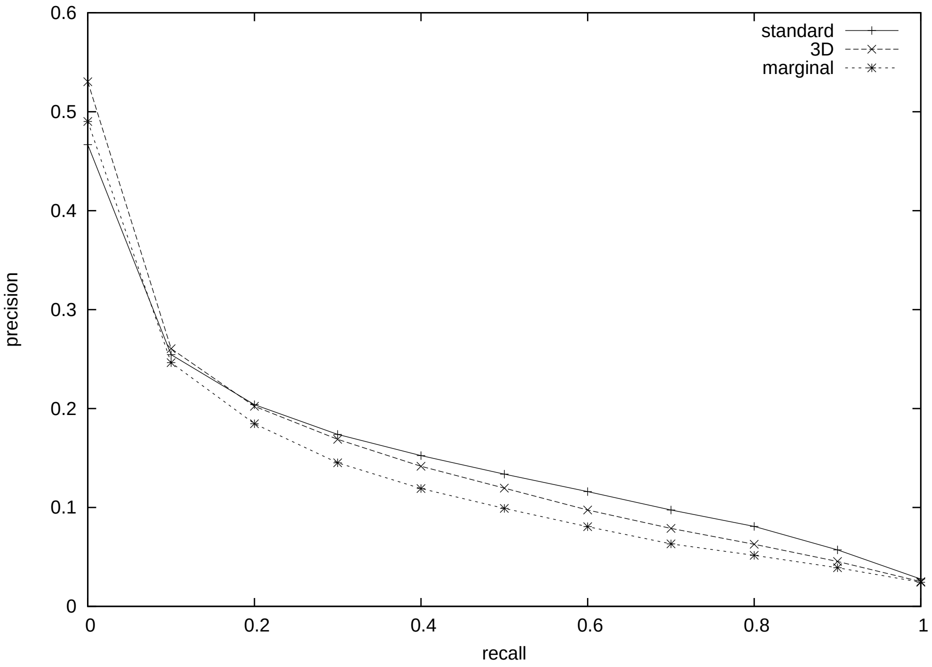


Figure 4.5: Precision-recall graph for combined Tamura features, Corel collection

colour histograms could then be used. This feature gave mean average precisions of 12.0% for a marginal histogram and 13.7% for the 3D histogram.

All combination methods gave a significant improvement over the single features. Precision recall graphs for these combined features are shown in Figure 4.5. Precision is similar for the three approaches at 10% recall with the difference widening at higher recalls.

4.4.6 Gabor

As with the other features there are two main questions that present themselves when applying the Gabor feature to a heterogeneous set of images in which texture patches of varying size, scale and orientation occur. The first is what configuration of filter bank best captures the scales and orientations occurring in the images. The second is how to apply these filters, either to tiled images or using a sliding window to build a histogram.

Tiling versus windowing

I evaluated the feature in two configurations across a range of scale and orientation values. The first scaled the filter dictionary to the size of the image or image tile. The effect of this should be to scale the response so that the same image of different size will give a similar value. The second approach was to use a fixed size filter. This was then applied to a sliding window over the image and a marginal

Scales	Orientations		
	3	4	6
2	13.1	14.0	13.9
3	11.0	11.4	11.3
4	10.8	11.4	11.2

Table 4.6: Mean average precision (%) for Gabor wavelets

histogram built using the response from each filter.

Initial results showed that scaling the filter size gave much superior results to the sliding window approach. Mean average precisions achieved with the sliding window were in the range 5–7%, approximately half that obtained with the tiling method. The results shown in Table 4.6 are for 7 × 7 tiling. It was found that tiling increased performance in a similar manner to the other features.

Choice of filter bank

The configuration of the filter bank has a direct impact on the resolution of scale and orientation that is captured by the feature. The results in Table 4.6 show that the best performance is achieved using a filter bank of 2 scales and 4 or 6 orientations, the difference between these two combinations was not statistically significant. Indeed the feature with 2 scales outperforms all others with the number of orientations tested.

This finding indicates that the best choice of filter bank is different from those used by other researchers. Most literature recommends 4 scales and 6 orientations although it is not clear if different filter banks have been methodically tested. It must be remembered that previous work was predominately aimed at texture classification rather than CBIR.

It was instructive to look at the filtered images for at the different scales. These indicated that, as for Tamura, response at coarser scales was predominately from objects rather than textures. With the structure of filter bank employed there is no advantage to be gained by increasing the selectivity by using more scales.

Normalisation

The results shown are for normalised features. Each element of the feature was normalised to have a median value of one across the collection. This gave equal weight to each element. The feature was also tested in an unnormalised configuration. The results from this showed that normalisation consistently improved performance by between 1 and 2%, this was statistically significant.

Feature	Single	Combined with HSV
Gabor-2-4	3.93	4.31
Co-occurrence homogeneity	2.85	3.03
Tamura standard all	2.57	3.43
Tamura CND	1.65	2.72
Tamura coarseness-2	0.97	2.49
HSV	1.91	–

Table 4.7: Mean average precision (%) for combined features with the TRECVID collection ($7x5$ tiling)

4.4.7 Evaluation using TRECVID 2003 video data

A range of the best performing features were run on the TRECVID 2003 data and evaluated using the published relevance judgements. The queries were run singly and then combined with a colour histogram feature, HSV, see Section 3.5 for details. The results are shown in Table 4.7. For comparison some features used for previous evaluations gave mean average precisions of: HSV 1.9%, convolution 2.2%; random retrieval would give 0.26% (Pickering and Rüger, 2003).

In this evaluation the texture features performed extremely well in comparison with previous benchmarks. Gabor gave the best results, 3.9% or 15 times better than random retrieval. Of the Tamura features the best performing was the combined standard feature. The top 3 performing texture features combined gave a m.a.p of 4.22%.

Combining each feature with the HSV feature improved average retrieval performance, over the individual feature, in all cases. The improvements were all statistically significant however, at an individual query level the benefits were both positive and negative. It is interesting that using simple combination of features gives varying degrees of improvement; being able to choose the optimum combination based on the query would be beneficial.

Experiments with tiling features again showed that features using this approach outperformed the windowed versions. Increasing the tiling improved performance in the same way as described in Section 4.4.3. The constant aspect ratio of the video keyframes meant that a tiling of 5×3 or 7×5 gave approximately square tiles. The latter gave the optimal retrieval performance and the results shown are for this tiling.

4.4.8 ImageCLEF 2004 Evaluation

The modified texture features were used for a submission to the image track of the Cross Language Evaluation Forum (CLEF). This work was presented in (Howarth et al., 2004, 2005).

This evaluation provided a valuable test for the features as the medical image collection, described in Section 3.2.3, presents a very different task. The nature of the images makes texture a key discriminator.

Feature	Mean average precision (%)
Gabor	35.3
Co-occurrence homogeneity	19.8
Tamura standard all	20.7
Tamura CND	18.4
Tamura coarseness-2	14.5

Table 4.8: Mean average precision (%) for texture features with the ImageCLEF collection

Table 4.8 shows the retrieval performance for a range of the texture features. The Gabor feature on its own outperformed all runs submitted to the evaluation with a mean average precision of 35.3%. The actual run submitted to the evaluation combined five features, both colour and texture. It achieved a map of 34.5% which was second in the evaluation, less than 1% behind the leader.

From the results we can see that the Gabor feature is a very good discriminator for this collection. A reason for this, is that a class of images within the collection (for example chest x-rays) is always posed in the same way. The result of this is that both the layout of an image, and the textures within it, will be very similar to the others in its class. Hence, the tiled Gabor feature, which discriminates both textural and structural information proves to be highly effective.

This evaluation confirmed that the results of the work testing and modifying the features is applicable over a range of image collections. The choice of parameters has produced robust features that perform well. The Gabor, co-occurrence and Tamura standard features in the configuration described consistently give the best retrieval results.

4.5 Conclusions and recommendations

This chapter describes the implementation, modification and evaluation of texture features for use in a query-by-example approach to image retrieval. The work uses three radically different feature types motivated by i) statistical, ii) psychological and iii) signal processing points of view. Each feature was evaluated and tuned on retrieval tasks using the Corel collection. Novel modifications to the features were proposed and tested at this stage.

The improved features were then re-evaluated with the TRECVID 2003 and ImageCLEF 2004 collections. This included testing the the performance of the texture features when combined with a colour feature. The ImageCLEF work included a submission to the evaluation forum where the features performed exceptionally well. This additional testing demonstrated that the work had identified and produced texture features that perform particularly well for the image retrieval task, providing robust performance across a range of data sets.

The evaluation with TRECVID 2003 data showed that the top three texture features performed better than previously used colour features. Combination with a colour feature boosted retrieval performance in all cases. The Gabor feature performed particularly well with both the largely monochrome ImageCLEF medical collection and the TRECVID news collection.

The work with each feature generated some concrete recommendations for using these feature types. The key findings are below:

Co-occurrence: With co-occurrence matrices it was found that considering vectors of length 4 pixels or less gave the best results. It appears that this scale of texture provides the best search discriminator with the real images used for this evaluation. Alternatively it may be that the co-occurrence feature itself is less effective at larger scales. In addition attempts to make the feature rotationally invariant degraded performance. It was best to generate separate matrices for each vector angle.

Tamura: The work with Tamura features led to some novel modifications. It was found that looking for large scale coarseness degraded performance. This contrasted with Tamura's original intention for this feature, of identifying the largest scale texture in the image rather than any smaller micro-textures. Limiting the size of textures considered was beneficial. In addition using a logarithmic scale to emphasis smaller textures improved the performance of the histogram feature. The performance of the standard directionality feature was boosted by using an entropy measure to find the degree of directionality in the angle histogram, instead of taking the second moments of the peaks. Two methods for combining the three individual features were tried. Overall, the simple approach of calculating the three standard features for each tile and concatenating these into a single feature outperformed encoding the features in terms of joint histograms.

Gabor: Rather unintuitively it was found that considering fewer texture scales gave higher retrieval rate. A filter bank of 2 scales by 4 orientations outperformed the commonly used 4 by 6 configuration. Applying the feature to image tiles was much better than using a windowing method to build a histogram.

Tiling: In all cases using the feature with image tiles, up to a maximum of 9×9, improved performance over either untiled or histogram features.

Throughout this study I have considered how best to cope with varying image sizes, scales, formats and orientations. This was working predominantly with images of roughly 10^5 pixels, i.e. 450 diagonal pixels. When working with larger images it is suggested to first down scale them to roughly this size before extracting the texture features from this thesis.

General principles for texture features

If this chapter only developed some good features for image retrieval then, although beneficial to practitioners, its value would be limited to those using these specific techniques. However, this work has given some insight into what texture characteristics are important for image retrieval. The summary below aims to identify some concise and general principles for constructing texture features. These are general guidelines for use when developing or adapting texture features for CBIR. They are intended as a starting point for ideas rather than being prescriptive.

Scale: In real images textures exist at relatively small scales. Larger artifacts in the image tend to be objects rather than distinctive textures. In general these smaller scales give better discrimination for image retrieval. Therefore texture features should aim to find and emphasise micro-textures rather than large scale variations.

Orientation: The orientations of repeating patterns within images can be important for image search. Making features that are rotationally invariant appears to be counter-productive (Section 4.4.4). The different angles should be kept separate within a feature.

Region: Tiling of texture features is beneficial. It is an effective way to isolate regions over which textures occur. In addition it captures some of the structure of an image.

Selectivity: For image retrieval the goal is not trying to classify exact textures but to generate a representation that reflects that an image that is a patchwork of textures. Increasing the amount of detail in a feature is beneficial, but only up to a certain point. Beyond this it can add noise and degrade performance. Using less selective discriminators may produce a more robust feature.

This chapter has described work on visual features for image retrieval. The features produced have been used very successfully for content based search systems, evaluation exercises and automatic image annotation. They have performed robustly with any image collections they have been tried with. The general guidelines developed for constructing texture features should prove useful for those developing new features or modifying existing ones.

Chapter 5

Localised similarity functions

5.1 Introduction

At the heart of any CBIR system are visual features, that have been extracted from images, and functions that are used to quantify the similarity between these features. The combination of these two components will drive the overall performance of a system. The previous chapter focused on visual features; this one moves on to the problem of what similarity function to use.

Two frequently studied research areas in CBIR are maximising retrieval performance using similarity measures and improving the efficiency and speed of search by applying indexing methods. Often the methods employed are mutually exclusive. The best performing similarity measures are usually computationally expensive and optimal indexing approaches can place many restrictions on what features and similarity functions can be used.

This chapter investigates how to localise the measurement of similarity. That is, emphasise points that are close to the query in some subspace of the full feature space. I show that this has dual benefits for CBIR, both improving retrieval performance and speeding up the search of high-dimensional features. The two approaches used are fractional dissimilarity and local similarity functions.

Fractional dissimilarity functions are a non-metric extension of commonly used L_p metrics while local similarity functions measure similarity based on a local subset of the feature space. Surprisingly these two classes of functions have a lot in common. These functions are applied to the task of image retrieval and their behaviour is investigated experimentally. The original work on fractional functions was published in Howarth and Rüger (2005a), and localised similarity in Howarth and Rüger (2005b). I also show how it is possible to identify parts of the feature space that are more *interesting* for visual search and how this can be used to further localise similarity, improve retrieval performance and speed up search.

The most significant limitations to the CBIR approach come from the high dimensionality of visual features and the effect this has on both search and indexing. This aptly named "Curse of Dimensionality" is discussed in the following Section. After this, Section 5.3 sets out some background material on the use of similarity within the context of CBIR. Section 5.4 then discusses the main idea of this chapter, localising similarity. This leads onto the specific functions used for this work. Experimental work, using the framework described in Chapter 3, was performed to support the use of local similarity. The results of this are set out in Section 5.5.

5.2 The Curse of Dimensionality – a problem for image retrieval

The high dimensionality of visual features has a major influence on the choice of similarity function for image retrieval owing to the "curse of dimensionality". The curse manifests itself in several ways. Donoho (2000) presents a interesting discussion of both the curse and blessings of dimensionality. He lists three problems that high dimensionality apparently makes intractable: accurately approximating a general high-dimensional function, integrating a high-dimensional function, and systematically searching through a high-dimensional space. It is the last of these that particularly affects image retrieval by both compromising the validity of nearest neighbour search and diminishing the effectiveness of indexing methods.

The "curse" term was coined by Bellman (1961) to describe the effects of the exponential growth of hyper-volume as a function of dimensionality. As a consequence the behaviour we understand and use in two or three dimensions breaks down as the dimensionality of a space increases. To gain some intuition we can consider the way volume changes as we look at a proportion of each dimension. Let us say that our feature space is described by a p-dimensional unit hypercube containing uniformly distributed data points. Given a query point, how much of the range of each dimension must we consider to capture a proportion of the data q? To enclose a fraction q of the unit volume the length will be $r = q^{1/p}$.

If we are trying to enclose 1% of the data, then in 10 dimensions this means we must consider 63% of the range of each dimension. If the dimensionality is 100 then this increases to 95% of each dimension, for 500 dimensions it is 99%. The graph in Figure 5.1, which shows edge length required to enclose up to 10% of the hypervolume, shows this. This behaviour is why it is often stated that all of the volume of a high-dimensional space is in the skin. As a result of the exponential growth in volume high-dimensional feature space is very sparsely populated. All of this causes significant problems for nearest neighbour search and indexing of high-dimensional vectors.

Effect on nearest neighbour

Beyer et al. (1999) carried out a theoretical analysis of the meaningfulness of a nearest neighbour in a

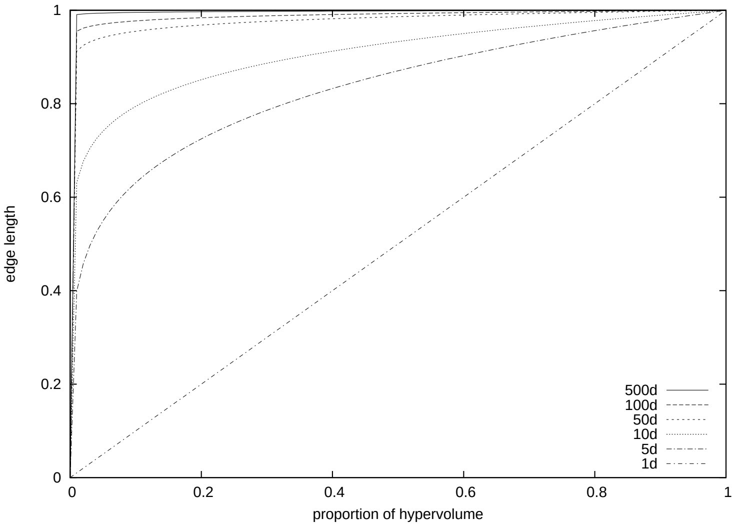


Figure 5.1: Fraction of volume enclosed versus edge length of sub-cube, for varying dimensionality

high-dimensional space. They proved that if

$$\lim_{p\to\infty} \frac{\mathrm{Var}(d_p(x,y))}{E[d_p(x,y)]} = 0 \ , \tag{5.1}$$

where $d_p(x,y)$ is the distance in p dimensional space between two random points drawn from the distribution, then for every $\varepsilon > 0$

$$\lim_{p\to\infty} P(D\max_p \leq (1+\varepsilon)D\min_p) = 1 \ , \tag{5.2}$$

where $D\max_p$ is the maximum dissimilarity between 2 points in a p dimensional distribution, and $D\min_p$ is the corresponding minimum dissimilarity. The implication of this is that when Equation 5.1 holds, then the distance of the nearest and farthest point from any query point converge to the same value. Effectively the contrast between data points becomes insignificant and as a consequence the results of a nearest neighbour search will lose their meaningfulness.

It is interesting to consider what conditions that are required for Equation 5.1 to hold and subsequently, if they do, then at what dimensionality the nearest neighbour starts to lose its meaningfulness. Beyer et al. (1999) demonstrated that this occurs with most reasonable data distributions and distance functions. The effects become significant very quickly for IID vectors. In addition, Equation 5.1 can hold for unique dimensions even where there is some correlation between all dimensions. Empirical evidence showed that nearest neighbour can become unstable, in the sense that the distance to the nearest and farthest points converge, with as few as twenty dimensions. If this occurs with a feature space and similarity measure used for image search then the idea of a nearest neighbour can lose meaning as a measure of relevance, as can the use of the top k nearest neighbours as a list of most relevant images.

Effect on indexing

Bellman (1961) originally described the problems with dimensionality after considering exhaustive enumeration on product spaces. If a dimension is split in half then with p dimensions there will be 2^p partitions. This means that with 100 dimensions there will be approximately 10^{30} partitions with this simple scheme. So unless there is a huge amount of data the probability of a point occurring in a partition will be very low. For a hypercube, containing uniformly distributed data, this probability equates to the volume of the partition. We can see from the graph in Figure 5.1 that the volume of a hypercube of edge length 0.5 tends to zero with increasing dimension. With a spherical range query things are worse. Even the largest possible query, which fits entirely within the cube, will have a volume that rapidly shrinks with increasing dimension. Here most of the volume will be in the corners of the hypercube, which are not enclosed by the sphere.

The consequences of this affect all high-dimensional indexing methods. Weber et al. (1998) set out a quantitative analysis of how high dimensionality affects the indexing of vector spaces. Their conclusions were that both partition and cluster based indexing methods lose effectiveness at high dimensionalities. They claim that, on average, existing indexing methods are outperformed by a simple sequential scan when the number of dimensions is greater than ten. At this point most indexing methods are well on the way to degenerating into a sequential scan.

They introduced a general cost model using the number of disk blocks that must be accessed as the cost of a query. Sequential scan is much more efficient than random access of disk blocks. They conservatively estimated the access time to be a factor of five faster, meaning that an indexing method had to access less than 20% of the disk blocks to be beneficial. In fact they showed that there is a dimensionality, for any indexing scheme, above which all blocks need to be accessed. They established a crude bound of 610 dimensions for all schemes. In practice this is much lower. Their work led them to suggest that an optimised linear scan was the best solution for high-dimensional indexing. Their method, the vector approximation file, is more fully discussed in Section 6.2.4.

Many indexing methods rely on the triangular inequality property of a distance metric to prune the number of elements that need to be searched. This leads to an alternative way of considering the effects of high dimensionality (Samet, 2006). As dimensionality grows the probability density of distances has a smaller variance and larger mean, i.e. becomes more concentrated. If we have a query point q and a indexing point r, from which we know the distance to every other point s, then it is possible to us the triangular inequality to eliminate points.

Thus, we can say any point s that satisfies $|d(q, r) - d(r, s)| > \varepsilon$ cannot be a distance of ε or less from the query. $d(q, r)$ is known, so we need to consider the distribution of $d(r, s)$. Unfortunately, if high dimensionality has taken effect, the probability density function of $d(r,s)$ will be large at distance $d(q,r)$, as all points tend to the same distance from each other. Worse than this, as the variance is low, even a small distance of ε around $d(q,r)$ will enclose a large area of the density function: $P(|d(q,r)-d(r,s)| \leq \varepsilon)$ will tend to one. Thus the probability of pruning any element will be very small.

Beating the curse

The preceding discussion points to a worrying conclusion for anyone wanting to build large scale content based image retrieval systems. It seems that both nearest neighbour search and indexing methods lose their effectiveness in high-dimensional feature spaces. Possible ways to mitigate these effects can be investigated by considering ways to slow down the convergence of Equation 5.1.

One ray of hope is that much of this analysis is based on uniformly distributed data, but real data is rarely uniformly distributed. It may be that the intrinsic dimensionality is much lower than the apparent dimensionality. Dimensionality reduction methods can be used to find these lower dimensional projections or embeddings. Of course it is still important that the nearest neighbours in these spaces are meaningful for our purposes. This is unlikely to be the case with global dimensionality reduction methods.

It is possible that the data is grouped into relatively tight clusters. This can occur when a collection contains very similar images. This is often the case for classification or image matching tasks, where there are almost identical images in the collection. With an image search task there may be clusters but these are likely to be more loosely defined and to overlap each other. In addition clusters may only be apparent in subspaces of the entire feature space.

The approach taken by this work is to try and identify similarity functions that will mitigate the effects of high dimensionality by preserving the contrast between data points at higher dimensionality. This is applied to image search and it is vital to ensure that the relevance of results is preserved. To do this we need to consider our knowledge of the characteristics of visual feature spaces. The approach used is discussed in Section 5.4. But first, in the next section, we consider commonly used distance functions and previous attempts to optimise these for image search.

5.3 Similarity and distance functions for content-based image retrieval

A similarity (or dissimilarity) function is a way of ordering the features given a specific query point. These can take many forms, but are usually described as a function that maps points of an $\mathbb{R}^n$ feature space to a one dimensional value. The result is typically seen as an indication of the degree of relevance of the collection images with respect to the query image and the feature. Usually several feature vectors derived from colour, texture, shape or other characteristics are searched concurrently. The similarities from these are somehow combined, and the results ranked to compute the top-matching images in a collection.

5.3.1 Commonly used distance functions

A large number of different distance and similarity functions have been used for computer vision. The variety of functions that have been employed for image retrieval is much less, with the Manhattan and Euclidean distances being very popular. The sections below give an overview of the main similarity function types.

Geometric

For image retrieval measures derived from the L-norms, Equation 5.9, are frequently used as a matter of course. These fall under the grouping of geometric functions. Other functions in this category include the cosine, d_c, and Mahalanobis, d_m, distances. Both of these use Euclidean geometry, the first represents the angle between two vectors while the second uses the covariance matrix of the data to both account for correlations between elements and normalise dimensions. The cosine distance is widely used in text retrieval as a measure of the distance between sparse vectors. The Mahalanobis distance is suited to classification tasks. They are defined as

$$d_c(x,y) = (x \cdot y)/(|x||y|) \quad \text{and} \tag{5.3}$$

$$d_m(x,y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)} \quad , \tag{5.4}$$

where x and y are vectors from the same distribution with covariance matrix Σ. The formulation of d_m is interesting because if the covariance matrix is replaced by a diagonal matrix of the variances, d_m becomes the normalised Euclidean distance, and if Σ is replaced by the identity matrix then d_m becomes the Euclidean distance.

The quadratic-form distance (Hafner et al., 1995) has the same formulation as the Mahalanobis distance but replaces the covariance matrix with a heuristically determined similarity matrix. Each element of the similarity matrix indicates the similarity between individual bins of the feature vectors. It attempts to capture the cross-bin similarity information within a histogram.

The histogram intersection distance (Swain and Ballard, 1991) is defined as

$$d_\cap(x,y) = 1 - \frac{\sum_i \min(x_i, y_i)}{\sum_i y_i} \quad . \tag{5.5}$$

It is a form of L_1 distance suitable for partial matches with histograms of different areas. It is equivalent to the L_1 distance if x and y are histograms of equal norm.

Information theoretic

Another major grouping of similarity functions is based on information theory. These include Kullback-Leibler divergence, or information gain; it gives an indication of the difference between two probability distributions. Traditionally described in a coding context, it can be used to measure the additional bits that are needed when using a code for a distribution that is different from the true distribution. For image retrieval a symmetric version of K-L divergence, Jeffrey divergence (Ojala et al., 1996), is normally used in preference. This has the advantage of being numerically stable and robust when comparing real distributions. It is defined as,

$$d_j(x, y) = \sum_i (x_i \log \frac{x_i}{z_i} + y_i \log \frac{y_i}{z_i}) \tag{5.6}$$

where $z_i = (x_i + y_i)/2$.

Statistical tests

Non-parametric tests such as the χ^2 statistic have been used for segmentation and retrieval tasks (Puzicha et al., 1997). The statistic is calculated between two vectors using

$$d_{\chi^2}(x, y) = \sum_i \frac{(x_i - y_i)^2}{z_i} \quad , \tag{5.7}$$

where z_i is as before. This function is not a metric. It is similar to the Euclidean distance, but each dimension is scaled by the average value of the two vectors on that dimension.

Ground distance

The Earth Movers Distance (Rubner et al., 1998) is a well known measure for comparing colour feature distributions generated from images. It is based on the amount of *work* necessary to change one distribution into another. To do this, each feature distribution is first clustered into k clusters; the feature can then be represented by a signature comprising of k tuples, each being the centroid and mass of a cluster. When comparing two signatures, one signature can be thought of as a set of holes and the other as mounds of earth. The distance is the minimum amount of work needed to move the earth into the holes. The work, in this case, is the mass moved times the distance it is moved. The minimum amount of work necessary can be computed using the transportation algorithm. An advantage of this function is that it is not as sensitive to binning and quantisation details as histogram measures.

5.3.2 Optimising similarity functions

The ideal similarity function would be one that mapped the feature space directly to the perceptual similarity space of the user. Ultimately the perception of a user is influenced by variables that may not be captured by low level features, or even be visual. Therefore with current techniques a perfect match in the perceptual space remains impossible. What the similarity function must do is consistently return a set of images that seem reasonably visually similar to the user. That is, a results set where the user can understand the visual link with the query. If this is achieved, then users will be able to learn how to use the system to achieve their search goals.

Researchers have worked on finding optimal similarity functions for image retrieval. Much of this work has been done using classification or supervised learning methods. Whilst this is useful, the classes or queries within a training set can never fully represent the variation present in a stream of real queries. For example, Vasconcelos and Lippman (2000) presented a paper giving a unified view of several common distance measures. They framed the retrieval task as a classification problem with a goal of minimising retrieval error. From this they could show that many distance measures were sub-optimal cases of the Bayesian criteria. They demonstrated that a maximum likelihood approach gave the best results. However, if we choose not to model predetermined classes then this approach is not practical.

Several broad approaches that are relevant for CBIR are discussed below. By examining what has been successful we can obtain a better understanding of how different similarity measures behave when used for image retrieval.

Evaluations

Where comparative evaluation of dissimilarity functions have been done these are usually in a framework of classification or image matching. The studies of Manjunath and Ma (1996) and Puzicha et al. (1997) applied a range of statistically based measures to small collections of largely homogeneous texture images. Rubner et al. (1998, 2000) concentrated on functions that included a cross-bin element, such as the earth movers and quadratic-form distances. Their proposition was that including information about the relationship between bins gave a measure closer to perceptual similarity. Their persuasive argument was not backed up by any serious image retrieval evaluation.

Zhang and Lu (2003) carried out an evaluation using the MPEG-7 contour shape database (1400 images) and the region shape database (3621 trademarks). They used two Fourier based features of 10 and 36 dimensions. The distance functions tested included all the geometric functions listed above plus the χ^2 statistic. Although a limited test, their results showed that the Manhattan distance and χ^2 statistic gave the best retrieval performance.

Puzicha et al. (1999) carried out one of the more thorough quantitative evaluations of dissimilarity measures. Interestingly they concluded that there was no measure with the consistently best overall performance, but that it depended on the specific task. They demonstrated how the selection of a similarity function based on a large scale evaluation could significantly improve the quality of results for a range of tasks, including retrieval.

Relevance feedback

Relevance feedback involves the user in the search process by getting them to indicate their preferred images from the initial results. The similarity function can then be modified to improve the ranking of the positive images. There are also approaches that modify the query itself. This is a very rich area of research. Essentially these methods are looking for ways to best combine multiple images, either at the query stage or by fusing results from individual images. Generally they use standard distance measures and combine these using a weighting scheme rather than modifying the distance function itself.

It is worth noting that although these methods can be successful they have a significant practical drawback. They rely on users marking a reasonable number of positive images. In practice users may not have the patience to do this. Systems that require negative images are even less likely to get a considered response from the user.

One method that works on the distance function itself was proposed by Ishikawa et al. (1998). They optimised a generalised version of the Euclidean distance, similar to the Mahalanobis distance. Their method iteratively optimises the query point and determines the similarity matrix. This is done by minimising the sum of the distances between the query image and the images marked relevant by the user. The similarity matrix is calculated from the covariance matrix C of the relevant images as $(\det(C))^{1/n}C^{-1}$. Unfortunately the performance for this method has been found to be rather unstable when used with sparse visual features.

Noise model

Sebe et al. (2000) considered what distance metric was justified based on a noise model. The image retrieval task they considered was one of finding images that were different due to handling and storage conditions of the original photographs. Thus, the noise was created by relatively small perturbations. This task is far removed from the image retrieval task tackled by this thesis, however, their work does give some insight into the behaviour of distance measures. They adopted a maximum likelihood approach. Given a discrete noise distribution n, the metric which maximises the similarity probability is

$$\sum_i \rho(n_i) \ ,$$

where ρ is the maximum likelihood estimate of the negative logarithm of the probability density of the noise. To investigate the behaviour of the estimate they used a method based on the influence function

(Hampel et al., 1986). This uses the derivative

$$\psi(z) = \frac{d\rho(z)}{dz}$$

of the estimate, to characterise the bias that a measurement has on the solution.

Where the noise is Gaussian, $P(n_i) \sim e^{-n_i^2}$, then $\rho(z) = z^2$ and $\psi(z) = z$. Thus the optimal metric is Euclidean. The function ψ shows that points more affected by noise get greater weighting.

With exponential noise $P(n_i) \sim e^{-|n_i|}$, then $\rho(z) = |z|$ and $\psi(z) = \text{sign}(z)$, and the best metric is Manhattan. Points get the same weighting irrespective of their noisiness (deviation from the mean).

By examining real noise distributions they determined that the tails were more prominent than with the exponential distribution. Therefore the image matching task should be better modelled using a fat-tailed distribution. They found that the Cauchy distribution closely matched the empirical distributions. This has the form

$$P(n_i) \sim \frac{a}{(a^2 + n_i^2)} \ ,$$

where a is the scale parameter, and gives

$$\rho(z) = \log(1 + (z/a)^2) \text{ and } \psi(z) = \frac{z}{a^2 + z^2} \ .$$

It is interesting that ψ initially increases before decreasing rapidly, thus outliers are largely ignored. From this they developed the *Cauchy metric*

$$d_c(x, y) = \sum_i \log\left(1 + \left(\frac{x_i - y_i}{a}\right)^2\right) \ .$$

It was demonstrated experimentally that the Cauchy metric achieved the best performance for their image matching task. Subsequently it was shown the Cauchy metric also performed well with a shape matching task (Sebe and Lew, 2002). The maximum likelihood approach was taken further by Yu et al. (2006), who developed a boosted distance metric for supervised classification.

Parametric model

Tarel and Boughorbel (2002) proposed a model of image retrieval and developed a scheme for deriving the best similarity measure from a set of measures of the form,

$$S_{(\alpha,\beta)}(x, y) = \sum_i |x_i^\alpha - y_i^\alpha|^\beta.$$

It is interesting to note that with $\alpha = 1$, the β parameter is equivalent to the p for the L_p-norms. They first showed empirically that an α value of 0.3 to 0.4 consistently improved average precision for an image retrieval task.

They assumed a parametric model of the variability of features within the same class and then proceeded to model the image retrieval task as a two class problem. R-precision was used as a performance measure and the goal was to minimise the retrieval error based on this. Their model assumed that each component of the feature came from the same distribution but was independently chosen. Using the central limit theorem the distribution for the total distance then converges towards a Gaussian as the number of components goes to infinity. This simplifies the analysis, as with sufficiently many dimensions the distributions can be approximately described by their mean and variance.

With $\alpha = 1$ they found optimal β values for several component distributions. Assuming that each component was from a Gaussian they showed that L_2 was the best distance function. With an exponential distribution the optimal function was L_1. The components of sparse histograms are often approximately exponentially distributed. Their results therefore concur with what has been found in practice, that L_1 generally outperforms L_2. The analysis could not be extended to values of $\beta < 1$.

Their analysis of optimal α values was more limited. A simulation for the exponential case showed that an α value around 0.4 gave the best precision.

Partial distance

Li et al. (2002) introduced a dissimilarity measure based on the hypothesis that each image in a collection may be relevant to a query in a different set of dimensions of the feature space. They called this a dynamic partial distance function. Given p-dimensional feature vectors x and y, they define

$$\delta_i = |x_i - y_i| \text{ for } i = 1, \dots p$$

and Δ_m as the set containing the m smallest δ_i values. The dynamic partial distance is then defined as

$$d_\delta(x, y) = \left(\sum_{\delta_i \in \Delta_m} \delta_i^r \right)^{1/r} \tag{5.8}$$

with parameters, m and r. By setting a value of $m < p$ the function will choose the m dimensions in which that particular image is closest to the query. The rationale is that the discarded dimensions will contain little similarity information. This method does not consider the distribution of the other images in the collection.

They applied this function to a retrieval task based on finding transformed versions of the same image, using a feature with 144 elements. They found experimentally that the best value of m was 114 and used $r = 3$. In this configuration the function performed significantly better than the Minkowski metric, i.e. with $m = p$. This evaluation was a little unfair on the Minkowski metric as it would almost certainly have given better results with a value of $r = 1$, as shown by Sebe et al. (2000) and my results in Section 5.5.2.

Summary

Several useful themes can be drawn from this review of distance measures. Both the noise and parametric models have shown the same optimal distance metrics in two cases. If the noise or distribution of a class in every dimension is Gaussian then the optimal distance is Euclidean; if the noise is exponential then the best distance measure is Manhattan. Sebe et al. (2000) went on to show that for the fat-tailed Cauchy distribution they could define an optimal metric. This implicitly gave less weighting to noisy points (outliers).

All of this analysis was done in the complete feature space. The underlying assumption was that similar images were clustered in all dimensions. The partial distance function took the different view that an image is relevant to a query in a subset of dimensions. With this model the assumption is that images are clustered in subspaces of the complete feature space.

5.4 Localising similarity

The ideal similarity function for content-based image retrieval would produce meaningful results in a realistic timescale. It would do this with different feature types and large scale collections. The problems caused by high dimensionality seem to make this an impossible task.

As discussed in Chapter 3, for a real image retrieval task the meaningfulness of results can only truly be determined by the user who issued the query. In the query-by-example paradigm, the user has one or more query images and wants to find other similar images. The relevance of images is based on the judgement of the user. This problem is much harder to define than other tasks such as image matching or object recognition. For instance with image matching the relevant images are very similar to the query, and often they have been subject to some noise or distortion. With CBIR the visual similarity is usually less obvious and the differences much greater. This makes it difficult to model the distribution of relevant images in the feature space. Indeed the relevant images for some user queries may have limited visual similarity. In these cases CBIR tends towards a browsing paradigm rather than one of query–results. The user then becomes an integral part of the system. It is important that the CBIR task is understood to put this work in context. The image retrieval task is very different from those normally associated with computer vision and image understanding. These differences have a significant effect on the effectiveness of different approaches and and how performance is measured.

The second, and often overlooked, performance measure is search time. This is made up of two main parts, retrieving the relevant data and computing the similarity. The similarity function can have a direct impact on both of these. The amount of data that has to be considered in the search is usually the dominant factor on the time taken. For a system where the data is stored on disk, loading the data will be the limiting factor. Similarity functions may lend themselves to efficient indexing or pruning of the search space. Where the feature database is in memory then the computation time is likely to become significant. The issue of efficiency is explored more fully in Chapter 6. The experiments later in this chapter use the proportion of data examined as a crude measure of efficiency.

This section discusses what factors influence the choice of similarity functions for CBIR including the notion of localising similarity. Fractional dissimilarity and local similarity functions are defined and related work highlighted. Finally query sensitive selection of feature components is considered. This is based on finding atypical points in the entire feature space.

5.4.1 Why localise?

The notion of similarity or distance is central to CBIR, and a large number of dissimilarity functions have been used. It is accepted that the choice of proximity function can have a profound effect on local topology. This is significant for CBIR as when querying a multimedia database we are normally interested in a small set of the closest images, rather than the ordering of images a long way from the query.

The choice of dissimilarity function is often made without much thought. The Euclidean distance metric has its basis in 2 and 3 dimensional space and in this context it is the physical distance function in a straight line. For higher dimensions it loses its significance, although it is frequently used without consideration of its applicability.

If we look at the visual features of images relevant to a query, we find that in a many dimensions the value of the feature is very different. The relevant images are close to the query in only a small number of dimensions. Hence, the information about similarity between the images appears to be concentrated in relatively few dimensions. These close dimensions are likely to vary from image to image even amongst the set of relevant images (Li et al., 2002).

This situation is very different to the image matching situation, where the similar images have been subjected to noise. In this case relevant images are likely to form clusters in the full feature space. This means that a similarity measure that considers all dimensions is likely to perform well. For a search task the images may well be very different. This means that the noise model used by Sebe and Lew (2002) is not valid for the CBIR task. In the absence of a particular noise model for specific visual features, it is noted that amongst all L_p norms, L_1 is the one which is most stable, i.e. least numerically influenced, against outliers and noise. Indeed, it was found in previous experiments that L_1 (Pickering and Rüger, 2003), which is equivalent to the histogram intersection for L_1 normalised vectors, consistently outperformed L_2, L_∞ and the cosine similarity function, which is nearly equivalent to the

Euclidean distance for small distances. One qualitative explanation for better performance of L_1 over L_2 is that it is less affected by outliers and therefore by noise in high dimensional data: in Euclidean space the distant components will dominate the distance measure; using L_1 gives near and far components the same weighting.

Developing this idea further leads to the goal of adding importance to components that are similar and removes emphasis from those that are different. I therefore propose that the important regions for a visual feature are those local to the query point in each dimension. In contrast, if an image is not local in a particular dimension we are not interested in the absolute difference, only that it is not a member of the local set. This is the essence of localising similarity.

This intuitively makes sense as the human visual system can detect small differences in neighbouring patches equally as well as large differences. Consider an image that has a colour distribution that is predominately green. The shade of green would be relevant for ranking similar images. However, the fact that other images are, for example, mainly red or blue should not affect their relative ranking: they are just different.

Given that there are sets of dimensions where relevant images are local in high-dimensional feature space, these will be manifested as small clusters in certain subspaces. The set of images relevant to a query may occupy more than one of these subspaces, with different images within the relevant set often being in a different group of subspaces. The technique of localising similarity will emphasise membership of these potentially interesting clusters. The result of this is that, where these clusters exist and are relevant, retrieval performance should improve. Ideally we would like to be able to detect these subspaces and focus the search within them.

5.4.2 Fractional dissimilarity

The first approach to localisation is that of fractional dissimilarity functions. Mathematically, all distances induced by the Minkowski norm L_p, $p \geq 1$, defined below in Eq. (5.9) are equivalent in finite-dimensional spaces in the sense that they result in the same topology for neighbourhoods: they define continuity of functions in the same way. However, as discussed earlier the performance of different L_p metrics varies greatly for retrieval tasks.

The L_p norm is usually induced by the distance

$$\mathrm{diss}_d^p(x,y) = \left[\sum_{i=1}^{d} |x_i - y_i|^p\right]^{1/p} = \| \, x - y \, \|_p \quad , \tag{5.9}$$

where d is the dimensionality of the space and p is a free parameter, $p \geq 1$. This can be extended by considering the case where $0 < p < 1$. From a mathematical perspective there is nothing new about this; however, its use in computer vision has been limited. Donahue et al. (1996) first used this function for robust shape matching. My work in Howarth and Rüger (2005a) was its first direct application to a CBIR task.

Functions of this type are often referred to as a quasi-norm or semi-metric. Strictly speaking the fractional functions defined by diss^p with $0 < p < 1$ are no longer distances in the mathematical sense as the triangle inequality is violated. The reason for this is that the ball with radius one under diss^p is no longer convex for $p < 1$, see Figure 5.3(a). This can have an effect on some indexing and partitioning schemes that rely on the metric properties. Nevertheless diss^p still conveys a sense of closeness, and we will refer to it as a fractional dissimilarity.

Violation of metric properties, including the triangular inequality, will not necessarily adversely affect the performance of image retrieval. Tversky (1977) argued that human ideas of similarity do not behave in the same way as metric geometry. If required it is possible to define a metric for spaces with $p < 1$ by applying a concave function. Hence, $d(x, y) = (\| x - y \|_p)^p$ is a metric defined on the vector space L_p. A proof of this is given by Jones (2001).

As a simple example of this, consider a two dimensional case with the points $a = (0, 0)$, $b = (1, 0)$ and $c = (1, 1)$. Now, for a fractional dissimilarity with $p = 1/2$, $d(a, b) = d(b, c) = 1$ and $d(a, c) = (1+1)^2 = 4$. Thus $d(a, c) \not\leq d(a, b) + d(b, c)$ so the triangular inequality is not satisfied. If the concave function is applied then $d(a, b)$ and $d(b, c)$ are unchanged, but $d(a, c) = ((1+1)^2)^{1/2} = 2$. So $d(a, c) \leq d(a, b)+d(b, c)$ and the triangular inequality holds. Of course changing the function in this way will have an effect on dissimilarity values and correspondingly retrieval performance may change from that of the non-metric distance.

Related work

Aggarwal et al. (2001) found several results applicable to both ordinary ($p \geq 1$) and fractional ($0 < p < 1$) dissimilarities. The first was that the absolute difference between the maximum and minimum dissimilarities increases at the rate of $d^{\frac{1}{p}-\frac{1}{2}}$. Secondly, they found bounds on the relative contrast which showed that fractional functions should usually have a better relative contrast than L_p norm distances. They carried out some experiments and showed that k-means clustering and a rudimentary classification technique performed better with fractional functions. This was using synthetic and real (non-image) data. It is interesting that they found a value of $p = 0.1$ to routinely give the best results. This should be contrasted with the results in Section 5.5.2.

Francois et al. (2005) investigated the effectiveness of fractional functions for image matching in the presence of non-Gaussian noise and concluded that fractional functions were beneficial. Recently Skopal (2006) devised an algorithm that finds a function that will enforce the triangle inequality for a general semi-metric, thus turning it into a metric. The functions are either of the form mentioned earlier, where the concavity is controlled globally, or a more flexible rational Bezier Quadratic curve. The compound function was then used to index the data with metric access methods, such as the M-tree index. He found that it was not possible to use this for exact search with small values of p, below 0.25. However, if the requirement was only for approximate preservation of similarity ordering then all the fractional functions tested could be indexed effectively using metric access methods.

5.4.3 Local similarity

The premise for the improved performance of fractional distance functions is that, when comparing two feature vectors, they emphasise dimensions that are close whilst reducing the noise from distant dimensions. This results in more meaningful nearest neighbour search when using high-dimensional visual features as was shown in (Howarth and Rüger, 2005a) and by the experimental results in Section 5.5. This idea can be naturally extended to similarity functions that consider only the locality of the query point in each dimension. They are effectively giving a weight of zero to those distant points and selecting a sample of the least noisy information for the measure.

We can define the following similarity function between two vectors x and y, both belonging to the set $X \subset \mathbb{R}^n$,

$$\mathrm{Sim}(x, y) = g\left(\sum_{i \in K(X,x,y,k)} \frac{s(x_i, y_i)}{z}\right), \tag{5.10}$$

where the $K(X, x, y, k)$ is the set of dimensions where y is local to the query point x, $s(x_i, y_i)$ is a local similarity function on $\mathrm{R} \times \mathrm{R}$ and z is a normalising factor. The function g acts on the result of the summation and may be varied depending on the form of $s(x, y)$. Defining the function in this way highlights the flexibility of this approach. There are many options available to vary both the local similarity function and the selection of the local neighbourhood.

The aim of this similarity function is twofold: to give an effective similarity measure for high-dimensional features and to only examine a small percentage of the data when doing so. $\mathrm{Sim}(x, y)$ is not aiming to be an approximation to another similarity measure, but a meaningful measure in its own right.

The local similarity function $s(x_i, y_i)$ will affect the local topology around the query point and therefore the search results. In addition, the complexity of the function will have an impact on the computational time. There will be a trade-off between these two factors.

Some possible choices of the local similarity function are similarity versions of the L_p-norms or fractional functions, a ranking from the query point or a voting function. A diagram showing some of these functions is in Figure 5.2. The voting function allocates a value of one to all objects within the local set and zero to others. The obvious advantage of this over other measures is simplicity. It is the outermost function on the figure. The ranking function is not shown, the distance for this is based purely on the ranking of objects from the query point. With the voting and ranking functions, g will be the

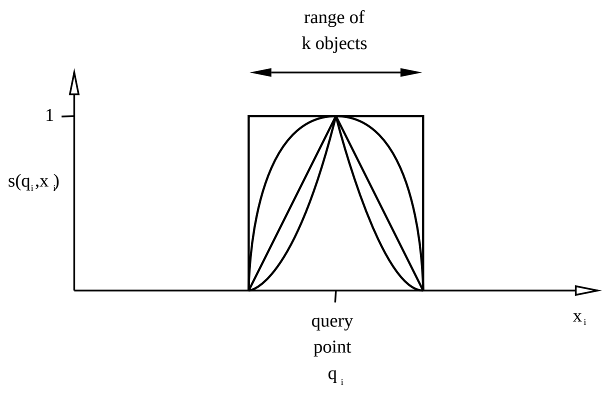


Figure 5.2: Similarity functions

identity function. For L_p functions it will be $g(x) = x^{1/p}$. There is no need for this form of g to be strictly adhered to: it can become another variation in the range of available functions.

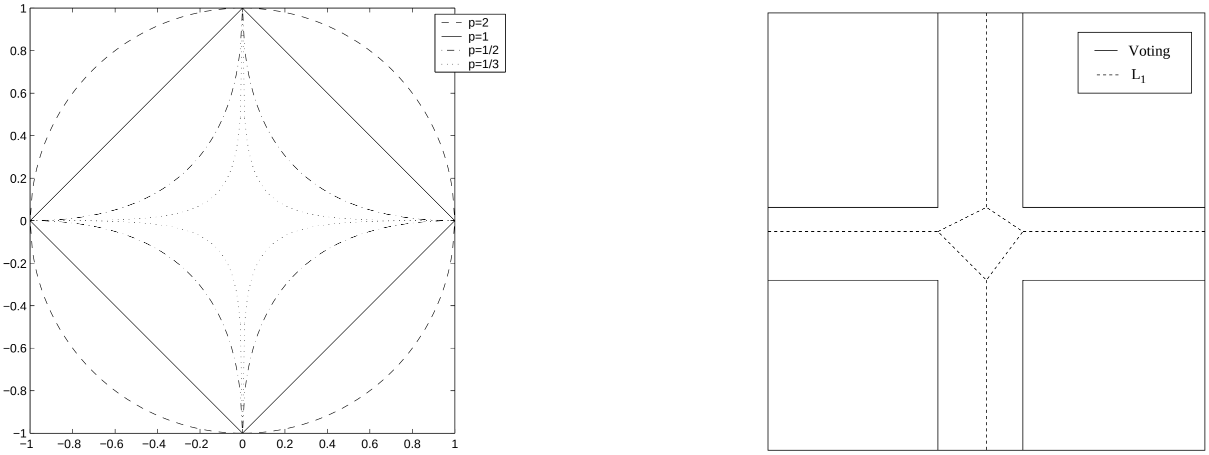
The second parameter in Eq. (5.10) is the function $K(X, x, y, k)$, which determines the set of dimensions where y is local to x. It is used to isolate the local set of objects for each dimension. This can be a selection of the nearest k points, or all the points within a certain distance. The first option was chosen for the experimental work as it enables the selection of a fixed percentage of the data. It also introduces another normalisation into the overall function by ensuring that each dimension carries equal weighting.

The choice of the local set will have a major influence on the performance of the function. Consider the voting function described above in a d dimensional space, and the set of k local points. Each dimension will have k votes and there will be a total of kd votes cast across all dimensions. An object receives a vote from a dimension if it is in the set of k nearest objects to the query point in that dimension. If v is the total number of votes an object receives across all dimension it will be the upper bound on the similarity for that object whatever local function is used.

The distribution of v across the data set will affect the discriminatory power of the similarity measure. Consider a data set with N objects. With $k = 1$ the only objects getting a vote will have to match the query exactly in that dimension. It is unlikely that many images will get more than one vote so that, other than exact matches, the results will be close to random. As k increases the number of objects with more votes, higher values of v, will increase. When $k = N$, all objects will be found in every dimension so every object will get d votes. Hence, if using the voting distance function, the discriminatory power will increase with k to a maximum, and then decrease to random at $k = N$.

Other similarity functions will have added discriminatory power. However, it is likely their overall performance will be strongly correlated with the performance of the voting function as the size of the local neighbourhood is varied.

It is interesting to consider the effect of the dimensionality, d, itself. Each object will get between 0



(a) Fractional functions

(b) Local functions

Figure 5.3: Unit balls in two-dimensional space

and d votes. As the dimensionality increases the possible variation in number of votes will increase; thus the contrast between nearest and farthest neighbours will increase. This contrast can be an indicator of discriminatory power. In this way the method of local similarity can be said to become more effective as the dimensionality increases. However, we must be careful with this claim as there is not necessarily a direct link between contrast and meaningfulness of results.

Figure 5.3(b) shows the unit balls for the voting and L_1 local functions with an arbitrary local neighbourhood. The lines on the figure indicate the points in the two-dimensional space that are a distance of one unit from the centre. For the similarity function, points inside the ball will have similarity greater than or equal to one, and correspondingly points outside the ball will have similarity less than one. The neighbourhood is not symmetrical because its absolute size depends on the distribution of the k points around the query point. From the unit balls it is easy to see the close relationship with fractional dissimilarity functions. The balls resemble a truncated version of those for fractional functions. The important parameter is the size of the local neighbourhood that is isolated by the function $K()$, or for fractional functions the value of p. From this we can hypothesise that the retrieval performance of local similarity functions is likely to share characteristics with that of fractional measures.

Related work

Previous work using a form of localised distance includes the iGrid index of Aggarwal and Yu (2000) and the bitmap indexing method of Cha (2003). Both of these used a fixed number of equally populated partitions to define the localities. Interestingly Aggarwal and Yu (2000) use a bin size inversely proportional to the number of dimensions. For a 500 dimensional feature this means that a locality of 0.2% of each dimension will be selected. This should be compared with the results in Section 5.5, which demonstrate that going below 5% of the data, for a comparable size of collection, can result in a large drop in the meaningfulness of results. They experimented with a classification task using non-image data. It is likely that the difference in performance is down to the characteristics of the feature space. The image search task, using visual features, appears to behave very differently to other data mining applications.

5.4.4 Metric properties of local similarity

It is interesting to note that if the local function is a metric, and subject to a simple boundary constraint, then the local similarity function can be formulated as a distance metric. This is a noteworthy result as it makes this local distance function compatible with the many algorithms and indexing methods that require metric properties.

To prove this by induction, we first to show that the local similarity function can be made a metric in the one-dimensional case, and that the required product metric can extend this to the n-dimensional case. The standard definition of a metric is given below.

Definition 5.1. *Let X be a set and d be a function $d\colon X \times X \to \mathbb{R}$, then d is a metric on X if it satisfies the following three conditions for all $a, b, c \in X$:*

(M1) $d(a, b) \geq 0$ *with* $d(a, b) = 0$ *if and only if* $a = b$

(M2) $d(a, b) = d(b, a)$

(M3) $d(a, c) \leq d(a, b) + d(b, c)$

As part of the proof we need to show that we can construct a suitable metric on the product set $X = X_1 \times X_2$ and, hence, the n-fold product set. We consider the case when $g()$ in Equation 5.10 is the identity function. This is a standard proof, shown below.

Lemma 5.2. *Let $(X_1, d_1), (X_2, d_2), \ldots, (X_n, d_n)$ be metric spaces and let the product space $X = X_1 \times X_2 \times \cdots \times X_n$. If we define a function $e : X \times X \times \cdots \times X \to \mathbb{R}$, where*

$$e(x, y) = d_1(x_1, y_1) + d_2(x_2, y_2) + \cdots + d_n(x_n, y_n)$$

then (X, e) is a metric space.

Proof. We first consider the case when $n = 2$ and show that (X, e) satisfies the conditions (M1)-(M3).

(M1) For all $x, y \in X$, $d_1(x_1, y_1), d_2(x_2, y_2) \geq 0$ since they are both metrics. Hence $e(x, y) \geq 0$.

For all $x \in X$, $e(x, x) = d_1(x_1, x_1) + d_2(x_2, x_2) = 0 + 0 = 0$. Conversely, if $e(x, y) = 0$ then $d_1(x_1, y_1) + d_2(x_2, y_2) = 0$. Since they are both metrics $d_1, d_2 \geq 0$, so they must both be equal to zero. Therefore, $x_1 = y_1$ and $x_2 = y_2$, so $x = y$.

Thus e satisfies (M1).

(M2) Again let $x, y \in X$. d_1 and d_2 satisfy (M2). Therefore.

$$\begin{aligned} e(x,y) &= d_1(x_1,y_1) + d_2(x_2,y_2) \\ &= d_1(y_1,x_1) + d_2(y_2,x_2) \quad = e(y,x) \end{aligned}$$

Hence, e satisfies (M2).

(M3) Let $x, y, z \in X$. Then using (M3) for d_1 and d_2,

$$\begin{aligned} e(x,z) &= d_1(x_1,z_1) + d_2(x_2,z_2) \\ &\leq (d_1(x_1,y_1) + d_2(y_1,z_1)) + (d_1(x_2,y_2) + d_2(y_2,z_2)) \\ &= (d_1(x_1,y_1) + d_2(x_2,y_2)) + (d_1(y_1,z_1) + d_2(y_2,z_2)) \\ &= e(x,y) + e(y,z) \end{aligned}$$

Thus e satisfies (M3). Therefore e is a metric on X.

Now consider an n-fold product $X = X_1 \times X_2 \times \cdots \times X_n$. Initially take the product of the first two spaces, which is a metric space, and then take the product of this with the third getting a new metric space. This process can be repeated $n-1$ times showing that (X, e) is a metric space.

□

We now redefine the local similarity function, Equation 5.10, as a distance function in one dimension.

Definition 5.3. *Let $d_L \colon X \times X \to [0,1]$, where $X \subseteq \mathbb{R}$. Let $x, y \in X$. A local distance metric is now defined on the set of points local to x, $d_l \colon K \to [0,1]$, where $K = K(X, x, k) \subseteq X$. The constant k determines the number of points that are in K. The members of K are the $k/2$ points closest to x on either side. If there are fewer than $k/2$ points on a particular side of x, then all the points on that side are in K and the number of points on the opposite side of x is not altered. The distance function, d_L, can now be defined as*

$$d_L(x,y) = \begin{cases} d_l(x,y) & \text{if } y \in K \\ 1 & \text{otherwise.} \end{cases} \tag{5.11}$$

The following boundary constraint also applies to d_l. Let $a, b, c \in X$, $b, c \in K_a$ and c be at the boundary of K_a, then the local metric d_l must satisfy

$$d_l(a,b) + d_l(b,c) \geq 1 \quad \text{and} \quad d_l(a,b) \leq 1 \tag{5.12}$$

We are now in a position to state and prove our theorem.

Theorem 5.4. *Let $X \subseteq \mathbb{R}$. The function d_L is a metric on X.*

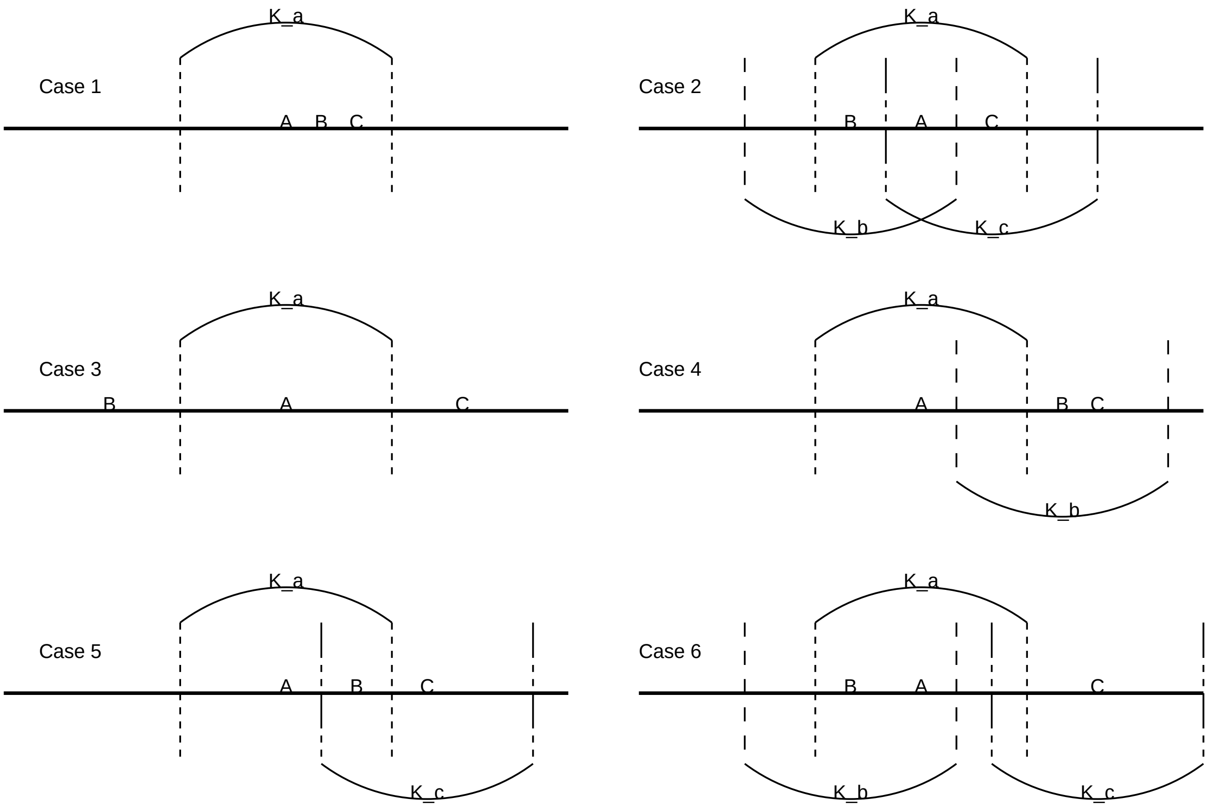


Figure 5.4: Cases for proof of M3

Proof. To prove that d_L is a metric we need to show that it satisfies the conditions (M1)-(M3).

(M1) Let $x, y \in X$. If $y \in K$ then $d_L(x, y) = d_l(x, y) \geq 0$ since d_l is a metric. If $y \notin K$, then $d_L(x, y) = 1$. Therefore $d_L \geq 0$.

For all $x \in X$, $x \in K_x$ and $d_L(x, x) = d_l(x, x) = 0$ since d_l is a metric. Conversely, if $d_L(x, y) = 0$, then $d_l(x, y) = 0$. Since d_l is a metric, $x = y$.

Thus d_L satisfies (M1).

(M2) Let $x, y \in X$ and the local areas to x and y be $K_x = K(X, x, k)$ and $K_y = (X, y, k)$. From the definition of K, $y \in K_x \Leftrightarrow x \in K_y$ and $y \notin K_x \Leftrightarrow x \notin K_y$.

There are two cases to consider. If $x \in K_y$ and $y \in K_x$ then since d_l is a metric $d_l(x, y) = d_l(y, x)$ and $d_L(x, y) = d_L(y, x)$. If $x \notin K_y$ and $y \notin K_x$ then $d_L(x, y) = 1 = d_L(y, x)$. Hence $d_L(x, y) = d_L(y, x)$ and d_L satisfies (M2).

(M3) Let $a, b, c \in X$. There are 6 cases to consider, these are shown diagrammatically in Figure 5.4.

Case 1: $b, c \in K_a$, $b \in K_c$ From the definition of K, $a, c \in K_b$ and $a, b \in K_c$. Using (M3) for d_l,

$$d_L(a, c) = d_l(a, c) \leq d_l(a, b) + d_l(b, c) = d_L(a, b) + d_L(b, c)$$

Case 2: $b, c \in K_a$, $b \notin K_c$ Using (M3) for d_l,

$$d_L(a,c) = d_l(a,c) \leq d_l(a,b) + 1 = d_L(a,b) + d_L(b,c)$$

Case 3: $b, c \notin K_a$, $b \notin K_c$

$$d_L(a,c) = 1 \leq 1 + 1 = d_L(a,b) + d_L(b,c)$$

Case 4: $b, c \notin K_a$, $b \in K_c$.

$$d_L(a,c) = 1 \leq 1 + d_l(b,c) \leq d_L(a,b) + d_L(b,c)$$

Case 5: $b \in K_a$, $c \notin K_a$ and $c \in K_b$. This uses the constraint from Equation 5.12

$$d_L(a,c) = 1 \leq d_l(a,b) + d_l(b,c) \leq d_L(a,b) + d_L(b,c)$$

Case 6: $b \in K_a$, $c \notin K_a$ and $b \notin K_c$.

$$d_L(a,c) = 1 \leq d_l(a,b) + 1 \leq d_L(a,b) + d_L(b,c)$$

Thus d_L satisfies (M3).

Therefore d_L is a metric on X. □

Theorem 5.5. *Let the product space $X = X_1 \times X_2 \times \cdots \times X_n$ and $x, y \in X$. We now define the function $d_L \colon X \times X \to [0,1]$ as*

$$d_L(x,y) = d_L(x_1,y_1) + d_L(x_2,y_2) + \cdots + d_L(x_n,y_n) \ .$$

d_L is a metric on X.

Proof. From Theorem 5.4, $(X_1, d_L), (X_2, d_L), \ldots, (X_n, d_L)$ are metric spaces. Using Lemma 5.2 the n-fold product space of metric spaces is itself metric. Hence d_L is a metric on X. □

Many of the obvious choices of local distance function, such as the absolute difference (Manhattan distance), are metric. A ranking function can be made metric as it is a map from the absolute distance. The voting function would appear not to be a metric as it violates the condition of (M1) that $d_l(x,y) = 0 \iff x = y$. However, with a small modification the local function can be changed into the discrete metric. This will make the compound distance function into a metric. The modified function is defined as,

$$d_l(x, y) = \begin{cases} 0 & \text{if x=y} \\ 1/2 & \text{otherwise.} \end{cases}$$

The value of 1/2 is required so that the constraint in Equation 5.12 is satisfied.

The ability to turn the function into a metric can be useful because a lot of indexing and clustering algorithms rely on this property. These include the many variants of metric access methods, described in Section 6.2.3, that can be used to index feature spaces.

5.4.5 Interesting parts of a feature

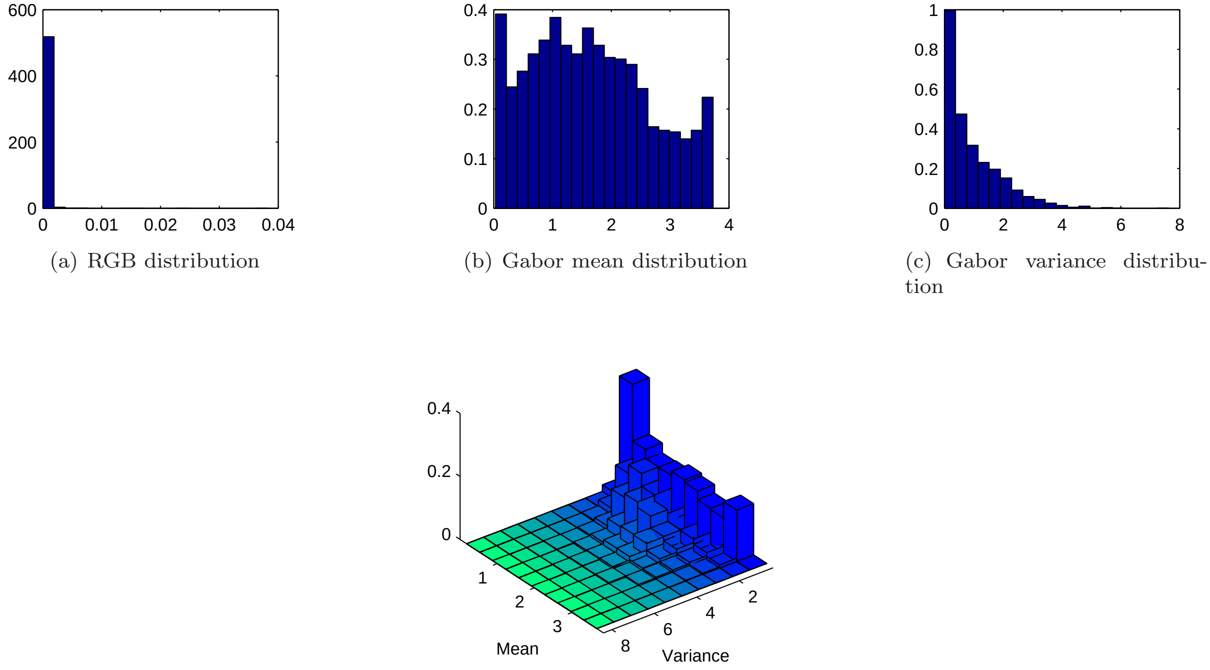
Local similarity functions allow us to store and manipulate each dimension independently. In effect we treat each dimension as a different feature. Taking this viewpoint we can select which dimensions are more important relative to a given query. Keeping this notion of independence of dimensions means we have to consider each separately. The only information available is the distribution of points and the position of the query point within the distribution.

Histogram style features occur frequently in CBIR. In fact nearly any feature can be converted into a histogram. An interesting characteristic of these is that within each feature vector a large number of bins have zero value, that is, they are not populated. Figure 5.5(a) shows a typical distribution[1] for the RGB feature. This distribution means that zero is a typical value for a histogram bin. If the query has the same value then the membership of the local neighbourhood has little meaning. It therefore makes sense to discard the dimensions where the query points are zero and only use those where the query value is atypical.

We are hypothesising that these atypical dimensions are the interesting ones for a specific query. This contrasts with some of the common approaches to finding clusters in subspaces. They generally seek high density areas and combine these to find candidate clusters. My approach takes the opposite approach by saying that the low density areas are most interesting. For example, the presence of a colour is more interesting than its absence. We can draw analogies to text retrieval where unusual words in a corpus have greater discriminatory power and are give more weight in the search.

The Gabor and convolution features are not histograms (see Chapter 3). However, we can take a similar approach to that used for histograms and focus on the least populated parts of the distribution, the tails, as these are most likely to differentiate the image. This method can be applied directly to the convolution feature. However, with the Gabor feature there is an added complication that each filter is represented by a pair of values, the mean and variance of the filter response. Examples of these distributions are shown in Figures 5.5(b) and 5.5(c) respectively. It is clear that there is no obvious typical value, particularly in the dimension representing the mean, so discarding of dimensions in this manner is

[1] All the distributions in Figure 5.5 have an area of one under the curve, however, note that the axes are scaled differently.



(a) RGB distribution

(b) Gabor mean distribution

(c) Gabor variance distribution

(d) Gabor joint distribution

Figure 5.5: Example distributions of feature values within a dimension

not straightforward. The two values for each filter are not independent. Ideally they should be combined into a joint distribution to identify the typical and atypical values. This is shown in Figure 5.5(d). By doing this we can use the variance dimension to select if a pair of dimensions (mean and variance) should be discarded. Although treating the joint distribution in this way may not optimal, it does provides a reasonable heuristic approach for discarding data that carries little information.

This discussion highlights how closely the design of the similarity function and resultant index should be linked to the characteristic of the feature. Also it indicates a potential improvement to our method as we could use a two-dimensional index, such as the R-tree, for these pairs of dimensions.

It is interesting that a related approach was used by Heller and Ghahramani (2006) to binarize feature vectors. They considered the skewness of each dimension. If a dimension was positively skewed then the images with a value above the 80th percentile were assigned 1 and the remainder assigned 0. For a negatively skewed distribution images with a value below the 20th percentile were given a 1. Their method finds its atypical values in the tail of the distribution, with the side of the distribution chosen depending on the skewness.

5.5 Experiments and results

This section describes experimental work investigating the performance of fractional dissimilarity and local similarity functions. With the latter the effect of selecting atypical dimensions is also evaluated.

5.5.1 Experimental setup

The experimental framework used was described in Chapter 3. Four collections were employed, Corel, TRECVID 2003, Getty and ImageCLEF 2004. Five features were used covering the major types. These were two joint colour histograms (RGB and HSV), the colour structure descriptor (HDS), the Gabor texture feature and the convolution feature. Average precision was used as a performance measure. This experimental setup covers a wide variety of the CBIR tasks. If findings are consistent across these collections and features, then this gives reasonable confidence that they are generally applicable to image retrieval.

5.5.2 Performance of fractional dissimilarities

The aim of this practical work with fractional dissimilarities was to ascertain if they improve the performance of content-based image retrieval systems. The experiments were designed to address the following questions:

- Do fractional dissimilarities increase retrieval performance for high dimensional visual features?
- How does the performance vary with the fractional parameter p?
- If there is an optimal p for a specific feature is this stable across different image data sets?
- Is it possible to predict the optimal setting for p from any characteristics of the feature or the resultant dissimilarity distribution?
- Should the features be normalised for the dissimilarity function?

Corel

The first set of experiments, with the Corel collection, were aimed at determining if fractional dissimilarity functions gave a significant retrieval performance gain across the range of visual features. The query set of multiple image queries was used. The results are plotted in Figure 5.6, which shows mean average precision (m.a.p.) retrieval against p.

The first thing to note from this graph is that all the features show an increase in m.a.p. for fractional dissimilarities ($0 < p < 1$). The most significant increases are for the RGB, HSV, HDS and convolution features. The Gabor feature is relatively flat across the graph, showing only a slight improvement in

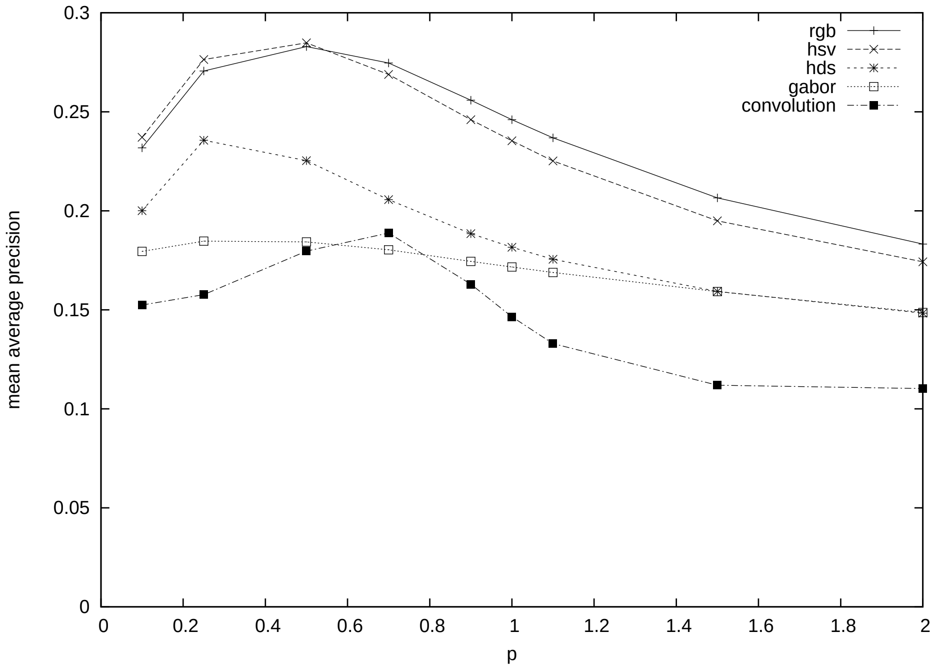


Figure 5.6: Retrieval performance using fractional dissimilarity, Corel collection

retrieval performance for fractional dissimilarities. The position of the maxima vary from feature to feature but all fall between p values of 0.25 and 0.75.

The HDS feature shows the maximum relative gain in m.a.p.. It increases from 18.2% at $p = 1$ to 23.6% at $p = 1/4$, a relative gain of 30%.

Effect of dimensionality

To further study the performance of fractional dissimilarity functions over different dimensionalities, RGB features with 4 to 512 dimensions were generated for the Corel collection. This allowed the examination of how the optimum p value varied with dimensionality. The results for single image queries are shown in Figure 5.7; multiple image queries showed the same characteristics.

The optimum p value increases from 0.25 to 0.5 as the dimensionality increases. This is perhaps counter-intuitive as it could be expected that the best p value would decrease as the number of dimensions increases. However, the generic behaviour remains the same across the wide range of dimensionalities investigated. The optimal p value occurs in a relatively stable interval. Indeed the fractional functions show a consistent improvement over L_1 at all dimensionalities. The largest gain relative to $L1$ performance was 27%, which occurred at 27 dimensions.

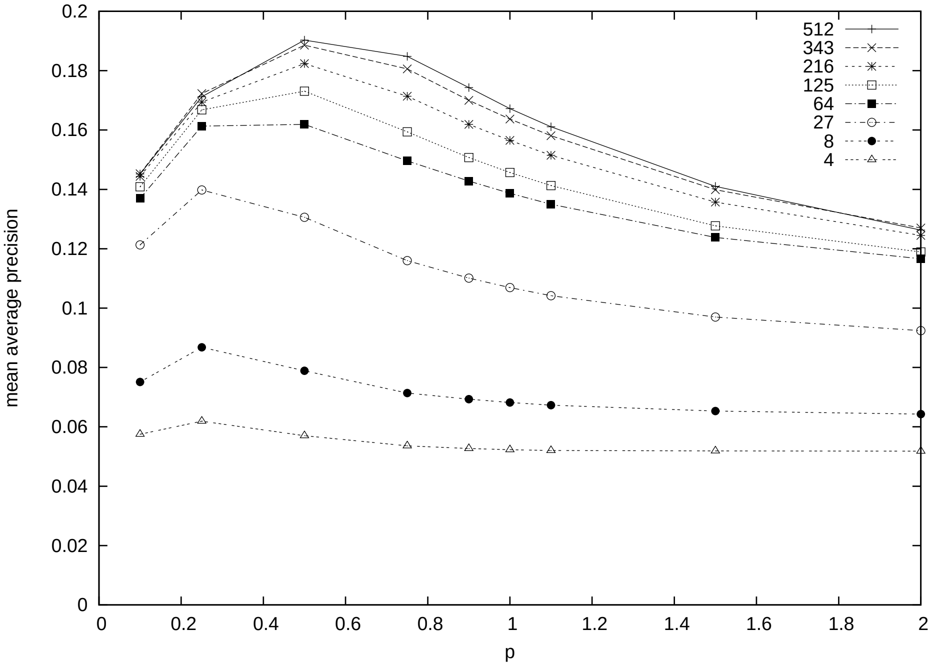


Figure 5.7: Retrieval performance using fractional dissimilarity with varying dimensionalities of RGB feature, Corel collection

Normalisation of features

The effect of normalisation of features was also tested with the Corel collection. Histogram features are routinely normalised to sum to one. This makes them all the same dissimilarity from the origin in L_1 space. Euclidean normalisation can be used when employing that distance function. It therefore seems plausible that normalising with respect to fractional dissimilarities may improve performance.

All the Corel experiments were run with with both normalised and unnormalised features. There was an absolute increase in m.a.p. of between 1–2% after normalising histogram features with respect to the distance function. This was statistically significant using a significance level of $\alpha = 5\%$. The effect with a non-histogram feature such as Gabor was less consistent but overall it showed around a 1% (absolute) decrease in performance. This is because the average level of the filter response is a significant discriminatory value. It seems clear that a gain in performance can be obtained by preprocessing histogram features in this way.

Getty

Retrieval based on the soft keyword classification of the Getty collection is a significantly different task to retrieval from the Corel collection. The results in Figure 5.8 again show improved retrieval performance for fractional dissimilarity functions. As with the Corel experiments RGB, HSV and convolution show

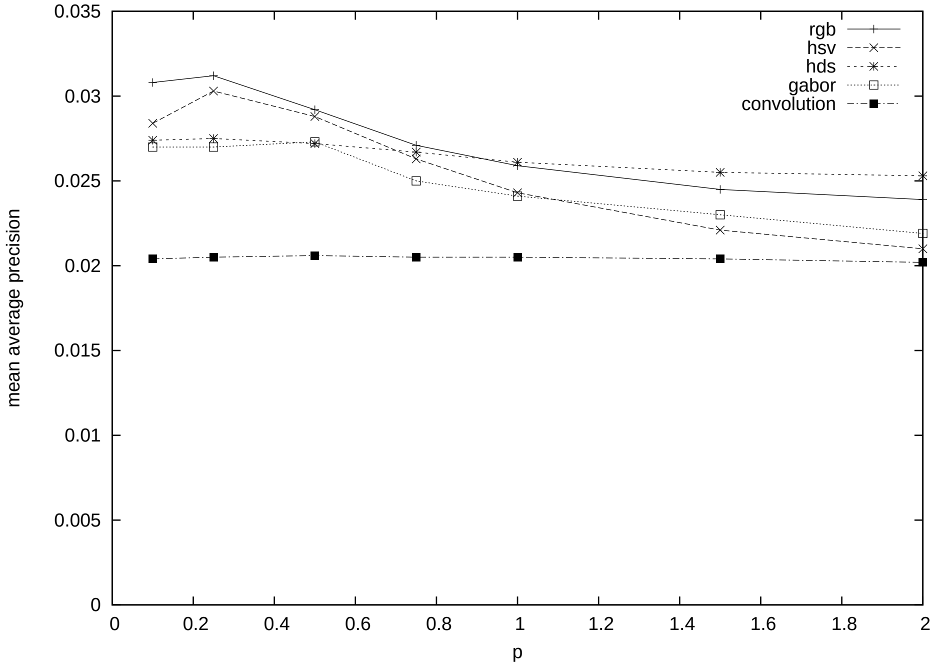


Figure 5.8: Retrieval performance using fractional dissimilarity, Getty collection

significant increases, HDS has a slight improvement while the others are flat across the range of p. The largest gain is for HSV, which is up 25%, from 2.4% to 3.0% m.a.p.. The peak improvements all occur within the range of p from 0.25 to 0.5. It is interesting to note that, in contrast to the results with other collections, performance does not reduce at the smaller values of p.

TRECVID 2003

The larger TRECVID collection presents another different challenge for image retrieval. The same set of features, as for Corel, were generated. The retrieval performance is shown in Figure 5.9. The results show a marked performance increase for fractional dissimilarity functions.

The overall results are very similar to those for Corel and Getty. RGB, HDS, HSV and convolution features show increased performance for fractional dissimilarities. Similarly, the performance for the Gabor feature does not improve, but remains constant down to $p = 0.25$. The maximum gain in m.a.p. is shown by the RGB feature which increases from 2.0% at $p = 1$, to 3.2% at $p = 1/2$. This is a relative increase of 60%.

The plots from the experiments have the same characteristic shape as for the other collections, with the maxima falling between 0.25 and 0.75. However, a detailed examination of the optimum p values for each feature shows that they have changed. This demonstrates that the optimum value of p is not independent of the data collection.

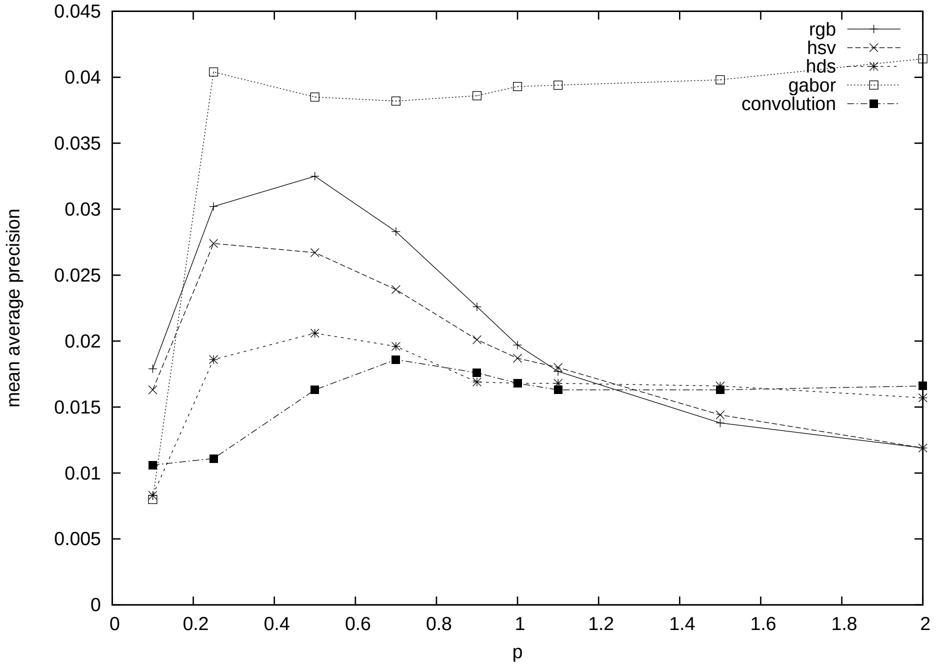


Figure 5.9: Retrieval performance using fractional dissimilarity, TRECVID 2003 collection

ImageCLEF 2004

The same features were used with the ImageCLEF collection for consistency. It would expected that the colour based features would perform less well with this collection due to its mainly monochrome nature. The results are plotted in Figure 5.10.

Examining the ImageCLEF results we can see that the general trend is similar to those from Corel, Getty and TRECVID. The convolution feature shows performance gains for fractional p values: this is a much larger relative gain than for TRECVID. The three colour features are relatively flat across the graph, dropping below their L_1 values with $p < 0.25$. The performance of the Gabor feature reduces slightly with fractional p values.

To explain these results we must consider the characteristics of the collection. It contains a large proportion of monochrome images, and because of the medical subject, contains groups of near identical images. For example, X-rays of a specific part of the body will always be composed in exactly the same way. Texture, shape are layout dominate the information within the image. The tiled Gabor feature discriminates texture and layout extremely well, hence its excellent performance. Convolution also detects shapes within an image. The effect of the monochrome images on colour features will be to reduce the dimensionality. Qualitatively this explains the reduced gain for the colour features.

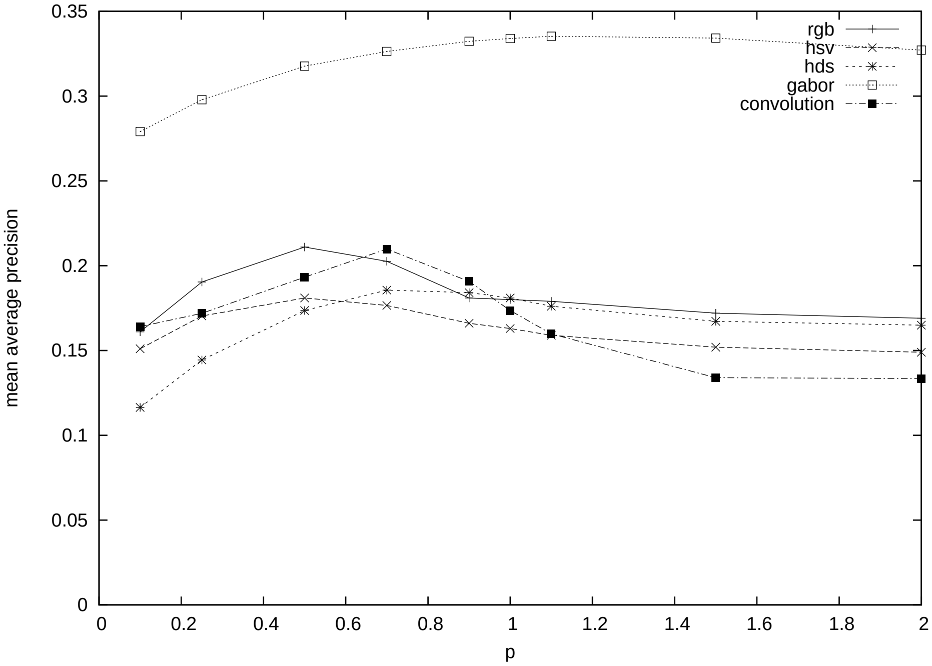


Figure 5.10: Retrieval performance using fractional dissimilarity, ImageCLEF 2004 collection

5.5.3 Performance of local similarity

An extensive range of experiments with local similarity was carried out. This Section contains a representative sample of results. The aim was to determine the effect of varying the size of the local neighbourhood and the actual local function used. In addition experiments were carried out to determine the impact of using only the dimensions where the query value was atypical.

The results graphs are of mean average precision retrieval against the fraction of data selected by the similarity function. This is the proportion of the entire data set that would need to be loaded from disk and can be taken as an indicator of the speed-up in query time. In the implementation each sorted dimension is stored contiguously on disk. This enables the retrieval of the minimum numbers of disk blocks required. Implementation and efficiency are discussed fully in the next chapter.

The region of the mean average precision graphs covering less than 40% of the data are most interesting as this level will give a significant speed-up. For proportions of data above 50% it was found that boundary and normalisation effects caused very variable performance. Some heuristics were developed and tested with the aim of smoothing out these variations. These were partially successful. However, results in this region have been excluded as they are misleading and are of no real importance when considering the goal of speeding up search. With 100% of the data boundary effects no longer occur. However, at this point the results may also be misleading. For example, a local L_1 function may give results very different from the usual L_1 distance because of the normalisation that occurs in the local

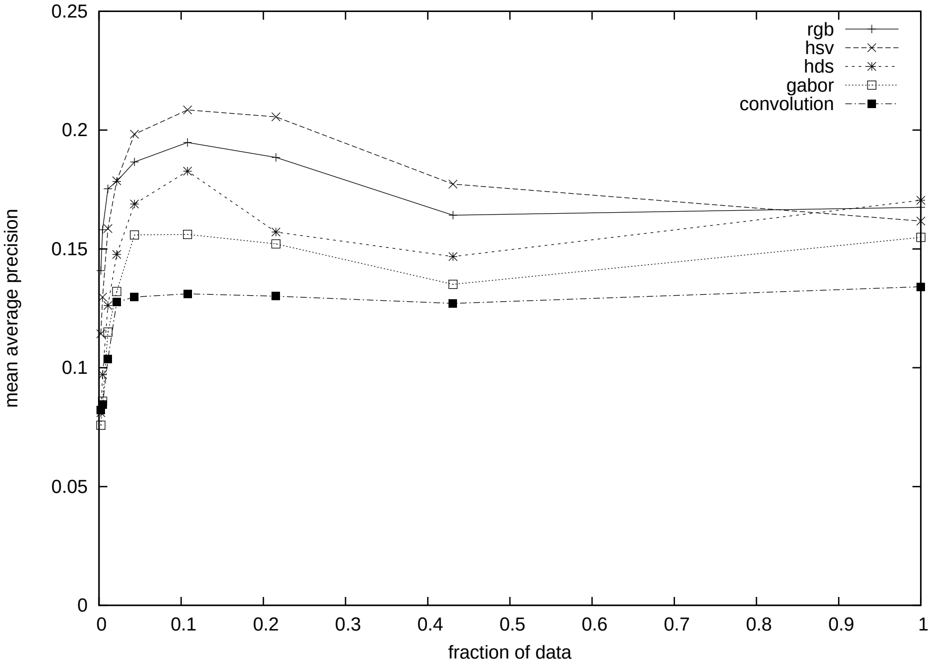


Figure 5.11: Retrieval performance using local L_1 function, Corel collection

similarity function.

Corel

Figure 5.11 shows the performance of the local similarity function, across the different features, using a Manhattan local distance function. These are averaged for a range of multiple and single image queries. From the graphs we can see that — although there is some variation — the retrieval performance remains the same down to 5–10% of the data. After this point m.a.p starts to drop off. For several of the features performance actually increases as less data is examined. Overall this is an exciting result as it means that up to 95% of the database can be ignored while maintaining retrieval performance. The absolute performance is not as good as the best achieved with fractional dissimilarities. However, it is between that obtained with an L_1 and L_2 metric.

TRECVID 2003

Figure 5.12 shows the results for TRECVID using a Manhattan local function. The results are similar in form to those obtained with Corel. Performance is maintained around L_1 levels down to 5–10% of the data, expect for the Gabor feature which shows a reduction in performance.

The graph in Figure 5.13 shows the retrieval performance with a range of local functions using the RGB feature. The most interesting thing to draw from this is, that with 0–50% of the data, the

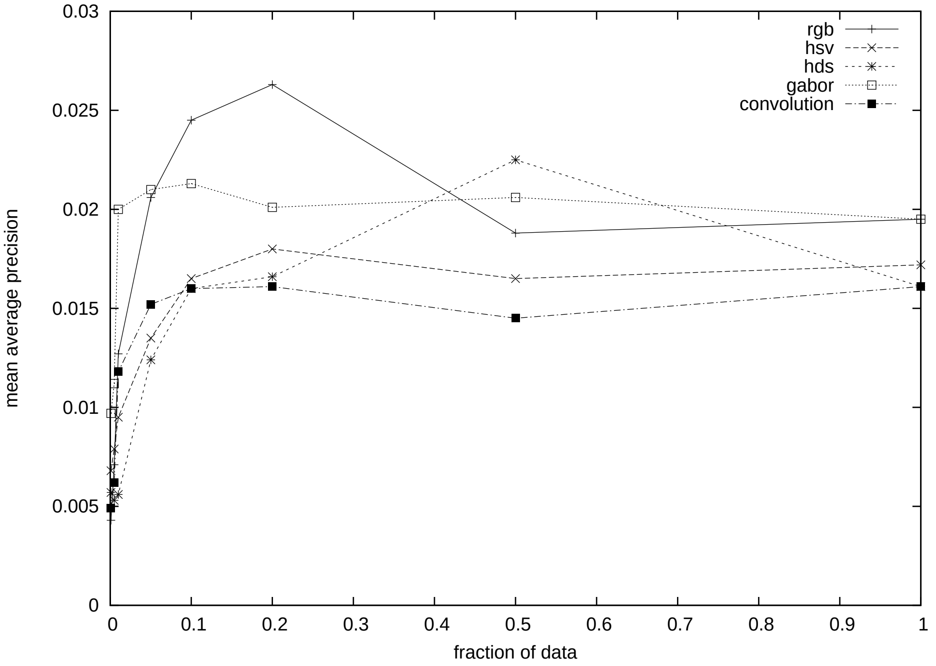


Figure 5.12: Performance using local L_1 function , TRECVID 2003 collection

performance of all the functions have the same characteristic shape. This indicates that the dominant factor is the membership of the local neighbourhood, rather than the form of the local function. As expected from 50–100% the performance of the voting function degrades rapidly as each dimension votes for all the objects. This is not a problem as this is not a region of interest.

Further experiments with other features and collections confirmed that they all exhibit the same characteristic. This is another exciting result as it means that the simplest local voting function will perform at a similar level to the more computationally intensive functions. This gives another avenue for increasing efficiency as there is no need to retrieve and process accurate distances. It therefore appears a good candidate for the fastest search method. There is still potential to exploit varying the local function in an image retrieval system but the voting measure is a good baseline that will be a challenge to beat consistently.

ImageCLEF 2004 and Getty

Figure 5.14 shows the results obtained with the ImageCLEF collection using a ranking local function. Again performance was maintained down to 5–10% of the data. The performance of all the features, except Gabor, is comparable with that obtained using a Manhattan metric.

Finally, with the Getty collection results are shown using the voting function, Figure 5.15. Performance above 40% drops off rapidly to random at 100%. However, between 5 and 20% performance is just

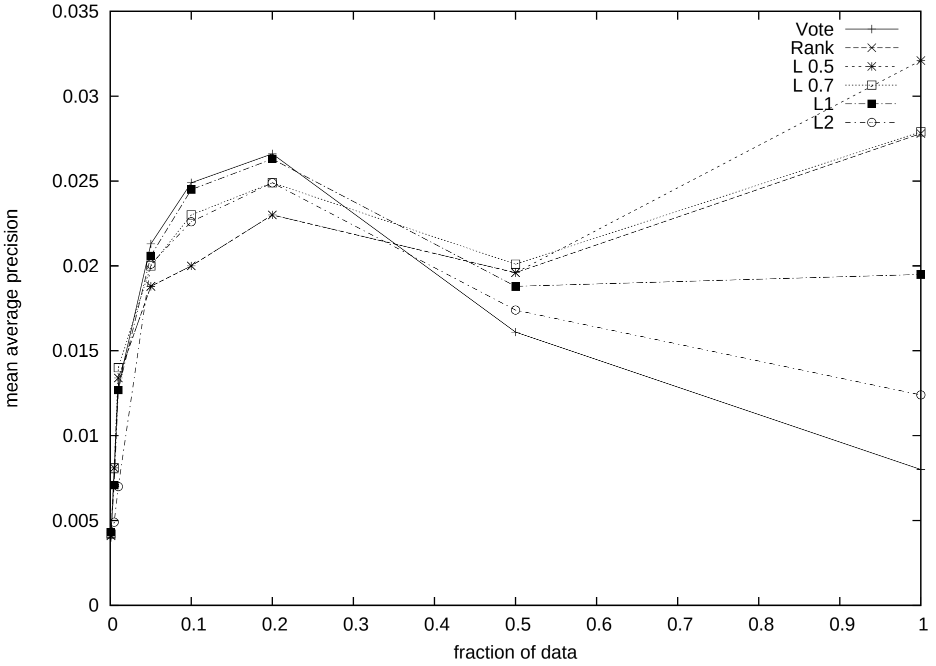


Figure 5.13: RGB performance for all local functions, TRECVID 2003 collection

Feature	Proportion non-zero (%)
RGB	22
HSV	49
HDS	16

Table 5.1: Proportion of non-zero elements in typical histogram features

below the maximum obtained with fraction dissimilarity functions and comparable with the Manhattan metric.

Selection of dimensions

An experiment was run with histogram features to compare the performance of either using all dimensions or only those where the query has a non-zero value. This was done using the Corel collection. The results were conclusive. The dimensions where the query was zero could be discarded with no loss in performance. Indeed in all cases there was an absolute improvement of around 1% in mean average precision. The effect on the amount of data looked at is significant. Table 5.1 shows typical proportions of non-zeros in the colour histogram feature vectors. These statistics were obtained using the Corel collection.

With non-histogram features experiments initially showed the effect to be similar. Dimensions were selected on the basis of how atypical they were. A large proportion of the dimensions, around 30%, could be discarded with a slight increase in performance. The proportion is less than that with histogram

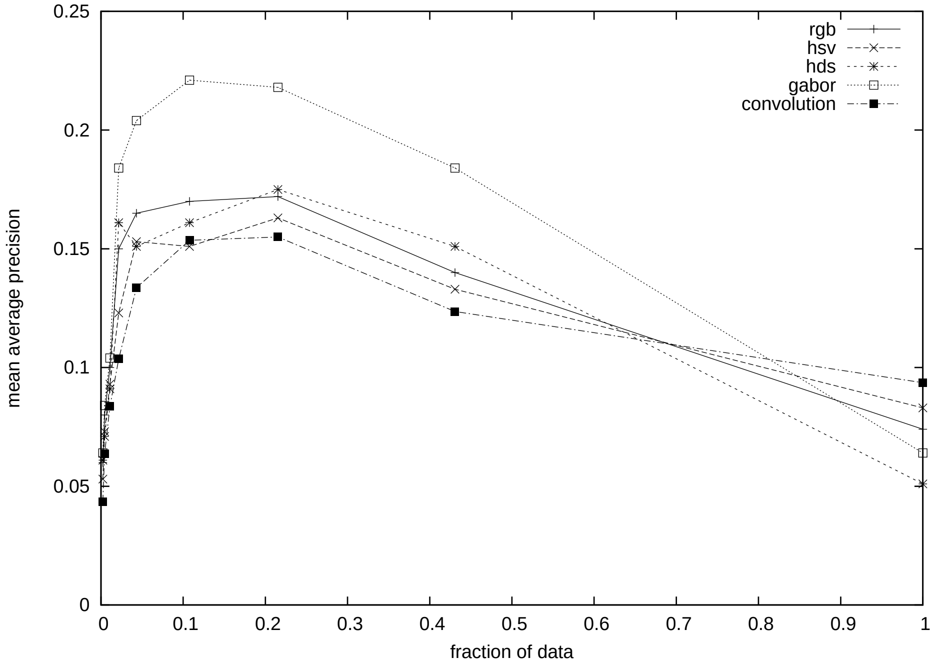


Figure 5.14: Performance using local ranking function, ImageCLEF 2004 collection

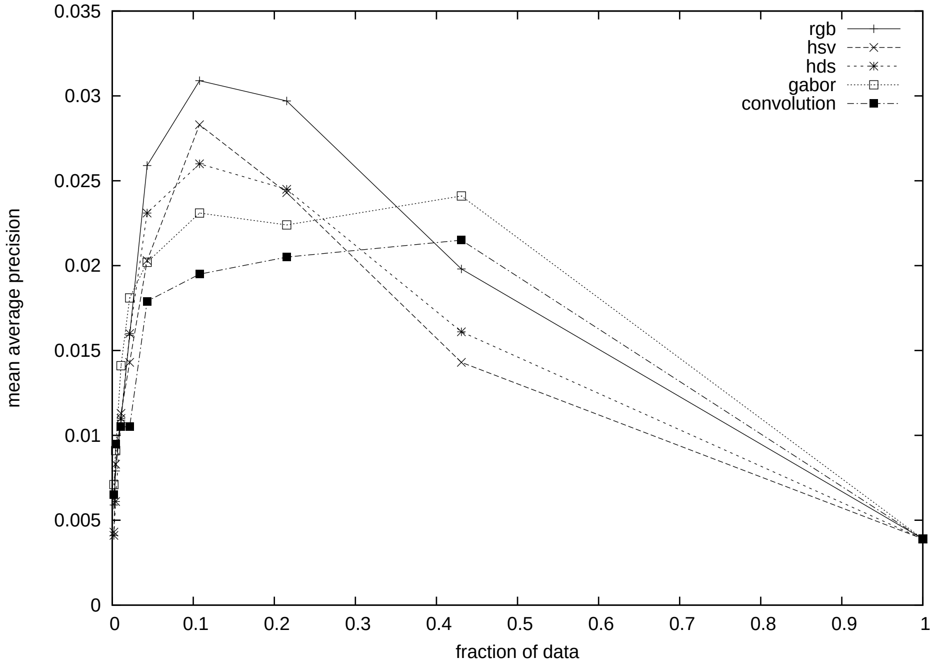


Figure 5.15: Performance using local voting function, Getty collection

features, but is still significant.

The same experiment was then performed but this time random dimensions were discarded. Interestingly a similar effect occurs: randomly discarding around 20% of the dimensions was not detrimental to performance. As the proportion of dimensions discarded was increased the performance gap between random and atypical selection widened. With the histogram features, randomly choosing which dimensions to discard caused a significant drop in performance.

The work by Indyk and Motwani (1998) on approximate nearest neighbours and Bingham and Mannila (2001) on dimensionality reduction, both use random projection in Euclidean space. They show that this method is an effective dimensionality reduction approach in high-dimensional space. Our results with random dimension selection in local similarity space concur with their findings. However, we have shown experimentally for CBIR that random selection is always beaten by an atypical selection method when used with localised similarity functions.

The consequences of these results are that further reductions can be made in the amount of data examined. Using a 10% width neighbourhood with histogram features we will only need to look at 1.6 – 4.9% of the data. With non-histogram features this rises to 7%.

Multiple image queries

Using several images as a query can commonly occur, for example, where a user selects several positive images from an initial set of results. The results in the previous section used multiple image queries where these were available. Presentation of these aggregated results hides some interesting behaviour that is revealed when single and multiple image queries are considered separately.

Section 3.4 describes two approaches to multiple image queries. In general, performance improves with multiple image queries where the images are visually similar. A multiple query can be compared to a query expansion. Each image is potentially within a cluster of relevant images in the feature space: different images can be in the same or different clusters. Pickering and Rüger (2003) showed that in an image retrieval context the k-nn approach outperformed the VSM model. K-nn uses both positive and negative images. In the experiments positive images come from the query and the negative images are randomly selected from the collection. This process makes k-nn results more expensive to compute than those using the VSM method.

The Corel collection has six sets of queries. Each set contains 630 n-image queries where n takes the values between one and six. Table 5.2 shows results for query sets using one, three and six images. The table shows mean average precision for the HSV feature using the different similarity functions. As a contrast to the previous results these were computed using the VSM approach. These results show the typical behaviour of the localised similarity function. The mean average precision for single image queries is below that of both L_1 and L_2, however it responds well to multiple images. For both three

Method	Number of images		
	1	2	3
Local - L_1	13.4	18.9	21.2
Local - vote	13.2	18.5	20.8
L_1	16.4	21.1	23.0
L_2	12.9	15.4	17.2

Table 5.2: Mean average precision (%) for multiple image queries using HSV feature – Corel collection

and six image queries the performance sits between that of L_1 and L_2.

5.5.4 Summary of results

Table 5.3 brings together the results for each method and collection. From this we can see that fractional dissimilarity functions outperform the other similarity functions in nearly all cases. The increase in mean average precision is largest when fractional functions are used with the colour histogram features. In a similar manner L_1 distances consistently outperform L_2. The L_p, $p > 0$, functions are all based on the same underlying formula. It would be expected that there would be a gradual change in performance as p is varied. This relationship can be clearly seen in the graphs in Section 5.5.2.

Local similarity functions are computed quite differently to the L_p functions, so it is interesting to consider their relative performance. In most cases their performance falls between that of L_1 and L_2. However, it is notable that with the TRECVID and ImageCLEF collection the mean average precision using the Gabor feature is significantly below all others. On the positive side, with the TRECVID collection the performance of local similarity with the RGB feature is well above that of L_1. In general there is a degree of variation in relative performance across collections and features. We can conclude that the performance of local similarity functions is comparable with that achievable using Manhattan and Euclidean distances.

5.6 Conclusions

This chapter has investigated and evaluated the idea of localising similarity. That is, in each dimension giving more weight to points that are close to a query point. Two classes of similarity function that have this characteristic were investigated using a content-based image retrieval task. These have the common property of contractive behaviour, that is, they emphasise the membership of small clusters that exist in subspaces of a complete feature space. The experiments showed that these similarity measures can improve retrieval performance and indexing efficiency within high-dimensional visual feature space.

The first approach was fractional dissimilarity. The results show that these functions give a significant improvement, in mean average precision retrieval, over the commonly used Euclidean and Manhattan

Collection	Method	Feature				
		RGB	HSV	HDS	Gabor	Conv.
Corel	Fractional	28	28	24	18	18
	Local	19	20	18	15	13
	L_1	24	23	18	17	14
	L_2	18	17	15	15	11
TRECVID	Fractional	3.3	2.6	1.7	4.0	1.8
	Local	2.6	1.8	1.6	2.0	1.6
	L_1	2.0	1.8	1.6	3.9	1.6
	L_2	1.2	1.2	1.5	4.1	1.6
Getty	Fractional	3.1	3.0	2.7	2.7	2.0
	Local	3.1	2.8	2.6	2.3	2.0
	L_1	2.6	2.4	2.6	2.3	2.0
	L_2	2.4	2.1	2.5	2.2	2.0
ImageCLEF	Fractional	21	17	17	30	21
	Local	17	16	17	22	15
	L_1	18	16	18	33	17
	L_2	17	14	17	32	13

Conv.=Convolution

Table 5.3: Mean average precision (%) for each similarity function and collection

distance metrics. The performance gains were consistent when using high dimensional visual features over four different image collections.

By experimenting across very different data sets it was shown that for fractional dissimilarity the optimum value of p for a feature cannot be determined by training on a single collection. It is linked to the combination of both feature and data set. The greatest gains in performance were achieved with sparse histogram features. These also benefited from normalisation of the feature with respect to the dissimilarity measure. Qualitatively there appears to be a link between the sparsity of the feature vector and how much a fractional dissimilarity function improves retrieval performance.

It was also demonstrated that a choice of $p \in (0.25, 0.75)$ improves mean average precision across most feature and data set combinations. To find the optimum p the dissimilarity function would need to be learnt for each collection. However, experimentally it was shown that a value of $p = 0.5$ can be expected to improve retrieval performance in nearly all circumstances. This is a robust choice for the parameter in circumstances where it is not possible to tune it to the specific collection and feature.

In the second approach I generalised a class of local similarity functions that are designed to address the problems associated with indexing high-dimensional feature space. The features are stored and indexed component-wise. For each dimension only those objects close to the query point are retrieved and then a local distance function is applied to this subset. Thus, the amount of data looked at can be dramatically reduced. Results showed that the performance of local similarity functions was at a similar level to that obtained using a Manhattan or Euclidean metric in the full feature space.

A local neighbourhood of 10% gives robust results in most situations, with the collections tested. It was shown that it is the membership of the local neighbourhood that drives the performance of this measure. This means that a simple voting function can give excellent retrieval performance. Alternatively, the local function can be varied at query time to squeeze the last bit of performance out of the system.

The local similarity function can be turned into a distance metric with use of a suitable local distance function. This was proved in Section 5.4.4. This result means that the metric can be applied to indexing and clustering methods that rely on this property.

With histogram features it was found an improvement in retrieval performance could be obtained by ignoring dimensions where the query value was zero. For non-histogram features, such as Gabor, selection of atypical dimensions gave a similar improvement. On average for histogram features 16–49% of the dimensions need to be considered and with non-histogram features 70%. This means that, for content-based image retrieval with the collections and features used, we can prune 93% to 98.4% of the data and still get comparable results to a full sequential scan.

Overall, the best fractional dissimilarity function will beat the retrieval performance of a local similarity function. However, as the size of an image collection grows the much greater search efficiency of local similarity may be a worthwhile trade-off for a small loss in average precision. In the next chapter we describe the practical implementation of a local similarity index. Using this implementation we can then investigate the retrieval performance and search speed of the local similarity index with much larger image collections.

Chapter 6

Searching large collections

6.1 Introduction

This chapter describes the implementation and evaluation of a local similarity index for CBIR. Different implementation strategies are discussed. The choice of which option is best will depend on the characteristics of the collection: particularly its size and how frequently it changes. A test implementation is described and then evaluated with large image collections. The aim of these experiments is to determine how both the retrieval precision and the time taken for an individual query vary, as the size of the collection is increased.

The next section reviews some alternatives for indexing feature spaces. These are considered and their strengths and weaknesses discussed. The review is not exhaustive, but aims to highlight the main ideas of high-dimensional search and indexing. This enables us to draw some conclusion as to what methods may be suitable for CBIR.

6.2 Indexing feature vectors

The practical issue with indexing high-dimensional spaces is how to reduce the amount of the data searched. If we can do this successfully it will have two benefits: reducing the number of distance computations and the amount of data loaded from disk or memory.

A single Euclidean distance calculation, for a p dimensional feature, will have p multiplications and $(2p-1)$ additions. These operations will take some time. However, the most significant bottleneck when searching any large database is the time taken to load the data from disk or memory. Accessing data that is in memory will be much faster than loading it from disk. Memory is becoming cheaper, and this has resulted in an increase in the amount usually available on a PC: 1GB is common. Many data sets are small enough to be loaded into this amount of memory. However, if we want to truly scale up CBIR, using multiple features, then it will be necessary to store the features on disk.

This remainder of this section gives an overview of indexing methods. In the conclusion we explain the reasoning behind approach we have taken for a CBIR search task.

6.2.1 Dimensionality reduction

It is likely that a real feature space may have an intrinsic dimensionality lower than the apparent data space. Often we make this so by building high-dimensional features. In certain cases it may be that not all elements of the feature are relevant. Dimensionality reduction methods aim to extract the significant information, within a feature, into a lower dimensional space. This is often used as a first stage in indexing, to reduce the data to a dimensionality to a degree that is manageable by a traditional indexing method.

Ideally a dimensionality reduction method should preserve proximity. That is for the mapping f, $d(x, y) \leq d(x, z)$ implies $d'(f(x), f(y)) \leq d'(f(x), f(z))$. Although this holds for translation and scaling in any Minkowski metric space, it only holds for rotation in a Euclidean space.

The most popular dimensionality reduction methods are based on singular value decomposition (SVD). These are transforms in Euclidean space and therefore preserve proximity. However, they are not proximity preserving for other distances measures such as L_1. SVD and its other versions (PCA, KLT) are effective and used widely as a pre-processing step. However, they are global methods that aim to preserve the main structure of the feature space. This means that a lot of the local detail, which may discriminate between images, is lost.

Several local dimensionality reduction techniques have been introduced to get round the problems of global SVD-based methods. The approach of Chakrabarti and Mehrotra (2000) finds locally correlated clusters and then performs dimensionality reduction on these. Other methods take a non-linear approach, aiming to find a lower dimensional manifold that the data lies on (Belkin and Niyogi, 2003). Another dimensionality reduction approach is to employ space filling curves. Lawder and King (2000) implemented a system using these for indexing multi-dimensional spaces. This was shown to be effective in spaces with up to eleven dimensions.

Although dimensionality reduction methods have been used with great success for some image search domains, such as face recognition, they have not been widely adopted for CBIR. A significant drawback for indexing large collections is that they are computationally intensive. Also, the transform to lower dimensional space can be sensitive to changes in the image collection, and hence require recomputing if there are updates to the data.

6.2.2 Hierarchical structures

A significant family of methods partition the feature space or data points into hierarchical structures. This section gives a brief view of the proliferation of these methods. A wider review is given in the survey by Böhm et al. (2001), while the recent book by Samet (2006) gives more detailed descriptions.

We first consider a class of methods that work well in relatively low-dimensional spaces. The simplest of these is the grid. This divides the space into squares, each then having a list of points within it. Developing this slightly we get the Quadtree of Finkel and Bentley (1974). This combines the grid and a binary search tree. The grid is no longer of equal squares but each block is decomposed when they contain more than a set number of points. The fanout of a quadtree is high, 2^p for a p dimensional space. This restricts its usefulness to low dimensionalities. The final variants of this type are based on the K-D tree of Bentley (1975). These partition the space dependent on the value of a single attribute (dimension) so can be represented by a binary tree.

The second class of methods are based on minimum bounding rectangles (MBR). These are a partition of the space defined by the minimum and maximum values in each dimension. The original index of the type was the R-tree developed by Guttman (1984). As points are added the space is split into nesting MBRs. In this way points are grouped into hierarchies and then stored in a disk based structure. As the structure develops the MBRs will start to overlap. A point will have a single MBR, but the area that this spans may be in several MBR's. This is the drawback of these structures – that they are not a disjoint decomposition of the space.

There have been many variants of these approaches that have proved successful in different circumstances. To make them more effective the bounding box overlap needs to be minimised and the fanout managed. Various algorithms have been developed to do this. They generally look at the increase in overlap following an insertion and minimise this. In addition the algorithms calculate the optimum axis and position to split a node when it overflows. Beckmann et al. (1990) developed the R^*-tree, which aims to do this by minimising the overlap with the adjacent MBR's. It carries out forced reinsertion when the leaf node overflows, rather than splitting it. The X-tree of Berchtold et al. (1996) takes an different approach. Where splitting an overflowing node would create too much overlap, it instead creates a supernode of two disk blocks.

White and Jain (1996) developed the SS-tree. This is based on minimum bounding hyperspheres (MBS) rather than rectangles. The idea behind this is that in Euclidean space query regions are also hyperspherical. The SR-tree (Katayama and Satoh, 1997) aims to reduce the volume of the minimum bounding boxes, and thus reduce node overlap. This is done by defining the bounding boxes by the intersection of MBRs and MBSs.

An alternative partitioning method is the Pyramid Technique (Berchtold et al., 1998). The hypercube is subdivided into pyramids with their tips at the centre of the data space. The pyramids are two dimensional and have one of the faces of the hypercube as their base. The data space is then subdivided by peeling off hypervolumes close to the boundary. This method works well with high-dimensional spaces, but is adapted only for the maximum metric, L_∞. The maximum metric returns the maximum value from the set of distances from the query point in each dimension. This is not a suitable metric for search using high-dimensional visual features as this maximum value is likely to contain little discriminatory information.

6.2.3 Distance metric indexing

Another major class of indexing methods is based on using the triangular inequality property of metric spaces to prune the search space. These methods identify a set of pivot (or vantage) points, and sort the other objects based on the distance from the pivots. This forms the index and can then be used to prune the search space. There are then two general methods: hyperplane partitioning as used in the Geometric Near-neighbour Access Tree (GNAT) (Brin, 1995), and ball partitioning used in the vantage point tree (vp-tree) (Yianilos, 1993).

As with the hierarchical methods the number of algorithms using distance based indexing has proliferated. The survey paper by Chavez et al. (2001) provides a good overview of these methods, and again the book by Samet (2006) provides more, in depth, details.

6.2.4 Sequential scan and compression

The methods described in the previous sections were all shown to perform well in the original papers. The evaluations were with different collections (usually non-image) and feature dimensionalities. They performed well up to medium dimensionality levels (10-20). In Section 5.2 we discussed the effects of high dimensionality on indexing methods that partition the feature space. Weber et al. (1998) showed that above a certain dimensionality all tree structures collapse to a point where every disk page must be accessed. When this happens a sequential scan of the data will be much more efficient as the I/O cost will be reduced (see Section 6.3.1 for a discussion of disk access times).

They established that for every clustering and partitioning method there is a dimensionality above which a simple sequential scan performs better. At this point the complexity of these methods will be $O(n)$, but added to this the disk access will be random. In their practical evaluation, they found that a sequential scan outperformed the X-tree and the R^*-tree for dimensionalities greater than ten. This led them to develop the VA-file, which accepts the fact that the linear scan is inevitable and attempts to optimise it using quantisation to compress the data. Using this they achieved search times of 12.5–25% of a full linear scan.

The VA-file was improved by using a better quantisation method (Ferhatosmanoglu et al., 2000). This method, the VA+ file, uses PCA to transform the feature space into new orthogonal dimensions.

The number of bits used for quantisation is varied depending on the significance of the dimension, and then clustering is used to determine a non-uniform quantisation scheme.

The compression of data to reduce the space requirement is an attractive proposition for indexing. Several researchers have combined compression with other techniques. The IQ-tree, introduced by Berchtold et al. (2000), is a hybrid of the R-tree and VA-file. It uses a set of MBRs pointing to leaf nodes containing compressed data. These can be thought of as VA-files for the data within the MBR. Below this level are pages containing the uncompressed data. By identifying the relevant MBRs for a query, the amount of compressed data scanned can be reduced when compared to the full VA-file. The A-tree (Sakurai et al., 2000) stores a compressed representation of bounding boxes of child nodes in its inner nodes. The children are defined relative to their parents, thereby allowing quantisation with fewer bits. These methods often perform better than the similar methods that do not use approximations.

6.2.5 Inverted files

Another group of methods take the approach of vertically decomposing the feature space into separate dimensions: Each dimension is stored and indexed separately. This gives a very flexible structure as dimensions can be treated differently depending on their significance.

Müller and Henrich (2004) created the inverted VA-file. This stores each compressed dimension at different quantisation levels, and only retrieves at the accuracy needed dependent on the query. They used this to significantly speed up searches using the histogram intersection measure.

The Branch-and-bound on Decomposed Data (BOND) system was developed by De Vries et al. (2002). It maintains a separate table for each dimension, containing all the vectors in the repository. The distance between a query point and the data points is accumulated by scanning the dimensions in turn. As partial distances of each vector are built up the lower and upper bounds on the complete distance can be computed. Those vectors that can no longer be in the nearest neighbour set are discarded, thus pruning the search space. This process is applied iteratively until the required number of nearest neighbours is obtained. One drawback of their method is that the pruned set will not necessarily be stored contiguously on the disk, so there may be a degree of random disk access.

Aggarwal and Yu (2000) introduced the iGrid index which uses the concept of local similarity. The function defined in Equation 5.10 is a generalisation of their approach. In the iGrid index the feature space was vertically decomposed and each dimension split into bins. The amount of each dimension selected for a query was inversely proportional to the dimensionality of the feature space. They demonstrated that for a non-visual classification task the results from the index showed improved meaningfulness over using the Euclidean distance. Cha (2003) also used a type of localised similarity function, but instead of storing dimensions independently used a bitmap representation. Evaluation of this showed improved speed of search but no results showing the meaningfulness of the search were presented.

6.2.6 Summary

This section has reviewed indexing methods for high-dimensional vectors. Efficient indexing is the key to scaling up CBIR, and for the search of high dimensional features in general. The main bottleneck for very large collections is the time taken to access the data. A selection of the huge number of hierarchical partitioning and clustering methods proposed for high-dimensional indexing have been reviewed. Although these are very effective for medium dimensionalities, it has been shown that for high-dimensional spaces they all will perform worse than an optimised sequential scan. It should also be taken into account that with CBIR we are interested in returning a ranked list of the nearest neighbours, not just finding a single nearest neighbour. This is likely to degrade the indexing performance of these methods even faster.

The sequential scan is the most stable method, giving repeatable performance. For this reason sequential scan is often used as a benchmark for other systems. It has the added benefit in this respect that it is easy to implement correctly and efficiently. Similarly the VA-file, which optimises a linear scan, can be regarded as a target that other indexing methods should aim to beat.

The implementation tested in this chapter uses an approach based on localised similarity functions. These store and index each dimension separately allowing the data searched to be pruned drastically. In addition, this data structure has intrinsic flexibility as each dimension can be treated differently at query time. In the previous chapter we showed that this method produced good retrieval results for a range of CBIR collections and features. In the remainder of this chapter we aim to demonstrate how effectively it can be applied to the task of searching large collections.

6.3 Implementation of an image search system

Image retrieval is a practical discipline driven by the requirements of users. When implementing a real system there are lots of issues that appear. This section explains what is important when working with localised similarity measures in a practical context. The aim is to give some alternative choices of data structures and algorithms that are available to the implementer of a CBIR system. These choices are driven by the characteristics of the collection, primarily its size and changeability. Within the discussion the implementation of the test system, used for scalability experiments, is described.

6.3.1 What slows down search?

From Figure 2.2 in Chapter 2, we can see that to perform a search a CBIR system needs to, compute features for the query image, compare these with the features in the database and then rank the images in order of relevance.

The generation of features for the query image takes a fixed time, so will not vary with the size of the collection. Complex features may take longer to compute, but this is not an issue considered in this chapter. It is the actual search time that increases with the size of the database. The search comprises of three stages: loading the features, performing the similarity comparison and ranking the output. Which of these components dominates the time taken will depend directly on the size of the collection.

To discuss this, we first need to consider the relative speed of components of PC hardware. CPU technology has continued to advance rapidly since its introduction. On new home computers CPUs have clock speeds of several GHz. Within the CPU itself there is a hierarchy of memory including registers, L1 and L2 caches. These operate together to make computation very fast. The on-processor memory is limited in size so there is often a requirement to fetch or save data to the main memory. This is normally random access memory (RAM). Here memory latency becomes a limiting factor; the speed of RAM is at least an order of magnitude slower than processor memory. As the data storage requirement increases it becomes necessary to utilise a secondary store, normally a hard disk. These are mechanical devices and their transfer rates are much slower than RAM. There is an added issue because data on disk is stored and retrieved sequentially. The retrieval time for a given disk page will consist of the seek time, where the disk head moved to the required block, and the transfer time. This means that randomly accessing data on disk may be up to ten times slower than retrieving data sequentially from contiguous disk blocks. More detailed discussion of these issues can be found in the latest edition of the classic undergraduate text by Hennessy and Patterson (2006).

If we restrict these discussions to the search of high-dimensional features routinely used in CBIR systems, we can discount the use of hierarchical indexing techniques. As described in Section 6.2 they can often be much slower than a sequential scan of the entire feature space.

So when considering a search of high-dimensional feature space, we have a large amount of data that needs to be compared. Computation of the relatively simple similarity measures is fast, so the most significant factor will be the time taken to access the data. Small collections can be loaded into memory in their entirety. Under these circumstances memory latency will be the limiting factor in the search speed.

As the number of images in the collection increases, it will get to a size when disk storage of features is the only option. Disk access is one to two orders of magnitude slower than main memory and this will now become the bottleneck for search speed. The simplest approach in these circumstances is to save the feature database in a single contiguous file and perform a sequential scan. Although this may appear naive, it can often be fast enough because disk access of contiguous disk blocks is much faster than when they are spread around a disk.

The local similarity index will reduce the amount of data that needs to be loaded. However, this will be at the expense of some additional search operations that also lead to the need for a degree of random

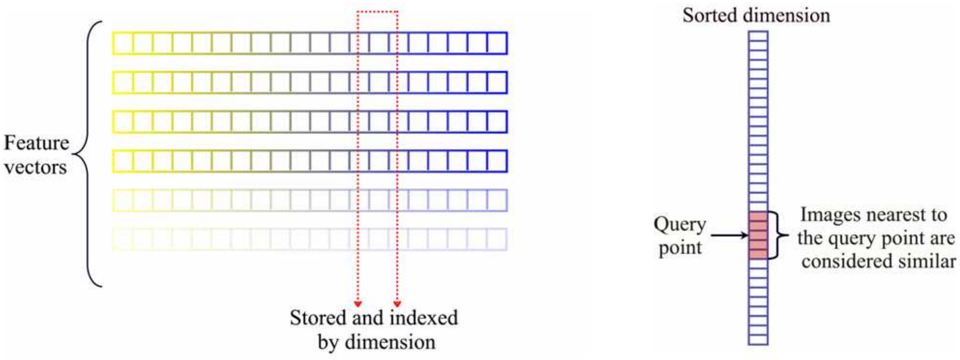

Feature vectors
Stored and indexed by dimension
Sorted dimension
Query point
Images nearest to the query point are considered similar

Figure 6.1: Local similarity index

disk access. In the next subsection the implementation of a test system is described.

6.3.2 The local similarity index

Local similarity functions consider each dimension of a feature independently. This means that the feature database can be decomposed vertically and each dimension indexed separately, Figure 6.1[1]. This structure has two advantages: there are very effective methods of indexing one-dimensional vectors, and the underlying data for each dimension can be stored sequentially.

To compute local similarity we need to find the objects that are within a certain range around the query point in each dimension. There are two stages to this, find the query point and retrieve the range.

At an abstract level, because of the need to retrieve ranges, the best data structure is one where the actual data for each dimension is stored contiguously. This can be indexed by a tree that provides $O(\log n)$ search. For a static collection we can achieve this by either storing the data in contiguous disk blocks or in memory. This was the design of the system used for the experiments in this chapter.

The implementation was kept simple to minimise the factors that may influence results. Each dimension consists of a set of points sorted by the element value. Each point was a tuple of a value and image id. The value was stored as a 4 byte float, and the id as a 4 byte integer. For the size of collections tested it would be possible to store the image id in three bytes, however, for simplicity this was not done. With histogram features we have already shown that it is beneficial to ignore zero valued elements. There is no need to store these at all. Similarly with the other feature types there is no need to store the points with “typical” values. Doing this saved considerable storage space. Each dimension was sorted by the feature value and then saved as a separate file on disk. These files were mapped into memory using the sys/mman library[2]. The loading and caching of the files into main memory was left to the hardware and

[1] Interestingly the Nobel Prize winning work of Hubel and Wiesel showed that the V1 cells of the visual cortex are organised in hyper-columns. The structure of the local similarity index resembles this orientation within the brain

[2] This library is specific to Linux, memory mapped files can be created on Windows using the CreateFileMapping function

operating system.

To find the query point, a binary search tree was constructed for each dimension file. These were stored in memory rather than on disk to enable rapid search. The leaf nodes of the tree were pointers to each page of the dimension file. This reduced the size of the binary tree needed. Each disk page of 4KB[3] can store 512 points (value/ id pairs). Once the required pages had been loaded into memory a binary search was performed to find the exact query position. As each dimension file was static it was easy to ensure that the associated binary tree was balanced.

Alternatives and extensions

The test implementation was based on indexing a static feature database. This would need to be rebuilt each time changes to the collection were made. If the collection changes frequently then it would be better to use a data structure that would allow easy insertions and deletions. A disk-based data structure that fits these requirements is the B+ tree. This is a type of B tree, which is a multilevel index with multiple keys stored at each node (Bayer and McCreight, 1972; Comer, 1979). The nodes are stored on individual disk blocks enabling efficient access. The B+ tree has all data stored in the leaf nodes, and has pointers between the leaf nodes so that they form a linked list. This is useful for range search. The algorithms that insert and delete points in the tree ensure that each node will be at least half full (except for the root). There will almost always be some space that is not utilised. This overhead is the trade-off for being able to easily update the collection.

The cache-oblivious B tree could be an alternative to using a B+ tree for storing and indexing dimensions. The details and benefits of these were explained by Bender et al. (2005). They are not dependent on any system-specific parameters and have been shown to be very efficient. In addition they use a packed-memory array to store the actual data. This would be ideal for the range search required for the local similarity index. This structure may give a speed advantage over the linked list of leaf pages in the standard B+ tree.

On a different note a ”filter and refine” approach could be adopted. This is where the top ranked results from the local similarity index are re-ranked using another similarity function with the full feature. This could be the fractional function, as described in Chapter 5, which may potentially give better results.

6.4 Experiments

The aim of the experiments described in this section was twofold: firstly, to determine how the retrieval performance varies for different local neighbourhoods and collection size; secondly, to measure search times for each configuration as the collection size is increased.

[3] This was the actual page size for the disk used. A larger page size may make disk transfer more efficient.

For these experiments it was necessary to construct a large, variable size, image collection with ground truth. The Web Collection described in Section 3.2.3 was created for this purpose. To provide queries and ground truth it was combined with the Corel collection. The test subdivision of Corel (4644 images) was included in each sub-collection, together with the required number of web images added to make it up to the desired size. Collections of 16, 32, 64, 128, 256, 512 and 1024 thousand images were created.

The design of this collection has its limitations. It is the combination of a fairly homogeneous collection (Corel) with one that is heterogeneous. This may make it easier to find relevant images than in a truly heterogeneous collection. As the size of the collection increases the proportion of relevant items to non-relevant will decrease. Each query has, on average, 69 relevant images. For the smallest collection the proportion of relevant images is 0.4%, this decreases to 0.007% for the largest collection. This variation does not necessarily mirror what would occur for a real query. These characteristics must be remembered when interpreting results obtained using this collection.

The choice of performance measurements should reflect the needs of a person doing a search. The user is likely to be most interested in what is returned on the first page of results, say twenty images. They may look deeper into the results, but are unlikely to go beyond five pages, or one hundred images. Under this assumption results beyond this point become irrelevant. We have therefore decided to use precision at twenty (P20) and one hundred (P100) returned images as evaluation measures. If a system returns relevant images in these ranges then the user's search may be satisfied. Alternatively, they will be able to refine their query with more suitable pictures, perhaps building a multiple image query. This search scenario may not fit all circumstances, but it gives a reasonable reference point.

Evaluation of response times was discussed in Section 3.3.3. The measure used was a wall clock time averaged over one hundred searches. Times for a linear sequential scan are given as a baseline. A time of one second is used as a target for an acceptable response time. Ideally we would want to be able to complete a multiple image query with several features within this target time.

Results are shown for the RGB feature. This is a joint colour histogram with 512 dimensions, see Section 3.5. Performance with other features showed the same general characteristics.

6.4.1 Retrieval performance

The set of 630 single image queries were run using local neighbourhoods of 10%, 5%, 1%, 0.5% and 0.1% of each dimension. The P20 and P100 results for collection sizes up to 1,024,000 images are plotted in Figures 6.2 and 6.3 respectively.

As expected the absolute results decrease with increasing collection size. With the smaller collections there are large differences between the precisions for different neighbourhood sizes. The largest neighbourhood of 10% gives the best results. This reflects the results in the previous chapter. If we consider that a 0.1% neighbourhood will only include 16 images out of 16,000 then it is not surprising that the

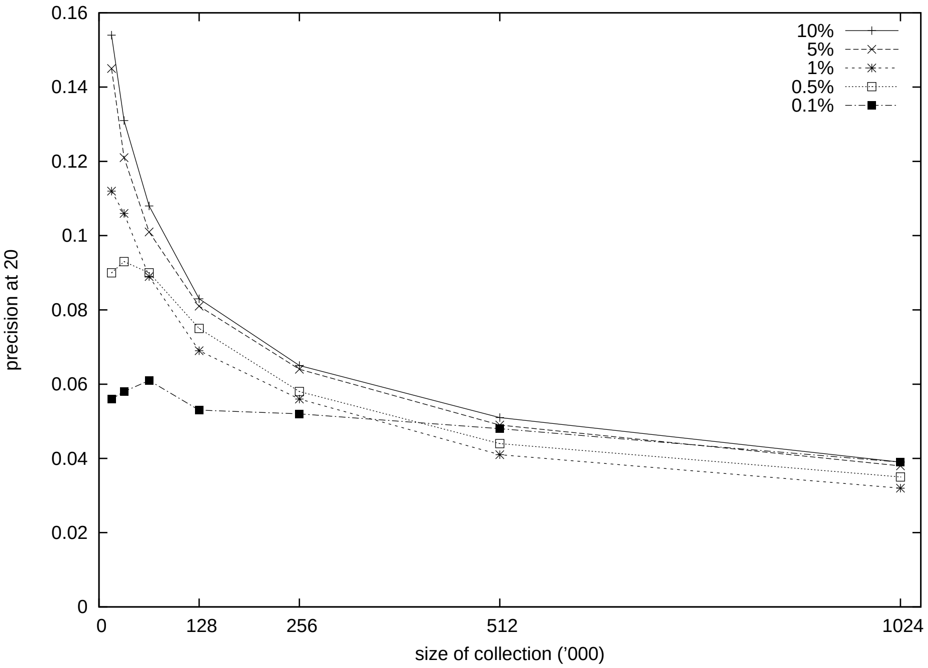


Figure 6.2: Precision at 20

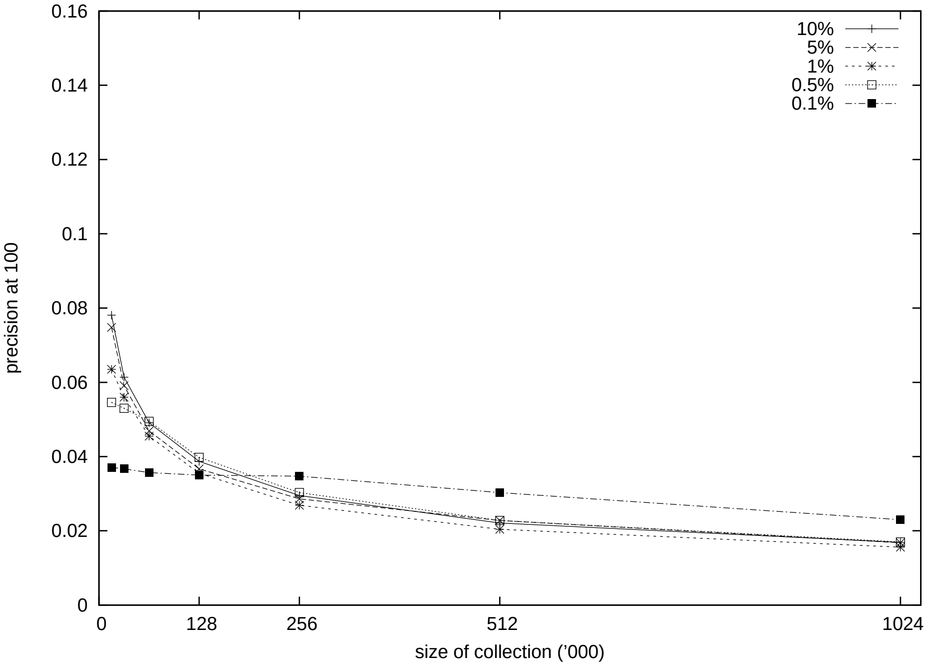


Figure 6.3: Precision at 100

precision is low.

As the collection size increases the P20 results tend to the same value. What is interesting is that the 0.1% neighbourhood gradually overhauls the others. By one million images it is giving the best precision of 4%. It should be noted, however, that the differences in precision at this point are not statistically significant. A P20 of 4 percent equates to an average of 0.8 of an image in the top twenty.

The P100 results show similar behaviour, except in this case precision for the 0.1% neighbourhood leaps above the others by a collection size of 256,000 images. It keeps an advantage of over 0.6% up to the maximum collection size tested. At this point there is an average of 2.3 images in the top 100. This performance is achieved using a neighbourhood of 1,024 points in each dimension.

Considering the P20 and P100 results together we can see that, in general, relevant images are retrieved near the top of the results list. Detailed examination of the results showed that with the largest collection size an average of one relevant image was retrieved in the top 30 results.

The use of averaged query precisions does mask some interesting details. For a collection size of 1,024,000 images the median query of the result set returns just one image in the top 100. A significant proportion of queries return no relevant images in the top 100. However, this is a common occurrence with all CBIR systems. There are “hard” queries where CBIR is unable to find relevant images. Overall, the performance of the local similarity index with large collections is good, and it should be able to satisfy a real user search need.

6.4.2 Speed of search

Measurements of search speed are not as precise and repeatable as precision measurements. There are many variables including the hardware, operating system, collection and queries. For example, caching occurs at disk, operating system and CPU level. The resultant behaviour is complex and difficult to predict. Therefore we have tried to interpret the results to determine general trends rather than treat them as precise measurements.

Experimental set-up

The experiments were run on a standard desktop PC with an AMD Athlon 64 3700+ 2.2GHZ processor and 1GB of RAM. The system had a single Maxtor 6L200SO disk. The specification of this was as follows: 8MB buffer, ATA-150 interface, 9ms seek time and 150Mb/s (18.75MB/s) data transfer rate. The operating system was Linux 2.6.19 and programs were compiled using gcc 4.1.

As discussed in the previous section the RGB feature was used for testing. An individual feature comprises of 512 floats (each taking up 4 bytes) thus has a size of 2KB. So a collection of 1,024,000 features will take up 2GB. This will be the size of the corresponding linear scan file.

Collection size	Sequential scan		Local index		
in ('000)	L_1	$L_{1/2}$	10%	1%	0.1%
16	0.097	0.13	0.016	0.0058	0.0026
32	0.19	0.29	0.027	0.0052	0.0047
64	0.39	0.41	0.054	0.012	0.0063
128	0.77	0.82	0.10	0.018	0.0084
256	1.5	1.6	0.18	0.031	0.015
512	27	28	0.32	0.057	0.028
1024	53	55	0.71	0.27	0.19
2048	–	–	2.8	0.79	0.62
4096	–	–	8.2	2.0	1.05

Table 6.1: Search time in seconds for sequential scan and local index

With the local index there is a storage overhead of a four byte integer id with each feature element. This doubles the storage requirement for each point. If all points were saved then the feature database would take up 4GB. However, zero values are not stored. This reduces the total file size to 0.9GB for a 1,024,000 image collection (on average 22% of each feature is non zero, see Table 5.1). To ensure that the test included cases where the file size exceeded the amount of main memory the maximum collection size was increased to 4,096,000 images. This was done by simply duplicating the feature database twice.

A set of 100 queries was used for speed testing. The results shown are averaged times for a single search. The queries were shuffled so that images of the same class did not appear consecutively. This ensured that the likelihood of a disk page being in the cache was minimised. This configuration should represent the worst case of each query being very different. In reality, if a user is browsing through the collection they may well search repeatedly with similar images.

Linear scan

The main component of the linear scan test system was a memory mapped feature file. All caching was left up to the operating system and hardware. During the tests the memory usage reached a maximum of 80% (0.8GB). The results are shown in Table 6.1 and graphically in Figure 6.4. They were broadly as expected. The time taken for a search increased linearly up to a collection size of 256,000. By interpolation we can see that the target search time of 1 second is reached with approximately 168,000 images. Beyond a collection size of 256,000 images the entire feature file could no longer be loaded into main memory. This meant that the speed bottleneck moved from being the latency of the RAM to the transfer rate of the disk. Correspondingly, the search time increased dramatically to 27 seconds for a collection of 512,000 images. Using the specified transfer rate for the disk of 150Mb/s, 1GB can be transferred in 56 seconds. The times recorded are half of this. Allowing for caching in main memory these results seem realistic. It should be noted that the additional computation needed for fractional dissimilarity functions, as opposed to the Manhattan distance, has a relatively small impact on the search

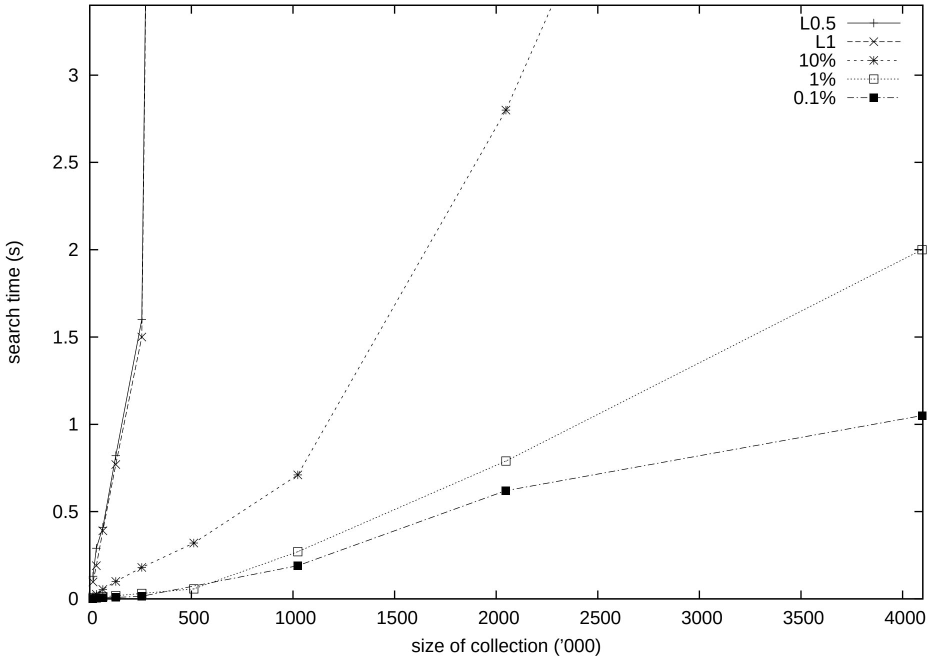


Figure 6.4: Search time in seconds for sequential scan and localised index

time. This means that the choice of L_p distance function can be made based on retrieval effectiveness rather than speed.

Local index

The implementation of the test local similarity index is described in Section 6.3. It is based on each dimension being stored in a separate memory mapped file. As before, caching was left to the operating system and hardware drivers.

From the tabulated results (Table 6.1) we can see that the local index is one to two orders of magnitude faster than the linear scan method. For a collection size of up to 512,000 images the dimension files fit in memory and the search times increase linearly.

Above this point the number of features becomes too large to fit them entirely in memory. They will therefore be loaded from disk as needed. There is a resultant jump in response time. For example, the search time for the 0.1% index increases from 0.028s to 0.19s. This is an increase of around six times for a doubling in collection size.

To interpret the results we need to consider the various bottlenecks that influence the search speed. On average, each query will have 113 non-zero elements and therefore needs this number of dimensions to be loaded. The seek time for each random disk access is 9ms, making a total seek time for a query of around one second. For a collection size of 1,024,000 features and a 1% neighbourhood a total of 9MB[4] needs to be loaded. Using the disk transfer rate of 18MB/s this should take 0.5s. For a neighbourhood of 0.1% the data transfer time is 0.05s and for the 10% locality it is 5s. Doubling the collection size will double the data transfer time but the random access time should remain constant.

The preceding discussion ignored the effect of any caching of the files in main memory. The results in Table 6.1 are considerably below those predicted for the collections of 1,024 and 2,048 thousand images. For the largest collection the time for the 0.1% locality search is around that of the total seek time. This indicates that a random disk access is needed for each dimension. For the 1% and 10% index the data transfer time has more influence. The total times are again less than those predicted, presumably due to efficient caching of data.

It should be noted that these results are averaged times. There was a significant variation in speed for individual queries. They could be up to double the average. It was seen that search times gradually reduced as the set of queries was run. When the query set was sorted by image class the average search time was roughly halved. These effects were due to caching of the dimension files.

Overall these results were in line with expectations. They demonstrated that for small collections RAM latency is the bottleneck, while for large collections the limitation becomes hard disk access and transfer times. With collections of one to four million images, and a local neighbourhood of 0.1%, we can see that the factors that dominate search time are the seek time of the disk and the number of dimensions. In effect the complexity of the search time is linear with the number of dimensions. For the test system it happens that the seek time for 113 dimensions equates to 1 second. With the effect of caching this allows the search time to meet the target of 1 second with up to four million images. In this way the performance of the system has exceeded the original goal.

6.5 Conclusions

The implementation of a test system and subsequent evaluation with large collection sizes has shown some interesting results. It should be noted that these results were from experiments with a single collection. The general trends should be informative, but any conclusions should be tempered by the lack of more exhaustive experiments.

The retrieval performance of the local index with larger collections, indicated that relevant images should be retrieved in the top 100 results. Therefore, the user should be able to find images of interest in the first few pages of results. With smaller collections the larger neighbourhood sizes gave better results. However, as the size of the collection was increased the performance of the various neighbourhood sizes tended to the same level. Indeed, the smallest locality of just 0.1% of each dimension gave the best

[4] 1,024,000 elements × 113 dimensions × 8 bytes × 1%

precision at 100 for collection sizes over 256,000 images.

In terms of search speed, the test system could search up to four million images in an average time of just over one second. This was on a fairly normal home computer using a 512 dimensional colour feature. When the feature database could not be held in memory, the experiments showed that for a 1% or 0.1% neighbourhood, the bottleneck in search speed was the seek time of the disk. The test implementation required that for each dimension a particular page needed to be found on disk. Once this point in a file had been found retrieval of adjacent blocks was much quicker. As the size of local neighbourhood was increased, the amount of data needed to be transferred from disk also increased. For a 10% neighbourhood this becomes the dominant factor influencing speed.

Further improvements in search time could be achieved by distributing the index over several disks and processors. The decomposition of features into independent dimensions means that the local index lends itself to parallelisation. In this way, the local similarity index could easily be scaled up for larger collections and multiple features.

Chapter 7

Conclusion

7.1 Summary of thesis achievements

The aim of this thesis was to research methods that allow CBIR to be scaled up to large collections, while maintaining retrieval performance. To do this three core components of CBIR were investigated: features, similarity functions and indexing. Throughout the work the nature of image retrieval was kept to the fore when developing methods or interpreting results. The task of image retrieval is full of ambiguity and approximations. Images can have multiple meanings, and the relevance of results is driven by a user information need that cannot be precisely defined. By exploiting the vagueness of the task, this thesis shows that combining straightforward methods produces excellent retrieval results.

In the first phase of work a detailed evaluation of texture features was carried out. Three exemplar texture features were chosen for experimentation: co-occurrence matrices, Gabor filters and Tamura features, based on statistical, signal processing and psychological paradigms, respectively. Each feature was adapted and tuned for the CBIR task. Some novel modifications to the Tamura texture features were proposed, evaluated, and found to be beneficial. As a consequence of this work we were able to determine general guidelines for constructing texture features for image retrieval. Interestingly, it was found that it is important to find the correct level of selectivity in features used for image retrieval. Refining features to be very detailed did not necessarily produce better retrieval results. For example, the Gabor filter is commonly used with a filter bank of 6 orientations and 4 scales for classification tasks. However, for image retrieval 4 orientations and 2 scales performed better. The important discriminators for CBIR turn out to be the micro-textures in images rather than the larger scale objects. In images of real scenes interesting textures occur at smaller scales. By using our own guidelines we were able to produce some robust and effective texture features that have performed well with a wide range of retrieval tasks.

The second part of the thesis focused on similarity measures for visual features. It was found that emphasising what was similar, and disregarding the very different, worked well. We referred to this type of approach as localising similarity. It characterises the two classes of functions investigated: fractional dissimilarity and local similarity.

The work presented in this thesis includes the first application of fractional dissimilarity functions to CBIR. These functions extend L_p-norm distances, by using values of $p \in (0, 1)$. This emphasises points that are local to the query in each dimension and reduces the noise from distant points. This makes the functions exhibit a contractive behaviour, which may reveal the membership of potentially interesting subspace clusters. A thorough evaluation across a range of collections and features showed that, for image search, fractional dissimilarity functions outperformed all other L_p-norm distance measures. Although an optimal value of p could be learnt for a particular collection and feature combination, we found that value of $p = 1/2$ gave robust performance in nearly all circumstances.

We also introduced a generalised local similarity function. This only considers a neighbourhood local to the query point in each dimension, thus selecting a subset of the least noisy data. It was found that, with commonly available test collections, this method achieved retrieval performance comparable to L_p-norm distances whilst using only 5–10% of the feature space. We also considered ways of selecting a subset of dimensions to search in. It was shown that an improvement in retrieval performance was obtained by choosing only the dimensions where the query has an atypical value. For histogram features this meant ignoring dimensions where the query value was zero, and for non-histograms it meant favouring the long tail of a skewed distribution. This finding allows the pruning of more data. In addition we proved that by imposing certain (nonrestrictive) constraints, a localised distance function can be made into a metric.

Our experiments with local similarity functions showed that it is the membership of the local neighbourhood that carries most of the similarity information. This result leads to the possibility of using a simple voting function as the basis for a fast search system. We implemented this type of local index and tested it with collections of over one million images. The results were gratifying in two ways. Firstly, it was found that, with large collections, contracting the size of the neighbourhood actually improves performance (measured as precision at 20 and 100 objects retrieved). A neighbourhood of one tenth of a percent of each dimension gave the best results. The average retrieval performance with one million images showed 2.3 relevant images retrieved in the top 100. This level of performance should satisfy a user given the tiny proportion of relevant images[1]. They are likely to find images that they are interested in within the first few pages of results. Secondly, by using this neighbourhood size it meant that the index was able to search four million images in just over one second. This satisfies the speed requirement for a real-time search (Section 3.3.3).

Overall, this thesis has investigated and brought together methods and ideas from three areas of

[1] random search would return 0.007 images in the top 100

content-based image search. We have shown that these approaches have a synergy. The combination of these strands of research has culminated in the implementation of a local similarity index that is an effective image search system for large image collections, in the order of several million images.

7.2 Implications for search index design

In practical terms, the end result of this work was a set of features and similarity functions that have been shown to perform well across a wide range of image collections. In this way we have produced the key elements of the CBIR tool chain. The experience gained through this work means that we can make some clear recommendations to implementers of query-by-example CBIR systems. The precise approach to take depends on the characteristics of the collection being indexed together with the user need. Our recommendations are:

- For a small collection, of less than 100,000 images, a full sequential scan can be run within a realistic user response time. A choice of a fractional dissimilarity function with $p = 0.5$ will give the best performance with most features. Histogram features should be normalised relative to the dissimilarity function.
- A larger static collection should be indexed using the local similarity index. The dimensions from the vertically decomposed features should be stored in memory or contiguously on disk as needed. A local region of around 0.1% should be used for collections of greater than 250,000 images. The number of dimensions used can be cut down by only considering those where the query has an atypical value, as defined in Section 5.4.5.
- A large dynamic collection should again be indexed using the local similarity index. However, the dimensions should be implemented using a data structure that allows insertions and deletions, such as a B+ tree. This will enable changes to be made to the collection more easily.

7.3 Limitations

There are several limitations on the results presented in this thesis. Many of these stem from the nature of content-based image retrieval and the construction of an evaluation environment.

Although we have made every effort to use a range of varied collections, see Chapter 3, we cannot test for the true randomness of generic image search. The results presented will always be biased by the collections used. This is particularly the case with the large collection used for the experiments in Chapter 6. This was a contrived collection, being built from two different collections. It is possible that the images from each collection are so different that they occupy different parts of the feature space.

This would have a consequential effect on retrieval performance. However, in mitigation, with these experiments, we are concerned with the trend of results with increasing collection size rather than the absolute performance level.

On a similar note, there is an argument against using diverse collections, in that they will make our methods too general, and thus end we will up with a mediocre result. If we concentrated on a particular domain then we could use much more sophisticated techniques and fit them to the problem. By aiming at a generic search task we may end up finding the “lowest common denominator”.

The use of average precision as a performance measure can be misleading. In some situations, changes in a few high performing queries can cause the averaged result to change a lot. Significance testing will capture this when comparing two results. With low average precision, a lot of individual queries may actually have zero precision. This situation was highlighted in Chapter 6. In some ways this stems from an inherent limitation with CBIR: queries and ground truth that are not visually coherent can never be satisfied by a content-based search.

Finally, in Chapter 6 we noted an important result: for optimal retrieval results, the proportional size of the neighbourhood decreases with increasing collection size. This behaviour may continue as the collection grows further, or this result may be specific to the experimental collection and its size. I feel that this point is very significant, and therefore have suggested future work to investigate this in the next section.

7.4 Future work

This thesis has covered a broad range of the content-based image retrieval landscape. Perhaps the most significant contribution has been to enable effective real-time search with collections of over one million images. I feel that the most interesting future work would be in areas that can improve and use this capability.

The work in Chapter 6 showed that as the collection size was increased to one million images the optimal size (as a proportion of dimension) of neighbourhood decreased. It would be interesting to see if this behaviour was repeated as the collection was increased further. If this is the case then this would be an important result as it would indicate that the complexity of the search is sub-linear. Investigating this would require careful construction of very large test collections.

Another important area is how to select dimensions based only on the information available in the query and collection. The approach taken so far was to select atypical dimensions in a relatively unsophisticated fashion. This worked well, but further gains in retrieval performance may be possible. Some preliminary work has been done using blind relevance feedback. A initial query was used to produce a candidate set of interesting images. These images were used to select dimensions that emphasise the membership of this set. These dimensions were then used to repeat the search. The initial results from this work were promising, but inconclusive.

There are several engineering approaches to speeding up search. For the test implementation, the factor limiting the speed of search was the time required for random disk access to each dimension. We have already suggested parallelisation of the index across several disks and processors to alleviate this. Another alternative is the use of solid state memory, such as flash drives. These do not have the mechanical components of a hard drive, hence have no penalty for random access. As this technology improves and becomes more affordable it may become a useful hardware solution. Although these approaches will almost certainly allow the further speeding up of search, they are, from a research perspective, not themselves that exciting. However, they would enable the experimentation to be moved to another level. The speed of retrieval should then be fast enough to allow complex methods to be applied to much larger collections than before.

One area worthy of study is how to best combine multiple features and images for a query. With the additional speed available it would be possible to employ many more techniques other than a straightforward linear combination of dimensions. This area is closely linked to a core component of CBIR that has been overlooked by this work: relevance feedback (Figure 2.2). Very large collections pose their own problems in this respect. For example, query expansion is a well known technique that has been effective for image search. Using it with the local similarity index will increase the number of localities utilised, this has the potential to either improve performance or simply add noise.

In Chapter 2 we discussed different search scenarios and how CBIR can be applied to these. I believe the future of CBIR will lie in the combination of content with context. That is, taking the visual relevance from a content-based approach and combining this with available contextual information[2] that is attached to or around the image. The local index will allow a content based search to be applied to a large collection. There are many alternatives for combining content and metadata searches. This could be done by either merging them into single index or combining the rankings from separate searches. The efficiency and effectiveness of the local index will allow these methods to operate on an entire collection rather than on small subset. Some interesting alternatives are given below.

- An appealing method would be to perform a metadata search and then use a large number of the top ranked results as an expanded query. In this situation the neighbourhoods could be kept small. This would ensure the search would be fast, and hopefully improve the results. It would require research into the best ways to execute multiple image queries. This method could be applied to the scenario in Chapter 2 where a collection has been partially annotated.

- Another approach that could be used to combine visual and contextual information is a *filter and*

[2] This refers to explicit information, for example filenames, URLs, EXIF tags, text around an image on a web page, etc., rather than implicit contextual information in the image itself such as scenery or cultural references.

refine method. This was mentioned in Section 6.3.2 as a way to combine two visual searches. It also has a huge potential for combining visual and metadata dimensions. The most straightforward method, from an implementation perspective, would be to use the local similarity index to perform a visual search on the complete collection. This would return a subset of the collection which could then be re ranked using metadata.

- Alternatively, the order of the searches in the filter and refine approach could be reversed. This case poses more questions. It would not be practical to build a new local index from the results of a metadata search. We would need to consider ways to use the existing data structure. One approach would be to incrementally increase the size of the neighbourhoods until a proportion of the metadata results were covered in each neighbourhood. This could be done simultaneously across all dimensions, so that when a target proportion was achieved in a number of dimensions, the search would be complete. This approach would mean that the size of the neighbourhood was chosen at query time dependent on the distribution of the metadata search.

Overall the prospects for content-based image search look promising. The ability to apply CBIR to large collections, as demonstrated in this thesis, opens up lots of exciting opportunities for further research that may improve the usefulness of CBIR.

Bibliography

C Aggarwal and P Yu. The IGrid index: Reversing the dimensionality curse for similarity indexing in high dimensional space. In *Knowledge Discovery and Data Mining*, pages 119–129, 2000.

C Aggarwal, A Hinneburg, and D Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the International Conference on Database Theory*, Springer LNCS 1973, pages 420–434, 2001.

L Armitage and P Enser. Analysis of user need in image archives. *Journal of Information Science*, 23 (4):287–299, 1997.

J Aslam, V Pavlu, and E Yilmaz. A statistical method for system evaluation using incomplete judgements. In *Proceedings of the ACM SIGIR Conference*, pages 541–548, 2006.

R Bayer and E McCreight. Organization and maintenance of large ordered indices. *Acta Informatica*, 1: 173–189, 1972.

N Beckmann, H-P Kriegel, R Schneider, and B Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD '90: Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, 1990.

M Belkin and P Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003.

R Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

M Bender, E Demaine, and M Farach-Colton. Cache-oblivious B-trees. *SIAM Journal on Computing*, 35(2):341–358, 2005.

J Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

S Berchtold, D Keim, and H-P Kriegel. The X-tree: An indexing structure for high-dimensional data. In *International Conference on Very Large Databases*, pages 28–39, 1996.

S Berchtold, C Böhm, and H-P Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. In *ACM SIGMOD International Conference on Management of Data*, pages 142–153, 1998.

S Berchtold, C Böhm, H Jagadish, H-P Kriegel, and J Sander. Independent quantization: An index compression technique for high-dimensional data spaces. In *International Conference on Data Engineering*, pages 577–588, 2000.

K Beyer, J Goldstein, R Ramakrishnan, and U Shaft. When is “nearest neighbor” meaningful? In *Proceedings of the International Conference on Database Theory*, pages 217–235, 1999.

E Bingham and H Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Knowledge Discovery and Data Mining*, pages 245–250, 2001.

C Böhm, S Berchtold, and D Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

A Bosch, A Zisserman, and X Munoz. Scene classification via pLSA. In *Proceedings of The European Conference on Computer Vision*, pages 517–530, 2006.

S Brin. Near neighbor search in large metric spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pages 574–584, 1995.

P Brodatz. *Textures: A Photographic Album for Artists & Designers*. Dover, 1966.

V Bruce, P Green, and M Georgeson. *Visual perception: Physiology, psychology and ecology*. Psychology Press, third edition, 1996.

C Buckley and E Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32, 2004.

F Campbell and J Robson. Application of Fourier analysis to the visibility of gratings. *Journal of Physiology*, 197:551–566, 1968.

G-H Cha. Bitmap indexing method for complex similarity queries with relevance feedback. In *Proceedings of ACM International Workshop on Multimedia Databases*, pages 55–62, 2003.

K Chakrabarti and S Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *The VLDB Journal*, pages 89–100, 2000.

E Chavez, G Navarro, R Baeza-Yates, and J Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

R Chellappa and S Chatterjee. Classification of textures using Gaussian Markov random fields. *IEEE Transactions on Acoustics Speech Signal Processing*, 4:959–963, 1985.

C Cleverdon, J Mills, and M Keen. *Factors determining the performance of indexing systems*. Cranfield, 1966.

P Clough, H Müller, and M Sanderson. The cross language image retrieval track (ImageCLEF) 2004. In *Fifth Workshop of the Cross–Language Evaluation Forum (CLEF 2004)*. Springer LNCS, 2005.

D Comer. Ubiquitous B-tree. *ACM Computing Surveys*, 11(2):121–137, June 1979.

W Daniel. *Applied nonparametric statistics*. Houghton Mifflin, 1978.

J Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision Research*, 20 (10):847–856, 1980.

P de Souza. Texture recognition via autoregression. *Pattern Recognition*, 15(6):471–475, 1982.

R de Valois, D Albrecht, and L Thorell. Spatial frequency selectivity of cells in macaque visual cortex. *Vision Research*, 22:545–559, 1982.

A de Vries, N Mamoulis, N Nes, and M Kersten. Efficient k-NN search on vertically decomposed data. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 322–333, 2002.

T Deselaers, D Keysers, and H Ney. Features for image retrieval – a quantitative comparison. In *DAGM 2004, Pattern Recognition, 26th DAGM Symposium*, number 3175 in Lecture Notes in Computer Science, pages 228–236, Tbingen, Germany, September 2004.

M Donahue, D Geiger, T-L Liu, and R Hummel. Sparse representations for image decomposition with occlusions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

D Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. Lecture at the AMS conference, "Math challenges of the 21st Century", August 2000.

B Dubuc and S Dubuc. Error bounds on the estimation of fractal dimension. *SIAM Journal on Numerical Analysis*, 33(2):602–626, 1996.

B Dubuc, S Zucker, C Tricot, J Quiniou, and D Wehbi. Evaluating the fractal dimension of surfaces. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 425(1868): 113–127, 1989.

P Enser and C Sandom. Retrieval of archival moving imagery - cbir outside the frame? In *International Conference on Image and video retrieval, CIVR 2002*, pages 206–214. Springer LNCS 2383, 2002.

H Ferhatosmanoglu, E Tuncel, D Agrawal, and A Abbadi. Vector approximation based indexing for non-uniform high dimensional data sets. In *CIKM '00: International Conference on Information and Knowledge Management*, pages 202–209, 2000.

R Finkel and J Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

D Francois, V Wertz, and M Verleysen. Non-Euclidean metrics for similarity search in noisy datasets. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 339–334, April 2005.

Getty Image Archive. http://creative.gettyimages.com (as of June 2005), 2005.

C Gotlieb and H Kreyszig. Texture descriptors based on co-occurrence matrices. *Computer Vision, Graphics and Image Processing*, 51:70–86, 1990.

N Graham. *Visual Pattern Analyzers*. Oxford University Press, 1989.

A Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

J Hafner, H Sawhney, W Equitz, M Flickner, and W Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):729–736, 1995.

F Hampel, E Ronchetti, P Rousseeuw, and W Stahel. *Robust statistics: The approach based on influence functions*. Wiley, 1986.

R Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

D Harman. Overview of the first Text REtrieval Conference. In *Proceedings of the Text Retrieval Conference*, pages 1–20, 1992.

M Hassner and J Slansky. The use of Markov random fields as models of textures. *Computer Vision Graphics and Image Processing*, 12:357–360, 1980.

D Heesch and S Rüger. NN^k networks for content based image retrieval. In *European Conference on Information Retrieval Research (ECIR, Sunderland, UK, Apr 2004)*, pages 253–266. Springer LNCS 2997, 2004.

D Heesch, P Howarth, J Magalhães, A May, M Pickering, A Yavlinsky, and S Rüger. Video retrieval using search and browsing. In *TREC Video Retrieval Evaluation*, 2004.

K Heller and Z Ghahramani. A simple Bayesian framework for content-based image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition – volume 2*, pages 2110–2117, 2006.

J Hennessy and D Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, fourth edition, 2006.

P Howarth and S Rüger. Evaluation of texture features for content-based image retrieval. In *International Conference on Image and Video Retrieval, CIVR 2004*, pages 326–334. Springer LNCS 3115, 2004.

P Howarth and S Rüger. Fractional distance measures for content-based image retrieval. In *27th European Conference on Information Retrieval*, pages 447–456. Springer LNCS 3408, March 2005a.

P Howarth and S Rüger. Trading precision for speed: Localised similarity functions. In *Proceedings of the International Conference on Image and Video Retrieval*, pages 415–424. Springer LNCS 3568, July 2005b.

P Howarth and S Rüger. Robust texture features for still-image retrieval. *IEE Proceedings on Vision, Image and Signal Processing*, 152(6):868–874, December 2005c.

P Howarth and S Rüger. Localised similarity functions for content-based image retrieval. *Submitted*, 2007.

P Howarth, A Yavlinsky, D Heesch, and S Rüger. Visual features for content-based medical image retrieval. In *Notebook of the Cross Language Evaluation Forum (CLEF, Bath, UK, Sept 2004)*, 2004.

P Howarth, A Yavlinsky, D Heesch, and S Rüger. Medical image retrieval using texture, locality and colour. In *Fifth Workshop of the Cross–Language Evaluation Forum (CLEF 2004)*. Springer LNCS, 2005.

P Howarth, J Magalhães, S Overell, S Rüger, and A Yavlinsky. SIRIL: A multidimensional browsing framework. In *MMKM Workshop: Multimedia Knowledge Management: Industry meets academia*, page 17, January 2007.

D Hubel and T Wiesel. *Brain and Visual Perception: The Story of a 25-Year Collaboration*. Oxford University Press, USA, 2004.

D Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the Sixteenth ACM SIGIR Conference*, pages 329–338, 1993.

P Indyk and R Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the 30th Symposium on the Theory of Computing*, pages 604–613, 1998.

Y Ishikawa, R Subramanya, and C Faloutsos. Mindreader: Querying databases through multiple examples. In *Proceedings of the Conference on Very Large Databases*, pages 433–438, 1998.

C Jacobs, A Finkelstein, and D Salesin. Fast multiresolution image querying. In *Proceedings of ACM Special Interest Group on Graphics (SIGGRAPH'95)*, pages 277–286, 1995.

A Jain and F Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 23(12):1167–1186, 1991.

F Jones. *Lebesgue integration on Euclidean space*, chapter 10, L_p spaces. Jones and Bartlett, 2001.

B Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8:84–92, 1962.

B Julesz. Textons, the elements of texture perception and their interactions. *Nature*, 290:91–97, 1981.

B Julesz, E Gilbert, L Shepp, and H Frish. Inability of humans to discriminate between visual textures that agree in second order statistics – revisited. *Perception*, 2:391–405, 1973.

N Katayama and S Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 369–380, 1997.

M Keen. Presenting results of experimental retrieval comparisons. *Information Processing and Management*, 28(4):491–502, 1992.

J Kekäläinen and K Järvelin. *Exploring artificial intelligence in the new millennium*, chapter User-oriented evaluation methods for information retrieval: a case study based on conceptual models for query expansion, pages 355–379. Morgan Kaufmann Publishers Inc., 2003.

J Keller, S Chen, and R Crownover. Texture description and segmentation through fractal geometry. *Computer Vision Graphics Image Processing*, 45(2):150–166, 1989.

B Kirkpatrick, editor. *Roget's Thesaurus of English words and phrases*. Longman Group UK Limited, 1987.

R Korfhage. *Information storage and retrieval*. Wiley, 1997.

W Kraaij, P Over, and A Smeaton, editors. *TRECVID 2006 Workshop*, 2006.

A Kundu and J-L Chen. Texture classification using QMF bank-based subband decomposition. *Graphical Models and Image Processing*, 54(5):369–384, 1992.

J Lawder and P King. Using space-filling curves for multi-dimensional indexing. In *Proceedings of the British National Conference on Databases: Advances in Databases*, pages 20–35. Springer LNCS 1832, 2000.

B Li, E Chang, and C-T Wu. DPF — A perceptual distance function for image retrieval. In *Proceedings of the International Conference on Image Processing*, pages 597–600, 2002.

J Malik and P Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America*, 7(5):923–932, May 1990.

B Manjunath and W Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

B Manjunath, P Wu, S Newsam, and H Shin. A texture descriptor for browsing and similarity retrieval. *Journal of Signal Processing: Image Communication*, 16(1):33–43, 2000.

B Manjunath, J-R Ohm, V Vasudevan, and A Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703–715, June 2001.

J Mao and A Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2):173–188, 1992.

D Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman, 1982.

B McCormick and S Jayaramamurthy. Time series model for texture synthesis. *Computing and Information Science*, 3:329–343, 1974.

R Miller. Response time in man-computer conversational transactions. In *Proceedings AFIPS Fall Joint Computer Conference*, volume 33, pages 267–277. AFIPS Press, 1968.

T Mitchell. *Machine Learning*. McGraw Hill, 1997.

H Müller, W Müller, D Squire, S Marchand-Maillet, and T Pun. Performance evaluation in content-based image retrieval: Overview and proposals. *Pattern Recognition Letters*, 22(5):593–601, April 2001.

H Müller, S Marchand-Maillet, and T Pun. The truth about Corel — evaluation in image retrieval. In *Proceedings of the International Conference on Image and Video Retrieval, London, UK*, volume 2383 of *LNCS*, pages 38–49. Springer, July 2002.

H Müller, W Müller, S Marchand-Maillet, T Pun, and D Squire. A framework for benchmarking in CBIR. *Multimedia Tools and Applications*, 21(1):55–73, September 2003.

H Müller, T Deselaers, T Lehmann, P Clough, and W Hersh. Overview of the ImageCLEF 2006 photo retrieval and object annotation tasks. In *CLEF workshop notes*, 2006.

W Müller and A Henrich. Faster exact histogram intersection on large data collections using inverted VA-files. In *International Conference on Image and Video Retrieval, CIVR 2004*, pages 455–463. Springer LNCS 3115, 2004.

J Nielsen. *Usability engineering.* Morgan Kaufmann, 1994.

P Ohanian and R Dubes. Performance evaluation for four classes of textural features. *Pattern Recognition*, 25(8):819–833, 1992.

T Ojala, M Pietikäinen, and D Harwood. A comparative study of texture measures with classification based feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.

M Ortega, Y Rui, K Chakrabarti, S Mehrotra, and T Huang. Supporting similarity queries in MARS. *ACM Multimedia*, pages 403–413, 1997.

R Picard and F Liu. A new WOLD ordering for image similarity. In *IEEE Conference on Acoustics, Speech and Signal Processing*, pages 129–132, 1994.

M Pickering and S Rüger. Evaluation of key-frame based retrieval techniques for video. *Computer Vision and Image Understanding*, 92(1):217–235, 2003.

M Pietikäinen, A Rosenfeld, and L Davis. Experiments with texture classification using averages of local pattern matches. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3):421–426, 1983.

K Porkaew, K Chakrabarti, and S Mehrotra. Query refinement for multimedia similarity retrieval in mars. In *Proceedings of the ACM International Conference on Multimedia*, pages 235–238, 1999.

J Puzicha, T Hofmann, and J Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 267–272, 1997.

J Puzicha, Y Rubner, C Tomasi, and J Buhmann. Empirical evaluation of dissimilarity measures for color and texture. In *Proceedings of the International Conference on Computer Vision — Volume 2*, pages 1165–1172, 1999.

T Randen and J H Husøy. Filtering for texture classification: A comparative study. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

A Rao and G Lohse. Identifying high level features of texture perception. *Computer Vision, Graphics and Image Processing*, 55(3):218–233, 1993.

Y Rubner, C Tomasi, and L Guibas. A metric for distributions with applications to image databases. In *IEEE International Conference on Computer Vision*, pages 59–66, 1998.

Y Rubner, C Tomasi, and L Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

Y Rui, T Huang, and S Mehrotra. Content-based image retrieval with relevance feedback in MARS. In *Proceedings of the IEEE International Conference on Image Processing*, pages 815–818, 1997.

Y Sakurai, M Yoshikawa, S Uemura, and H Kojima. The A-tree: An index structure for high-dimensional spaces using relative approximation. In *VLDB '00: International Conference on Very Large Data Bases*, pages 516–526, 2000.

G Salton. The state of retrieval system evaluation. *Information Processing & Management*, 28(4): 441–449, 1992.

H Samet. *Foundations of Multidimensional and Metric Data Structures.* Morgan Kaufmann, 2006.

N Sebe and M Lew. Robust shape matching. In *Proceedings of the International Conference on Image and Video Retrieval*, July 2002.

N Sebe and M Lew. *Principles of visual information retrieval*, chapter 3: Texture features for content-based retrieval, pages 51–86. Springer, 2001.

N Sebe, M Lew, and D Huijsmans. Towards improved ranking metrics. *IEEE Transactions on Pattern Analysis and Machine Learning*, 22(10):1132–1143, 2000.

T Skopal. On fast non-metric similarity search by metric access methods. In *International Conference on Extending Database Technology*, pages 718–736. Springer LNCS 3896, 2006.

A Smeaton, W Kraaij, and P Over. TRECVID 2003 — An introduction. In *TRECVID 2003 Proceedings*, pages 1–10, 2003.

A Smeulders, M Worring, S Santini, A Gupta, and R Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

J Smith. Image retrieval evaluation. In *IEEE Workshop on Content-based Access of Image and Video Libraries*, 1998.

J Strand and T Taxt. Local frequency features for texture classification. *Pattern Recognition*, 27(10): 1397–1406, 1994.

M Swain and D Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

H Tamura, S Mori, and T Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, 8(6):460–472, 1978.

J-P Tarel and S Boughorbel. On the choice of similarity measures for image retrieval by example. In *Proceedings of the Tenth ACM Conference on Multimedia*, pages 446–455, 2002.

K Tieu and P Viola. Boosting image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

D Travis. *Effective color display.* Academic Press, San Diego, CA, 1991.

A Treisman. Features and objects in visual processing. *Scientific American*, 255(5):114–125, 1986.

M Tuceryan and A Jain. Texture segmentation using Voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):211–216, 1990.

M Tuceryan and A Jain. *Handbook of Pattern Recognition and Computer Vision*, chapter 2. Texture analysis, pages 207–248. World Scientific Publishing Co., 2nd edition, 1998.

M Turner. Texture discrimination by Gabor functions. *Biological Cybernetics*, 55:71–82, 1986.

A Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

J van Gemert, J-M Geusebroek, C Veenman, C Snoek, and A Smeulders. Robust scene categorization by learning image statistics in context. In *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, pages 105–112, 2006.

C van Rijsbergen. *Information Retrieval*. Butterworths, 2nd edition, 1979.

N Vasconcelos and A Lippman. A unifying view of image similarity. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 38–41, 2000.

VisTex Database. VisTex database. http://vismod.media.mit.edu/vismod/imagery/VisionTexture/, 1995.

E M Voorhees and D Harman. Overview of the eighth Text REtrieval Conference (TREC-8). In *Proceedings of TREC*, pages 1–33 and A.17 – A.18, 1999.

R Weber, H-J Stock, and S Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional space. In *Proceedings of the Conference on Very Large Databases*, pages 194–205, 1998.

T Westerveld and A de Vries. Experimental evaluation of a generative probabilistic image retrieval model on 'easy' data. In *Proceedings of the SIGIR Multimedia Information Retrieval Workshop*, 2003.

J Wezska, C Dyer, and A Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man and Cybernetics*, 6:269–285, April 1976.

D White and R Jain. Similarity indexing with the SS-tree. In *International Conference on Data Engineering*, pages 516–523, 1996.

F Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1(6):80–83, December 1945.

P Wilkins, P Ferguson, and A Smeaton. Using score distributions for query-time fusion in multimedia retrieval. In *Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 51–60, 2006.

H Wold. *A Study in the Analysis of Stationary Time Series*. Almqvist and Wiksell, 1938.

A Yavlinsky. Behold Image Collection. www.beholdsearch.com, 2005.

A Yavlinsky and S Rüger. Efficient re-indexing of automatically annotated image collections using keyword combination. In *Proceedings of SPIE Volume 6506. Multimedia Content Access: Algorithms and Systems*, 2007.

A Yavlinsky, E Schofield, and S Rüger. Automated image annotation using global features and robust nonparametric density estimation. In *International Conference on Image and Video Retrieval*, pages 507–517. Springer LNCS, 2005.

P Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.

J Yu, J Amores, N Sebe, and Q Tian. Towards robust distance metric analysis for similarity evaluation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 316–322, 2006.

D Zhang and G Lu. Evaluation of similarity measurement for image retrieval. In *Proceedings of the IEEE International Conference on Neural Networks and Signal Processing (volume 2)*, pages 928–931, 2003.