UNIVERSITY OF SOUTHAMPTON

Content Based Image Retrieval : Analogies with Text

by

Mike Westmacott

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

October 2004

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Mike Westmacott

This thesis describes research into an area of content based image retrieval (CBIR), that of feature indexing for the purpose of rapid retrieval. The techniques in this thesis draw from the field of text IR and demonstrate that individual image extraction algorithms can be optimised for use with an inverted index, which could lead to CBIR systems capable of sub-second retrieval times on collections of millions of images.

A novel global feature algorithm, QMNS, is presented, which is capable of capturing both colour and texture information in a spatially insensitive manner. Images are divided into regular patches from which dominant colour modes are derived using the mean shift algorithm. The RGB bi-modal colour space is quantised giving a set of labelled feature terms with associated frequencies and the terms inserted into an inverted index. Terms in the index are retrieved with a TF*IDF algorithm.

The performance of QMNS and the index is measured by comparison with an RGB colour histogram, an RGB CCV histogram, and the unquantised MNS features. Precision and recall results show that the indexed feature performs equally as well as the other algorithms for an image collection of photographic images. An analysis of the distribution of each type of feature term was performed, showing that Zipf's law holds in each case. Quantisation parameters for the algorithms were varied, demonstrating that a tradeoff exists between vocabulary size, the average precision, and the speed of retrieval.

This thesis indicates that the current generation of highly successfully text IR systems, which are implemented using inverted indexes, could provide the basis for very rapid image and multimedia retrieval. The optimisation techniques shown can be used for any quantisable feature, and allow retrieval to be performed without specialised comparison algorithms.

Contents

List of Figures

List of Tables

Acknowledgements

I dedicate this thesis to my father, my mother, and my family. Without their love I would not have had the opportunity to carry out this research.

I thank Paul Lewis as my supervisor who has shaped the work presented here, and offered much assistance during my time at the University of Southampton. I also thank those countless others who have provided help and ideas, at conferences, in the lab, the coffee room, and at the bar. Finally I thank HP for the excellent range of equipment they have provided the IAM research group.

This research could not have been carried out without the research funding provided by EPSRC (Studentship 0031080C), for which the author is very grateful.

Chapter 1

Introduction

1.1 'Finding Out About'

Rik Belew's book on search engine technology, entitled 'Finding Out About'[13], emphasises the cognitive aspects of information retrieval, and as such identifies a key aspect of the discipline. There are many applications where information retrieval is required in order to replace human operators with machines, and a highly desirable property for these machines to own is the ability to act as humans. When designing machines to replace humans it is therefore of prime importance to understand how a human would perform such a task.

Possibly the reason that as humans we are as intelligent as we are is our ability to find out about things - we are adept at the skills of knowledge acquisition, storage, transformation and dissemination. Using our senses of vision, hearing, touch, smell and taste we are able to interpret and understand our environment, and to determine, through beliefs about our environment and capabilities, appropriate actions that allow us to achieve certain goals that we may have.

Of these senses, our vision is perhaps the strongest and most acute. It is certainly that which presents us with the most information on a daily basis, through the recognition of objects around us and our inherent abilities in written communication. Our brains allow us to interact with our physical environment by taking visual stimuli and interpreting these as salient objects in a three dimensional world. We are performing a continual process of information retrieval as we look at scenes that surround us, comparing objects that we see with objects that we have memories of.

Whilst machines that have vision and intelligence comparable to our own are still a dream, we can use the knowledge we have about how we perform our own information retrieval to create machines that assist us in searching through the huge quantities of information contained in our libraries, offices, hospitals, and of course, in the world wide web.

The field of CBR (Content Based Retrieval) will hopefully allow us to achieve some of these goals. There are, however, many aspects that must be overcome. Of particular importance to this thesis is the concept of 'information overload' - the overwhelming amount of information which is generated and stored on computer systems. The manifestation of this is the difficulty of attempting to locate desired information. It is the task of the information retrieval community to solve such problems. The now mature text retrieval field has already demonstrated that good quality retrieval from massive corpora is possible (consider the Google web search engine), and with advances in natural language processing is getting better.

This thesis aims to demonstrate to the reader that technologies developed for text IR (the inverted index and associated retrieval mechanisms) are applicable to image, and other multimedia, retrieval systems. Whilst not demonstrated, it is theorised that careful development and analysis of features[1], their information domains, and the distribution of 'feature terms' will allow a multimedia database to store millions of documents, and to respond to client's requests in sub-second time.

1.2 Aims and Objectives

The key aims and objectives of the research presented in this thesis are as follows:

- To demonstrate that sub-second content based image retrieval is possible from massive image collections:
  - Use available feature extraction algorithms, or develop a novel global colour feature extraction algorithm.
  - Develop a technique for transforming image features into a form storable in an inverted index.
- To provide techniques that allow analysis of the distribution of the feature space of different features:
  - Identify how changing a feature extraction algorithm's parameters change the distribution of points in feature space.
  - Investigate the effects of different techniques of feature space quantisation.
  - Present optimisations for indexed feature descriptors.

[1]Throughout this thesis the term 'feature', when used alone, will refer to either a semantic feature (for example *coarseness* is a textural feature), or to the data that is generated by a feature extraction algorithm for a particular image (also referred to as *feature vector* or *feature descriptor. The surrounding paragraph will determine the context.*

- To demonstrate whether feature term retrieval is flexible and extensible.
  - Show the potential for heterogeneous feature storage.

1.3 Thesis Structure

This thesis is split into the following chapters:

- **Chapter 2 - Fundamentals of Information Retrieval:** Background material on image features, feature indexing, the inverted index and retrieval evaluation.
- **Chapter 3 - Multimedia Information Systems:** Background on applications and architectures of content based retrieval systems, and examples of a number of old and new MIS.
- **Chapter 4 - INVISTOR - An Inverted Index Multimedia Search Engine:** This chapter describes the software applications that were developed for this research. These include the image processing API, the CBIR image indexer and query processor, the results analysis module and the index analysis module.
- **Chapter 5 - QMNS - The Quantised Multimodal Signature:** The QMNS feature algorithm forms the basis for a lot of the work in this thesis. This chapter describes how the features are extracted and represented, and presents tests which analyse the retrieval performance of QMNS and other colour features. The distribution of feature terms is investigated, and it is shown that it is the most common terms for QMNS which are the best for retrieving images. Work presented in this chapter is published in [142].
- **Chapter 6 - Generalised Feature Indexing:** Extending the work from the previous chapter, this chapter looks in more detail at the distribution of feature terms, how changes in the underlying QMNS parameters effect this, and how to optimise the index by pruning terms which are not good discriminators. Work presented in this chapter is published in [141]
- **Chapter 7 - Conclusion:** The results of the research are outlined, and the aims and objectives listed above are shown to have been fulfilled by this thesis, presented as relevant contributions to the field of content based retrieval. Future work is discussed in the context of the work that has been completed.

Chapter 2

Fundamentals of Information Retrieval

2.1 Introduction

The use of computers to either assist, or replace, human operatives is dependant on a thorough understanding of the duty that their counterparts must perform - current technology is a long way from the ideal of generic, adaptive, machines capable of any task. The type of task will determine many of the functional requirements of the computer system which must be designed carefully from the outset.

The field of computer vision and image understanding encompasses tasks where a computer is required to take data in the form of images and motion video[1] and extract salient information from it. There are two main application areas for such systems: Those that are for autonomous systems, requiring no interaction from an operator, and those where the system provides a (possibly) interactive service for an operator.

Both applications require that the system is capable of analysing an image (or a sequence of images in the case of video) and extracting succinct information that may allow for discrimination of objects and structure within that scene. They may also be required to apply knowledge they have to identify, and reason about, those objects. This process will therefore need to involve activities such as delineation of regions within the image, in order to determine object boundaries, and the comparison of the image, or parts of it, with other known objects and images.

It is this process, where the content of images are compared to find similarities (and hence similar objects), that forms a core part of computer vision[2], and is the topic of

[1]Both image and video media may originate from sources which are non-visual, such as x-ray, radar, or sonar.

[2]Cognition and intelligence form the counterpart here, allowing a system to reason about similar objects, and to then label with attributes those objects.

this thesis. The broad topic is known as Content Based Retrieval (CBR), and is a key part of a wide variety of subjects ranging from text retrieval to digital libraries. The following sections of this chapter introduce these concepts and the seminal, and state of the art, techniques that researchers have developed over the past few decades.

The second section introduces the processes involved in CBR and some of the research projects and real-world applications that use it. The third section looks at the key entities that are involved in one form of CBR - document based retrieval - and how these are used to rate the similarity of images and other media. The fourth section looks at text retrieval, which is itself a form of CBR, and the techniques that allow massive databases of text to be searched incredibly rapidly. The next section moves the focus back to image retrieval, and reviews image processing algorithms for extracting low level features. Sections six and seven are about the indexing of these features, and present some of the core concepts that shall be explored throughout this thesis. The last main section discusses how the quality of results produced by CBR systems should be evaluated, in order to compare algorithms and systems. Finally the chapter concludes with questions about the future of CBR and presents the hypothesis again, this time in the context provided by this background work.

2.2 Content Based Retrieval

Content Based Retrieval (CBR) is a technique that will retrieve documents from some form of store, or archive, such that the retrieved documents satisfy the informational requirements of a given query. This process typically involves extracting information from media in the query document, and comparing this with the information known about the media and documents in the archive. CBR results are normally ranked according to how 'similar' each document is to the query. Text IR is a form of CBR - documents containing the same, or similar, terms as the query string are returned, usually ranked by a function of the frequency of the matching query terms.

As human beings we perform a form of CBR on a daily basis in almost all activities. Every time we look at something we are comparing the visual stimuli from our eyes with experiences we have in memory in order to find the best match. The processing capabilities of our brains, however, are significantly more complex than current CBR techniques, and we have the ability to associate much higher level concepts with stimuli than a computer can. CBR is currently able to provide a measure as to how similar two objects are - but it cannot say whether, for example, one picture of a cup shows the same cup as another picture.

Exactly how we perform such actions is still unknown, but there is extensive research into how we represent such associations between stimuli and experience in our minds. Semiotics, as founded by the linguist Ferdinand de Saussure (1857-1913) and Charles

Peirce (1839-1914), is the study of signs. Too complex a subject to describe here, it provides a foundation for describing communication in terms of language and culture. Put very simply, a sign is some representation of an object, or concept, in the real world. The word CAR is a sign for an object that, in our culture, is deemed to be a car. The pixels that constitute the area in an image corresponding to a picture of a car are also a sign for car, and together are called a signature of the car.

Given that there are an almost infinite number of image based signs for objects and concepts (let alone the other media in which signs exist, such as audio), it is unsurprising then, that sign based retrieval is a difficult task. As humans we have the intelligence to interpret our stimuli as signs very easily, but with computers we must compare a signature that we wish to identify with all other known signatures. We assume that if we find two similar signatures then the two objects that they signify must also be similar. It is these ideas that form the basis of CBR.

2.2.1 Applications of Computer Vision and CBR

The applications for which a vision and CBR system could be useful are endless. The following list includes some of the more typical applications, and research projects that have demonstrated them:

- **Medical Imaging:** Medical imaging systems - PACS (Picture Archiving and Communication Systems) - are designed to provide imaging support for clinical use. Originally defined by Huang in [64], these MIS allow operators to archive and retrieve the many different modes of image that are acquired by devices such as X-Ray, MRI and CAT scanners. These systems have to cope with huge quantities of image data, which may be requested from remote sites, and so the storage and network functions of the system must be designed appropriately. CBR in PACS often involves image registration, the process of mapping 3D data from images of one source to those of another.

- **Web Retrieval:** Web based retrieval would provide access to the vast multimedia content available on the Internet. Such services would be provided by search engines such as Google, or by content portals such as Yahoo and MSN.

- **Digital Archives:** Many institutions have large archives of books, artwork, and other physical artifacts which would benefit from the advanced cataloging and retrieval functions of a computer system. Digital archives (such as those developed for the Artiste and Sculpteur projects) provide access to archive material that has been digitised and stored. Once the information is available on computer, analyses may be performed to extract more data, or to cross-reference. The Artiste project employed techniques such as colour and texture analysis and crack detection and classification to assist art researchers, restorers and enthusiasts in their work. Sculpteur extends the Artiste system by providing digitising and retrieval of 3D models.

- **Manufacturing Inspection:** Components can be checked by vision systems as they come off assembly lines. In this case the system may be looking for the presence, or absence, of elements in the component, or it may be measuring elements to check that they have been produced correctly.
- **Autonomous and Robotic Systems:** The military are the biggest investor in research in this area, developing surveillance equipment and weaponry that are capable of performing their functions autonomously. To this end DARPA (the American Defence Advanced Research Projects Agency) are offering a $1 million prize to the first team that can build a vehicle to complete a 150km course in the Mohave desert[3]. Deep sea exploration, mining, and space exploration are other examples of applications where the practicality of sending humans is a limitation.

All of these applications share four common stages:

- **Acquisition:** Images (and other media) must be acquired through some source. This may be through a camera, or other sensor, or it may involve using existing media. Images are simply arrays of pixels, and so any sensor capable of generating an array of discrete data is a candidate for image acquisition. In the case of non-visual sensors, such as x-ray or radar, post processing may be required to create an image suitable for viewing by people.
- **Analysis:** Once acquired the images are analysed according to application domain specifics: Salient information that will in some way assist the application is extracted. The x-ray component of a PACS may be required to delineate certain bone structures, or if the source is that of a security scanner it may attempt to identify hazardous or prohibited articles.
- **Storage:** The information extracted from the images, and maybe the original images, are required to be stored. Stored images and the extracted information can be catalogued, aiding future retrieval by showing positive examples of objects. The indexing of this information can then allow for rapid retrieval of the images in the archive.
- **Retrieval:** Without mechanisms for using the stored information, the application would be of limited use. This is perhaps the most important aspect, for which the other stages are only preparatory. Retrieval may be performed by browsing

[3]At the time of writing (April 2004) the furthest a vehicle managed has travelled on the course is 7km.

through catalogues, by providing an example of what is required, and even by sketching a composition.

Now a well established research field, computer vision systems are now enjoying a large amount of high profile commercial success. bio-metric scanners that identify people based on the unique pattern of their iris have been installed at Heathrow airport as part of a trial by Virgin Atlantic, British Airways, and EyeTicket Corporation to allow rapid boarding of flights for first class business customers [7]. Less highly regarded amongst the public are a new generation of speed cameras which read the licence plate of speeding cars and automatically issue a ticket.

A major reason for the increased commercial interest has been due to the ever reducing cost and size of electronic equipment required to provide such facilities. There is now much interest in personal assistant computers incorporating computer vision which could continually record what you do (via a camera worn in a suitable location, for example in glasses) and then analyse the imagery and record certain relevant pieces of information. They could, for example, record the text of a restaurant menu so that it could be recalled at a later date and emailed. Nokia have already developed software called Lifeblog [8] which organises data from a phone and its camera into a multimedia blog.

2.2.2 Paradigms of Information Retrieval

The method by which we search for information is crucial to the quality of the results gained, and will depend on the scope of results which we deem will answer our query. Some queries require a very specific answer - perhaps a single document - whilst others may be less definitive, being answerable by any document which has a topic which is the same as the query's. Below are listed some of the methods by which information may be retrieved.

- **Query By Example:** Perhaps the most common query form, QBE systems require an example which they compare with those in their database. They answer the question: "Find me documents *like* this".
- **Query By Sketch (or by Humming):** QBS requires slightly more interaction from the user, and allows them to generate a proxy document for which they wish to find similar documents. In image retrieval this could involve laying out coloured shapes to indicate where particular colours should occur in the desired documents [65]. In audio this might involve humming a tune to indicate the direction of the pitch (i.e. up, or down) [56, 18].
- **Browsing:** Browsing is perhaps the retrieval technique which is used by most people on a daily basis when attempting to search for anything (not on a computer), and is primarily an iterative process. The first iteration involves locating

the collection of items in which the desired object exists, for example, a particular book will probably exist in a library. The following iterations reduce the size of items through which to search. Within a library this will lead the searcher to the appropriate section (perhaps with the aid of Dewey decimal catalog codes) to row of shelves, then to a shelf and then to the books which are of interest.

- **Navigation:** The concept of retrieving information by navigation is restricted purely to the domain of computing, and is best illustrated by the web, and HTML documents. Normally a link on a web page is static - it provides a path from the current document to another. A generic link, when clicked upon, will provide a selection of documents to navigate to. The documents will be determined, not statically, but dynamically, by using the link anchor as a query passed to content based retrieval engine.

2.3 Documents, Features and Signatures

In information retrieval we are typically looking for an answer to some sort of question, presented in the form of a query. We will direct our query at an appropriate source of information, and with luck, and a well posed query, we hope to find that piece of information which we desire. Unfortunately, where information is stored digitally, and unlike 'querying' a human counterpart who is able to construct just the right answer, we are unlikely to find the exact piece of information we require.

Documents provide the core entity for information storage and retrieval. A document can be defined as a source of information that shares in common some topic, and that has been authored explicitly to record - or document - that concept. A book contains writings that all pertain to the same topic, or in the case of fiction, all pertain to the same, fictitious reality. An image contains representations of a particular scene, and a 3D model represents a physical dimensions of an object, be it real or conceptual.

This simple definition contrasts with the dogmatic dictionary definition, and does not reflect the richness of modern multimedia documents which are available. An alternative, more general, definition is that a document is a collection of signs all associated by a common topic, or concept (which is itself a sign).

2.3.1 Documents and Media

Documents have always consisted of a combination of text and image, and such forms of recording events as the hieroglyphics of the ancient Egyptians show the use of signs directly. Today multimedia authoring allows us to combine different media very easily into a document, and hypermedia allows us to control the order in which these media are viewed, providing a personalised view of the information.

Electronic documents come in a wide variety of types and formats. Most share in common the ability to combine different types of media at different locations, spatially and possibly temporally. Some formats allow for interaction with certain elements of the document In order to keep the model of a document simple such interactions shall be ignored, and a document will be considered to be a static entity[4].

Techniques for decomposing documents are necessary since information within them is spatially (and temporally in the case of video) organised - that is that pieces of information close together are more likely to be on the same topic than those that are far apart.

Many documents are composed of different types of media, and depending on the document format, the different media elements may be in different formats. Text may be plain ASCII, RTF, HTML or one of the many other formats. Images could be in bitmap or vector format, and possibly the most complex media type, video, comes in a myriad encodings, sizes, frame rates, and audio formats.

Any information system that deals with complex documents must have some method for parsing the media elements, so that they may be analysed and information extracted from them. How the system stores this extracted information, and whether it maintains document structure information (which may incur a high storage cost), is determined by the application of the system.

2.3.2 Features

The lowest level entity involved in information retrieval is the feature. A feature can be described as 'an entity which is a representation of some salient aspect of the document'. In text the features that are used correspond to characters, words, phrases, sentences, chapters and other constructs that we typically use in natural language. Such features have an immediate 'semantic value', corresponding almost directly to the concept that they are a sign for.

A good, and useful, feature is one which is both salient (important) and relevant in a given context. In the human vision system there are a limited number of cues with which we are able to interpret a scene presented to us. In particular our visual sensitivity to horizontal and vertical straight lines is particularly strong. It is a form of both conditioning and of inherent ability that we are able to accurately interpret and understand our surroundings using our vision. Inherent abilities in the human vision system seem

[4]Documents, such as those in the HTML format published on the WWW, which allow interactive navigation are still considered, since navigation does not change their authored content.

to take second place as we grow older and as our brains become familiar with visual cues they encounter.

As with biological vision systems there are a number of primitive features that are used in computer vision: colour, texture and shape. We can use the colour and texture in an image to segment regions that we believe might be salient objects, or we could trace edges that are found in convoluted representations of the image which may also give rise to shape. We can combine these primitives to form more complex features, or we can employ the information that they provide to focus on a particular area. When designing a feature for use in image retrieval there are two key factors which must be taken into account:

- *Context:* The information domains in which the stored documents and the user's query exist. In set terms this is an intersection of the topics that exist within documents of the corpus and the typical information requirement of the user issuing a query.
- *Retrieval Time:* The size of the corpus, at both intra- and extra-document levels, and the speed with which results should be delivered to the user. Where some applications may demand rapid response times by their very nature (such as real time medical and military systems), others suffer simply from the desired response time of their users - now!

Image and other multimedia features vary in the level of abstractness that they convey. Some image features (described further in section 2.5) simply provide a count of the number of pixels of a certain colour which are present, whilst others are capable describing much higher level concepts, such as the location, size and density of muscle tissue inside the heart. The first type of feature lies in the broad application domain, whilst the second is clearly in a very narrow application domain. In [125], and presented in table 2.1, Smeulders et al define the two application domains succinctly.

Their original table has been annotated to include some examples of typical applications. In general we know that the narrower the domain of the application, the more specific, and hence the less data each feature needs to carry. It is important to note that whilst a feature may describe a complex entity within a specific context, the feature itself may be very simple. In [145] the primary information required is the size, defined by width and breadth, of *Candida* yeast cells. The template matching used by the feature extraction algorithm is relatively complex - requiring a model of the size and shape of the cell and the optical density. The resultant data output is a very concise representation for each of the cells as a six place tuple containing: $< loc_x, loc_y, width, breadth, angle, confidence >$.

	Narrow	**Broad**
Variance of content	low	high
Sources of knowledge	specific	generic
Semantics	homogenous	heterogenous
Ground truth	likely	unlikely
Content description	objective	subjective
Scene and sensor	possibly controlled	unknown
Target application	specific	generic
Type of application	professional	public
Tools	model-driven, specific invariants	perceptual, cultural, general invariants
Interactivity	limited	pervasive, iterative
Evaluation	quantitative	qualitative
System architecture	tailored database-driven	modular interaction-driven
Size	medium	large to very large
A source of inspiration	object recognition	information retrieval
Typical applications	radar - esp. defence medical imaging	web document retrieval digital archives and libraries

TABLE 2.1: Narrow versus broad domain in image retrieval (Smeulders et al [125])

2.3.3 Signatures

The signature of a document is a representation of features contained within that document. A signature is usually restricted to one type of feature, and so a document will have as many signatures as types of feature extracted. Ideally a signature should be unique to a document, but it should also be able to identify documents that have similar content.

An important aspect of a signature is that it ideally needs to be a compact representation of the signified document. Small signatures may be stored easily and searched rapidly, but can only store a small, succinct amount of information - that which is extracted by the feature.

2.3.4 Document Similarity

Once document signatures have been generated and stored, retrieval of documents may be performed. In a QBE system the user presents a query document for which they wish to find other similar documents. A signature will be generated for the query which must be compared with the stored signatures, and the results are presented to the user in order of similarity. Measuring document similarity is a process of signature classification, and it is the features that determine which information the classes are based upon.

All signatures can be represented as either a single, or multiple points, within the n-dimensional space of the feature algorithm. As such, points that are close are similar signatures, hence represent documents that are similar - according to the information extracted by the feature algorithm.

Imagine that one has a particular query that needs to be answered. Consider an abstract information space where all the documents that answer the query are close together, and those that do not are farther apart. The feature that is best suited to answering the query is the one where the useful documents are also close together in the feature space. The feature, however, may not be any good for other queries.

There exist a great number of methods for measuring how far apart points are in Euclidean space. The following list presents some of the most often used metrics:

- **Ln Measures.** The L_n measures share the common form:

  $$dist_{Ln}(x, y) = (\sum_i (|x_i - y_i|)^n)^{\frac{1}{n}} \tag{2.1}$$

  The greater the value of n, the more a large difference in one component will differentiate the two points. As n is increased the function will approach $\max(|x_i - y_i|)$ (known in some literature as the *Chebychev distance). These measures are typically used where the values provided are interpreted as being points, and not vectors, in space.*

  - L_1 **(City block, or Manhattan Distance):** *The distance is simply the sum of the differences for each component, which in 2 dimensions is seen as two perpendicular lines.*

    $$dist_{L1}(x, y) = \sum_i |x_i - y_i| \tag{2.2}$$

  - L_2 **(Euclidean Distance):** *Possibly the most often used measure, the L2 distance measure returns the shortest line distance between two points:*

    $$dist_{L2}(x, y) = \sqrt{\sum_i (x_i - y_i)^2} \tag{2.3}$$

- **Correlation Metrics.** These metrics are used to compare vectors and return values between zero and 1, unlike the L_n metrics. They are used where the direction of the vectors being comapred is considered important.

  - **Uni-polar:** uc() ranges from 0 to 1, where 1 is identity. To use as a metric (where 0 indicates no difference between two entities) $1 - uc()$ must be used.
  - **Bipolar:** bc() ranges from -1 to 1, where 1 is identity. To use as a metric, either $1 - bc()$ or $\frac{1-bc()}{2}$ must be used.
  - **Uncentered (cosine):** This metric treats the two vectors as points and calculates the angle between them:

    $$dist_{cosine}(x, y) = 1 - \frac{xy'}{\sqrt{x'x}\sqrt{y'y}} \tag{2.4}$$

where x' and y' are the transposed forms of x and y.

- **Vector Set Comparison measures.** In the situations where a single signature is composed of multiple points a slightly different approach must be used. Points in the two signatures must be paired in some manner so that some predicate is satisfied in each case. Various names exist for this type of operation, including *joins* and *stable marriage matching*.
  - **Hausdorff Distance.** This measure compares each point in the source set to each point in the target, and then takes the distance from the point in the source that is farthest from any point in the target to the point in the target that is closest to it. The directions of the sets are then reversed and the maximum value taken as the distance. [32, 67].
  - **Stable Marriage Matching.** Given two equally sized sets of n objects, m and w, two nxn matrices are created where each row contains rank values for the corresponding object in the opposite set. The rank values are created according to some predicate, and each must contain a distinct value. In [54] Gale and Shapley state that there exists a valid and stable marriage for any combination of the rankings. A stable marriage occurs when there are no two pairs $\{m_i, w_j\}, \{m_k, w_l\}$ such that w_l is ranked higher than w_j by m_i, and m_i is ranked higher than m_k by w_l.
  - **Earth Mover's Distance:** This algorithm measures the distance between two distributions, calculating the minimal amount of work required to transform the source distribution into the target. It also provides for partial matching [114, 115].

As well as these generalised point and vector distance metrics, many CBR techniques employ their own algorithms. Notably there are a large number of metrics dedicated to the comparison of colour histogram features [129].

All of these metrics are only useful for comparing identical feature spaces, and so for a document composed of multiple media, or for a single media element with multiple features, a query will return multiple similarity values. Where the result of a query is to be presented to a human user these classifications must be combined into a single hit list.

2.3.5 Classification and Clustering

Simple similarity based searches using point comparison, as shown in the previous section, are a highly valuable tool, and are used to find document signatures which are like the query's. This is a simple form of classification - identifying which documents are in the same class as the query. By proactively classifying and labelling documents according to the signatures that they contain the probability and ease of finding documents that match a query are increased - if one document in a class matches then the others are in some respects guaranteed to also match. Immediately we have reduced the number of signatures we must search through.

Classification of feature vectors involves finding the clusters of points in the feature space, and assigning each point to one or more clusters. There are a large number of techniques which can be used, ranging from the minimal spanning tree algorithm [16], the K-means group of algorithms [84, 121] to the mean shift algorithm [51, 28, 41]. A thorough review of classifiers is provided in [71].

The spanning algorithm initially treats all points in the space as individual clusters, and then using a distance measure appropriate to the data, finds the two clusters in the space that are nearest each other and merges them. This process is repeated until either the required number of clusters is formed, or the clusters reach a maximum size (either in number or volume).

The K-means group of algorithms start by taking the n points in space that are farthest apart, with n being the number of desired clusters, and assigning these points as the initial clusters. Next the clusters are iteratively assigned the points that are closest to them, after each point is added to a cluster its centroid is recalculated. The overall effect is that the clusters will each move towards (but not reach) the centroid of all the points.

The mean shift is another iterative algorithm which will cluster points. The method looks at each point in the population in turn and examines other points that fall inside a window centred on it. The mean position of this subset of points is calculated, and the current point is shifted by that vector. The process is iterated for each point until the mean move is less than a designated threshold. The movement of the points will form clusters whose original points will not be outside the window centred about the cluster centroid.

2.3.6 Combining Classifiers

The author of a particular CBR algorithm that can classify 90% of documents correctly may believe that his technique is best and does not require the use of another algorithm that can only classify 20% of documents correctly (given the same sets of documents). If the correct documents classified by the second algorithm were part of the set correctly classified by the first algorithm, then the first algorithm is clearly superior and using the second can not improve the results. If however, the patterns of classification do not overlap, and the second algorithm can classify documents that the first can not, then there is a good case for combining the results of the two.

The field of classification crosses into many other different fields, providing mechanisms that allow decisions to be made when multivariate, or multidimensional data is found. There are two primary approaches for decision making:

- **Centralised:** A single inference mechanism receives all data inputs (that have possibly been pre-processed) and provides a decision so to the classification of the data input given predetermined classes.
- **Distributed:** Each data input is processed and a decision made. These decisions and then passed to a second level inference mechanism which produces a final decision.

The second approach applies to most systems since 1993, and is referred to as *classifier combination*. According to Kittler et al [71], there are two reasons for combining classifiers - efficiency and accuracy. Simple classification tasks may be achieved by using multistage classifiers, such that classification stops once an object has been identified to a desired level of certainty. More complex objects may require additional features and additional procedures and rules.

The key inference mechanisms that are used in basic AI and CBR are listed below:

- **Knowledge Rules:** Decisions are based on boolean algebra. Each condition is either fulfilled, or not - there is no allowance for errors in measurement. For example: (Condition A) AND (Condition B) $\Rightarrow$ Condition C
- **Fuzzy Logic:** A simple extension of knowledge rules where a condition is assigned a value that determines how well it has been fulfilled - there are various degrees of class membership. Membership functions are used to determine how much a particular object belongs to a class, based on a measurement. The function returns 0 for no membership at all, and 1 for complete membership. Various rules similar to those for boolean algebra exist for both conjunctive and disjunctive reasoning.
- **Probabilistic and Belief Approaches:** These techniques work on the ability to model the degree of belief that a particular event has occurred. Belief results from uncertainty, which may be due to a lack of information. Common models are the classical probability model, the Bayesian model, upper and lower probabilities model, and Demster's model. [60] introduces logical models of inference, and [39, 122, 124] provide background material for the Demster-Shafer approach.
- **Bayesian Networks:** An extension of the classical Bayesian probability model, where the interactions amongst knowledge entities are explicitly modelled. This makes Bayesian Networks simple to use and apply to other existing models. In [108] Pearl originally introduced the concept of the inference network.

- **Neural Networks:** Using the neurons found in many living organisms as a foundation, neural networks are trainable structures that accept data measurements and provide a classification of the data. Each neural node can have multiple inputs, and when the sum of those inputs exceed some threshold, the neuron triggers and sends messages on to other neurons. By applying known data to the inputs the network can be weighted so that it can classify unknown data. Neural networks are often attributed to the work of Frank Rosenblatt [113] who developed the perceptron neural network in 1957.

Content based retrieval is ultimately a classification activity. We wish to determine whether a document fits into the same information classes, which we have in our mind, as our answer. We may not know what these classes are, but we will know whether a document fits them. By creating a good model of classes of documents we are able to better group them so that when we find one document that answers a query (by the user's standards) we have also found many others.

2.4 Text Document Retrieval

A great deal of work has been performed in the field of Content Based Text Retrieval (CBTR), beginning with such early work as Dewey's decimal system. His system allowed books containing similar content to be grouped together, so allowing a primitive form of CBR. Of course, this technique only provides insight into the broad topic that a book is concerned with, and requires manual categorisation.

CBTR has a distinct advantage over other forms of CBR, for example image retrieval, because of the inherent semantic content of words. The word WORD[5] brings about ideas of language, groups of characters, news (e.g. *word of mouth*) and others. These concepts, whilst not being immutable, are defined by our language and culture, and maintain a very high level of semantic interpretation.

Text retrieval systems are commonly known as *search engines* and all operate on the same principles. The system is split into three core components: the indexer, the index and the retrieval engine.

The indexer has two functions: inter- and intra-document parsing. Inter-document parsing involves the acquisition and maintenance of files that will form the corpus of the searchable index. This component is best called a document management system, and may perform actions such as crawling the web and downloading files, controlling access to the stored documents and other ancillary functions. The intra-document parser will extract the actual terms and phrases that are to be indexed.

[5]The author shall assume the standard linguistic approach of using small caps to denote that it is the concept of the written word, and not the word itself, that is the object of reference.

The index is the data structure that stores information about documents and the words that occur within them. It can be simple, and indicate only the presence, or absence, of a word in a document, or it can be complex and store the location of each occurrence of a word, and even the locations of similes. Section 2.7 explains a particular type of index - the inverted index in detail.

The retrieval engine provides the user with access to the documents stored in the index through queries. A query could be a couple of keywords, or it could be a complete document. In either case the retrieval is performed in a similar manner - documents in the index are compared with the query, and, in general, those documents that contain the same words as the query are returned, in a ranked hit list.

2.4.1 Lexical Analysis

Intra-document parsing is performed by the lexical analyser, its task being to extract and count all the tokens within the document. In a plain text retrieval system, the document would at this stage be blocks of plain text with other media elements removed, and so tokens will be sequences of characters.

The types of sequences that could be extracted will vary depending upon the search requirements of the application's users. Some may require just words, or words and numerics, and mixed character sequences.

- **Words:** Regular words that appear to be part of a sentence. The case may be important for distinguishing names, and so tokens such as INTERNET and INTERNET could be treated as being different.
- **Numbers:** There may be a requirement to record numbers. This may appear to be a burden to the index, however the string token may be converted into a more compact integer or floating point representation as required.
- **Mixed:** As with numeric tokens, alphanumeric tokens may be indexed according to the search requirements.
- **Phrases:** Some retrieval systems may also need to index phrases that relate to the search application's information domain. Technical search engines, and applications requiring indexing of peoples full names are examples of such systems.

2.4.1.1 Parsing Process

The parsing process requires some form of grammar in order to operate. The grammar is specified by defining a finite state machine which will determine when a sequence of

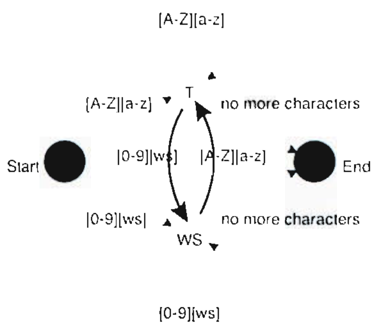

FIGURE 2.1: A Simple Text Parsing Finite State Machine, capable of extracting English words.

characters legitimately forms a token. A simple parser to extract only words will have a very simple grammar:

- Start
  - If current character is a letter, ignore the case, add to current token, and proceed to next character.
  - If current character is a number, whitespace or other then finish the current token. Increment the count for the current token, start a new, empty, token and proceed to next character.
- No more characters: Finish

This simple algorithm, illustrated in diagram 2.1, is a simple finite state machine that will parse text. The letters *ws* indicate a whitespace character, such as a space, or punctuation. Execution starts at the left hand node of the diagram, and depending on the character moves to either the top or bottom node. When execution is at the top node it will stay there until either there are no more characters left in the text, or a non alphabetical character is encountered. In either case on leaving the top node a new token will have been parsed.

2.4.1.2 Token Handling

When a new token is found in the text it must be handled accordingly. Not all grammatically valid tokens will actually be a desirable token, and may be discarded based upon the granularity requirements of the index. This helps to reduce the size of the index and makes it more concise.

These stop-words (also known as *noise words*) are a list of words which it is not desirable to store in the main index. These are words which are very common - words which it can be expected will occur frequently in every document. Such words do not aid retrieval since they do not *discriminate* between the documents in an index, a point originally noted by Luhn [82]. Some of these are *function words* that may be important to the syntactic structure of sentences, but carry little meaning on their own - for example A, THE, IT, and AND. Not all frequently occurring words are noise words, with the words TIME, WAR, HOME, LIFE, WATER and WORLD being among the top 200 words in English Literature [48].

The noise words are stored in a list known as a negative dictionary, and during the indexing process may be stored in a secondary negative index so that should a particular query require some of them, then they are available but are not affecting the performance and conciseness of the main index.

2.4.1.3 Morphology

Many of the words encountered in a text will present the same root meaning, but will have had syntactic pre-, or post-fixes applied depending on their exact meaning within the sentence. Stemming is the process of removing these surface markings. Since IR is normally only concerned with the presence or absence of a particular sign, the different words ENGINEER, ENGINEERS, ENGINEERING, ENGINEERED, and RE-ENGINEER can all refer legitimately to the sign ENGINEER, which invokes concepts of the design, construction, industry and of engines.

The lexical analyser will have certain generic rules to remove suffixes that denote plurality (-S, -ES etc), past tense (-ED), ownership (-'S, -S), and others, in the form of a context-sensitive transformation grammar [48]. This will be backed up by other rules for words which do not adhere to the grammar, for example - WOMAN/WOMEN.

2.4.2 Text Indexing

When complete, the lexical analysis will provide a list of terms, possibly stemmed, and categorised as noise, or non, and the frequency with which they occur in the text - the document frequency, d_f. Advanced systems may also contain for each word a list of the locations where that word occurs (the Wordnet system [45]), and pointers to similes (IBM's STAIRS system).

There exist three primary techniques for storing the information provided in this list - the inverted index, signatures, and bitmap files [119, 13], each having certain advantages and disadvantages. In [149] Zofel et al provide a good comparison of the first two approaches, by examining the effect of different types of compression of the different indexes.

- **Inverted Index:** Otherwise known as an inverted file, this data structure consists of a list of terms present in the text corpus. For each term there is a list of the

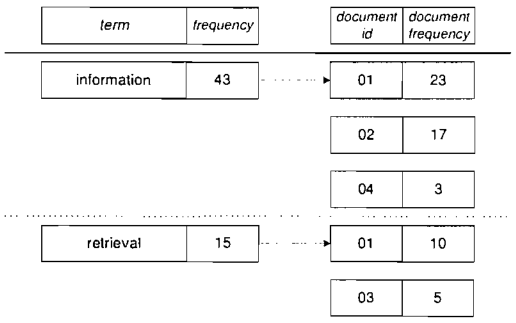

FIGURE 2.2: A Simple Inverted Index

documents in which that term occurs and the corresponding document frequency, and depending on the granularity of the index, additional information such as term location. Diagram 2.2 shows an example of a simple inverted index. Section 2.7 discusses this data structure in greater depth.

The inverted index is suitable in situations where the corpus is very large, or where additional information is required about each individual occurrence of a term. It is very fast to retrieve documents when there are only a few query terms, and in general slower for large numbers of query terms, but certain implementations (such as distributed indexes, see section 2.7.4) can allow document-document similarity to be calculated very rapidly.

This type of index suffers from poor update performance, due to optimisations which are applied to the postings list - which can be very long in a large corpus. Typically a secondary update index is used to store incremental changes to the index, which is then merged with the main index periodically.

- **Signature Files:** This type of index employs hashing techniques to create a signature for a document which allows the calculation of the probability that a particular term will occur within a document in the index. After lexical analysis is complete each term is assigned a binary hash string. One technique for generating the hash is to use three different functions that return a value that is at most equal to the number of bits in the hash string. This will mean that the string can have 1, 2, or 3 bits set in it.

  Even when a hash collision occurs, there is no need to change the hash because the second stage of indexing involves combining all the hash strings for all the terms in a document to produce the signature. The hash strings are or'd together to produce a final signature. To test if a term is present the hash string for the query term is generated and if all the relevant bits are set in the signature then it was probably present. Other combinations of term hash strings might have created the same pattern as the query term, and the probability of this can be calculated. This probability will be a function of the number of the number of documents and terms in the corpus, the length of the hash string, and the hashing function.

If the signature is of a suitable size then many documents can be eliminated rapidly from a query because the correct bits aren't set. The other documents must be scanned to determine whether the term does actually occur. Various enhancements have been developed for signature files including *bit slicing* where the bits that correspond to the same terms are stored together on disk - a technique which makes the index very similar to an inverted file.

- **Bitmap Files:** These are the simplest way to store document-term information, and consist of large bit strings with as many bits as there are documents in the corpus. Presence, or absence, of a term in a document is shown by setting the bit for the document at the appropriate document's index in the term's bit string. These bitvectors are perfectly suited to Boolean retrieval since the vector for each term in the query is combined using the appropriate Boolean operation. The result is a bit vector which indicates the documents that satisfy the query.

  Since the number of documents could be very large each bitvector can be compressed, however a compressed bitmap file is very similar to an inverted index, and the performance advantage is lost. Bitmap files are not regularly used because of the sheer size that they occupy - N documents * n terms bits.

2.4.2.1 Term Weighting

Not all terms in a document will be related to the subject that a document is about, in particular the stop-words. There will, however, be certain words in the document which are definitely about the subject, for example the title, the introduction chapter, or the abstract. If we know about the structure of a document, then we can apply a weighting to these elements as they are inserted into the index (we could apply a weighting later, however this weighting does not take into account their location structurally in the document). To make these terms more important we could choose to count each term in the title 10 times, and perhaps each term in the abstract 5 times, or apply a weighting that is some fraction of the size of the document.

2.4.3 Text Retrieval

The previous section illustrated some of the data structures that are commonly used in text retrieval and the advantages and disadvantages that they have. Aside from the differences in retrieval performance their purpose and uses are nearly identical. The process of retrieval involves looking for terms that are present in the query and finding which documents also share those terms.

The signature and bitmap techniques offer some limitations in querying since they do not store the document frequency for terms, and as such two documents that contain the same query terms, regardless of frequency, will be judged to be equally similar. The inverted index, however, is able to rank documents based on the number of times that the query terms occur. A thesis often attributed to Luhn's seminal work [82] is that a term must be relevant to a document's content if that term occurs often within the document.

As such, a document that contains a query term once will be rated lower than another that contains it many times. This effect is enhanced by lexical operations such as stemming since it is the stem and not the word that is counted.

Another consideration in text retrieval is that a document that contains frequently a term that occurs rarely within the corpus as a whole must almost definitely contain content that is related to that term. From this it can be deduced that the terms that are best for retrieval are neither those that occur frequently within the corpus, nor those that occur rarely, but those that lie in the midrange. By removing noise words the shape of the curve is changed so that it becomes close to an exponential curve.

This premise lies behind much of the thinking that forms the basis of some of the retrieval techniques in the following sections.

2.4.3.1 Boolean Retrieval

Documents are retrieved according to the presence, or absence of terms. An expression is formed using Boolean algebra to determine which documents match, for example:

information AND retrieval AND (bitmap OR signature)

Enhancements to the simplest type of Boolean retrieval, which only determines whether documents match, or do not match, the query, include term-document frequency ranking and fuzzy Boolean matching. The first technique ranks documents according to a function (normally the sum) of the frequencies of query terms, so the more often a document contains a query term, the higher its ranking. Fuzzy Boolean matching again employs the term-document frequency, but uses fuzzy Boolean membership functions (see section 2.3.6) where the frequency of the terms determines the size (and hence fuzziness) of sets. Lee et al provide a comparison of these two extensions to Boolean retrieval in [78].

2.4.3.2 Weighted Retrieval

Much like extended Boolean retrieval with only conjunctive operators, weighted retrieval uses the term-document frequencies available in the index. Since this technique relies on these frequencies it is only a viable option for an inverted index. For each query term the term frequencies are retrieved from the appropriate postings list in the index.

The simplest scoring function sums the term-document frequency tf_{ij} (for further mathematical definitions, refer to the mathematical section of the glossary):

$$score_j = \sum_{\forall\, i \in (Q \cap T_j): i \neq 0} tf_{ij} \tag{2.5}$$

This is a very simple form that can be used for keyword queries, since it does not take into account the frequency of query terms. The general form of a weighted retrieval extends equation 2.5:

$$score_j = \sum_{\forall\, i \in (Q \cap T_j): i \neq 0} qw_i \cdot qr_i \cdot w_{ij} \cdot r_{ij} \tag{2.6}$$

Here qw_i refers to a *weight* applied to the query term i, and qr_i is a *relative term frequency* value. The other values w_{ij} and r_{ij} refer to the weight and relative frequency for term i in document j. The relative term frequency values are calculated from the term frequencies, and can simply count the presence of a term,

$$r_{ij} = 1 \tag{2.7}$$

or the actual frequency in the document,

$$r_{ij} = tf_{ij} \tag{2.8}$$

or a normalised frequency, shown here according to the number of terms in the document, tn_j,

$$r_{ij} = \frac{tf_{ij}}{tn_j} \tag{2.9}$$

and can incorporate a logarithm to minimise the effect of large frequencies,

$$r_{ij} = 1 + log_e tf_{ij} \tag{2.10}$$

Values for the relative query term frequencies, qr_i, can be calculated similarly. The document frequency can be used to promote documents that contain many occurrences of a query term which occurs in few other documents. According to Luhn and Zipf's relevant hypotheses (section 2.4.3), this weight needs to be high when the document frequency is low, and so the inverse is used. Typical calculations are:

$$w_{ij} = log_e(1 + \frac{N}{df_i}) \tag{2.11}$$

$$w_{ij} = log_e(1 + \frac{tf^m}{df_i}) \tag{2.12}$$

and

$$w_{ij} = log_e(\frac{N - df_i}{df_i}) \tag{2.13}$$

where N is the number of documents in the corpus and tf^m is the largest term frequency in the index. Again, values for the query term weighting, qw_i, can be calculated similarly. Whenever the term frequency tf_{ij} increases monotonically, and the document frequency df_i is used in a monotonically decreasing way (as in equations 2.11 to 2.13), the similarity heuristic is known as TFxIDF - Term Frequency times Inverse Document Frequency.

There are many different combinations for the weightings and relative term frequencies, however a commonly used combination for keyword queries is:

$$\begin{aligned} qw_i &= log_c(1 + \frac{N}{df_i}) \cdot 1 \\ w_{ij} &= r_{ij} = 1 + log_e tf_{ij} \end{aligned} \tag{2.14}$$

Here the relative query frequency qr_i is 1, since query terms should only occur once, and is multiplied by the inverse document frequency of the term. The term weighting is equal to the relative term frequency.

2.4.3.3 Vector Space Models

If we consider that V, the vocabulary, contains all possible terms, and create an n-dimensional space, where n is equal to the number of terms in V, then vectors in this space can represent documents. We can then use the metrics defined in section 2.3.4 to measure the similarity between a query and a document. Using the Euclidean, L2, distance we have:

$$score_j = \sqrt{\sum_i (qw_i \cdot qr_i - w_{ij} \cdot r_{ij})^2} \tag{2.15}$$

If the relative term frequencies have been normalised then this method will compensate for different length documents, but if they haven't then documents closer in length to the query will automatically be favoured. In most situations the direction of the vector is more important, and we can measure the difference in direction between two vectors using the *cosine rule*. For two vectors X and Y the angle θ between them can be calculated:

$$cosine(X,Y) = cos\theta = \frac{X \cdot Y}{|X||Y|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} \tag{2.16}$$

Substituting $x_i = qw_i \cdot qr_i$ and $y_i = w_{ij} \cdot r_{ij}$ gives the cosine similarity heuristic for documents.

2.4.4 Advanced Text Retrieval

The techniques for text retrieval presented above provide a common framework that is adopted by most text retrieval applications. There are a number of advanced features that have been developed to extend this framework.

- **Relevance Feedback:** A query specified by a user may not always be well formed, or the terms used relate to a synonym that is semantically related, and so the document desired is not returned, but others that are of some relevance are. By allowing the user to indicate which documents that have been retrieved by a query are genuinely relevant, and those that are not, the retrieval system can adjust the vector of the query by moving it towards the centroid indicated by the relevant vectors.

  Conversely it is also feasible to adjust the index so that documents are moved toward common query vectors. If an index is searched repeatedly using similar queries the documents considered relevant to that query by users will be moved towards the query centroids. Documents that are rarely retrieved or are marked non-relevant move to the fringes of the query space.

- **Spatially Sensitive Retrieval:** Quite often the terms in a query will form a phrase, or part of a sentence, or paragraph, that the user wishes to be present in the target document. In such a situation a document that contains the query terms many times, but very spread out, should be rated less relevant than one which contains the terms once only, but clustered very close together (and even better, in the specified order).

  To support such a query the index will need to record term location information, and depending on the granularity and target size of the index will record the location of each occurrence of the term. The character, term, sentence and paragraph position within the document are all viable location descriptors which could be included in the index.

- **Adaptive Information Retrieval:** Manual indexing, and categorisation, of documents provide superb resources that are coherent and effective in use. The number of corpora which have actually received thorough attention is, however, very low. Automated document classification tools attempt to perform the same task for digital collections.

  Because there are some collections that have been manually indexed they provide data with which to design, test and train automated systems. Classification systems attempt to cluster document vectors in term space. A query that is within the cluster will then retrieve all the documents in the class.

  All of the techniques shown in section 2.3.6 are applicable here.

- **Content Based Navigation:** Described briefly in section 2.2, CBN provides linking in a hypertext environment such that selecting a word in a document will provide a list of links to other suitable documents in which that word occurs (derived directly from the postings in the index). In order to reduce the number of documents shown the system may only display those that are in the same document cluster (see previous item) and contain the same term.

This section has introduced the very basics of text retrieval, which is a very broad topic. Section 2.7 examines the inverted index in greater detail.

2.5 Image Features

The concept *feature*, as introduced in section 2.3 denotes some element of an image such that it is useful for discriminating between images. For each application where content based retrieval of images is required, features that provide a high discriminating power should be selected. Some applications may require more than one feature for identification of objects, and could have a highly specific feature , or one that may be very general. Medical imaging requires features that are highly accurate and are capable of stating a confidence factor for features extracted [19, 140]. General purpose retrieval for large databases of unrelated images requires an approach that is less exact and must take into account the image as a whole.

2.5.1 Colour Features

Colour forms a significant part of human vision, and without it many everyday tasks would prove very difficult. We are able to distinguish objects very effectively based on colour alone. Lai and Tait [131] have used our perception of colour to directly index and navigate around retrieved images. This was extended to create the CHROMA [76] image browsing and navigation system. Mojsilovic presented a vocabulary and grammar of colour patterns in [95] obtained through subjective experiment and demonstrated through colour features.

The simplest form of colour feature is perhaps the colour histogram [129] as created by Swain and Ballard. The histogram is created by a quantisation of the colour space, and typically each axis is divided by 4, so that in an RGB histogram there are 64 separate partitions, or bins, in the colour space. Each of the bins is assigned a label, and for every pixel colour that falls into that bin the bin value is incremented. The resultant signature is a 64 dimensional vector that can be compared to others by simple Euclidean distance, or by more complex methods like the quadratic formula [46] and histogram back projection [129].

The image histogram is a very flexible construct and has been used extensively for image retrieval. The Colour Coherence Vector (CCV) [107] takes a two stage approach to generating the signature. Firstly the image is segmented by reducing the number of colours in the image, and then the pixel values that are in regions sized above a particular threshold area are stored in a *coherent* vector, and those that are under the threshold size are stored in an *incoherent* vector. Results are presented that show that the CCV is better at retrieval than the simple histogram.

Different images of real life objects will rarely show the same colours in a histogram. This is due to changes in surroundings, ambient and directed lighting, orientation and scale. It is particularly necessary to attempt to make retrieval invariant to illumination, and colour features that are immune to lighting changes are much sought after. Funt and Finlayson take the original work by Swain and Ballard and histogram the ratios of RGB values to provide colour constant indexing [52]. Drew et al [42] investigate the transform that the RGB channels undergo when illumination is altered and demonstrate a linear transformation that may be applied to features to make them illumination invariant.

2.5.2 Textural Features

Texture in images is a function of the intensity, or colour, difference between pixels in an image. Tamura et al [130] were perhaps the first to extract textural features in images that the human visual system is highly tuned to. They showed that coarseness, contrast, directionality, line-likeness, regularity and roughness are attributes that we are sensitive to, but they could only implement sensors for the first three.

The characteristics of texture that Tamura highlighted have proved to be fundamental to textural analysis. The Fourier transform will turn a signal into representation of the frequencies within that signal, and for images the signal is the 2D discrete set of pixels. Not all signals make sense in the frequency domain, but the regularities and frequencies displayed by textures are ideal. A simple texture feature would involve histogramming the discrete Fourier signal of an image. The more advanced Wavelets [135] display better localisation and the resultant signal is generally smaller than the equivalent Fourier.

2.5.3 Shape Features

Shape features are extracted from regions in images that have been segmented in some way. Shapes can be represented in many ways, from chain codes that follow individual pixel outlines to splined vector formats. Simpler representations can also be used, such as the quad-tree which is a binary representation of a $2^n x 2^n$ grid, or descriptors that describe the boundary of an object.

2.5.4 Hybrid Features

Hybrid features use a combination of techniques to form a retrieval system. Pecenovic takes colour histogram and wavelet texture descriptors and incorporates them into a single index using LSI (Latent Semantic Indexing) [110, 109] and presents encouraging results. In [138] Wang uses a variation of the TF/IDF text term weighting technique with features - RF/IPF (Region Frequency / Inverse Picture Frequency) to index regions within images, and combines this with text retrieval. Paek also uses a similar approach in [106], however his system requires training in order to classify regions and objects within a scene so that they may be labelled and introduced to the index.

2.6 Feature Indexing

A key problem with content based retrieval lies in the process of searching through signatures to determine their similarity with the query. In a large collection this $O(n)$ operation can take a long time, and be compute intensive. By indexing the signatures it is hoped that the retrieval speed can be significantly improved, and also that the size of the signatures is reduced.

2.6.1 Representation of Features

Any signature can be represented as a point, or points, in a Euclidean space of some number of dimensions. In the first case document comparison is a simple matter of point to point comparison, however multiple feature vectors require a more complex approach such as the stable marriage matching algorithm [88].

2.6.2 Multi-dimensional Indexing

Feature vector similarity may be calculated as the distance between two points, or the distance between a point and the centre of a volume. Other tests may be required, such as whether a point is inside a volume, or whether a volume is inside a volume. In [53] Gaede, et al, gives a clear review of what is required from a multidimensional database, and then introduces a large number of techniques for storing and retrieving such data.

The majority of the data structures designed to store multi-dimensional data use the approach of dividing space into sections, variously called buckets, cells and bins. Each cell may be subdivided when it has too many entries, or the space may be completely re-divided. By searching through each cell descriptor first it is possible to determine which cells do, or do not, contain useful data, so that they may be searched thoroughly. Many of the structures employ a hierarchical, or tree, structure, so that whole branches of cells may be eliminated from the search very rapidly.

The B-Tree (Balanced Tree) [11] is such a structure, but is designed to store only one-dimensional data. Each node in the tree has a lower and an upper bound for the number of children (descendants) that it can contain. A node can only be created if there are sufficient entries to go into it - until then the entries are held in other nodes on the tree. When a node reaches its upper bound it must be split - a procedure which may recurse through the whole tree. This guarantees that the branches of the tree are loaded fairly equally.

GRID files [102] subdivide multi-dimensional space using orthogonal grids. In two dimensions this can be seen as a series of infinitely long horizontal and vertical lines, splitting the space into rectangles. Each of these cells corresponds to a data-page (the smallest single unit retrieved or written in a single operation by a disk operating system) inside the GRID file, so it is the operating system that determines the number of entries in each cell. When a cell exceeds the size of its data-page it must be split by introducing another hyper-plane into the space - splitting any other cells that are orthogonal to the splitting cell. There are a number of variants on the GRID file, the Two-Level GRID file [62] where the first grid is a directory that points to a second level of GRID files, and the twin GRID file [66] which employs two different grids in the same space, points are allocated to the file that best accommodates them.

A multi-dimensional version of the B-Tree is Antonin Guttman's R-Tree [59] which uses hyper-cuboids as cells, stored in a balanced tree. The upper and lower bound for each dimension must be stored for every node, so the actual size of the nodes (not including actual data) becomes an issue for the data structure. This more complex tree requires significantly more complex algorithms for inserting and removing data, for splitting and merging cells when their bounds are breached, and for searching through the tree.

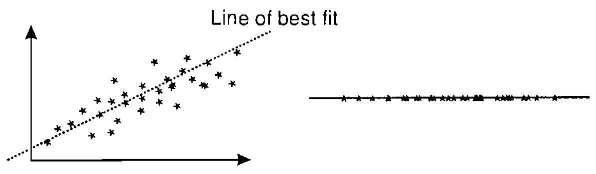

FIGURE 2.3: Dimensionality Reduction

Many variations on the R-Tree exist: the R*-Tree [12] which has modified insertion and splitting algorithms, the Hilbert R-Tree [69] that uses the integer value of the Hilbert space filling curve to locate points close to each other, and the distributed GPR-Tree (Global Parallel R-Tree) [50] which is designed to work on a CoW (Cluster of Workstations). Berchtold et al [17] take a different approach by pre-computing the distance between points, which demonstrates high efficiency. Their technique is also dynamic and allows points to be inserted (but not removed). Another variation of the R-tree is the vantage point tree [30] which automatically selects the appropriate start position and scope to search through in a tree structure. Yet another is the M-Tree [29] for which a generalized hyperplane approach is taken for splitting the N-space.

These are just a few of the data structures and algorithms, however there are many more in existence, each claiming to be more efficient at some aspect of multi-dimensional data storage than the others - and unfortunately there are few comprehensive reviews of the techniques, and no independent performance tests. Which algorithm to use will depend on the dimensionality, the quantity, and the distribution in space of the data to be indexed. Other factors such as the speed of insertion, retrieval, update, deletion and the target size of the data structure must be considered.

2.6.3 Dimensionality Reduction Techniques

Another solution which can be used in conjunction with multidimensional indexing involves reducing the number of dimensions in feature space. The principle behind this is that within the data there may exist correlations that would allow dimensions to be collapsed without much detriment to the information held within them. Figure 2.3 shows an example of a 2-dimensional cluster which would be suitable for collapsing into 1 dimension.

- **PCA - Principle Component Analysis:** PCA directly identifies the planes of correlation in the data through the generation of eigenvectors. The small eigenvectors correspond to the minimal variations in data (the axis perpendicular to the line of best fit in figure 2.3) and may be discarded. The result is a transformation matrix through which is applied the raw data and generates the lower dimensionality data. PCA is the most often used reduction technique [133, 70]

- **K-L transform:** The Karhunen and Loeve transform [139, 134] is nearly identical to PCA, but achieves the same results through a different set of computations.
- **Kohonen Self Organising Maps:** An ordered form of neural network, the self organising map [72] will adapt its weights by dimensionality reduction.

Because many multi-dimensional index structures do not operate well with high dimensional data, due to the overhead incurred by management of the bucket or cell structures, data is often reduced, using the techniques above, first. Ultimately there is a limit to the dimensionality of data before it becomes too expensive to process - hence the well known phrase 'the curse of dimensionality'.

2.7 The Inverted Index

Introduced in section 2.4.2, the inverted index has many different variations and optimisations. It is a very flexible data structure and is perfectly suited to parallelisation. As with many data structures, if it is to work well then it must be ordered, and in particular it must be kept ordered. The first sub-section here describes the nature of the term data that is stored in inverted indexes for text IR, the second describes structural variations, and the third presents some of the typical optimisations that can be used to enhance retrieval.

2.7.1 Zipfian Distributions

A property of many types of groups of object is that their distributions obey a form of the power law. There are many groups which contain a small number of the object and a small number of groups which contain many of that object. In his 1949 book, *human behaviour and the principle of least effort* [148], George Kingsley Zipf observes this in many elements of the universe, including galaxy sizes, city populations, and the distribution of words in natural language.

Known as Zipf's law, this form of the power law provides an excellent model for text based term indexes, and allows quite accurate approximation of posting list lengths. The law states that the frequency rankings of terms in a text are inversely proportional to the corresponding frequency:

$$log\ f_r \approx C - z\ log\ r \tag{2.17}$$

Where f_r is the frequency of the term at rank r, z is the exponent coefficient (near to unity) and C is a constant. There has been some work in identifying C for different

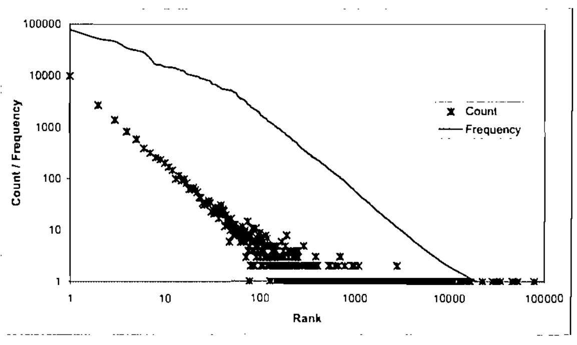

FIGURE 2.4: A Comparison of Rank-Count and Rank-Frequency Zipfian Analysis

natural languages, with Gelbukh and Siborov showing that English and Russian texts could be identifiable by the term distribution alone [55].

In order to determine the distribution it is necessary to analyse the number of collections which have a particular size. Chen provides a good technique by assigning an index to sequential observations of the data [27], which has been used throughout this thesis for the analysis of term distributions. He presents two representations of the distribution: Rank-count and rank-frequency. Four entities are observed in the analysis (this excerpt taken from his paper):

- word count n: the number of occurrences of a certain word contained in a text
- count frequency $f(n)$: the number of words of each count
- word rank r: the cumulative frequency of words of the same or greater count
- rank frequency $g(r)$: the number of words of the same rank

Figure 2.4 illustrates the two types of distribution for the same data. The rank-count data cannot be plotted as a curve due to the lack of uniqueness in the count variable, unlike the rank-frequency data which is shown as a clear curve. The shape of the curve is significant, and this is investigated later, in chapter 6.

This simple model allows the length of postings lists to be approximated (the document frequency of a term in a corpus is proportional to its overall term frequency), so when given a corpus of text of a known size and language it is possible to approximate the size of the index.

Whilst a simple index may benefit only slightly from such knowledge, a distributed index (see section 2.7.4) can be managed such that the size of each part of the index can be predicted.

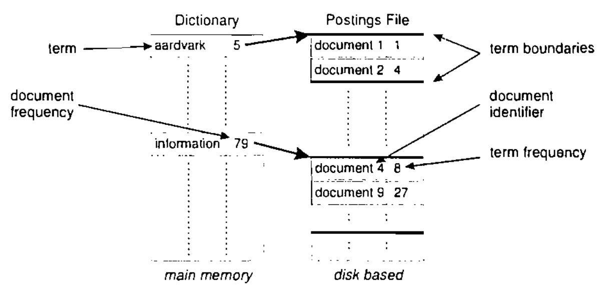

FIGURE 2.5: Realisation of Inverted Indexes

2.7.2 The Index Structure

The index may be a simple structure, but its effectiveness lies in the implementation used. Figure 2.5 shows how the index structures are typically configured. The main index is stored in main memory whilst the posting lists are stored as files on secondary storage. The main index will contain a pointer for each term which specifies where the postings start, and also a pointer to the posting in a secondary update index.

The update index is required because rebuilding a postings list is an expensive procedure, without which the postings file would become fragmented. An index will typically be rebuilt at a time when the load on the index is minimal (or the index is offline) and the update index has grown to a size where it is no longer efficient. This approach guarantees that the main postings file will be contiguous, and that a posting list can be retrieved sequentially from disk.

The implementation of the data structures varies, however the majority of systems use a B-tree to store the main dictionary, which for each term points directly to the postings list, be it in memory, on disk, or on another node.

2.7.3 Index Optimisations

Small, and fixed, document collections only require optimisations for the index retrieval operation, since updates to the index are rarely (if ever) required. Large collections containing many gigabytes of documents require more care when they are generated, and the indexing must be performed on disk. Frieder et al provide a clear review of efficiency considerations for text indexes in [49].The topic of such optimisations is very complex, and outside the scope of this work, and so this section will concentrate on optimisations to the index file that benefit retrieval performance. There has however been much research into dynamic index update strategies: Cutting and Pedersen were among the first to develop algorithms [34], and Clarke et al [31] describe another distributed system.

The majority of index retrieval time is spent in searching through postings lists, and it is through the careful construction of these that the best performance increases can be achieved. In many retrieval algorithms the term frequency for each document is used to calculate the amount that a term contributes to the document's score (in particular the TFxIDF algorithms). By ordering the postings list by decreasing frequency the number of postings that need to be retrieved can be reduced, since there will come a point in the list where the term frequency is so low that adding it to a document's score will no longer change the rankings of the retrieved documents [21]. Similarly the index is queried in descending order of query term frequency (should the query terms have frequencies greater than just 1).

2.7.3.1 Index Compression

Reducing the size of the index on disk can save in both storage and access time. A significant amount of work has been done in this field by Moffat, Witten and Bell, in various publications [94, 143]. This can be achieved by compressing the postings data, and can result in considerable savings. The information stored in a simple posting is the tuple $< docID, freq >$, requiring 8 bytes if both fields are integers. If the document ID list for the posting is ordered by descending frequency then the difference can be stored.

Rather than storing thesis Δ values as smaller integers, we can encode them into variable bit length representations. One such binary code is the unary code which takes a non zero, non negative, integer x and encodes it as $x - 1$ 1's followed by a single zero. The unary code for 4 is then 1110. When the delta values are small this code is very efficient, but when they are larger it can consume a very large number of bits. An alternative is the γ code which represents x in two parts. The first part uses unary coding to represent $1 + log\ x$ and is followed by a code of $log\ x$ bits that represents the value of $x - 2^{log\ x}$. For 4 this gives the two part code 110 00. There are a number of other suitable codes, including the δ and Bernoulli codes.

2.7.3.2 Limited Retrieval

A technique that can be used in conjunction with others is to restrict the amount of terms retrieved by a query. The removal of stop-words from a query significantly reduces the time to retrieve postings lists and also the number of documents that are retrieved.

Zig-zag database joins allow significant improvements in retrieval time where a conjunctive query has more than one term. Consider a query with two terms, one term which occurs frequently in many documents, and one which is rare. Retrieved documents must contain both terms, and so it is the intersection of the two postings lists that are required. The zig-zag join allows selective retrieval of postings from the long list given the set of documents from the short list.

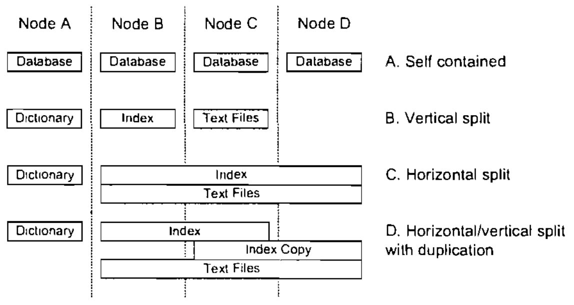

FIGURE 2.6: Strategies for distributing indexes

2.7.4 Distributed Indexes

Much work has been done in the field of distributed retrieval engines, mostly for inverted indexes, and is split into two primary groups: Parallel building of inverted indexes, and parallel retrieval from inverted indexes. In the first category is the work of Ribeiro-Neto et al [112] who describe a number of algorithms that run in a distributed shared-nothing architecture, also Melnik et al [92] who examine the use of a pipelined architecture with an embedded database system. In the second category, Tomasic and Garcia-Molina [132] examine the performance of queries and identify the variables which most strongly influence response time and tradeoff, Mamalis et al [86] implement a parallel system that runs on a distributed memory multi-processor environment, and Frieder et al [49] look at general considerations for parallelisation.

Distributing the postings files (and the main dictionary, which points to the postings) can benefit small, as well as large, indexes. The index can easily be split so that the postings for different terms are spread across multiple machines, allowing the retrieval of some to be performed concurrently. Martin et al [89] describe some of the strategies for dividing the index in an early parallel system that adapted IBM's STAIRS retrieval engine. Figure 2.6 illustrates four such strategies:

- A. Here each node contains a complete database - the dictionary, index and text files. Each node contains a subset of the database, and each must be queried separately.
- B. A vertical split allows each node to be dedicated to a particular task, in this case a node for the dictionary, index and the text files.
- C. A horizontal split spreads one or more of the functions across multiple nodes, allowing some parallelisation of the function.
- D. Another type of split creates redundancy in the functions such that concurrent operations on the index can be balanced across the nodes.

Each of these strategies has its advantages and disadvantages, and the choice of which one to use will depend on the demands required of the system in terms of: Its ability to be updated (static or dynamic index), the size of the collection (a few megabytes or many gigabytes) and the query load (occasional keyword queries or multiple document comparisons).

2.8 Evaluation of Content Based Retrieval

The performance of retrieval systems can be measured by how many documents returned for a particular query are actually relevant to a query. If the query is very simple like a boolean text query then the correctness of a retrieval can be exactly calculated, however a more human query such as *"Find me documents that are on the same subject as this document"* is incredibly subjective. To solve this problem collections of documents must be created where distinct categories are manually created. Only documents from within the known categories are used as queries, with a substantial amount of filler material used to bulk up the corpus. This approach has been used with much success in the text retrieval community for conferences like TREC (Text REtreival Conference) where there is a standard corpus of documents and categories, and a well defined system for retrieval engine evaluation.

2.8.1 Precision and Recall

Measurement of the accuracy of a query performed with known categories is achieved using *precision* and *recall* metrics [85, 119, 136, 81, 126]. The precision of a query is defined as the ratio of relevant documents to non-relevant returned by a retrieval system[6]. Recall is a measure of how many documents from the relevant category were returned, and is defined as the ratio of relevant retrieved documents returned to the total number of relevant documents.

$$precision = \frac{retrieved\ relevant}{retrieved} \tag{2.18}$$

$$recall = \frac{retrieved\ relevant}{relevant} \tag{2.19}$$

Precision and recall are normally represented on a graph on opposing axes. This data is created by calculating the precision and recall up to each of the rankings in turn, so forming a graph containing as many points as documents retrieved. Equation 2.20 is

[6]Precision and recall metrics assume that there is a limit to the number of documents returned - the size of the *hit list* returned. Perfect recall will always be achieved if there is no limit to the number of documents returned

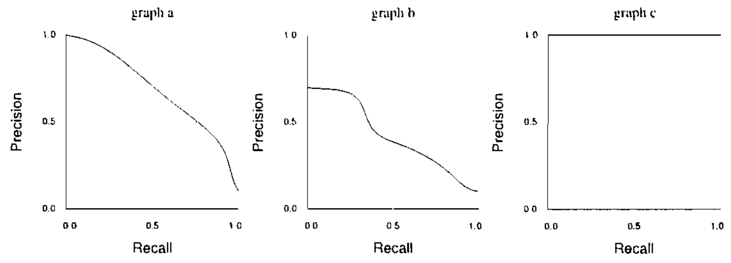

FIGURE 2.7: Precision and Recall Graphs

called the *precision at n*, where n is a particular point in the hit list. Similarly *recall at n* is calculated as in equation 2.21. Here rel_i is 1 if the i^{th} document is relevant and zero if not, and R is the total number of relevant documents for the query.

$$precision_j = \sum_{i=1..j} \frac{rel_i}{j} \tag{2.20}$$

$$recall_j = \sum_{i=1..j} \frac{rel_i}{R} \tag{2.21}$$

This is, however, not entirely sufficient since it does not take into account how well the documents were ranked. A perfect retrieval would return all relevant documents ranked sequentially from the top hit. In figure 2.7 graphs a and b are for two queries where the same number of relevant documents are retrieved, but in the second graph the ranking is not perfect. Since the first few documents in the second query were not relevant the initial precision is not 1.0, and so the graph starts low. A perfect retrieval as in graph c where only relevant documents are retrieved would form a straight line ($precision = 1$). By calculating the area under the graph we obtain the *average precision* for the query. This metric does take the ranking of the relevant documents into account and so provides an ideal measure for retrieval effectiveness.

Taking n to be the number of documents in the hit list, average precision is defined as:

$$\textbf{Average Precision} = \sum_{j=1..n} \frac{precision_j \cdot rel_j}{R} \tag{2.22}$$

As well as the average precision there are also other measures that can be calculated from precision and recall:

- *R-Precision* is the precision at rank R where recall becomes 1 - when R is equal to the number of documents in the category.
- *Initial Precision* is the precision when recall is equal to 0, as calculated by interpolation.

- *Fallout* is a measure of how quickly precision drops as recall is increased, and is defined as: $fallout = \frac{nonrelevant\ retrieved}{nonrelevant\ total}$. Recall - fallout curves can be plotted to show this measure.

2.9 The Future of Content Based Image Retrieval

The potential of content based image retrieval systems is very great, and in certain domains is being fulfilled. In narrow domain applications, where much is known about the entities which must be identified, it is possible to extract features and to reconstruct a certain amount of a scene in real time, but only where the acquisition of the scene is controlled.

The far harder task of retrieving images where the content of both the query and the searchable database is uncontrolled is still beyond the horizon. Whilst the paradigms of query by example and query by sketch are proving to be very useful in providing a basis for retrieving using content, they still have fixed notions of content and use a very limited semantic domain.

We are far away from understanding how the human brain interprets visual stimuli, and associates them with abstract and concrete concepts and ideas that it has learnt and experienced. Bridging the semantic gap that exists between low level features and high level concepts is not impossible, but will take time.

The growth of the Internet, and the advances in technology which provide much faster bandwidth network connections are increasing the feeling of information overload. It is true that the capacity of storage devices is increasing at a similar rate to the speed of processors, and also to the quantity of data produced by image acquisition devices. This results in situations where data can be produced far faster than it can be analysed. Text search engines on the web struggle to keep up with continually changing information, and users desire only the most up to date information in their searches.

A possible solution to this is to ensure that as much as possible of the feature extraction is done at the point of origin, and that feature information is stored together with the original data. The choice of feature would then become very important. A suitable feature might offer a sensible blending of colour, texture and shape information. The MPEG-7 content description layer [87] is making some progress towards offering such content description for video as an additional stream.

Other future technologies which would place a strain on systems are those which have a requirement for continuous content retrieval. Applications such as video security surveillance, automated control systems (in particular vehicle control), autonomous robotic devices (such as those deployed on extraterrestrial planets) and personal assistants (when combined with a personal software agent) must not only be able to continuously receive and interpret information, but they are also required to store new information, and, to an increasing degree, generate semantic associations.

The combination of such future technologies will result in requirements for larger databases to store information, and the need to be able to share information. A common problem with current CBR techniques is that the feature algorithms must be available not only for the feature extraction, but also for signature comparison. Whilst some signatures can be compared simply as vectors in n-space it is common for features to have complex comparison algorithms. The heterogeneity of features creates restrictions on how extracted information can be used. The work presented in this thesis shows an index in which heterogeneous features may be stored together, and additionally demonstrates that the *feature terms* that it uses are highly portable.

2.10 Summary

In this chapter a lot of material has been introduced, covering a wide variety of topics. Information retrieval has been shown as an activity exercised by humans on a continuous basis, as a fundamental aspect of our society today. Applications of information retrieval are not only restricted to humans, but to autonomous systems. The way in which queries are presented to the information source is discussed, demonstrating the need to have an understanding of the structure and scope of stored information.

In order to make the information accessible to the target user suitable features must be selected. The information requirements of the user, and the domain of the information may be well defined, which would allow the use of a specific feature. A suitable algorithm can be developed to locate this feature, allowing queries to return precise answers. On the other hand, the corpus of information may be formed from very diverse sources, with equally diverse demands from the user. The features required in this situation would need to be less specific, resulting in less precise answers. These ideas lead to the definition of narrow (specific) and broad (general) information sources and queries - an important factor in the design and implementation of a retrieval system.

Regardless of the type of feature used, a content based retrieval system must be capable of measuring the similarity between the features of a user's query, and the features contained in the stored documents. Classification and clustering techniques are vital to this process, providing quantitative measures for the similarity of the features. Techniques are shown which draw from Euclidean mathematics and artificial intelligence.

Techniques in text retrieval have been evolving for a number of decades, and a significant amount of literature has been produced on the field. The extraction of tokens and identification of terms which are equivalent is highly important, and will define the granularity of a text index. The field of linguistics provides mechanisms such as morphology for finding the root meaning of words which can increase the precision of retrieval and also reduce the size of the index.

The storage of extracted terms is a key factor in the success of a text retrieval system. Different data structures offer different performance characteristics and need to be selected based on the application: Quantity of documents, frequency of index update (addition and removal of documents), the type of query being performed and the volume of queries. The index and query terms can be weighted to emphasise, or lower, their importance. Various automated weighting schemes have been devised which use properties of the term distribution to determine the relevance of terms in text.

The distribution of text terms is shown to adhere to a form of the power-law - Zipf's law. This law relates the frequency of terms to the number of terms which occur with a particular frequency. This law is important in the construction of inverted indexes because it will allow the size of the index, and the speed of retrieval, to be estimated.

A variety of different types of image feature types are illustrated. Colour forms the basis of all image features, and leads to second order features like texture. Image features consist of multi-dimensional data which leads to difficulties in the management and comparison of them. Multi-dimensional indexing attempts to overcome some of these problems by transforming the data into a more manageable form, or by providing complex data structures which support comparative queries.

Given a complete content-based retrieval system, with a corpus of documents, suitable features and an index, it is necessary to quantify the usefulness of the retrieval. Precision and recall metrics provide a measure of the accuracy of the documents returned by a hitlist, and the average precision gives an overall measure, taking into account how close to the top of the hitlist relevant documents are.

There are many topics in CBR, of which only those relevant to this thesis have been examined. For a more comprehensive review of the field, with greater attention to image features, the reader is referred to Smeulders paper - Content-Based Image Retrieval - at the End of the Early Years [125].

Chapter 3

Multimedia Information Systems

3.1 Introduction

The aim of research in the field of content based retrieval is to provide rapid, coherent, and useful, access to the wealth of information which we now store in computer based systems. Returning to the ideas outlined in the introduction chapter, we wish to provide systems with which people are able to *find out about* topics which are of interest to them. It is also absolutely necessary that the information is stored in a flexible manner which allows not only for the querying of it, but for its growth and transformation. Multimedia Information Systems (MIS) offer complete computer based solutions for the tasks.

Such systems can be largely divided into two main categories - academic and research systems, and commercial and military (or other government) systems. In the research domain the aim is to either investigate the worth of an information retrieval framework, or to provide a fixed framework which facilitates the development of other components of the system. Commercial systems will transfer research technologies, and engineer an application that may be accessed concurrently by many different users, possibly in real time, through an interface that allows the user to express their information requirements adequately and accurately (within the relevant information domain). There are also the hybrid systems which lie somewhere between the two categories, providing both a practical environment in which to test techniques and algorithms, and also a viable information retrieval application.

MIS provide two core areas of functionality: Information storage, and information retrieval. Emphasised in systems whose primary objective is to archive material, the storage aspect of MIS is equally as important as the retrieval aspect. According to a review of human information production, Lyman and Varian [83] state that medical and astronomical applications already produce many petabytes of data yearly, with the volume increasing in line with Moore's law.

This chapter presents the types and architectures which MIS adopt, examining the types of image corpus which is stored, the domain of the queries, and the storage and retrieval architecture of the index. Section 3 presents some of the classic CBR systems, including QBIC, MARS and MAVIS.

3.2 Types and Architectures of Multimedia Information Systems

Section 2.2.1 introduced some of the applications of computer vision and the environments in which they operate. Whilst these systems all have a common set of components their architectures will all be quite different. There are a number of factors which must be taken into consideration when designing the architecture of an MIS system, which can be split into two overall groups - the source of the information (i.e. documents and images) and the user (not necessarily human) from whom queries are elicited and responses returned:

- **Information Source:** Where the source information originates from is very important and will define many aspects of the system.
  - **Domain and Scope:** Is the information domain in which the processing is to be performed restricted (narrow) in any way? Does the retrieval system have to interface with any domain dependant systems? If the information domain is restricted then is the scope of the domain liable to change?

    Where the domain is restricted, the information extracted by features will also be restricted. This means that the algorithms used are liable to be more complex but will also be highly specific. Measures of accuracy will become important as a qualitative assessment of the objects that are being recognised needs to provided.
  - **Volume and Detail:** What is the quantity of information to be analysed? What is the format of the source information - are images of a high resolution? How much information must be extracted from the source?

    If the source is providing very large numbers of images then this will mean that the system will need to be able to provide sufficient processing power to analyse them. This becomes a serious consideration where the system must process incoming data in real time - a factor which applies to video sources. It may be required of a system to archive the original source, to allow for normal viewing alongside feature data, or for additional post processing.
- **Target Users:**

- **Response Time and Availability:** How many concurrent users are expected? Is the system running in real time? Will users expect to wait for a result to be returned? Must the system be available 24x7?

  The answer to the first question will have a large impact on the architecture of the system, for if multiple concurrent users are accessing the system then additional loads will be placed on the retrieval engine. If the archive is updateable by users then an update mechanism must be provided on the index to prevent updates from blocking retrievals.

  A real time system such as the EyeTicket biometric system [7] must respond within a matter of seconds and must have an effective uptime of 100% which will lead to architecture that provides fail-over systems. If the processing demands are sufficiently slow then an offline queuing system could be used, like that in the Artiste and Sculpteur systems.

- **Retrieval Accuracy:** Does the system have to return a true or false response to a query, rather than a set of matching results? Could harm be caused by inaccurate results?

  Certain applications will require an automated system to be entirely autonomous and so must make decisions based upon the results of retrieval (comparison) operations. Where this is true there must be a clear understanding of the consequences that will occur should an incorrect judgment be made. For example, in a medical PACS which diagnoses patients, an incorrect result that is automatically processed could result in harm to the patient, and on the other hand a missile might be designed to commit to striking when it has positively identified its target. In general it may be better to err on the side of caution and provide false negatives rather than false positives.

3.2.1 Information Retrieval Environments

3.2.1.1 Medical Systems : PACS

Medical Picture Archiving and Communication Systems [64] present a number of the issues described above. Perhaps the most difficult of these problems to cater for is the heterogeneity of the systems involved. Image acquisition sources in a clinical environment will be very varied, including Computed Axial Tomography (CAT), Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), X-ray as well as regular photography. Each sensor system must be linked into the PACS so that image files are correctly associated with an individual clinical session and patient, for which the records may reside on another external database. A single examination may produce a few hundred megabytes of image data which must be stored and analysed.

Not only must the data be stored, but images and records must also be secure which could lead to problems where an institution has policies about holding and transmitting clinical data off site. The storage facilities themselves must be both reliable and permanent, which leads to the use of hierarchical storage systems. Incoming images will be stored onto disk (RAID) arrays where they may be kept whilst analysis is performed. Once this is complete they may be moved onto tape based offline storage until they are required again.

The emphasis in PACS is on the provision of rapid storage of incoming media from heterogeneous sources, with analysis and retrieval functions placed by necessity in second. Major developments in the area include GRiD [47] technologies to support distributed processing of images, the Semantic Web [137] to enhance interoperability, and a variety of novel image processing algorithms. One active research project combining elements of these fields is MIAKT (Medial Imaging with Advanced Knowledge Technologies) [35, 146, 63].

3.2.1.2 Digital Libraries

The key aim of digital libraries is to provide electronic access, and all that that entails, to archives of items. The most literal instance of a digital library is one which contains textual material - both documents that have been produced electronically as well as those that have been captured from paper or other media. The web sites of academic journals provide advanced retrieval functions to allow precise location of papers from often very large collections. In such a system the library may not actually contain the documents themselves, providing only an information retrieval service without the storage facilities.

The Malibu project [61] defined a framework for sharing disparate resources between academic institutions, centred on the humanities disciplines. It offers an infrastructure to combine distributed collections into a hybrid library that is accessed by common tools. Citeseer [77] provides a highly specialised digital library that indexes research papers, extracting citation information using ACI (Autonomous Citation Indexing) and providing a clean and easily navigable interface.

The architecture of such systems is as varied as the subject matter which is contained on them. Some may offer indexing services, whilst others will provide only the framework required to connect others together. It is the development of a uniform and coherent which poses the biggest challenge, requiring methods for querying the composite systems and presenting the returned information. Agent architectures have proven to be successful in this area, with ontologies providing the means to interpret information from heterogeneous sources.

3.2.1.3 Web Retrieval

The public internet search engine Google [20] has achieved a well deserved amount of success and is currently not rivaled by any of its older competitors (Altavista, Yahoo, WebCrawler, etc.). Its success can be largely traced to a clever ranking algorithm, an architecture that has scaled successfully [9], and an aggressive crawling strategy. The PageRank technology developed by Google will rank highly documents (and complete websites) which have a high number of links to them from other pages. This in effect states that people consider a site to be relevant and worthwhile because they are referencing it, even though this takes no account of the actual content of the site. The Google index provides a document cache containing text only copies of each page in the index from the last time that they were indexed. This means that a hit for a page which may have changed since it was last indexed can still be viewed.

Given the enormous quantity of documents available across the internet, web retrieval engines must employ state of the art techniques in order to be successful. Not only do they have to index millions (if not billions) of HTML pages, but they must keep popular pages in the index up to date, and they must serve millions of requests that must be answered in a matter of seconds. Google's chosen architecture is one of massive levels of parallelisation using inexpensive servers. A hierarchy of machines provide front cluster (network) and rear cluster (index processing) balancing, with mirrors situated across the globe.

This approach is not only highly scalable, but it is sustainable. As equipment prices drop and processing power rises new servers are purchased and placed onto the network, moving older machines into less demanding roles. So far this approach has enabled the search engine to keep pace with the expansion of the internet, both in terms of the volume of web pages and the number of users.

3.2.1.4 Life Recording

This aspect of information retrieval has only started to emerge as a viable research field in the past few years due to the continued miniaturisation and reduction in cost of electronics. The aim of this technology is to allow the continuous recording of one's own life, with two main outcomes: The user is provided with an enhanced memory through the retrieval functions of the system, and can publish a multimedia diary of their life (to those who wish to know about it).

An element of such systems is the ability to determine what the user wishes to remember - it would be very ambitious to request of the system that it record all events - and to reduce that information to a manageable quantity. Incorporating a personal assistant would remove the need for intervention by the user in determining the events that are recorded (and the detail in which they are recorded), but introduces room for errors in judgment on the part of the system.

An obvious requirement for a life recorder is large amounts of storage, but this should not be physically located with the recording device itself - which needs to remain portable. Instead the system needs to reduce the information in real time as much as possible so that it can either be uploaded across a wireless network, or, where a link is unavailable, can be saved in the limited local storage.

HP's CoolTown project [10], the European Equator project [1] and others have been spearheading research into the applications of portable (and indeed wearable) computers. Companies such as Xybernaut are selling compact personal computers which can be worn on a belt, and are supplied with high resolution display monocles.

3.2.1.5 Consumer Content Delivery

The general public are a good reference point for the maturity of technology - complicated and unusable systems are rejected by the market. Personal content delivery is starting to become a reality with the availability of broadband home network connections, and cheap digital AV recorders (set top box DVD and hard disk recorders). The Sky+ package provides a hard disk recording system and a stream of channels from which the user selects the programs they would like to watch. They are then able to watch at their leisure.

The ability to provide a useful content layer is therefore becoming highly desirable - not least because the media industry themselves would be a primary user. The MPEG-7 [87] standard is attempting to provide flexible descriptors, and has support for some image features. The BUSMAN (Bringing User Satisfaction to Media Access Networks) project [6, 68, 144] looks at how such content description streams can be successfully managed in heterogeneous environments, with the potential for media to become transformed or corrupted on its course from source to destination.

Some of the objectives which the BUSMAN project hopes to, and indeed has, achieved are:

- Access to and delivery of multimedia content on consumer devices (digital television, computer systems, information kiosks, mobile devices).
- The integration of delivery channels such that content will be transformed from the source to the destination seamlessly.
- Content protection throughout the delivery channel by advanced watermarking techniques capable of withstanding transformations.

- Content based search facilities with a wide range of content descriptors.

The architecture developed for this project has two key server components, and two client types. The *input unit* part of the server provides ways to submit content to the system and the *annotation client* allows professional users to annotate the content - adding, for example, semantic and shot boundary metadata. The *information server* connects to the *end-user terminal* and provides content tailored for the terminal type (two types have been developed: fixed PC desktop and wireless, mobile, terminals), and the query engine functionality.

Data is embedded into video content using watermarks tailored to both MPEG-2 and MPEG-4. These techniques will insert a Digital Item Identifier (DII) into the video stream by using a narrow band signal over the wide band of channel of the video signal which attempts to minimise visible artifacts being introduced. The DII contains a Busman Content Tag (BCT) which will allow a server to determine the appropriate metadata for the content without having to send metadata with the content itself.

The metadata used with BUSMAN is in accordance with the MPEG-7 standard, with visual descriptors (

3.3 Image Retrieval Multimedia Information Systems

3.3.1 QBIC

QBIC [46, 101] was the first commercially available CBR system, available as a extension to IBM's mainframe DB2 database system. It provided both image and video retrieval together with a browseable user interface and a novel query by sketch interface. The system is broadly divided into four subsystems - image and video analysis, media database, feature index, and user interface.

Media Analysis Subsystem

QBIC has three levels of feature extraction: Scenes, which include (global) spatial colour and texture features; Object, which includes location, shape and user defined colour and texture features; and Video, which includes object and camera motion features.

Object extraction in QBIC is performed using a combination of manual and automatic methods, as required by the user. A foreground/background (threshold) model is provided, for which the results may be augmented by selecting background pixels to enhance the thresholding. A region growing algorithm similar to a flood fill is also provided, where the user initiates the region with the selection of one or more pixels. The threshold of the homogeneity constraint that determines whether the pixel is part of the object or not, is user adjustable. The third technique is based on snakes, which the user can sketch around an object before allowing the algorithm to iteratively adjust the shape of the snake to the object.

Video features are limited to shot detection (shot boundaries exist where global colour change or global motion exceeds a threshold) and motion detection. Motion in the image is calculated using a specifically developed algorithm that computes the global view transformation while remaining insensitive to local changes caused by objects and brightness changes [120]. Object identification is performed by examining the motion flow for coherent areas of motion; if this is lacking and the global motion is uniform, then camera motion is assumed. Identified objects become candidates for the object extraction techniques described above and are stored in a layered structural representation.

Image Features

A number of colour coordinates are extracted: (R,G,B), (Y,i,q), (L,a,b) and MTM (Mathematical Transform to Munsell) [93]. A complex series of algorithms are used to extract a colour histogram, which begins with the generation of a $16^3 = 4096$ bin histogram. This histogram is populated with RGB values from the image or object, and then the MTM coordinate of each cell is calculated, and these coordinates clustered using the minimum sum of squares to find the 'best' 256 colours, called super-cells. Histograms are compared by calculating the weighted Euclidean distance of histogram vectors, allowing the user to, for example, reduce the influence L has in an (L,a,b) query.

Described in detail in [43], the texture extraction technique employs Tamura's notions of contrast, coarseness and directionality (see section 2.5.2). The algorithms were modified to make them computationally less expensive and so more viable for an enterprise scale system. Texture comparison is also by means of weighted Euclidean distance

Shapes in QBIC are described by a combination of heuristics: Area, circularity, eccentricity, major axis orientation and moment invariants. Each shape is also stored in the database as a binary pixel mask. To support the query by sketch function, images are reduced to single band luminance, then passed through a Canny edge filter. The image is then reduced to 64x64 pixels using a filter that maintains edge structures.

User Interface and Queries

A query may be formed of an entire image, an object, or a set of object attributes. Both colour and texture pick palettes are provided, as well as a shape creation tool (with various drawing tools) and the query by sketch interface. Shape creation can be based on an existing image in the database, which allows the use of the snake outline tool.

Indexing

The index in QBIC is built as a modular component so that it could be updated as faster multidimensional indexing techniques are found. The indexing used by QBIC in its early phases relied on either a flat file, or Starburst relational database, for the image features, which were retrieved sequentially. The R*-tree was tested, and interest was shown in linear quadtrees and grid-files (see section 2.6).

3.3.2 MARS

The MARS project [65, 105, 104, 103] has been highly successful, and produced many papers (over 45 at the time of writing) on a large number of CBIR topics. Of particular note are the contributions to video representation [147, 118, 116] and multidimensional indexing [23, 22]. Like QBIC, MARS/IRS (MARS Image Retrieval System) is divided into four primary systems, supported by the Postgres SQL database server.

Image Analyser

The early versions of MARS used three salient image properties, and did not provide support for video. An HSV colour histogram provided the colour feature, the colour model being chosen due to its perceived uniform colour space. The histograms are compared using Swain's intersection method [129] which allows better matching where query and target histograms have a subset of similar colours, for example in images with the same central object on different backgrounds.

Texture similarity was provided by a 3D texture histogram, the components being Tamura's coarseness, contrast and directionality. The neighbourhood around each pixel is examined for the strength of the three measures, and the appropriate bin incremented. The decision to use a histogram was made because of the potential for indexing, unlike Tamura's scalar approach.

Shapes are extracted by clustering in the spatial-colour-texture space provided by the colour and texture histograms using c-means clustering. Like k-means this approach involves iteratively pairing c starting cluster centroids to their nearest unpaired neighbour. The process is repeated until convergence occurs. The resulting areas will be divided until they are spatially contiguous, or discarded if they fall below a threshold area. A threshold is set in the colour-texture space to determine objects and background using an attraction (gravity) based method [117].

Images are represented on two levels - the global and the object. The global descriptors consist of fixed text metadata and free text, and low level image properties (colour and texture) for the entire image. Objects have the same descriptors for their pixels, but also include the shape only properties of area and centroid.

Indexer

Initially the indexing subsystem used flat files and R* trees, however the index did not scale to the large image databases and complex feature spaces required by the project.

Later efforts include a localised method for principle component analysis [24] and a hybrid tree that combines techniques from data partitioning structures (for example the R-Tree family) and space partitioning structures (for example the kDB-tree). The index also contains a text processing component, which is used for textual metadata associated with images and objects.

User Interface and Querying

The query interface provides three types of query: simple, complex and similarity. In a simple query a colour and texture palette are provided to allow the user to select colours and textures that should be present in the images retrieved. Complex queries may be constructed by creating objects with attributes which are desired in the target images. A graphical interface allows shapes to be created, to which colour and textures may be assigned. The last query type is the global similarity query, in which all features, be they global or object, are compared with the query image.

3.3.3 VisualSEEk, MetaSEEk and WebSEEk

VisualSEEk provides a comprehensive diagramming query system which will return images based on the relative proximity and location of similar objects in the query. Regions in images are extracted using colour set back-projection [127], and a 166 colour histogram (created by transforming RGB to HSV and quantising into 18 hues, 3 saturations, 3 values and 4 greys) is derived. Regions are compared by non-spatial attributes first, then the locations of the identified regions from the candidate images are transformed into a 2-D string which is compared with that of the query image's regions.

WebSEEk [26] extends the architecture of its predecessor with web spiders that download web pages, following links, and retrieving images and video to index. Unlike VisualSEEk, WebSEEk does not store the images themselves, only the images features and metadata that it has extracted. Similarly MetaSEEk [15] is a web based system, but has no active image analysis components. As a meta-search engine it selects target visual search engines and scripts queries to them based on the query that the user gives to it. A core part of its functionality is the ability to learn the characteristics of its target search engines, and provide relevance feedback during the query process.

3.3.4 VideoQ

The key attribute in VideoQ [5] is that of motion, which can be sketched in the user interface. This allows video objects (defined as sets of contiguous regions of pixels that are homogenous according to a low level feature that display consistency over a number of frames) to be retrieved based primarily on their motion. Care is taken in the work to ensure that the size of video data does not affect the user's retrieval experience, and so candidate video shots that match a query are displayed to the user as key frames. It is shown that in most cases this is sufficient for the user to determine whether the shot will actually be a desirable match.

3.3.5 MAVIS

The Microcosm system [36, 37] explored hypermedia techniques for linking documents, and navigating between them. The linkbase and generic link was proposed and demonstrated, a function that performs a query against known documents based on the text contained in the link. The MAVIS (Microcosm Architecture for Video, Image and Sound) project [79] delivers the same functionality for non-textual media. The user is allowed to select multimedia elements in a document with which the system will execute a content-based retrieval query against other media elements in the database, resulting in a list of generic links. The MAVIS II [80] system moves into the semantic domain by adding a multimedia thesaurus that extends the functionality of the linkbase. Media elements in the database can be linked to concepts, which are stored hierarchically.

3.3.6 Artiste and Sculpteur

Both of these projects have developed digital libraries for museum artifacts. The first, Artiste [57, 25, 3], produced a system for content based retrieval and analysis of paintings. A core framework and API [4] for image analysis and indexing was developed. An important factor in the design of the system was to be able to process and store very large images - of up to 10k^2 pixels, for which low level feature processing demands are very high. QMNS, the algorithm presented in chapter 5, was implemented as a feature for Artiste, and the API was used throughout this thesis to compare the retrieval performance of the tested features.

The Sculpteur (Semantic and Content-based mULtimedia exPloiTation for EURopean benefit) project [2, 58] introduces technology from the semantic web community, combined with 3D object storage and retrieval, to provide a comprehensive, interoperable MIS.

3.3.7 Viper and GiFT

The name of the group that performed the research, and of an implementation of a CBR system, Viper (Visual Image Processing for Enhanced Retrieval) has three primary assets: MRML, the Multimedia Retrieval Markup Language [100], provides an abstraction of content based retrieval systems. Many aspects of the field have been covered, including: support for multiple query paradigms, relevance feedback, metadata, and evaluation. The Viper retrieval system [128, 98] uses MRML to provide image retrieval from an inverted index. Whilst the low level features themselves are not novel (a HSV colour histogram and real, circularly symmetrical Gabor texture filters) they demonstrate good retrieval at very high speeds.

The GiFT (GNU image Finding Tool) [99, 97] is a framework for content based retrieval, built with modular components for multimedia storage, processing, feature extraction and indexing, retrieval and evaluation. Access to the framework is via an MRML enabled client. The Viper system was re-implemented as a plug in for this framework.

3.4 Summary

There are many different uses for content based retrieval, as illustrated by the second section of this chapter. It is very hard to think of any computer based application where content based retrieval would not be useful, be the source of information a relational database system, or a video feed. Architecturally the systems are all very similar. There is a component to store documents, one to analyse them, a index to store the features in, a query processor and some form of user interface.

Some architectures are designed for flexibility, to be adaptable and allow the addition of new components, and, indeed, to research the suitability of the architecture itself. Others are intended for proof of concept of one, or more, of the individual systems. In order to understand how well the system is performing it is necessary to evaluate retrieval, and it is in this area that the field is currently lacking. The Viper group have been involved in the Benchathlon network [96] which is attempting to form an evaluation community and resources similar to that provided as part of the TREC text retrieval conference series. Until a standardised evaluation technique is available, it will be very difficult to compare CBR systems.

Chapter 4

Invistor - An Inverted Index Multimedia Search Engine

4.1 Introduction

This chapter presents the systems that were used for the research work presented in the next two chapters. Using a mixture of C, C++ and Java, the systems create a complete platform for generating, testing, and analysing binary signature and feature term based indexes. Split roughly into three categories the systems cover feature extraction and indexing, index data analysis and retrieval evaluation.

This chapter is divided into four main sections after this introduction. The first describes the overall architecture of the system. The second section presents the largest component of the system - the CBIR image indexer, which extracts the actual features from images (and potentially any document) and stores them in the inverted index. The third part describes the techniques and algorithms used to extract data about the distribution of the feature terms held in the index, and the fourth describes the Java program that generated the metrics used to measure the performance of the retrieval system.

After these sections, in section 4.6, there is a short discussion on potential improvements to the system, issues of scalability and the architecture of a future system. Finally the chapter ends with a brief conclusion.

4.2 The Architecture of Invistor

The architecture of the Invistor system is divided into two parts - the systems that perform processing, and the databases. Figure 4.2 presents an overall view of the architecture, showing each of the key components in the system. On the left are the

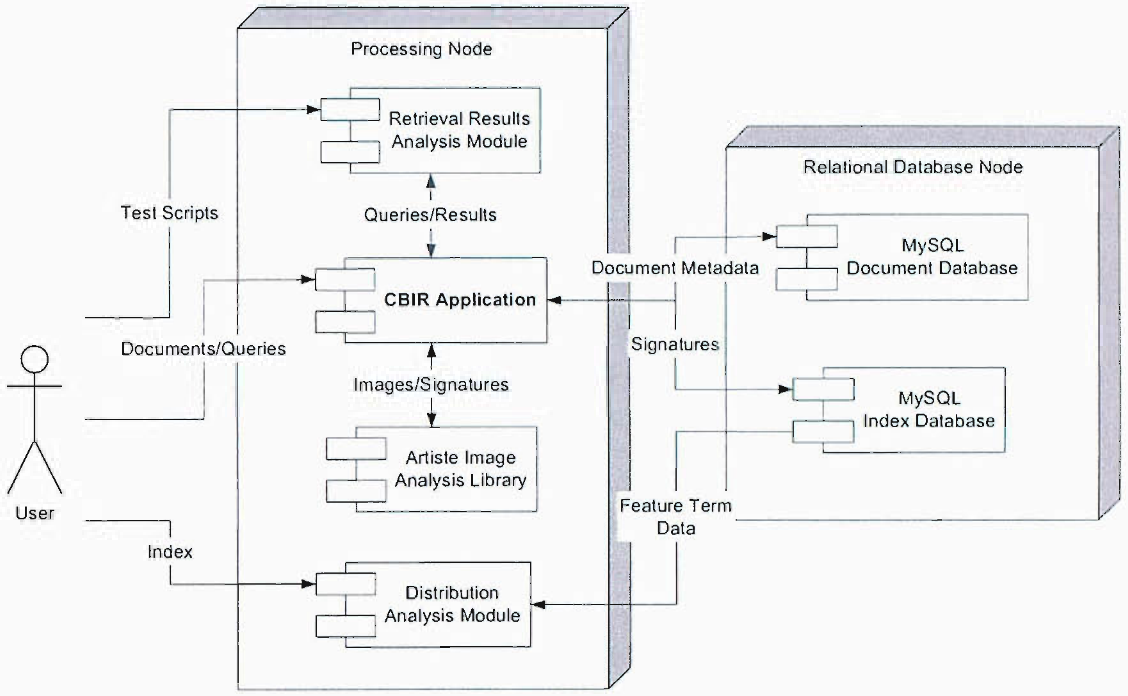

FIGURE 4.1: Invistor - Overall Architecture

processing modules, including the CBIR image indexer, the retrieval results analysis module, the Artiste image analysis library and the index data analysis module. On the right are the two databases, one of which stores metadata about the documents and the other which contains the main index. The information that is transferred between the modules is shown by arrows.

The CBIR image indexer is responsible for most of the processes that are involved in the system. A user is able to submit images to the indexer which are processed by the Artiste image analysis library. The library outputs signatures in either a binary form, or a feature term form. These signatures are then inserted into the metadata and index databases. A user is also able to query the system by submitting a single image to the indexer which is processed and then compared with the other images in the index. Results are provided in a formatted HTML report, and a tabulated text format.

The retrieval results analysis module accepts test scripts containing categories of images which form the ground truth for recall-precision metric generation. The module will then execute a series of queries by calling the indexer and analysing the results that are returned by it. Once all tests are complete it produces a report of the recall-precision data.

The distribution analysis module performs only one task, which is to generate a Zipf distribution for the data in a specified feature term index.

4.3 The CBIR Image Indexer

Written in C, the CBIR image indexer provides the core functionality for the whole system. Controlled via command line options in a UNIX environment it has five commands: Initialise database, index one image, index multiple images, compare two images and query image against index.

- **Initialise Database:** This command will create a document metadata database in MySQL, and will either create a blob index for normal signatures, or an inverted index for feature term signatures.(See the next section for details on the signature data structures).
- **Index Single Image:** This command takes a single image filename and the codes for the desired Artiste image processing class, and generates the signature. If a feature term signature is required a switch can be enabled that will cause the image processing class to generate a feature term signature, which may be inserted into an inverted index. The name of a valid database is also passed with this command.
- **Index Multiple Images:** Extending the previous command, this function allows the user to index all the images listed in a file.
- **Compare Two Images:** The first of the two database querying commands, this function will compare two images, returning a similarity score.
- **Query Index:** The last main command will query the index and generate a hitlist for the best retrieved images. If the query image is already in the index its signature is retrieved and is compared against all signatures. When the comparison is complete the first image is ignored since it is the query itself.

4.3.1 The Relational Database Index

The indexer stores information about images, and the signatures of the images, in a relational database - MySQL. The use of such an RDBMS removed the need for designing and implementing some complex data structures. The database system provides a reliable and efficient means for storing data, and allowed efforts to be focussed on the core functionalilty of the systems.

The database system is used for two purposes - storing metadata about the images submitted to the index, and storing the actual signatures themselves. Since the indexer supports both raw binary signatures (of an arbitrary format, known to each image processing alorithm) and feature term signatures, two versions of the database were created. Figure 4.3.1 shows the two tables required for the binary signature index (top) and the three tables required for the feature term index (bottom). Table 4.1 describes

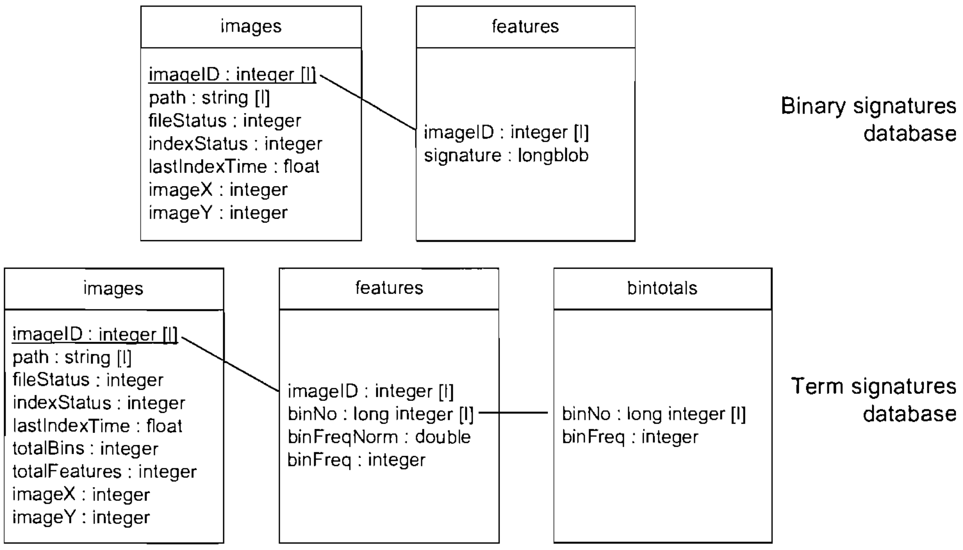

FIGURE 4.2: Relational Tables: Top. Binary Signatures, Bottom. Feature Term Signatures. Underline indicates primary key, [I] indicates indexed field

the purpose of each of the fields in the tables, indicating where fields are common, or not, to the two types of index.

Table	Database	Field	Description
images	both	imageID	A unique ID for each image document
		path	The path to the image
		fileStatus	Status of the file - exists, not exist, corrupt
		indexStatus	Status of the signature - unindexed, indexing, indexed
		lastIndexed	Date and time that the last index of the image started
		lastIndexTime	Time (in millis) that the last index took
		imageX	Width of the image in pixels
		imageY	Height of the image in pixels
	term	totalBins	The total number of different terms in the image
		totalFeatures	The total number of features in the image
features	both	imageID	The image ID
	binary	signature	The binary feature signature
	term	binNo	The unique term identifier
		binFreqNorm	The normalised frequency of the term in the image
		binFreq	The frequency of the term in the image
bintotals	binary	binNo	The unique term identifier
		binFreq	The number of documents in which the term occurs

TABLE 4.1: Index database table definitions

4.3.1.1 The Image Metadata Table

The metadata table stores a small amount of information about each image that is submitted into the index. The path indicates the location of the file, and may be in any format. The CBIR indexer uses the UNC file naming conventions. Only the dimensions of the image (in pixels) are recorded since it is assumed that all of the feature extraction algorithms provide their own methods for converting different images from their original colour model to that which they require.

A flag to indicate the status of the file is provided, so that the indexer is able to indicate whether the files submitted for indexing are valid files and has the following values: *unknown, ok, does not exist, corrupt, non-image*. The indexer can determine some of the possible states together with the image algorithm returning either corrupt, or non-image should it be unable to generate a valid signature for the file.

Also provided is a flag to indicate the status of the signature in the index, should it exist. There are four possible index states: *unknown, indexed, indexing, unindexed*. Where the file status indicates anything other than an ok status the index flag is set to *unindexed*. Since multiple copies of the indexer may be run concurrently (see section 4.3.3.1) the indexer is able to set the index status to either indexed, or indexing. This also allows another CBIR client to perform queries whilst an index update is being performed.

The metadata table for a feature term index includes two further fields that record data about the feature terms in the image. The *totalTerms* field shows the number of unique terms within the image (the vocabulary size), and the *totalFeatures* field shows how many feature terms existed in total in the image.

4.3.1.2 The Index Tables

The index tables used to store the signatures for the image are entirely different for the two types of index. The binary index uses only one table, whilst the feature term index uses two.

The Binary Index The binary index stores signatures which are of an arbitrary format determined by the image processing algorithm used. A single entry in is stored in the table for each image, containing the image ID and the binary object (blob).

The Feature Term Index The data structures required to build an efficient and reliable inverted index are complex. The preferred technique for building an inverted file is to use a B-Tree like structure and a flat file. This B-Tree provides rapid access via the data key to the information stored in the flat file, and should ideally exist entirely in main memory. Since MySQL provides good indexing on tables it was decided that there was no requirement for a bespoke inverted file implementation. The index provided by MySQL is implemented as an abstract B-Tree that can store any datatype, provided that a suitable comparison for the datatype is available.

To simplify the index further each term posting is stored separately. This allows the index table to adopt the structure shown in table 4.1. The normalised term frequency is provided because it can not always be calculated from the term frequency and the image dimensions. This will occur when a feature extraction algorithm does not always generate the same number of features for a particular image size. The QMNS algorithm, for example, does not create features for some patches, whereas the histogram algorithms (the RGB histogram and the CCV) always count every pixel in an image.

The second feature term index table, *termtotals*, provides data for scoring algorithms which require the *document frequency* of a feature term. This table contains only two fields - the term number and the document frequency.

The term number field in both tables is indexed causing MySQL to generate a B-Tree for the table keyed against it. When querying the SQL table for all postings that are for a particular term the database will locate the desired entry in the table index and is then able to return those postings without further searching. In the current implementation this does, however, still incur a significant overhead in disk accesses.

4.3.2 The Artiste Image Processing Library

The Artiste image processing API [4] was developed by Stephen Chan for the Artiste project to allow for a modular, extensible, architecture. Implemented as two C++ classes, the interface provided two functions that an image processing algorithm had to implement: The ability to generate a signature for a given image, and the ability to compare two signatures. Each algorithm implemented in the API had to supply its own image processing library, however it was the VIPS (Vasari Image Processing System) that was used by most researchers.

Here follows a list of the image processing modules created, and the authors of the module:

- RGB Histogram (Dupplaw): A simple RGB histogram capable of quantizing each axis of colour space independantly.
- CCV (Chan): A colour coherence vector histogram that provided 8 levels of quantisation.
- MCCV (Chan): The CCV above, but extended with a multiresolution spatial encoding.
- QMNS (Westmacott): The QMNS algorithm, as described in chapter 5,section 5.2.
- PWT (Fauzi): A Pyramid Wavelet Transform algorithm for extracting texture features.

- DWT (Fauzi): A Discrete Wavelet Transform algorithm for extracting texture features.

The two classes provided are the ImageProcessor, which contains the code to generate and compare signatures, and the FeatureVector class, which contains code to store the signature as a machine independant binary object (in database terminology a BLOB).

The Artiste project used the modules by linking them with the database, and referencing each one individually. Since the modules were in a state of continuous development, and more were being written, a library to contain them in was created. Each of the modules was altered to take advantage of the inheritance features of C++ so that any module had as its parent either an ImageProcessor class, or a FeatureVector class for image processing functions and signature storage functions respectively. The library is implemented using the Linux dynamic C library, which allows binaries to be loaded dynamically during program execution. Because this library is written in C, and the Artiste API is in C++, it was necessary to implement class factories that would return a reference to the required Artiste class.

The final stage in creating the completed Artiste library was to provide one further function on the feature vector interface which would convert a binary signature into a list of feature terms. Since each binary signature is unique to its image processing algorithm, this function is implmented by each feature vector class. The output from this function is an array containing the term number, the document frequency and the normalised document frequency. Each of the image processing algorithms has a unique range of term numbers which allows heterogenous features to be stored in a single index.

The completed library provides a means for any researcher to access the image processing algorithms using one library file via a short number of calls (provided they use a Linux environment and program in C or C++), together with a great deal of flexibility over which image formats may be processed.

4.3.3 The Index Operation

Regardless of the technique used to start indexing, each individual image is indexed in the same manner, with some differences depending on whether a binary or feature term signature is required. The indexing operation follows this simple algorithm:

- **File Check:** The file supplied is checked to see whether it is a valid file (as opposed to an incorrect filename or a directory) on the file system, and also to find if it is already in the index or not. If the file is invalid then execution ceases here. If it is valid then an entry in the document metadata table is created (or replaced if the filename already exists) and the indexing flag is set to *indexing*.

- **Signature Generation:** The appropriate Artiste image processing modules will already have opened during initialisation of the indexer, and now the valid file is passed to the module.
  - **Format Check:** The image processing module will attempt to load the image file and convert it into a format suitable for feature extraction.
  - **Feature Extraction:** Next the module will generate the set of features for the image.
  - **Signature Creation:** Once the extraction process is completed the module will store the feature data in a feature vector class, and will return control to the indexer.
- **Signature Indexing:** The indexing varies depending on the type of signature required:
  - **Binary Signature:** The indexer will take the feature vector class returned by the image processing module and request the binary signature. This binary signature is then encoded as an ASCII string for insertion into the MySQL feature table.
  - **Feature Term Signature:** The indexer will request the quantised, labelled, version of the binary signature. The feature vector class will generate the set of feature terms and frequencies and package them as an array for return to the indexer. Once returned the indexer will insert each feature term into the index:
    * The posting is inserted into the *features* table.
    * The relevant record in the *termtotals* table is incremented by 1.
- **Index Update:** The indexing flag in the *images* table is updated to indicate *indexed*, and the other fields in the record are populated accordingly.

4.3.3.1 Parallelisation of Indexing

The generation of image features is a very compute intensive process which called for some method for parallelisation of the indexing process. The chosen method allows multiple instances of the indexer to compete to index images. Each instance of the indexer is passed the same list of files to index together with the database in which the index is to be stored.

Each indexer will perform a file check on all the files in the list. This is necessary since each indexer process may be running on different physical machines. Once each indexer has a list of valid files to index it will begin indexing, checking the index status of each image in the database, and generating a signature for an image if it is not present, or is marked as unindexed. In order to maintain sychronisation of the processes the following procedure is used when indexing:

- **Lock Table *images*** This is required so that no other process can insert a new record, or update an existing record whilst the current process is checking it.
- **Index Check** If the image to process is not in the index then create a new record and mark the index status as *indexing*. If the file exists and is unindexed then mark the index status as *indexing*. For all other cases the table is unlocked and the next image in the list is processed.
- **Unlock Table *images***
- **Generate Signature**
- **Lock Table *features (and termtotals)*** This lock is required so that any processes which may be querying the index do not retrieve partial signature data.
- **Insert Signature**
- **Unlock Table *features (and termtotals)***
- **Lock Table *images***
- **Update Metadata** The last stage is to update the metadata in the *images* table and set the index flag to *indexed*
- **Unlock Table *images***

4.3.4 The Query Operation

The difference between binary and feature term retrievals is quite substantial, warranting an explaination for each. In both cases the retrieval is performed and then a hitlist generated. The hitlist holds entries that contain the filename of the image and its similarity score. This list is sorted using a quick-sort routine. By default the 100 most similar images are listed to the console together with their similarity scores, and if required an HTML results file may be produced which provides icons of the images. The time taken to index the query image, and the overall retrieval time are recorded and included in the output. Where a binary signature database is being queried the time taken to compare each individual signature is also recorded.

4.3.4.1 Binary Signature Retrieval

Binary signature retrieval is performed sequentially and all signatures in the database must be compared. The database is queried for all images which have a valid signature, which returns individual records on demand to the indexer. Each record contains the image details (filename and dimensions) and the signature. The test signature is extracted and a new feature vector class created with it. The test feature vector class and the query feature vector class are then passed to the image processor for comparison. The image processor will return a similarity (or dissimilarity) score for the test image which is recorded. Once all images have been processed the hit list is generated, sorted and the results output.

4.3.4.2 Feature Term Signature Retrieval

Feature term retrieval considers only the images in the database which have features in common with the query. Once the query feature terms have been extracted then retrieval is completed by generating a score for each image which has the same features as the query. The method with which the amount a particular term counts towards an image's score is discussed in chapters 5.1 and 6.1, and will, to a high degree, determine the quality of the retrieval operation.

Inspired by text retrieval techniques the different feature term scoring algorithms require different information about the terms and the images to which they belong. CMR1 requires no information about the frequency of terms since it counts only the number of terms common to the query and test images. CMR2 requires the query term frequency, and CMR3 requires both the query and test image's normalised frequencies. CMR4, which is a TFxIDF scoring algorithm, requires the query and test image frequencies (not normalised) and also the document frequency for each query term with which the TF/IDF weightings are calculated.

4.4 The Index Data Analysis Module

The purpose of the index analysis module is to generate a *rank-frequency* (see 2.7.1) curve for an inverted index. Using this data it is then possible to test whether the index conforms to a Zipfian distribution, and also to approximate the index size for a given number of images.

The algorithm used to calculate the rank-frequency curve is taken from the work of Chen [27], which compounds different approaches for rank-frequency distribution calculation, and provides techniques for analysing the different sections of such distributions.

As a module that was not intended for frequent use, the analysis module was written in Java, and with no advanced data structures. As such the performance of the module, which performs a number of loops over large arrays, is not excellent, but is entirely satisfactory for its purpose. The module is written as two java classes - the analysis class which accepts as input term-frequency tuples and outputs rank-frequency data, and an executable class which provides a method for connecting to and extracting term-frequency data from a particular index via the command line.

This module was used to generate the rank-frequency graphs for the RGB Histogram, the CCV and the QMNS algorithm that are shown in section 6.3. For each algorithm the quantisation level was varied, which has a positive correlation with the vocabulary size, and an index generated. Each of these indexes was then analysed using the analysis module, which produced the data for the graphs.

4.5 Retrieval Results Analysis Module

The final member of the Invistor collection of CBIR components is the retrieval results analysis module. The decision to use the precision-recall (see 2.8) family of metrics to measure the quality of image retrieval from an index, and the requirement to be able to test multiple indexes meant that a flexible method of results generation and analysis was required. The solution needed to be able to apply the same sequence of queries against multiple indexes and generate overall recall-precision metrics for each test, and group of tests.

Inspiration was taken from the Text REtrieval Conference (TREC) which has been running for over a decade. The conference is dedicated to the evaluation of new text retrieval algorithms and techniques, and for each conference sets of documents ranging in size from a few gigabytes to a hundred gigabytes are provided. Contributors index these collections and query their retrieval engines using secondary sets of query documents. Within the collections are ground truth categories - subsets of documents which have been manually indexed - with which the quality of retrieval may be measured.

Queries from each ground truth category are executed against a collection and recall and precision metrics are calculated. Once all queries in a category have been executed then the average precision for that category can be calculated. This metric indicates how good the retrieval engine is at retrieving documents from that category. The averge precision over all categories is then calculated to provide a final, overall, indicator of the quality of the system.

This is the basis behind the results analysis module, which can perform queries in groups and will calculate recall-precision and average precision metrics over different indexes with different algorithms. In order to be as flexible as possible the module runs executable programs and reads the results which it prints to the console (as long as these are in a particular format).

4.5.1 Input File

The input file contains three distinct sections. The first defines the executables that will be used in the tests, the second defines the ground truth categories and the last section sets the combinations of executables, categories and databases that will form the tests.

Because the analysis module was developed in parallel with the core image analysis components it was necessary to ensure that there was plenty of flexiblility in the way in which each executable could be controlled. A simple method was to provide a replacement mechanism that could insert the filename of a query, the name of an SQL database containing an index and the name of a results file from which the query results are read.

Figure 4.3 shows an excerpt of the executables section, showing two different algorithms. The CBIR image analysis module provides two command line switches (-mid and -fid) for selecting the Artiste image processing and signature modules, query and database selection switches (-query and -database) and another for setting the output results file. The -v option enables verbose output.

```
EXECUTABLES: 3
EXECUTABLE: rgb 1
cbir -init
cbir -query \% q -database \% d -res \% r -mid 1010 -fid 1009 -v
EXECUTABLE: qmns-cpr1  1
cbir -init
cbir -query \% q -database \% d -res \% r -mid 1008 -fid 1007 -q 1
>>  -mns rapid_qmns -v
EXECUTABLE: qrgb 1
cbir -init
cbir -query \% q -database \% d -res \% r -mid 1010 -fid 1009 -q 2 -v
```

FIGURE 4.3: Executables section. The $\gg$ symbol denotes a broken line continued from above.

The first executable uses the RGB colour histogram algorithm and will store and retrieve the signature from a binary index. The second executable uses the MNS algorithm, and enables feature term retrieval from an inverted index by giving the -q switch. The parameter supplied with this switch determines the CMR feature term scoring algorithm; for example -q 1 selects CMR1. The -mns switch selects one of the preset MNS parameter configurations. Also provided (but not illustrated here) are switches which allow complete control over MNS parameters.

The third executable uses the RGB colour histogram again, but this time with a feature term index and the CMR2 scoring algorithm. Whilst the CMR (Colour Mode Retrieval) algorithms were originally designed for retrieving quantised MNS features (hence *mode*) they were also perfectly applicable for use with RGB and CCV feature terms.

The next section defines the ground truth categories, with each category being defined by two sets of filenames. The first set of filenames must correspond with images in an index for whom it has been determined share some commonality. In the case of the ground truth categories used throughout this thesis the image categories share high level semantic features (objects, layout and colour). The second set of filenames are for queries that shall be executed against an index. Images from the first set that are retrieved will allow precision and recall metrics to be calculated (see the next section). See Appendix B for the complete pink flowers category.

```
CATEGORIES: 1
CATEGORY: PinkFlowers 4 3
general/06340031.jpg
general/06340032.jpg
general/06340033.jpg
general/06340034.jpg
QUERIES:
general/06340031.jpg
general/06340032.jpg
pinkflower001.jpg
```

FIGURE 4.4: Categories section.

The last section sets the actual tests themselves, declaring the executable that will be used in the test, the categories which will be tested, and the databases against which each category will be tested. In all the tests run for this thesis only one database was used at a time because the recall-precision metrics are calculated across individual tests.

```
TESTS: 1
TEST: rgb-db1 2 1
rgb
PinkFlowers
BeachScenes
RGBDB1
```

FIGURE 4.5: Categories section.

4.5.2 Precision and Recall Calculation

The results analysis module accepts two parameters when run: a valid input filename and a file to write results to. Tests are executed in the order in which they appear in the input file, as are categories within each test, and the queries within each category.

Each executable is expected to output query results in the form of a hitlist formatted as 1 result per line, containing the rank of the result, the filename of the result and the similarity score, delimited by whitespaces. The CBIR program could return a result for all images that are in the database, however there is little point since it is guaranteed that all images from a category would be returned. A user will rarely look at all the results returned by a search engine, instead being only interested in the top results. The default hitlist length in returned by CBIR is 101 results.

The reason for returning 101 results (ranked 0 to 100) is that the query image is in the database, and is always expected to be returned first - since the signature in the database will be identical to the signature generated for the query. This assumption can only be employed where the signature extraction algorithms are deterministic and are guaranteed to always return the same signature for a particular image. This determinism was exploited to speed up the testing process by examining the database for the existence of the query signature. If it exists in the database then the signature is retrieved and the database queried, otherwise the signature is generated first. Note that this does not affect the speed of the retrieval itself. This functionality was only implemented for binary signature indexing, and not for the feature term index.

Once the executable has returned the hitlist is searched for the existence of any of the other images from the query's category. The position in the hitlist of any matches are recorded and a new PR object instantiated to calculate the PR values.

4.5.2.1 PR Class

This class provides for storage, and calculation, of PR metrics. The only data that is required for PR calculation is the size of the hitlist, the number of relevant images in the category that the query belongs to. and the positions in the hitlist at which relevant images occur.

From this information it is possible to plot each point on the graph - the precision and recall components being calculated as in equations 2.20 and 2.21 (chapter 2, section 2.8). This single hitlist graph will typically be very jagged in appearance, with characterisic steps moving from the upper left down to the lower right.

The class is able to calculate the average precision for the query which is equal to the area under the PR graph, using equation 2.22. Another method allows the averaging of multiple PR objects, which is used to combine query results from a category into a single graph, and then the category results into an overall test graph. An averaged PR object is created by taking interpolated precision points at regular intervals along the recall axis from each of the source graphs and finding the average. In all cases the recall axis is sampled at intervals of 0.1, ranging from 0 to 1. The sample at recall value 0, is the initial precision, and provides an indication of the relevance ranking of the top few hits. Similarly the sample at 100% recall indicates the precision that would be obtained by extending the hitlist until all relevant documents have been retrieved (if they haven't already).

4.5.3 Output File

After each category within a test is complete the precision-recall data from the PR objects for each of the queries are output. Formatted in a similar fashion to the the TREC results, each query has the following data output:

- Interpolated Precision-Recall: 11 point interpolated recall-precision points, from 0 to 1 in the recall axis.
- Recall at N Documents: The recall value in the hitlist every 5 documents.
- Precision at N Documents: The precision value in the hitlist every 5 documents.
- Raw Precision-Recall: The precision-recall values at each relevant retrieval.
- Average Precision: The AP for the query.

The individual query PR values are then merged together to form a category PR graph, and the points from this are output together with the category average precision. Similarly, each category PR graph is averaged to form the overall test result.

4.5.4 Subimage Testing

As well as full image matching, the algorithms needed to be tested for their subimage matching capabilities. In such tests precision and recall are meaningless since the retrieval must be evaluated on the presence, or absence, of just one image - the source (or parent) of the subimage.

Subimage testing is enabled by providing just one image in the test category, the parent image, and one image in the query section of that category - the subimage to be matched. A command line switch changes the evaluation metric from precision and recall to the subimage comparison metric (section 5.4.2). This metric evaluates to 1 for a hitlist in which the subimage parent is at rank 1, and 0 if the subimage parent is not retrieved at all. A metric to calculate the subimage performance over a series of subimage queries is also provided, evaluating to 1 where the subimage parent is retrieved at rank 1 for all queries, and 0 where the subimage parent is never retrieved.

4.6 Scalability and Future Work

The Invistor system has a very simple architecture. The Artiste image processing modules have allowed different feature types to be used, but there is no modular indexing capability. The two index types were implemented separately, and allowed little flexbility in their operation. Any future system would require significant architectural reworking, and would be better replaced by an exisiting research framework - such as the GiFT (see section 3.3.7). This freely available package already contains indexing components that could be modified to test alternative indexing and retrieval strategies, and a common GUI which operates on the MRML which would not require any modifications.

The index itself is also very modest, a reflection of the work presented in this thesis which is less concerned with the implementation of an index but with the data which is stored within it. This said, the index would benefit from any of the compression techniques described in section 2.7.

4.7 Summary

The research in this thesis required a system capable of supporting binary signatures and an inverted index, and performing precision and recall evaluation of queries. The collection of applications written fulfil these requirements adequately, allowing the desired research to be carried out effectively.

This chapter has described the different components of the system:

- The CBIR image index which uses the Artiste image processing library and a relational database to store signatures and features terms.
- The results analysis module - a Java application that executes groups of queries with the CBIR indexer, and generates precision and recall data for test sets.
- The term distribution analysis module - another Java application that reads term data from an index and calculates the Zipfian distribution from it.

Chapter 5

QMNS - The Quantised Multimodal Neighbourhood Signature

5.1 Introduction

The chapter presented here introduces the core research in this thesis. A novel global colour algorithm, QMNS, is presented which is implemented for the Invistor retrieval application as a module in the Artiste API.

The process by which QMNS features are extracted is discussed, then the index operations (insertion and retrieval) are described. Following this section is a description of the evaluation of the test algorithms, covering the test image collection and ground truth categories, and the individual test scripts executed. The first part of the test determines the best parameters for the MNS algorithm, comparing with an RGB colour histogram and an RGB colour coherence vector histogram. The best MNS configuration is selected, and put forward into two further tests - full image and subimage tests. Both of these tests use the image collection at different scales. The results for these tests are then presented and discussed, and it is shown that the retrieval algorithm based on TF*IDF is most successful.

An extension to the QMNS algorithm is given, which increases the feature's term vocabulary. The distribution of the new vocabulary is discussed and a series of tests evaluating the different types of QMNS terms are analysed. The results demonstrate that for QMNS it is the common terms which are most important for retrieval, and not the rarest.

The work presented in this chapter has, in part, been published in [142].

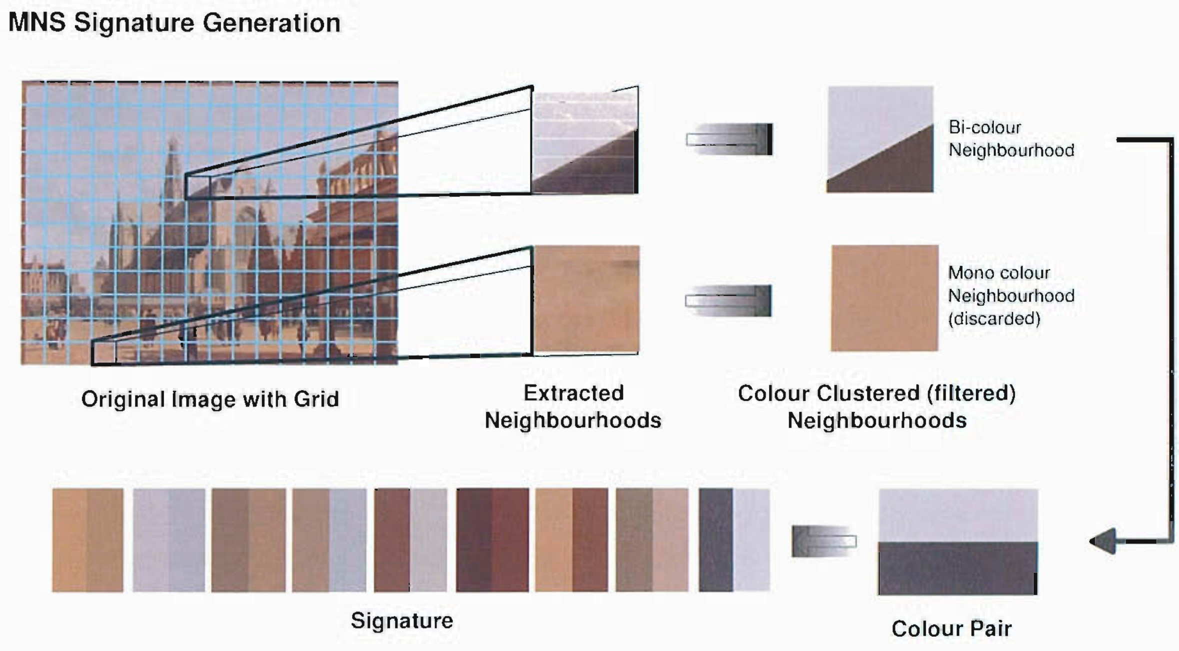

FIGURE 5.1: Generation of MNS Signature

5.2 MNS - The Multimodal Neighbourhood Signature

The MNS algorithm [90, 74, 73, 91], created by Jiri Matas and Dimitri Koubaroulis of the CVSSP group at the University of Surrey, was originally intended for indexing video sequences for the purpose of rapid content-based retrieval, and also to allow tracking of objects through the sequences. As a statistical (low-level) image feature MNS provides only hints as to the actual content of the image, in the form of colour pairs that are prevalent in the scene. These colour pairs form a signature for keyframes from video data which are stored in a sequentially accessed database.

Aimed at applications such as archive footage recovery for television program production and at home for video on demand, the algorithm would be used on massive amounts of video data. The size of the frame signature generated by MNS is typically very small at around 100 bytes for an average frame. The retrieval algorithm that computes signature similarity compensates for differing lighting conditions. As well as invariant colour features the algorithm is not affected by changes in scale, rotation, translation, and it is resistant to noise. Generation of the signature is very rapid, without requiring any spatial segmentation or filtering. In [73] a high match percentage is claimed on the Simon Fraser University [75] image database, and with their own data, a hit rate (percentage correct of images returned for a query) of 92% on a database of video frames is reported in their papers.

As with all low level image feature algorithms, MNS is split into two distinct processes - signature generation (involving feature extraction) and signature comparison. The extraction of the colour pair features follows a three stage process. Firstly the image is split into a grid of neighbourhoods which are perturbed by a small random amount to avoid aliasing problems, and then the colour distribution of each neighbourhood is identified using the *mean shift* algorithm [51, 28, 41, 40]. Neighbourhoods that contain two significant modes (the largest two modes must occupy at least seven eighths of the area of a neighbourhood) have the bi-modal value stored as a six dimensional vector representing the RGB colour pair. This space is denoted RGB^2 space. The set of colour pairs that is most representative of the image is then found using the mean shift in the RGB^2 space, providing the final signature.

The signature comparison process is a form of the stable marriage matching problem [88] between two (not necessarily equally sized) sets in order to determine the distance (or how similar) the two sets are. This problem attempts to match pairs of elements from the two sets such that each pair satisfies some predicate. This is quite similar to the Hausdorff distance [32] which is a measure of the similarity between two sets, and also to the similarity join in databases [53]. The original MNS algorithm was aimed at video databases and so the author developed similarity predicates that were based on physical surface reflectance characteristics, such as the *diagonal model of illumination change* [52]. Here the two sets are composed of the six dimensional colour-pairs from a query image and a test image. The matching algorithm first builds a matrix of all the pairwise distances between the two feature sets, then orders these values in a list. It then moves along this list from minimal pair distance to maximal taking the first occurrence of each feature in the query set and adding the distance to the image's score. The algorithm penalises any unmatched query features by adding a fixed penalty for each unmatched feature - this implies that the number of query features should be lower than the number of test features and so the algorithm is trying to match a sub-image. The more query features that are matched to test features, the lower the overall score, and the more similar the signatures and hence images.

5.3 QMNS - The Quantised MNS and the Colour Mode Retrieval Algorithms

In order to label colour pair features for inclusion in an inverted index the RGB^2 feature space must partitioned in some manner. The most logical method and the best starting point was to create evenly sized bins in the feature space like those made in the three dimensional space for colour histograms. Initially we chose to use a division of 4 units per axis giving $4^6 = 4096$ bins in the feature space, although other divisions were examined for suitability. We treat these partitions, or bins as being equivalent to terms in a text index, and are sometimes referred to as *atomic index items*. Whilst *bins* and *terms* are synonymous we shall use *bin* to describe the partitions in feature space and the word *term*, or phrase *feature term* to refer to a particular bin in the feature space that has a frequency of features associated with it.

5.3.1 Implementation of the Index

A generalised inverted index in its raw form is quite a simple structure, making implementation straightforward. A straightforward implementation will however, not supply the performance that is required from a data structure that may hold hundreds of thousands of data entries. The main consideration for designing a fast inverted index is to consider the top level - the term level. A typical query will be to retrieve all the postings for one, or a number of, terms. Rather than adopt the most common and efficient approach, a B-Tree of terms pointing to a flat file of postings, we have chosen to implement the index in a relational database. This has allowed us to concentrate on the functionality of the index application and the retrieval algorithms, and leave the data management, multi-user transactions, back-up and user-control to the DBMS.

We chose to use the MySQL relational database primarily due to its availability. Whilst MySQL does not yet support all SQL (Standard Query Language) commands or nested statements, BLOBS (Binary Large OBjectS) are a supported data type, and the tables can be indexed allowing for very fast access. In [111] Putz describes advanced techniques for creating and maintaining an inverted text index using a fully featured T-SQL (Transactional-SQL) relational database server.

Two approaches for creating the index as tables are available - one which uses two tables and another that uses just one. The first type is more efficient in terms of storage, however under MySQL the time performance difference between the two is negligible due to table and column indexing. We chose the second option since it is less complicated to update and allowed us to make modifications to our algorithms very easily. Section 4.3.1.2 describes the relational tables used in the index.

5.3.2 MNS Quantisation

The last phase of MNS signature generation is to find the modes of all the colour pairs in RGB^2 space using the mean shift. Since this has a clustering effect we have omitted it from the quantised signature generation. A simple partitioning algorithm, shown in figure 5.2, transforms an N dimensional vector into an N digit decimal coded representation of the feature space. Each of the digits ranges from 1 (indicating the bin that is closest to zero for a particular component) to the number of bins required per component. To illustrate take the RGB colour pair $< BLACK, WHITE >$ which is represented in RGB^2 space by the point $(0, 0, 0, 255, 255, 255)$. When the feature space is quantised by 4 it becomes a value in the bin represented by 111444.

```
double MAX:             // Maximum value of components in the vector
double VECTOR[]:        // Array containing vector values
int    COMPONENTS;      // Number of components in the vector
int    BINS:            // Number of partitions in each axis
int    bin;             // The decimal coded bin


bin = 0;                                              // Set bin label to zero
for (int i=0; i<COMPONENTS; i++)                      // For each component in the vector
{                                                                                                   10
    for (int x=0; x<BINS; x++)                        // And for each partition
    {                                                 // Calculate which partition the value is in
        if( ( VECTOR[i] >= x * (MAX/BINS) ) && (VECTOR[i] <= (x+1)*(MAX/BINS) ) )
        {
            bin += (int)(BINS - x) * pow(10,i);       // Add appropriate bin at appropriate power of 10
            break;                                    // Next
        }
    }
}
```

FIGURE 5.2: Feature Term Labelling Procedure

Since the term is in an integer form a single term lookup in the index table will be an atomic integer lookup - MySQL uses the table index to find all occurrences of an entry in a column which due to the nature of MySQL's index will be in a group. This means that all the postings pertaining to a particular term will be recovered very rapidly.

A second index table is employed to record the frequency of a particular term - its *document frequency* which is required by the TF/IDF scoring algorithm presented in the next section. Together with a metadata table that describes each image and its current index status this table completes the relational database component of our inverted index.

5.3.3 Operations on the Index

Rather than develop all four methods of the CRUD (Create, Retrieve, Update and Delete) database paradigm we found it was not necessary to define an Update algorithm - since our database is not intended for rigorous updates a delete followed by a create suffices.

5.3.3.1 Term Insertion

The insert operation places a new entry for an image into the database. Firstly the metadata table is queried to check whether another process is already trying to index the given image. If there is no entry one is created and the entry marked as *indexing*. The MNS signature is then generated and quantised. At this point the inverted index and document frequency tables are locked for both read and write operations. Each feature term is then inserted into the inverted index table, as the four place tuple $< ImageID, BinNo, NormFreq, IntFreq >$. ImageID corresponds to the unique integer created for the image in the metadata table. BinNo is the integer number of the bin that the feature term refers to. The IntFreq component is the frequency of the feature term - that is the number of colour pair features that occurred within the bin that the feature term refers to. The floating point NormFreq component is the IntFreq value normalised by the maximum possible number of individual features in the image, which is the total number of neighbourhoods in the image. After each term is inserted into the index table, the document frequency total table must be updated for that feature term. The two index tables can now be safely unlocked. To end the operation the metadata entry for the image is set to indicate a successful index operation.

5.3.3.2 Term Deletion

The delete operation will remove a term from the index and decrease the document frequency for the feature term across the entire index. All tables are locked for the table updates. The posting for the term is retrieved from the index table and then deleted. Next the terms document frequency is retrieved, then decremented by the frequency retrieved in the posting and then updated. The tables are then unlocked.

5.3.3.3 Term Retrieval

The retrieval operation should not be confused with an image query. A single index retrieval will retrieve all the postings for a particular term - the retrieval term. All postings for the term are retrieved from the inverted index table using an SQL query. A single term retrieval does not require the index tables to be locked.

5.3.4 Querying the Index

A complete image query consists of a term retrieval for each feature term from the query image, and then a process of merging the postings to form a similarity ranked list of images in the database. Each of our four QMNS algorithms takes a different approach to this merging and scoring process. The four CPR (Colour Pair Retrieval) algorithms follow the same outline process, as illustrated below. The first step is to generate all the QMNS feature terms for the query image. For each of these feature terms a term retrieval is performed on the index, and then for each of the postings returned a running total of image scores are updated according to the particular CPR algorithm. The following is an outline description of the matching algorithm:

- Generate QMNS signature and extract feature terms for the query image
- FOR ALL feature terms in the query signature with a non-zero frequency
  - Retrieve Document Frequency for query term
  - Perform SQL query for postings matching the query term on the inverted index table

V	the set of all terms
D	the set of all image
N	the number of images
g	the number of neighbourhoods in the query
f_j	the number of neighbourhoods in image j
Q	the set of query terms
T_j	the set of terms in image j
q_i	the frequency of query term i
t_{ij}	the frequency of term i in image j
q'_i	the normalised frequency of query term i
t'_{ij}	the normalised frequency of term i in image j
d_i	the document frequency of term i
c	the document frequency cut off fraction

TABLE 5.1: Definitions for the CPR Algorithms

- FOR ALL postings returned
    * IF CPR1 THEN Increment posting image score by 1
    * ELSE IF CPR2 THEN Increment posting image score by the integer *query* term frequency
    * ELSE IF CPR3 THEN Increment posting image score by weighted, normalised term frequency
    * ELSE IF CPR4 THEN Increment posting image score by TF/IDF weighted *query* term frequency

- Rank by ascending score: Highest score is most similar

5.3.5 CPR1 - Original Algorithm

CPR1 is the simplest algorithm, awarding 1 for every feature term in an image that is also in the query image. Since the score is integer this version of the algorithm suffers when the results are ranked - many images will share the same rank.

$$score_j = \sum_{\forall i \in (Q \cap T_j) : i \neq 0} 1 \tag{5.1}$$

5.3.6 CPR2 - Sub-Image Matching

CPR2 takes advantage of the term frequency information stored in the postings in order to favour images that contain the query as a sub-image. For each feature term that is also in the query image an image will have its score incremented by the integer *query* term frequency. This gives greater weight to terms which occur frequently in the query image. The algorithm is designed to best match images that contain the query image at any size - it is scale invariant.

$$score_j = \sum_{\forall\, i \in (Q \cap T_j) : i \neq 0} q_i \tag{5.2}$$

5.3.7 CPR3 - Full Image Matching

This version of the algorithm is aimed at situations where a full image, scale independent match is required. For each feature term that is also in the query image an image will have its score incremented by a *weighted normalised* term frequency. The term frequency is normalised by the maximum number of features in the image.

The query terms are weighted such that the closer the query term frequency is to database image term frequency the higher the score awarded. This benefits images where a feature term is present in the same ratio as the query. We have also adopted a process of eliminating feature terms that are common across the corpus, the document frequency cut-off point, denoted by c, for these terms ranging between 25% and 75% of the images in the corpus. This is analogous to removing stop words in text, and has the effect of speeding up the retrieval by reducing the number of index lookups that need to be performed. Equations 5.3 and 5.4 give the normalised term frequencies for a query image term and a test image term.

$$q'_i = \frac{q_i}{g} \tag{5.3}$$

$$t'_{ij} = \frac{t_{ij}}{f_j} \tag{5.4}$$

$$score_j = \sum_{\forall\, i \in (Q \cap T_j) : i \neq 0, d_i < \frac{N}{c}} q'_i \cdot \left(1 - \frac{|q'_i - t'_{ij}|}{max(q'_i, t'_{ij})}\right) \tag{5.5}$$

5.3.8 CPR4 - Weighted Full Image Matching

The fourth algorithm has its roots in text retrieval, being based on the TF/IDF weighted scoring method for inverted text indices. We have employed this technique since analysis of the distribution of QMNS features terms throughout our primary image database showed large numbers of features that occur in the majority of images. Whilst using a cut-off for some of these common features is useful, it does not penalise other terms that still occur in large quantities. Like in text retrieval these feature terms are deemed to not have very good discriminatory power. We chose to use the log of the inverse document frequency (equation 2.11).

Category	Description	Images
Beach Scenes	Images containing scenes of beaches	15
Pink Flowers	Images of pink flowers on a background of foliage	10
Buildings at Night	Images of famous buildings illuminated at night	20
Sea and Sky	Images containing views of the sea and or sky	47
Sea World	Images from Sea World	6

TABLE 5.2: The categories selected from the General image collection

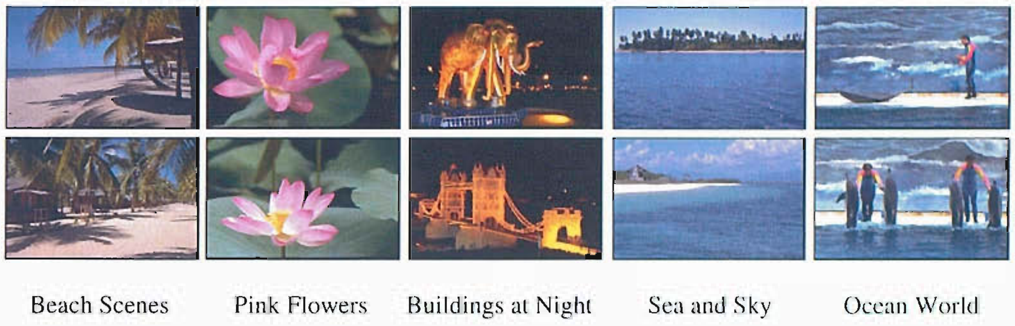

FIGURE 5.3: Example Images From the Collection

$$score_j = \sum_{\forall i \in (Q \cap T_j): i \neq 0, d_i < \frac{N}{c}} q_i \cdot log(\frac{N}{d_i}) \tag{5.6}$$

5.4 Evaluation of QMNS

5.4.1 Image Collections

One image dataset was used in the testing. It contains 769 digitally photographed images of a wide variety of subjects and locations and has been titled the 'general' collection. Whilst the images are very diverse there are also a number of distinct image categories contained within. Table 5.2and figure 5.3 show examples of these categories. The image collection has also been scaled to form two collections that are half and double the size of the original set in order to be able to test the effectiveness of algorithms when querying a collection that is at a different scale. The scaling algorithm used was a bilinear interpolation.

5.4.2 Evaluation Metrics

In order to measure the effectiveness of the algorithms precision and recall metrics (described in section 2.8) have been used which provide a clear, overall indicator to the performance of the algorithm. These metrics, represented by graphs, are interpolated and averaged across each query to provide results for each of the image categories and tests.

Subimage tests require a slightly different metric since it is only a single image that we are attempting to retrieve. Given an ordered set that contains N images returned for a sub image query, containing the correct full image at rank i, a suitable metric is:

$$\frac{(N+1)-i}{N} \tag{5.7}$$

Which will give 1 for a perfectly ranked image and zero for an image that is not retrieved at all. If we perform T subimage tests, and use r_i to record the number of queries for which the correct image is retrieved at rank i then an overall metric is:

$$\frac{\sum_{i=1..N} \frac{(N+1)-i}{N} \cdot r_i}{T} \tag{5.8}$$

5.4.3 Test Harnesses

Two test harnesses have been developed for running tests using each of the algorithms and for extracting precision and recall data. The first harness, a signature generator and storage application named CBIR, uses image processing modules written for the Artiste project [57] to create signatures which are stored as binary objects in a MySQL database. The Artiste image processing API supports two main functions - signature generation and signature comparison. In normal use the database may be queried sequentially to build a ranked list of images in the database that are similar, presented as an HTML document. The CBIR test harness also contains the QMNS specific code, and in the application's second mode access to the inverted index in MySQL is enabled.

The second test harness is a Java application that runs individual queries using CBIR and generates precision and recall data. This simple program takes lists of files that are known to belong to a category *a priori* and sets of queries that should be run, the databases to query and the algorithms with which to query them. All query data is saved and the program will generate all precision and recall data for a test, interpolate and calculate average and overall test results.

5.4.4 Computer Hardware

Primarily one computer system was used for all development and testing. This system is an HP LT 6000r Server with Six 700MHz Xeon processors, and 1.7GB 133MHz ECC SDRAM. This is supported by ten 36GB disks (10k rpm Ultra2) and two 74GB disks in a RAID configuration.

5.5 Test Specification

The evaluation of QMNS was been divided into three separate areas. The first test was needed to help determine the best parameters for the MNS algorithm on the image sets being used. The second test measured the effectiveness for retrieving similar images from a database when using a complete image as a query. The third test was to determine how well the algorithms perform at retrieval from a dataset when using a randomly selected sub-image extracted from some of the images. Every test was performed on each of the three differently scaled image sets.

5.5.1 Test Algorithms

The algorithms tested are as follows: 18 variations of the MNS algorithm, including 2 that use a 5 dimensional chrominance feature rather than the 6 dimensional RGB^2 one. There are then 8 versions of the QMNS algorithms - 1 for each of the CPR1, CPR2 and CPR3 variations, and then 5 for the CPR4 variation. A table presenting the parameters changed is given in Appendix A, table A.2, followed by a written description of the algorithm. Two variations of the QMNS algorithm are called *3 Bin QMNS* and *5 Bin QMNS*. The numbers refer not to the total number of bins, or terms but to the divisions along each dimension of feature space.

Before testing was performed all the signature indexes needed to be created. Thirteen different databases were generated for the MNS variations, five were generated for QMNS, and one database each for the CCV, RGB Histogram and MCCV algorithms. This process was repeated for each of the image scales, resulting in 63 signature indexes. Once all the indexes had been generated, metadata was extracted from the databases. For the MNS and QMNS algorithms and variants this included index time and feature statistics, but for all others only time statistics were recorded.

Test 1 : Variations of parameters within the MNS algorithm.

- TEST OBJECTIVE
  - To determine the best parameters of the base MNS algorithm for different datasets.
- TEST METHOD
  - For each of the adjustable parameters in MNS, perform tests with the parameter varied. Measurements are based on precision and recall metrics. Refer to Appendix A for the parameters descriptions.
  - Use all five categories and use all images in each category as a query. Perform retrievals and calculate precision and recall metrics for each test, then average for each category, and then for each algorithm.
  - The test shall be run against each scale of the three image collections.
- TEST OUTCOMES

- Precision and recall data for each of the different variations of the MNS algorithm at individual query, category and test levels.
- Minimum, average and maximum retrieval speeds for each different parameter change at each image scale.

Test 2 : Evaluation of each QMNS algorithm for full image retrieval.

- Test Objective
- To determine whether CPR3 and CPR4 display better full image retrieval performance than other algorithms.
- Test Method
- Use all five categories and use all images in each category as a query. Perform retrievals and calculate precision and recall metrics for each test, then average for each category, and then for each algorithm.
- The test shall be run against each scale of the three image collections.
- Test Outcomes
- Precision and recall data for all QMNS algorithms, CCV and an RGB Colour Histogram at individual query, category and test levels.
- Minimum, average and maximum retrieval speeds for each algorithm at each image scale.

Test 3 : Evaluation of each QMNS algorithm for subimage retrieval.

- Test Objective
- To determine whether CPR2 is better than other QMNS, and MCCV algorithms at sub-image retrieval.
- Test Method
- Use all images in the randomly extracted subimage collection as queries for each of the QMNS, Base MNS, RGB Histogram and MCCV algorithms.
- The test shall be run against each scale of the three image collections.
- Test Outcomes
- Subimage metrics for all QMNS algorithms, MCCV and an RGB Colour Histogram.
- Minimum, average and maximum sub-image retrieval speeds for each algorithm at each image scale.

The first test was run with 13 variations of the MNS algorithm. The parameters for the MNS variations are explained and given in Appendix A. The first and second tests used all of the images from the 5 categories in the general collection as queries with the precision and recall metrics from each query being averaged across each category, and then the averaged values averaged across the whole test. The third test for sub-image accuracy used a 100 sub-images extracted randomly from the images. The sub-images were restricted to a minimum size of 64x64 pixels or 10% of the original image dimensions and a maximum size of 50%. Measuring the effectiveness of sub-image queries was a matter of performing a query and finding the rank of the full image that the sub-image was extracted from. Overall effectiveness was calculated using equation 5.8.

```
------------------------------------------------------------
Category:PinkFlowers Query Image:general/06340031.jpg
------------------------------------------------------------

Recall - Precision values:
at 0 1
at 0.1 1
at 0.2 1
at 0.3 1
at 0.4 1
at 0.5 0.833
at 0.6 0.75
at 0.7 0.778
at 0.8 0.8
at 0.9 0.191
at 1 0.191

Recall:
at 5 docs: 0.4
at 10 docs: 0.8
at 15 docs: 0.8
at 20 docs: 0.8
at 25 docs: 0.8
at 30 docs: 0.8
at 35 docs: 0.8
at 40 docs: 0.8
at 45 docs: 0.8
at 50 docs: 0.9
at 55 docs: 0.9
at 60 docs: 0.9
at 65 docs: 0.9
at 70 docs: 0.9
at 75 docs: 0.9
at 80 docs: 0.9
at 85 docs: 0.9
at 90 docs: 0.9
at 95 docs: 0.9
Precision:
at 5 docs: 0.8
at 10 docs: 0.8
at 15 docs: 0.533
at 20 docs: 0.4
at 25 docs: 0.32
at 30 docs: 0.267
at 35 docs: 0.229
at 40 docs: 0.2
at 45 docs: 0.178
at 50 docs: 0.18
at 55 docs: 0.164
at 60 docs: 0.15
at 65 docs: 0.138
at 70 docs: 0.129
at 75 docs: 0.12
at 80 docs: 0.112
at 85 docs: 0.106
at 90 docs: 0.1
at 95 docs: 0.095

Recall and Precision:
0.1 1
0.2 1
0.3 1
0.4 1
0.5 0.833
0.6 0.75
0.7 0.778
0.8 0.8
0.9 0.191

Average Precision for All Points:
Avg 0.735
```

FIGURE 5.4: An Example of the Precision and Recall Data Produced

5.6 Results

The Java test harness outputs test results in a format similar to that output by the SMART database used in conjunction with the TRECEVAL program developed for the TREC (Text REtrieval Conference) conferences. For each test in a batch the test description is printed together with a list of the image categories that are to be tested against and the executable file to be used. Following this the precision and recall data for each query (as illustrated in table 5.4) are presented followed by the category averages. Once all the category results in a test have been printed the overall average results are listed.

When the test harness is processing subimage results the precision and recall data are not presented since they are not relevant. Instead the ranking of the correct full image for a query is given, or zero if it was not in the top 100 results.

5.6.1 Test Group 1 - MNS Variations

Figure 5.5 and table 5.3 present the indexing speeds for the different MNS variations compared with the three histogram algorithms - RGB Histogram, CCV and MCCV. Eight of the variations (Base MNS, Fixed MNS, Rapid MNS (Random 0.5), Enhanced MNS, Reduced MNS, Large Mode MNS and Small Mode MNS) all have approximately the same indexing times for each of the three scales of image collection. This is easily

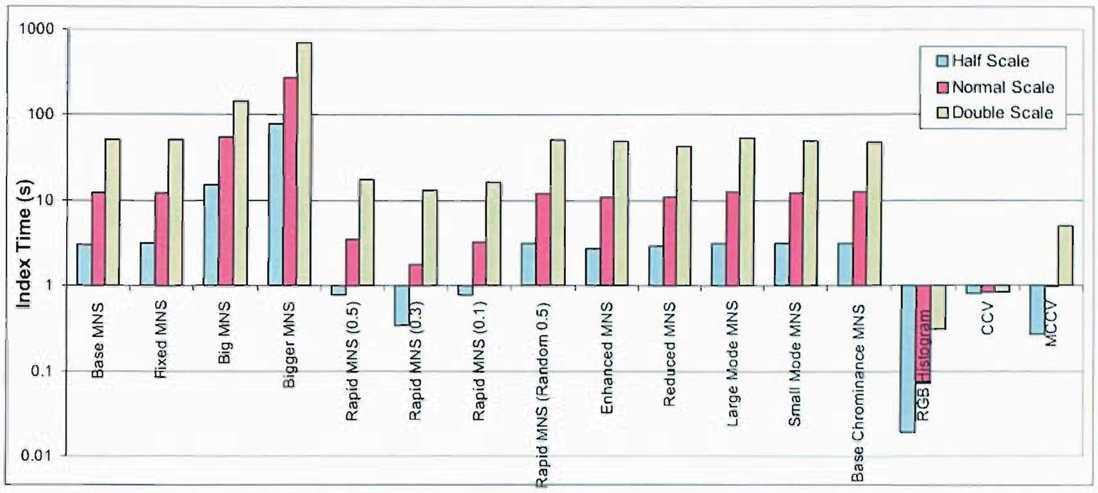

FIGURE 5.5: MNS Variation Indexing Times

Algorithm	Indexing Time (s)			Average Colour Pair Features		
	Half	Normal	Double	Half	Normal	Double
Base MNS	3.092	12.395	51.025	8.437	10.808	10.663
Fixed MNS	3.138	12.263	50.660	8.363	10.718	10.584
Big MNS	15.095	55.251	146.919	6.316	9.286	11.408
Bigger MNS	77.865	276.060	700.809	3.792	6.540	9.514
Rapid MNS (0.5)	0.795	3.488	17.845	8.137	10.574	10.155
Rapid MNS (0.3)	0.348	1.808	13.375	9.155	11.646	10.957
Rapid MNS (0.1)	0.774	3.299	16.268	6.049	8.283	6.618
Rapid MNS (Random 0.5)	3.159	12.303	50.793	8.481	10.743	10.644
Enhanced MNS	2.773	11.091	48.820	26.814	45.628	69.523
Reduced MNS	2.967	11.230	42.682	2.192	2.239	1.817
Large Mode MNS	3.165	12.564	52.355	9.027	11.403	11.752
Small Mode MNS	3.152	12.464	49.798	7.395	9.849	9.299
Base Chrominance MNS	3.143	12.625	48.403	14.140	22.917	26.613
RGB Histogram	0.019	0.072	0.311			
CCV	0.818	0.828	0.829			
MCCV	0.265	0.986	5.085			

TABLE 5.3: MNS Variation Indexing Times and Average Feature Statistics

explained by the MNS generation process - the parameters for these variations do not alter or add to the overall complexity of the process. The parameter altered in Big MNS, and Bigger MNS has a profound affect on the indexing time. These two variations have a neighbourhood size of 16x16 pixels and 32x32 pixels respectively, compared to 8x8. This results in 256 and 1024 pixel RGB triples being passed to the mean-shift algorithm for modality derivation instead of just 64. Our implementation of the mean-shift is not efficient and suffers from $O(N^2)$ complexity ([33] presents an efficient algorithm for the mean-shift). Since the number of neighbourhoods varies linearly with the area of neighbourhood to total image area these two results are unsurprising.

An alternative approach is to use less of the pixels in the neighbourhood to determine the modes, and the first three Rapid versions of MNS show a satisfactory decrease in indexing time for images, however Rapid MNS (0.1), which uses a tenth of the pixel data for each neighbourhood, is slower than Rapid MNS (0.3) which uses a third. The results for this variation are invalid due to the incorrect configuration of the parameter set; the results are included here for completeness.

All the versions are much slower at indexing images than the histogram techniques -

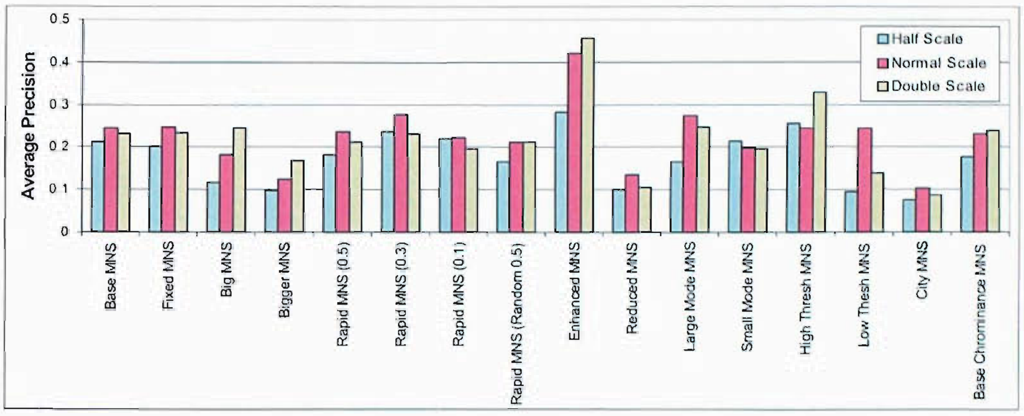

FIGURE 5.6: MNS Variations Average Precision

probably due to the poor implementation of the mean-shift algorithm and other overheads associated with calculating neighbourhood modality. Attention is drawn to the CCV results, which do not display an increase in indexing time that is exponentially proportional to image size - in fact all three scales of image are indexed in the same average time. This is most likely due to a process of normalising the image to a particular size before extracting the coherent and non-coherent histograms.

Figure 5.6 presents the overall average precision values for each of the MNS variations at each of the image collection scales. The majority of the variations display retrieval accuracy that is slightly lower for both the half and double scale collections, although there are some exceptions to this. Increasing the size of the neighbourhoods has a negative effect on accuracy, more so for the Bigger MNS algorithm, although both of these variations display better retrieval on the larger scale collection. This can be expected since the query images are taken from the normal sized collection, and so using neighbourhoods that are double the normal size on images that are also double the normal size will result in approximately the same feature set for the same query and database.

Analysing only half of the pixels in each neighbourhood results in only slightly lower precision levels, as demonstrated by Rapid MNS (0.5). Using a third of the pixels results in better retrieval rates, and this could be explained by a reduction in noise levels that result in a more accurate representation of the colour modes of a neighbourhood. Rapid MNS (0.1) offers worse performance since the sampling rate is too low and the colour modes become less accurate, however the tradeoff between retrieval accuracy and indexing speed is still worthwhile.

Creating more image features by reducing the size of the window used by the meanshift at both the neighbourhood and feature clustering stages results in a considerable increase in performance. Retrieval rates for Enhanced MNS with the double scale dataset are nearly twice those of the Base MNS algorithm. The normal scale is also considerably better, however performance for the half scale set do not show the same level of increase.

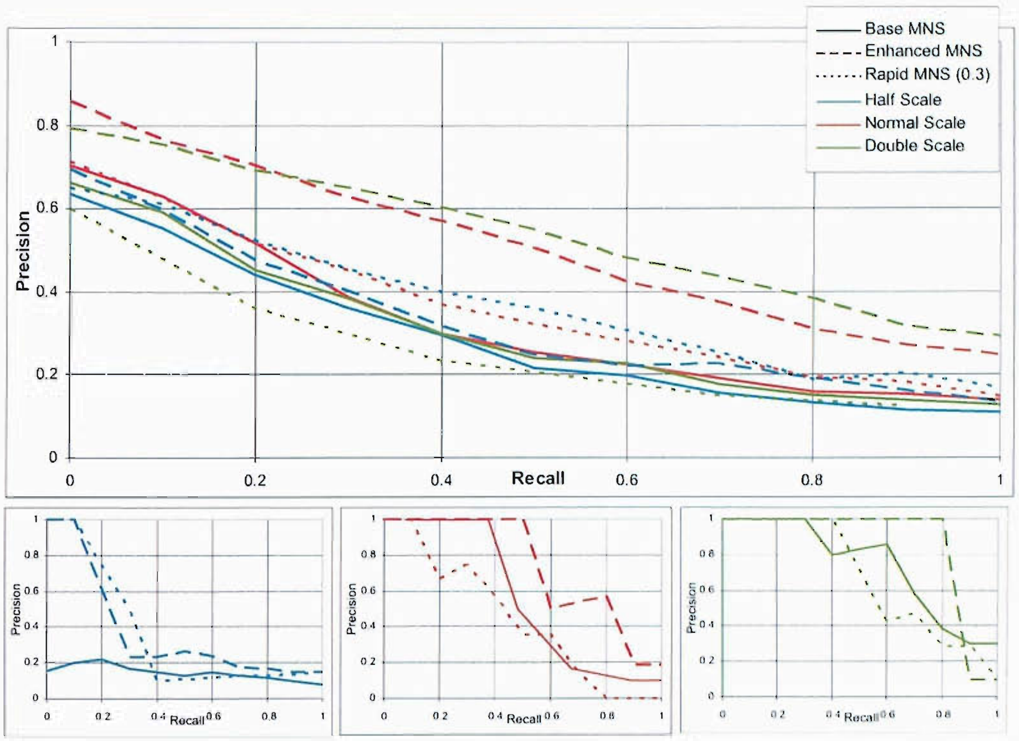

FIGURE 5.7: MNS Variations Test Average and Individual Query Precision and Recall Curves

Figure 5.7 presents four recall-precision curves for three of the MNS variations. The main graph shows overall averaged and interpolated curves for an entire test run. The three smaller graphs show curves for a single query at the three different image scales. Since the area under a recall-precision curve is the average precision a response that extends into the top right indicates that a particular algorithm is good. In the individual queries this can be seen very well with the Enhanced MNS algorithm being clearly better than the other two for the normal and double scale collections - in both cases achieving perfect precision for over half of the documents in the category being queried.

The results from this test have shown that increasing the number of features that are extracted from the image has a very positive effect on retrieval rates, and that using less of the pixels in a neighbourhood still provides very good results. A variation that employs both of these parameter changes should be tested to see whether the same retrieval rates for Enhanced MNS are achieved with the same indexing speeds as Rapid MNS (0.3). The Enhanced MNS variation has been put forward to be tested with the QMNS algorithms

5.6.2 Test Group 2 - Full Image Tests

Since both MNS and QMNS algorithms are similar apart from signature generation, the indexing times for QMNS shown in figure 5.8 are nearly the same as for MNS. This

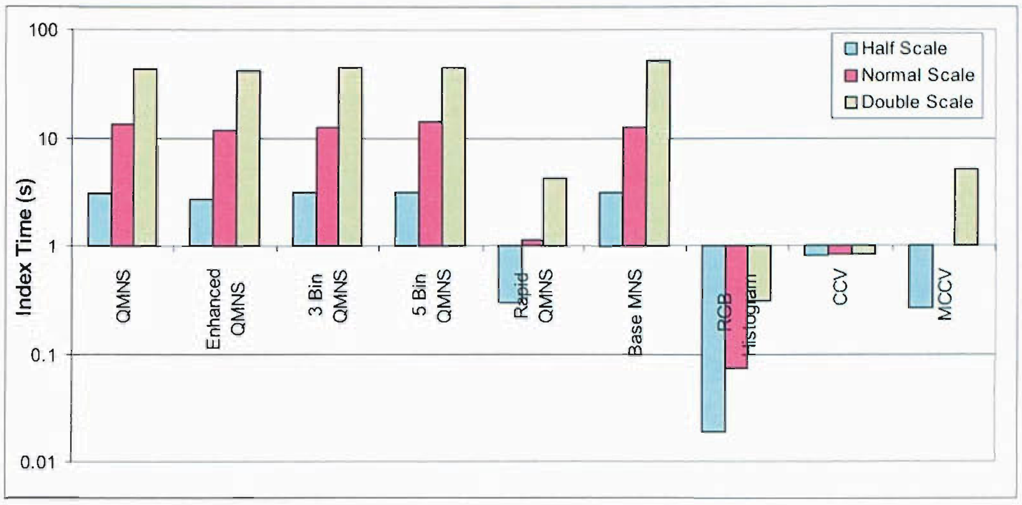

FIGURE 5.8: QMNS Variation Indexing Times

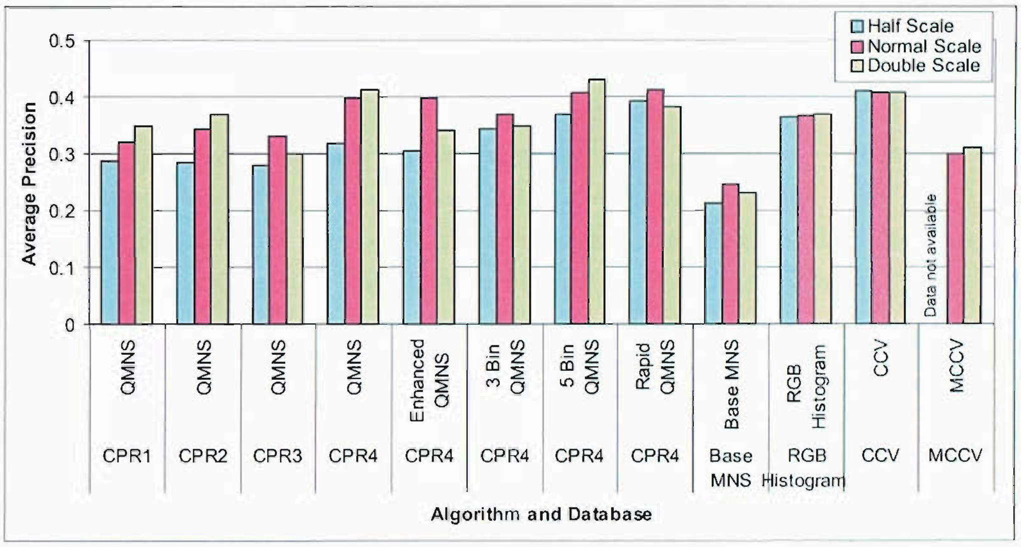

FIGURE 5.9: QMNS Variations Average Precision

second test compares the QMNS algorithms (both indexing and retrieval parts) against a CCV and a Multi-Scale CCV. RGB histograms display excellent indexing times as do CCVs. The Multiscaled CCV (MCCV) is indexed in the same time as Rapid QMNS (using the Rapid MNS (0.3) variation).

Figure 5.9 shows average precision values for the different combinations of datasets and retrieval algorithms (the database used is shown vertically above the retrieval algorithm) and clearly demonstrates that QMNS has better precision than just MNS alone. Whilst the first three CPR algorithms show little difference in performance, CPR4, which uses logarithmic term weighting, outperforms them. Using the same signature generation parameters as Enhanced MNS does not make a significant difference to the retrieval for CPR4, in fact overall the results are worse for Enhanced QMNS than for the Base

Algorithm	Index	Overall Test Average Precision				Average Retrieval
		Half Scale	Normal Scale	Double Scale	Overall	Time (s)
CPR1	QMNS	0.286	0.320	0.348	**0.318**	11.649
CPR2	QMNS	0.285	0.344	0.369	**0.333**	11.650
CPR3	QMNS	0.279	0.330	0.301	**0.303**	11.575
CPR4	QMNS	0.319	0.397	0.412	**0.376**	11.575
CPR4	Enhanced QMNS	0.304	0.398	0.341	**0.348**	10.624
CPR4	3 Bin QMNS	0.344	0.369	0.350	**0.354**	11.490
CPR4	5 Bin QMNS	0.369	0.408	0.431	**0.403**	11.533
CPR4	Rapid QMNS	0.393	0.413	0.382	**0.396**	1.522
Base MNS	Base MNS	0.212	0.245	0.232	**0.230**	14.122
RGB Histogram	RGB Histogram	0.365	0.367	0.369	**0.367**	0.198
CCV	CCV	0.410	0.408	0.407	**0.408**	0.463
MCCV	MCCV	[illegible]	0.300	0.311	**0.306**	12.526

TABLE 5.4: QMNS Overall Average Precision and Average Retrieval Times

QMNS index. The normal scale results for Enhanced QMNS/CPR4 are good, but the half and double scale results are worse.

Using 3 bins to partition each dimension in the 6D feature space, which results in less possible feature terms, also has a slightly detrimental effect when CPR4 is used for retrieving the features, yet using five has a positive effect - overall reaching slightly above QMNS/CPR4. A very encouraging result is for Rapid QMNS/CPR4, which together give very good retrieval performance with good indexing times.

Table 5.4 gives overall average precision and average retrieval times. The retrieval time *includes* the time taken to generate the image signature, and clearly shows that using the QMNS inverted index is as effective as using a sequential index. Whilst the time for the histogram features is very low the difference between any of the QMNS indexes and retrieval algorithms is nearly three seconds, with Rapid QMNS being over twelve seconds faster.

These results are disappointing in the sense that the CPR3 retrieval algorithm was designed to provide scale independent retrieval, yet overall it scores the worst in these tests, behind even CPR1 the prototype index retrieval algorithm. Overall CPR1, CPR2 and CPR3 offer full image retrieval performance that is very similar. Like the MNS Variation tests, the results for the double scale collection are sometimes better than the normal scale collection, but rarely worse than the half scale collection. It is highly likely that this is due to the way the MNS algorithm extracts features from neighbourhoods which stay the same size, no matter what the image scale is. This could have a positive effect in some cases since a particular area in an image will be larger at a large scale, and so more neighbourhoods will occupy the same area than for a smaller scale image. CPR3 uses normalized feature frequencies in order to combat this scale problem, although the results do not indicate that this has helped the situation.

The results for CPR4 are very good, especially when used with the Rapid QMNS indexing algorithm, and make any further enhancements to the algorithms very promising.

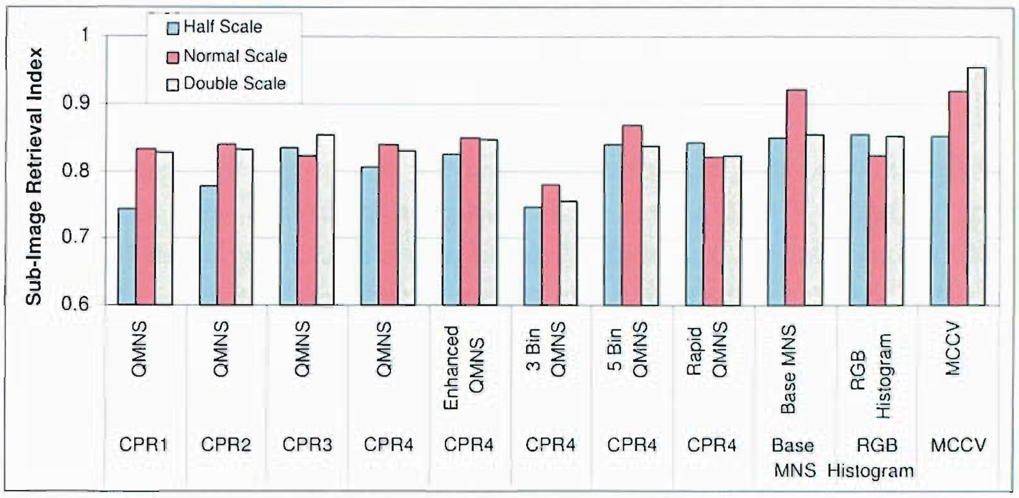

FIGURE 5.10: Subimage Retrieval Metrics

Algorithm	Database	Retrieval Index			
		Half Scale	Normal Scale	Double Scale	Average
CPR1	QMNS	0.744	0.833	0.827	0.802
CPR2	QMNS	0.778	0.839	0.832	0.816
CPR3	QMNS	0.834	0.824	0.853	0.837
CPR4	QMNS	0.806	0.839	0.830	0.825
CPR4	Enhanced QMNS	0.825	0.850	0.847	0.841
CPR4	3 Bin QMNS	0.747	0.780	0.755	0.761
CPR4	5 Bin QMNS	0.839	0.868	0.838	0.849
CPR4	Rapid QMNS	0.842	0.820	0.822	0.828
Base MNS	Base MNS	0.849	0.921	0.854	0.875
RGB Histogram	RGB Histogram	0.855	0.822	0.853	0.843
MCCV	MCCV	0.852	0.919	0.955	0.909

TABLE 5.5: Subimage Retrieval Metrics - Averages

5.6.3 Test Group 3 - Sub Image Tests

Graph 5.10 and Table 5.5 presents the subimage retrieval metrics (equation 5.8) from the second test. Like in the previous test, the four CPR algorithms achieve very similar results with the QMNS index. CPR3 does however have a marginally higher average score, even though CPR2 was designed to be better at sub-image retrieval. Again using the Enhanced index results in better performance, again by a slight amount.

The index created with 3 bins offers worse average performance, yet the index with 5 bins offers better than average performance. The explanation behind this again may lie in the difference fact that more bins results in features that are more distinctive, and hence are better discriminators.

Again the Rapid index performs very well, proving that it is a very practical and useful improvement over the original signature generation variation.

The three control algorithms all perform better compared to the QMNS variations, with the Base MNS algorithm outperforming QMNS. Interestingly Base MNS does not perform well at all scale levels, and a very good score for the normal scaled collection is not reflected in the half and double scale collections. A general explanation for why any MNS algorithm does not work well in such a situation is provided by the positioning of neighbourhoods. Whilst the neighbourhoods are shifted randomly from their regular grid layout the positioning of the random subimages could be in any location, and so the neighbourhoods lie over different features.

The MCCV algorithm outperforms all the others and would perhaps have performed better, if not for some anomaly that appeared in the results. MCCV scores highly, yet almost half of the correct images are ranked at third, none at second and only four in first. The same is also true for the other results, although to a lesser extent. This would appear to indicate that there was something wrong with the test harness that produced the results, however closer inspection of the test data indicates that the results are indeed correct, and that another unknown factor seems to have caused the anomaly.

These results show that whilst using the QMNS inverted index does not create improved subimage retrieval it does provide effective retrieval. A disappointing result is that CPR2 does not outperform the other CPR algorithms as it was intended to.

5.6.4 Results Analysis

In the testing it was demonstrated that CPR2, an algorithm specifically designed for subimage retrieval, was not as good at subimage retrieval as CPR3, an algorithm designed for scale independent full image retrieval.

The algorithms for each of the CPR algorithms are shown in section 5.3.4. All of these scoring techniques share a common procedure, which is the process that determines the intersection of the two image's term sets. Given the intersection, CPR1 awards 1 for each shared term, a simple technique that was designed to be equivalent to the MNS scoring process (5.2). CPR2 was designed to award a high score to an image that is a subimage of another target image. It was assumed that the intersection between the two image's term sets would correspond to the set of terms for the whole query subimage and the set of terms that lay in the area of the target image that corresponded to the query.

CPR3 on the other hand was designed to award a high score to an image that contained proportionally (in terms of area) the same number of terms as a target image. As such the target image could have been a different scale to the query since the normalisation applied to the term frequencies removed the physical size constraints. There was an underlying assumption that two images identical apart from scale would have similar normalised term frequencies. To ensure that the best score was awarded each term was weighted so that the closer the frequencies of the query and target, the higher the score.

CPR4 is based on research from the text information retrieval community and weights terms according to how common they are throughout a corpus. Common terms are deemed to have a low informational content and are given low weightings. A rare term that occurs in both a query and another document is given a very high weighting, since the two documents share something in common that few others do.

The results from a thorough evaluation of these techniques showed that CPR2 was in fact worse than CPR3 at subimage retrieval and also that for full scale image retrieval CPR2 was better than CPR3. Whilst CPR4 was better than CPR2 at subimage and better than both CPR2 and CPR3 at full image retrieval, the two key algorithms we designed performed against our expectations.

5.6.4.1 Analysis

In order to test the full image retrieval capabilities of the algorithms it was necessary to use an image collection that had images at different scales. As such each image was scaled up by a factor of two and down by a factor of a half. Only the normal sized collection was used for queries and the images for the subimage testing were also extracted from this collection. The images were all scaled using the bicubic resampling algorithm provided with Adobe Photoshop.

Whilst this initially appeared to be sufficient further analysis has shown that merely resampling images does not provide ideal test collections. In an end user application a CBIR system may be required to retrieve images that have been created from different capture sources, such as digital cameras, scanners or other sources. The actual level of semantic detail between two images of the same object, or scene will be different, and indeed greater for the image that has a higher resolution. Using software to resample an image acquired from a single source does not provide an accurate representation of scale change. Using a bicubic filter to increase the size of an image results in the blurring of pixels and when decreasing the size artifacts that would not be present in an original small scale image are introduced.

The effect of image reduction is not as serious as enlargement since information is being removed, and there is no way that any algorithm can add information to an image without knowledge of the object, or scene that the image represents. The MNS algorithm suffers badly from the blurring effect of enlarged images very badly, since the patch size used is constant. As the image is enlarged colour boundaries and textures are blurred and once the image is scaled by a factor equal to the pixel width, or height, of a patch, every patch in the image will become unimodal.

In MNS the RGB^2 features will gradually separate and become more sparse, but when quantised into QMNS terms such artifacts begin to cause problems. Since terms in QMNS are not associated to one another once a feature in RGB^2 space moves over a bin partition it becomes a new unrelated term. Whilst there will often be enough terms in an image to mask this effect it is the terms that occur infrequently that carry a high discriminatory power that are affected the most.

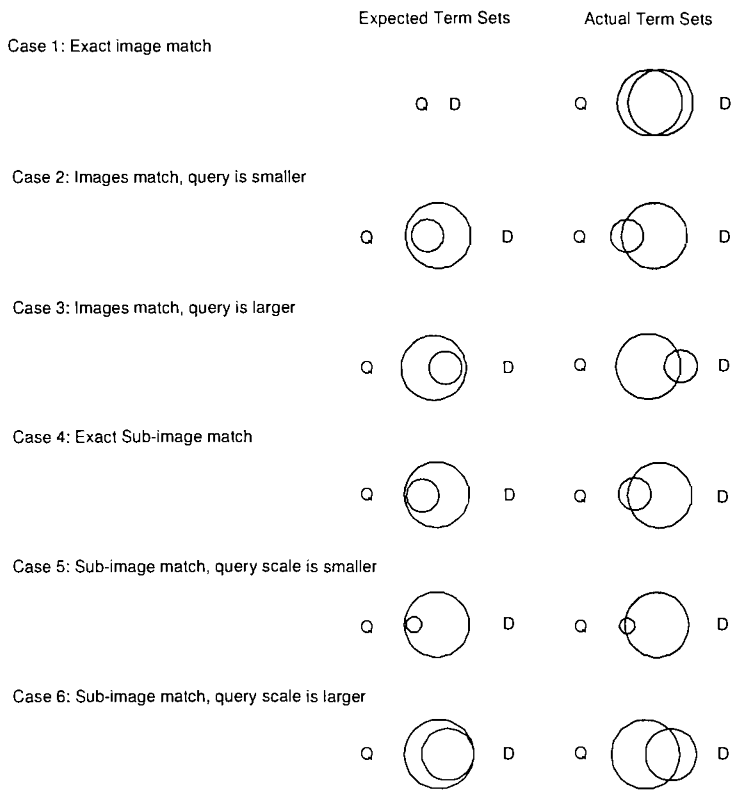
Expected Term Sets
Actual Term Sets
Case 1: Exact image match
Q D
Q D
Case 2: Images match, query is smaller
Q D
Q D
Case 3: Images match, query is larger
Q D
Q D
Case 4: Exact Sub-image match
Q D
Q D
Case 5: Sub-image match, query scale is smaller
Q D
Q D
Case 6: Sub-image match, query scale is larger
Q D
Q D

FIGURE 5.11: QMNS Term Intersections - Correct Query Cases

Diagram 5.11 presents the six cases for a query and correct target image for sub-image and full-image retrieval together with Venn diagram representations of the QMNS term sets. It is the intersection of the two sets that is considered in the scoring process. The left hand column gives example images, the central column shows the ideal sets that might be expected and the right hand column gives the sets as they are more likely to be.

The overall effect of this is that the actual intersection of the two sets does not necessarily correspond to the image area shared by both images. Both images may have terms that aren't present in the other, and for subimage queries the shared terms might not just be from the common area, but could be from anywhere in the image. Diagram 5.12 illustrates how terms from the image could be located in the sets.

The the location of the subimage within the target will change the effectiveness of the query. The example shown in figure 5.12 shows a subimage that does not encompass an entire segment of an image. Figure 5.13 shows a subimage where an entire object is within the subimage. The terms that form the main object area are not present in the area outside the subimage, making this image very well suited to the CPR2 algorithm, but far less so for CPR3. CPR3 picks out another very similar image before the correct target because it has more of the green colour in it.

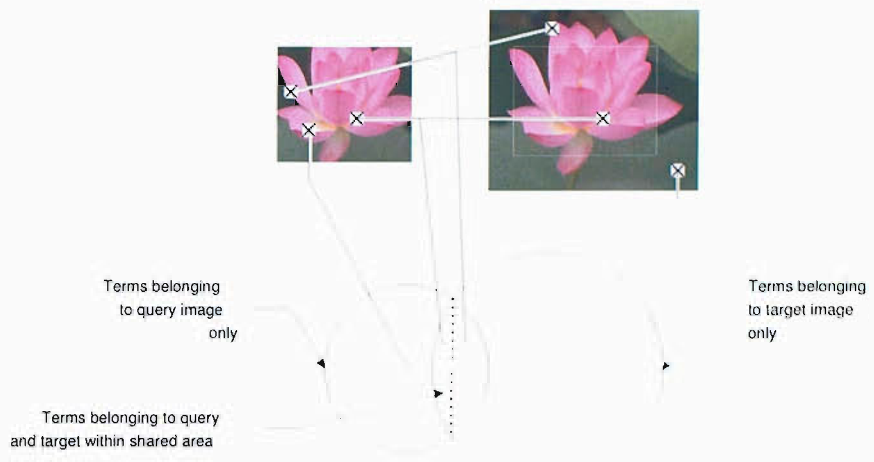

FIGURE 5.12: QMNS Term Distribution

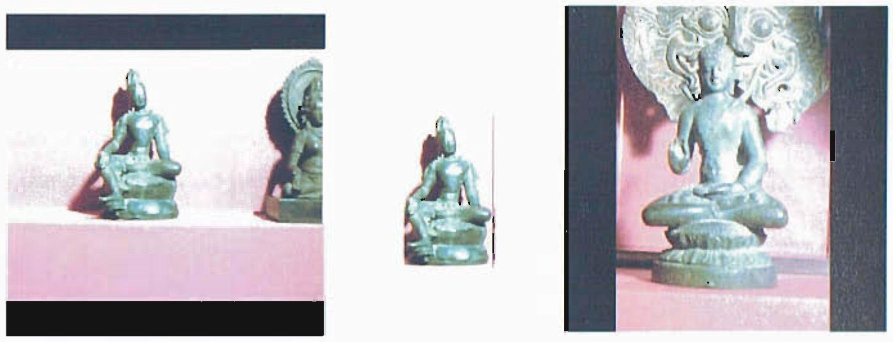

FIGURE 5.13: A Good Subimage for CPR2 Retrieval

A particularly important factor in the success of subimage retrieval with CPR2 is that the number of terms present in the query subimage should be fairly low, and that they do not occur in the large majority of images in the collection. If there are a large number of terms then it is highly likely that there will be a large number of images that share at least some of them in common.

Another general problem that exists is the file format used to store images. As a lossy image storage algorithm the JPEG encoding system modifies changes the information in an image so that it is compressed more. JPEG uses a DCT (Discrete Cosine Transformation) operation on pixel blocks that are 8x8 pixels - the same as MNS neighbourhoods. Figure 5.14 shows an extract of an image and the same image enlarged and enhanced, clearly showing the artifacts introduced through encoding. The image collection used in testing QMNS has been encoded using JPEG.

5.6.4.2 Solutions

The scoring mechanism could itself be altered to allow for better retrieval by introducing penalties for images that do not match all of the terms in a query image. The MNS

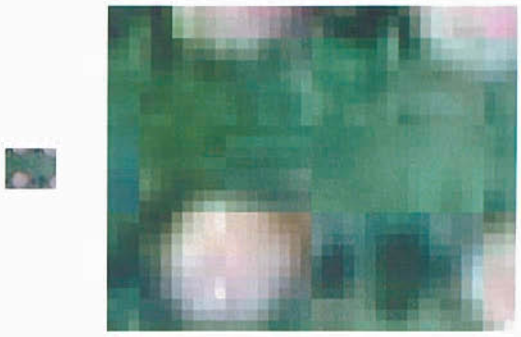

FIGURE 5.14: JPEG Artifacts

scoring algorithm adds a fixed penalty for every query feature not matched to a target image feature (within a certain distance in RGB^2 space), and adding a fixed penalty for every QMNS term not matched in query is equivalent. Terms that do occur in both images may also be penalised if the frequencies are not valid for the query. A subimage should not have more of a particular term than a correct target image and so CPR1 and CPR2 could also penalise excessively large query frequencies when the term is also present (in lower frequencies) in the target image. Whilst some correct, and relevant, images would be penalised because of the sensitivity of the QMNS quantisation procedure to slightly differences in the position of patches (as described in the previous section) non-relevant images would be penalised even more.

A system where spatial information was recorded together with the QMNS terms would considerably help subimage retrieval by indicating which terms should occur together in an image and which shouldn't. This, however, would be a very difficult task to approach and would result in an equally difficult implementation.

Ultimately the end application should be considered further. The subimage tests run for the QMNS evaluation used subimages that were selected at random from the test collection, and whilst this was still a valid approach many of the subimages were not of high level (semantic) objects in the original image. Many were from areas of images that contained little detail, and did not correspond to salient regions, or objects. A subimage of an object in a scene is far more likely to contain features that are not present elsewhere in the image than a randomly selected subimage. Since CPR2 is sensitive to regions in images that are homogenous future subimage queries should be designed to better reflect the needs of the user.

A typical users subimage query would be of an object in an image, and the purpose of a query would be to retrieve images in which that object occurred - for example the original image. A subimage query might also be to retrieve images in which a particular texture occurs, in which case the query might be of a homogenous region.

5.6.5 Test Conclusion

There are many tests that could have been run, however the original aim of this work was to show that retrieval performed using a QMNS feature stored in an inverted index and retrieved using a CPR algorithm was as good as using a standardized technique such as an RGB Histogram, or a CCV.

The first test was to determine indexing times and shows that whilst MNS is a slow algorithm for signature generation, there are improvements that can be made that decrease the indexing time without affecting retrieval effectiveness, and that significantly increase the retrieval effectiveness without affecting indexing time. Interpolated recall and precision graphs for overall test results and graphs for individual queries are shown with explanations for the particular responses.

The second test introduces the QMNS CPR algorithms and also the two different versions of the index. A QMNS index based on the Enhanced MNS and the Rapid MNS (0.3) variations are introduced to see whether the results achieved in the first test are carried through to the second test. The comparison was against an RGB Histogram, a CCV and a Multiscale CCV. All versions of QMNS perform better than just the Base MNS variation, and offer retrieval that is approximately equal to the other three algorithms. The test demonstrated that CPR3 is not as well suited to scale invariant full image retrieval as CPR1 and CPR2, but that CPR4 is the best.

The final test compares the same algorithms as the second, using one hundred randomly selected subimages. Algorithms are rated according to how many of the correct images are retrieved at high level rankings. The results are disappointing since the CPR2 retrieval algorithm was intended to be better at subimage matching than the others, and the difference between all QMNS versions is negligible.

Overall the tests do demonstrate that the QMNS and CPR algorithms do achieve the same and better levels of retrieval accuracy, as measured using recall and precision metrics. The scale invariancy of the algorithms appears to be good, although the extent of the scale testing was not rigorous. A particular aspect that was not examined in these tests was how well the inverted index managed with differing sizes of image collection in terms of retrieval speed. This needs to be tested in the future, since it is a key requirement of the index that it scales well.

5.7 Multimodal Patch Retrieval

After testing the configurations possible with bi-modal patches, we extended QMNS to accept uni- and tri-modal patches, where the modality refers to colour. Implementationally this was a matter of adjusting the rules which determine what the clusters found by the mean-shift. The binary signature was already capable of storing multiple modes for each feature, which left only the modifications to the quantisation process. For each mode the term label was generated by concatenating the relevant 2 digit bin number from each colour axis together, resulting in an integer of the form $RRGGBB$. The label for each mode was then concatenated forming a 6, 12, or 18 digit label. The frequency for each bin was incremented for every feature present in the bin of that modality. Since the algorithm no longer just retrieved colour pairs, the CPR (Colour Pair Retrieval) name was dropped in favour of CMR - Colour Mode Retrieval.

5.7.1 Multimodal Patch Retrieval Testing

Two types of test were performed for this component of the research: Full and sub-image retrieval. Both types of testing included only the original scale image set with the same image categories as before.

We chose to test the effectiveness of our algorithms at subimage retrieval by creating a set of subimages selected by potential users from the image collection. By using people to select the subimages, rather than extracting random regions from random images, the test set created contains objects that are representative of typical queries - in almost all cases the selection bounds an object in the image. Ten subjects were selected and asked to choose 9 images from the complete image collection at random, and one further image was provided from the categories for each of the subjects. The subjects were then able to select a region in each of the images at random (care was taken to ensure that no two subjects selected the same object in an image) which formed the subimage.

We compared our algorithms, and each of the parameter variations, against a 64 bin RGB colour histogram, a 64 bin CCV and a 64 bin multiscalar CCV [25]. Two of the parameter sets were selected from the analysis presented in the previous section. The first used 8x8 pixel patches with a minimum mode size of 6 pixels and a mean shift window of 22. The second used 16x16 pixel patches with a minimum mode size of 10 pixels and a mean shift window of 22. For both of these configurations indexes were generated that contained the following combinations of modalities: uni-, bi-, tri-, uni- and bi-, bi- and tri-, and uni-,bi- and tri-modal, resulting in 12 different configurations. Each of these variations was tested with each of the four CMR retrieval algorithms, forming a complete test set of 48 QMNS tests and 3 histogram tests.

5.7.2 Analysis of Multimodal Terms

In text retrieval, word terms carry a great deal of semantics on their own, unlike our colour *feature terms*. However, both types of term share the notion that a term that

Algorithm	Time(s)
RGB	0.109
CCV	0.426
MCCV	1.001
QMNS, 8x8, modes:1	1.207
QMNS, 8x8, modes:1,2	1.318
QMNS, 8x8, modes:1,2,3	1.324
QMNS, 16x16, modes:1	4.556
QMNS, 16x16, modes:1,2	4.639
QMNS, 16x16, modes:1,2,3	4.872

TABLE 5.6: Average Image Indexing Times

occurs infrequently throughout a corpus as a whole, yet occurs many times in one particular document, is very likely to be of importance to the information content of that document. That term is said to have a high discriminatory power.

In QMNS the modalities of the terms directly affect their discriminatory power. When a quantisation factor of 4 is used with the 3 channels of RGB there are only 64 (4^3) possible uni-modal terms, but 4096 (4^6) bi-modal and 262144 (4^9) tri-modal terms. The higher the modality the higher the information content and the more discriminating the term. However, there will also be far fewer of these high-modality terms.

There are only a fixed number of patches, and hence individual terms, in an image, and so it is desirable to maximise the number of high-modality terms that are created by the MNS algorithm and also the quantisation procedure. QMNS has a number of parameters that directly affect generation of the MNS features: The size of a patch controls the maximum number of patches and hence features in an image, the size of the mean shift window controls the colour clustering of pixels within a patch, and an integer parameter determines the minimum number of pixels that must belong to a cluster for it to be classed as a mode.

By increasing the window size the size, and separation, of the mode clusters will increase. This should result in an increase in the number of patches that have a low number of modes, and to a point, the number of different terms and their frequencies will increase before decreasing as the mean shift blurs the patches - ultimately resulting in 100% unimodal patches.

Once all MNS features have been generated the quantisation factor will determine the number of unique terms (the vocabulary) of an image. It is desirable to have a vocabulary where each term has the best balance of discrimination and occurrence, and if, by using a high factor, there are too many terms, each term may be highly discriminatory but may only occur a few times across an entire corpus. A low factor would lead to a small vocabulary of terms that would occur many times in all images. Either situation will lead to poor retrieval results, and so the best balance needs to be determined.

Quantisation	Uni-Modal		Bi-Modal		Tri-Modal		Proportion
Factor	Maximum	Actual	Maximum	Actual	Maximum	Actual	
3	27	23	729	211	19683	1517	0.0857
4	64	52	4096	561	262144	271	0.0033
5	125	93	15625	1222	1953125	423	0.0008

TABLE 5.7: Vocabulary Sizes for Differing Quantisation Factors. Parameters used were 8x8 pixel patches, 22 window size and minimum 6 pixels per mode.

The remainder of this section presents an analysis of QMNS parameter changes, and the effect this has on the distribution of terms in the corpus. Indexes were generated using the collection of 769 photographic images known as the 'general' collection.

The generation of QMNS feature terms is not as fast as the generation of any of the three histogram algorithms, however it is still a relatively fast procedure, as shown in table 5.6. The index times for the larger 16x16 pixel patches are slower because the mean shift algorithm (an $O(N^2)$ algorithm) is slow when used with a large number of data points.

In total 43 different parameter changes were made. 23 of the variations used 8x8 pixel patches, and the other 20 used 16x16 pixel patches. In the 8x8 group the minimum mode size was varied from 5 to 8 and the mean shift window size from 11 to 55 (in steps of 11), and additionally the quantisation factor was tested at 2,3 and 5 divisions. The 16x16 group had a minimum mode size variation from 10 to 25 in steps of 5, and the same mean shift window changes.

Table 5.7 shows the maximum number of terms created by differing factors, and also the actual number generated in the index. As the quantisation factor increases the proportion of terms actually present drops exponentially, as expected.

The two graphs in figure 5.15 illustrate how changing the two main parameters affects the distribution of terms in the index. In both graphs the left axis corresponds to unique terms - the vocabulary - and the axis on the right corresponds to the actual number of terms, both values as generated in our image collection. As the window size is increased the number of unique terms increases across all modes, up to a value of 33, where it starts to recede again. The number of uni-modal patches continually increases due to the blurring effects of the mean shift, coupled by a reduction in the number of bi-modal and tri-modal patches. Ultimately all patches in the image would become uni-modal.

Increasing the minimum pixels per mode has the effect of reducing the number of different terms in the index, which occurs as the number of pixels required for a cluster to be classed as a mode increases. At values of 7 and 8 there are no tri-modal features, however at lower values their numbers increase rapidly, and become more abundant than the unique uni- and bi-modal patches. The total number of tri-modal features in the index doesn't increase as rapidly, and at its greatest, peaks at 15687, representing just 1% of all the features in the index. This indicates that there are too many different tri-modal features in the index, and they occur too infrequently to have a high discriminatory power in retrieval.

When the terms from a complete index are ranked by frequency and plotted on log scales they form nearly straight curves, indicating that they fit to a Zipfian distribution [148], as do other collections such as the frequency of words. This distribution of text terms led to the use of a log weighting in the TF/IDF weighting algorithm, and the presence of such a distribution in our feature terms indicates that CMR4 should be well suited to feature term retrieval.

Our approach in this analysis has been to take a starting point, namely the parameter set of 8x8patches, shift window of 22 and 8 minimum pixels per mode, and vary the parameters in order to determine the best distribution of feature terms. Our aim was to achieve a balance of uni-, bi-, and tri-modal terms, such that the discriminatory power of each was highest. A tradeoff that took into account the blurring effect of the mean shift at large patch sizes and large window sizes, the increased level of noise added by a low minimum pixel count per mode and a suitable quantisation factor led us to adopt the following two parameters sets for the evaluation:

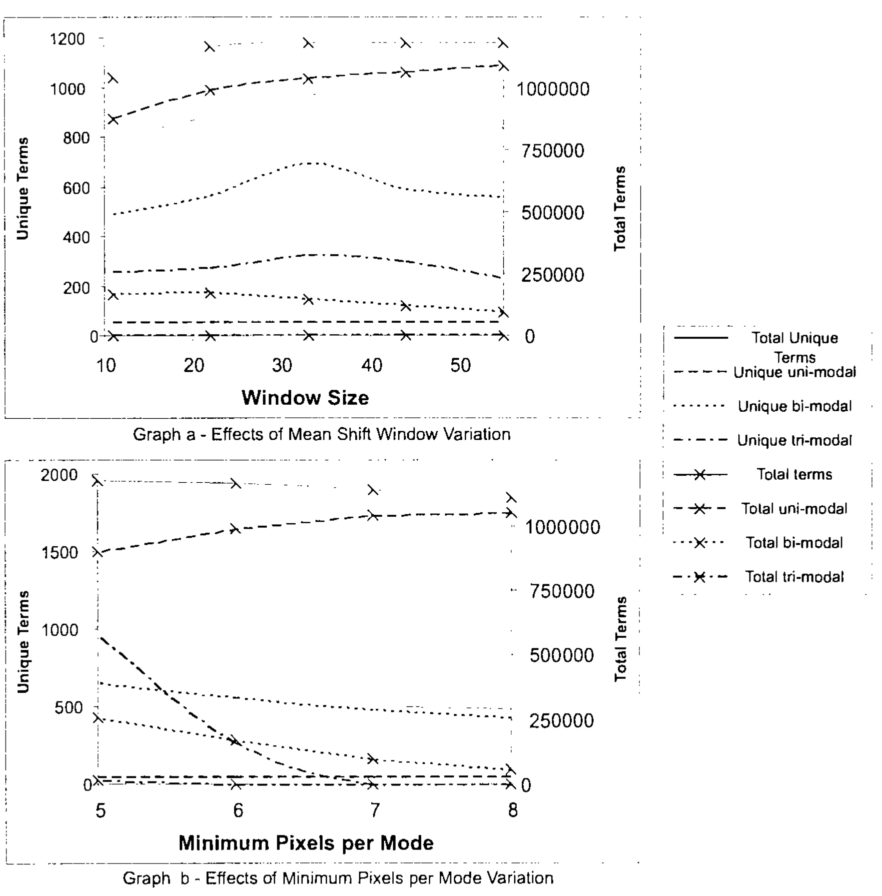

FIGURE 5.15: Top: Effect of Modifying the Size of the Mean Shift Window. Minimum pixels per mode fixed at 6. Bottom: Effect of Modifying the Size of the Minimum Pixels per Mode. Mean shift window size fixed at 22.

- 8x8 pixel patches, 22 mean shift window, 6 minimum pixels per mode
- 16x16 pixel patches, 22 mean shift window, 10 minimum pixels per mode

Due to the large number of tests run, not all results will be shown in this section. Instead the most important comparisons are presented. These include comparisons between the CMR retrieval algorithms, the modes used in retrieval, and the size of patch used.

5.7.3 Full Image Tests

Figure 5.17 shows an example of a full image query. Along the top of the figure are the 6 images from the *SeaWorld* category, the image on the far left being the query. Below them are the top 8 images returned by 5 of the algorithms tested. The parameters for the QMNS examples were 8x8 pixel patches with a window size of 22. All three modes were used in the retrieval.

Figure 5.18 shows recall and precision curves for a single query (shown in figure 5.17), averaged values for all queries in a category and averaged values for all categories. The further to the top right that a recall-precision curve extends, the better the retrieval. In the example all of the algorithms perform very well, with CMR4 just possessing the most area under the curve. The category average shows how the different algorithms start to separate out, with CMR4 as the clear best. Graph(c) shows the averaged curves for all queries over all image categories, and how the CCV algorithm is the overall best.

In figure 5.19, chart(a) shows that using uni-, bi- and tri-modal patches in an image provides significantly better retrieval results than tri-modal patches alone. When 8x8 pixel patches are used very few tri-modal patches are extracted from an image, resulting in the very low score for the first variation. This is also shown in chart(b), where a full comparison of modes is given. Each of the tests was run using 8x8 patches and a window size of 22. The use of uni- and bi-modal patches alone provides very good retrieval capabilities, and when combined together the results are better still. As expected from

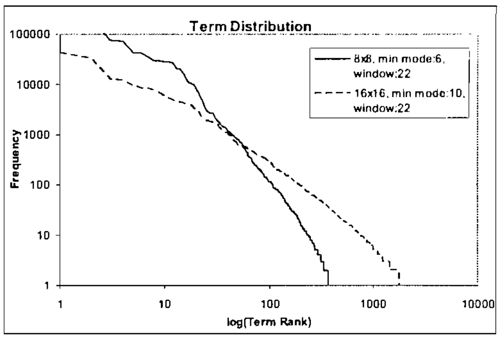

FIGURE 5.16: Frequency Distribution of Terms Across a Corpus

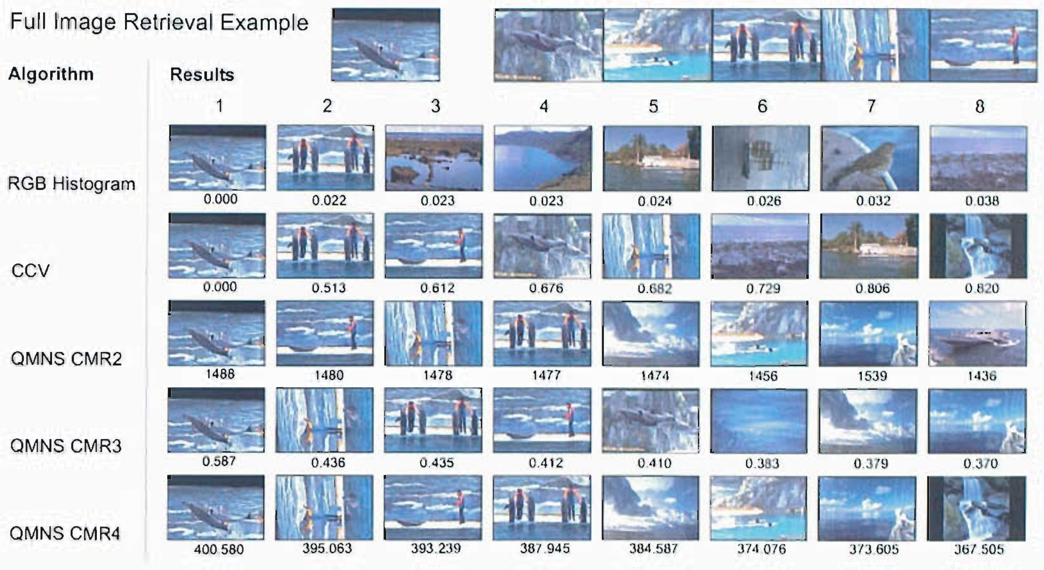

FIGURE 5.17: Full Image Query Example

the analysis of term distribution, using tri-modal patches makes no difference to the results at all, only slowing down the retrieval time by adding more feature terms to the query.

5.7.4 Subimage Tests

Figure 5.20 presents an example of a sub-image query in the same manner as the full-image query example. At the top is the sub-image used as the query and its parent image. The RGB histogram and CMR3 both attempt to match the image signatures based on the proportion of the image that contains a certain feature. As such both retrieve images that contain a large proportion of features present in the query as well as other unrelated features. The MCCV performs in a similar fashion, but has retrieved the correct image first due to the use of multiple CCV histograms from the parent image. Both CMR2 and CMR4 retrieve images that contain the query features at any frequency, including images that have a substantial number of other features.

Whilst the MCCV is the best algorithm for retrieving the correct parent image at the top rank (42% of the images, compared with 31% for CMR2 and CMR4 and 19% for CMR3) the overall effectiveness of each algorithm at retrieval is measured using equation 5.8 and by plotting the percentage of images retrieved to rank N. Figure 5.21 shows retrieval curves for each algorithm. Similar to recall and precision curves, the area under the curve indicates overall effectiveness with a larger value corresponding to a better algorithm.

MCCV is the best up to rank 10 after which the number of images correctly retrieved starts to drop, whereas for CMR2 and CMR4 this drop off is more gradual. By rank 20 CMR4 has retrieved 4% more correct images than MCCV and 3% more than CMR2.

Measurements such as this are very useful since a typical user may not wish to view more than 20 images before deciding their query has not been successfully.

Test results between different parameter sets appear similar to the full image results. Retrieval using only tri-modal patches proves to be poor, especially when using 8x8 pixel patches, due to the low number of terms. As before, when comparing across one parameter set and only varying the modalities of patches retrieved, the use of both uni- and bi-modal patches is the best combination. The characteristics of CMR2 ensure that it performs better than the other CMR algorithms, although this is by a small margin. CMR3 performs worst, confirming the need to restrict its use to full-image retrieval.

Of the four CMR algorithms CMR4 is the overall best, being equivalent to CMR2 at subimage retrieval, a task for which it was purposely designed. It is also better than both CMR2 and CMR3 at full image retrieval, but due to changes in the parameters CMR3 is often not the best algorithm at this task. The results for CMR4 compare to text retrieval results that show that TF/IDF is suited to both sub-document queries (such as keywords) and full document comparisons.

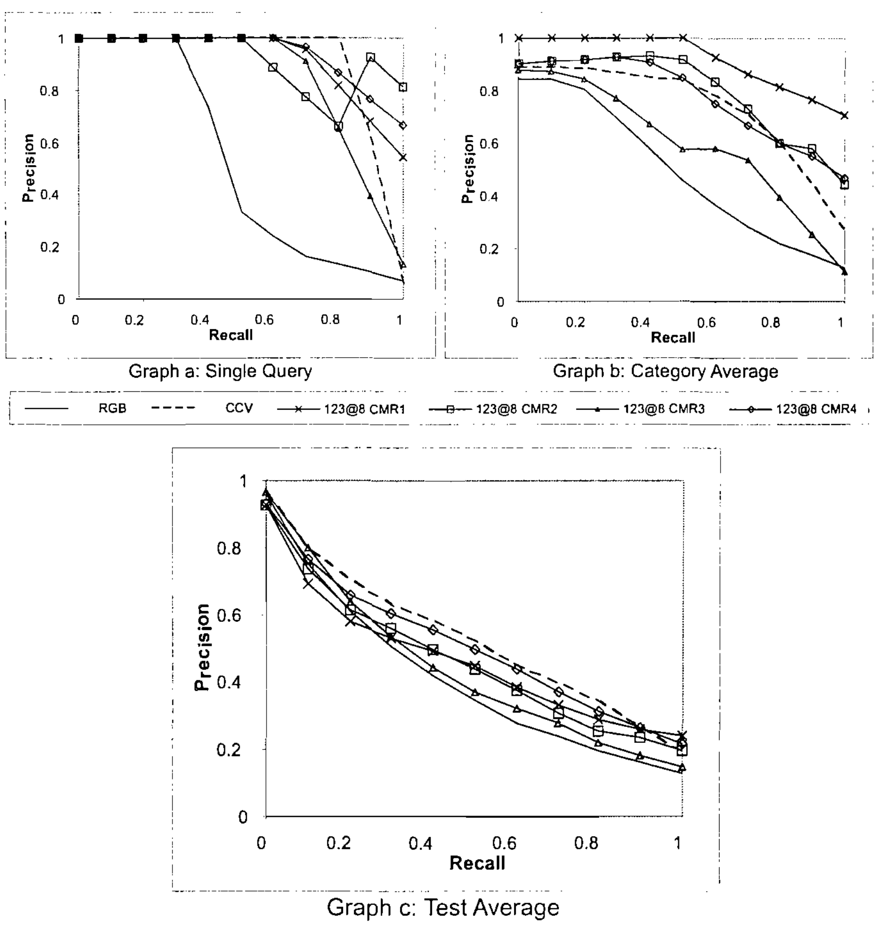

FIGURE 5.18: **Full Image Recall and Precision Curves:** The recall and precision data shown was taken from the tests for QMNS when the parameters were as follows: 8x8 pixel patches, modes: 1,2 and 3, window size:22

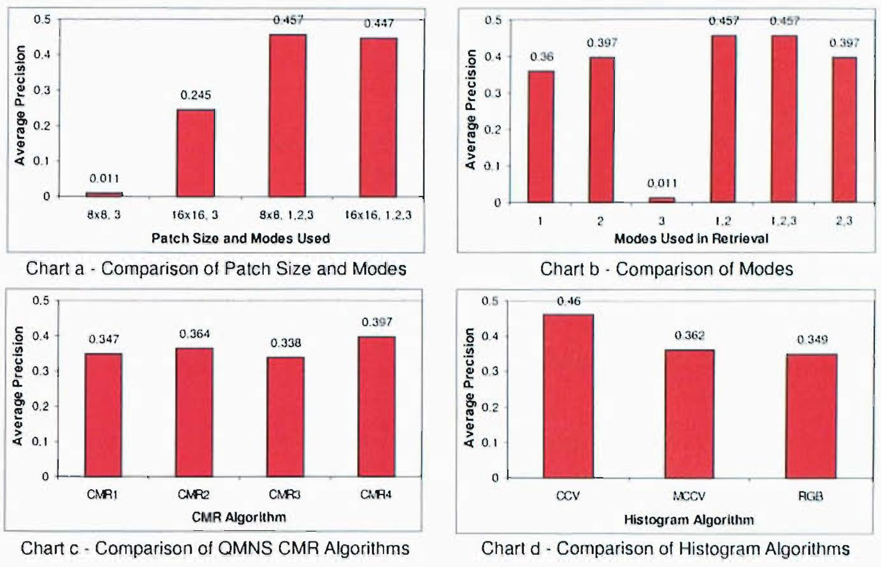

FIGURE 5.19: Full Image Test Results

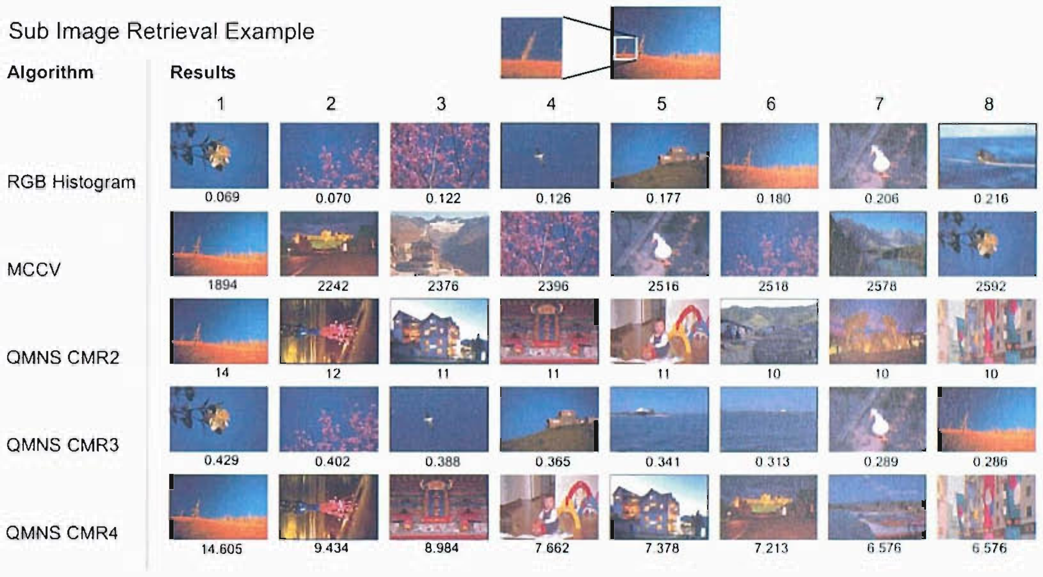

FIGURE 5.20: Subimage Query Example

On the use of different colour modes for retrieval, table 5.8 shows a quick comparison of one parameter set which is consistent with others. Tri-modal patches are of little use on their own, and when combined with others do nothing to enhance performance. Using bi-modal patches alone is better than just uni-modal patches, but when combined there is a distinct improvement.

This section has presented an analysis of the distribution of colour feature terms stored in an inverted index. Parameters that directly affect the distribution were varied and the effects measured. The aim of the analysis was to determine the parameter set that would provide a balanced distribution of feature terms between the modalities of the colour features. In order to be effective in retrieval, features must be selected that have a high

Modes Used	Sub image score			Full image score		
	CMR2	CMR3	CMR4	CMR2	CMR3	CMR4
1, 2	0.805	0.737	0.8	0.412	0.389	0.457
1, 2, 3	0.805	0.737	0.799	0.413	0.389	0.457
2	0.720	0.558	0.706	0.364	0.338	0.397
2, 3	0.719	0.558	0.706	0.364	0.338	0.397
1	0.608	0.661	0.618	0.324	0.334	0.36
3	0.010	0.01	0.01	0.016	0.013	0.011

TABLE 5.8: Retrieval Effectiveness of Modes. Full image metric taken from the results for 8x8 pixel patches using a window of22 and minimum pixels per mode of 6 when using CMR4. Sub image metric taken from the same parameter set when using CMR2.

power of discrimination, and, based on the analysis, parameter sets were chosen. Four retrieval algorithms are compared, using two common types of example based queries - the full image and sub image queries. Comparisons are also made between the use of uni-, bi- and tri-modal feature terms. The retrieval algorithms CMR2 and CMR3 were designed to be superior to the original CMR1 algorithm at sub image and full image retrieval respectively. Whilst both are indeed better than CMR1, in many cases CMR3 is not better than CMR2 at full image retrieval. The fourth algorithm, CMR4, that is derived from work from the text retrieval community, proves to be equal to both CMR2 and CMR3. An evaluation of the performance of some of the parameter sets has also demonstrated that using the more specific tri-modal terms is not as effective as using either uni- or bi-modal terms. The combination of the two lower order terms is shown to be more effective than either on their own, or when combined with the higher order term.

5.8 Summary

This chapter has introduced QMNS - the Quantised Multimodal Neighbourhood Signature. We have extended the MNS algorithm by first quantising the original feature

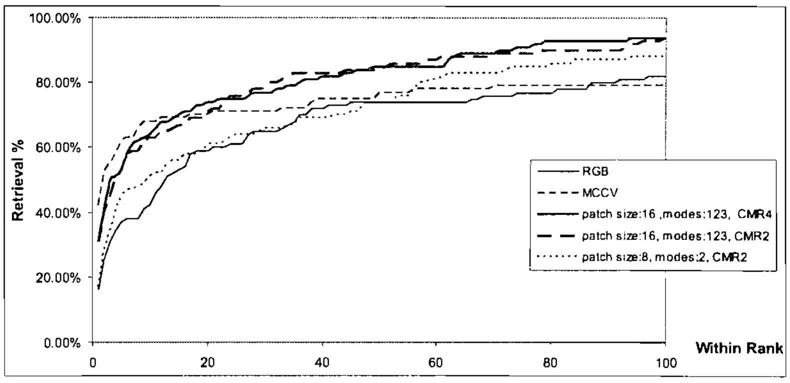

FIGURE 5.21: **Sub Image Query Curves:** For each algorithm the curve shows the percentage of the 100 subimages that were retrieved within rank N

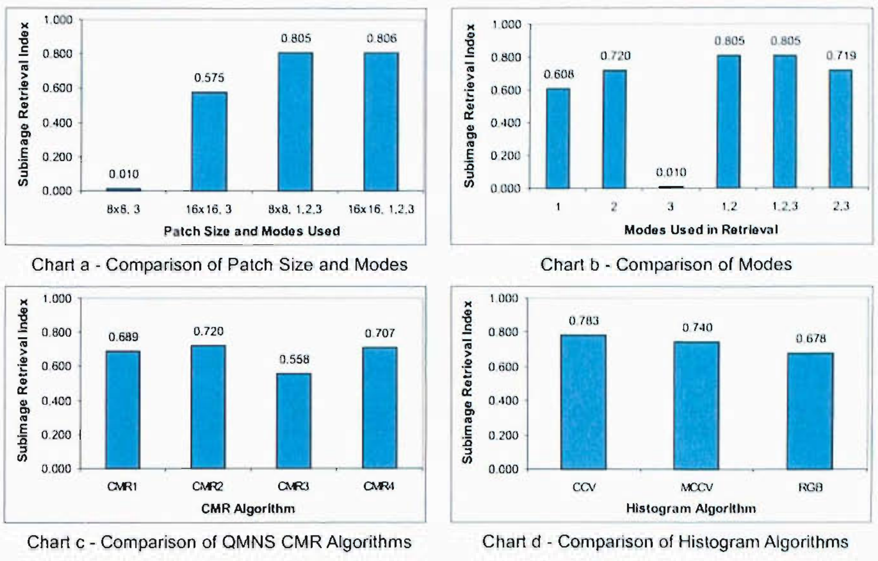

FIGURE 5.22: Sub Image Query Charts

space, and then by allowing different modalities of colour of the underlying RGB feature. Tests have been run to compare different types of retrieval algorithm including TF*IDF style scoring.

The key findings in this chapter have been:

- MNS Signature Generation: The number of neighbour pixels passed to the mean shift can be reduced to 30% without a negative effect on the retrieval accuracy, and if the mean shift window size is set at 11 the average precision is almost doubled.
- Algorithm Comparison: The base form of MNS does not perform as well as either the RGB or CCV histograms, but with the modifications stated in the previous point the results are equivalent. The four forms of QMNS retrieval are shown, and it is CPR4, which uses TF*IDF scoring which is the best. Subimage retrieval for the QMNS algorithms is at best equivalent to results provided by an RGB histogram, although the normal MNS algorithm is better. The MCCV is shown to be much better than any of the other algorithms.
- QMNS Extension: Quantising multi-modal QMNS features leads to better performance when the uni- and bi-modal patches are used. Tri-modal patches offer very poor performance due to their rareness in images.

Chapter 6

Generalised Feature Indexing

6.1 Introduction

The tradeoff between retrieval accuracy and retrieval response is one that needs to be considered carefully, and is highly dependant on the retrieval application. Where some systems may require very accurate retrieval (for example medical imaging) and real time queries are not an issue, others require very fast response times (for example internet search engines) but may not need to retrieve all relevant documents.

Sequentially comparing the signatures of a small number of documents is inexpensive, but the large number of documents that applications are required to search through, and the amount of information that we desire to search through, makes this approach redundant. A solution is to index documents in some manner, so that the amount of information that must be searched through is considerably less per document. Section 2.7 introduced some of the techniques for indexing image signatures so that they may be retrieved more rapidly. The section following this shows how the text retrieval community (and indeed all of us) have been indexing text documents for centuries, and how this has been translated into the inverted index which allows modern text retrieval to be so rapid for such large corpora.

The previous chapter introduced the global, colour, image feature QMNS, that employs an inverted index in which to store image features. Whilst not the first feature indexed in such a manner, it is nonetheless novel in its use of colour patches. The conversion of the feature into a format that was acceptable by the index was a relatively straightforward process, and rather than showing a degradation in retrieval performance, the feature generally performed better. What were the processes involved in the indexing of this feature, and how can this be applied to other features?

Inverted indexes store terms - any distinct, countable, element of a document, be it a word in the English language or the RGB colour 255,0,0. The second section looks at the space in which features exist and categorises the different techniques available for discretising this space so that feature terms can be generated. and stored in the index. QMNS was quantised using a simple, regular, algorithm. but could it have benefited from a different technique? When populated by all the documents in a corpus, the feature space of a good feature will contain clusters of documents - showing that those documents share some form of content.

Given a particular feature, quantisation algorithm and document collection there are still many changes that can be made to improve retrieval from an inverted index. In particular the coarseness of quantisation will change the number of unique terms that the feature space is divided into. The fourth section looks at the tradeoff to be made between the number of terms (directly related to the retrieval speed) and the retrieval accuracy.

In text information retrieval the terms specificity and exhaustivity refer to the power of discrimination of the index. An index can be said to be specific if users are able to retrieve those few documents that are highly relevant, whilst an exhaustive index will contain many more less relevant topics. This relates directly to the recall and precision metrics discussed in chapter 2.8.

6.2 Term Distribution Tests Index Generation

The previous chapter examines a large number of QMNS configurations, which amongst other variables, look how the modes are extracted from patches, and what constitutes a mode as opposed to noise. This showed clearly that the Enhanced MNS version, which produces a larger number of modes due to a small mean-shift window size, is very good. This section extends the testing of mode creation by introducing another variable - the threshold pixel value that determines whether a cluster forms a mode.

For this test set the window size was varied from 5 to 35 in steps of 5, the mode pixel threshold from 2 to 10 in steps of 1, and the quantisation level (bins) from 2 to 9 in steps of 1. This produced 504 indexes, all of which were tested with the 5 image categories. In this set only the bi-modal patches are extracted.

For each index the Zipfian distribution was generated, and the logest (a function that determines the power-law coefficient), average precision, term count and vocabulary sizes collated. To many to present here, the results have been reduced to illustrate the effect on the retrieval performance and distribution that the variables have.

The first comparison is between the minimum pixels per mode and the window size. The quantisation level is fixed at 8 bins Figure 6.1 shows this data set in two graphs, both have the minimum pixels on the x axis, and the window size as the series. The graph on the left shows the resulting average precision, and that on the right shows the vocabulary size - the two measures directly relating to retrieval performance. In all window sizes the vocabulary size naturally drops as the number of pixels required to form a cluster increases, however the average precision does not drop as dramatically.

The second pair of graphs, in figure 6.2 holds the windows size at 20, and shows the number of bins in the series. We see that the higher the number of bins, the better the average precision, up to 8 bins. The higher quantisation factor has a direct effect on the vocabulary size, but the minimum pixels per mode has an interesting effect on this measure, causing peaks and troughs in the curve.

The last pair of graphs in figure 6.3 also shows that the quantisation level alone does not have a huge effect on the average precision, but does on the vocabulary size. The large peaks in the vocabulary at high and low window as the quantisation level increases size demonstrate a particular sensitivity in the clustering process.

The data shown in this section serves to underline the need for quantitative analysis of features when quantising. The changes in the variables show definite trends, however there are many local maxima and minima occurring irregularly that might disrupt attempts to model the performance or vocabulary size. A key unknown variable that will influence this data is the pixel colour distribution of the image collection as a whole; given a different set of images the overall curve trends may be similar, but with a different local structure.

An overall view of the tests is presented in the following four graphs. Figure 6.4 shows the Logest and the total number of terms plotted against the average precision for all 504 indexes. Both graphs show a strong positive correlation, as does the right hand graph in figure 6.5 for vocabulary. On the left of this figure is a plot of vocabulary against the Logest of the distribution which, apart from a few outliers, shows a very clear correlation between the two measures.

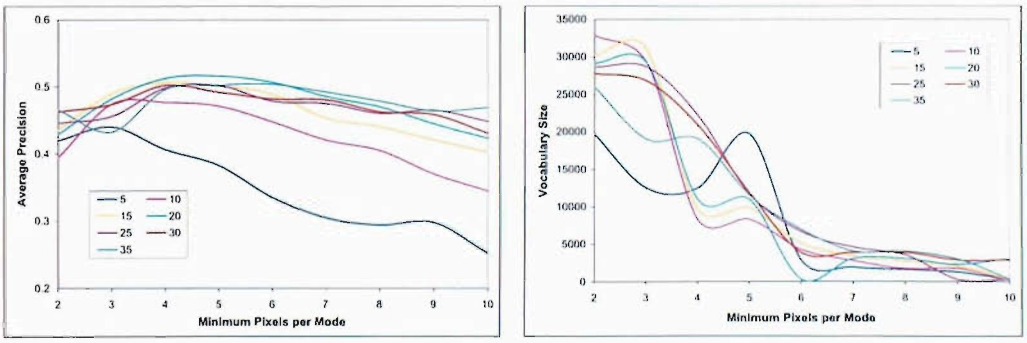

FIGURE 6.1: Minimum Pixels per Mode vs Window Size (series) at 8 Bins

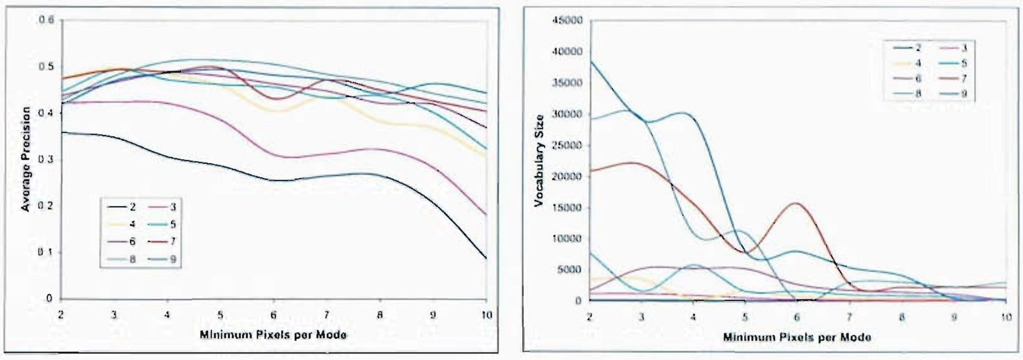

FIGURE 6.2: Minimum Pixels per Mode vs Bins (series) at Windows Size 20

6.3 The Distribution of Feature Terms

The second chapter introduced the fact that the English language, and indeed any natural language, has an interesting distribution of terms. There are only a few words which occur very frequently, yet a great number which occur very infrequently.

Such Zipfian distributions have a natural compatibility with inverted indexes. This is because those terms which occur very frequently are not good discriminators, and are unable to distinguish between documents - which makes the index efficient to use, since there the less frequent terms will have much shorter posting lists, and allows terms to be removed completely from the index. Exactly why the Zipfian distribution is important will be explained in the next section. Presented here is an analysis of the distribution of feature terms, which will allow a particular feature to be engineered to fit into an index such that the trade-off between retrieval speed and retrieval accuracy is balanced.

Like text indexes, inverted indexes that store feature terms can be pruned so that terms that are not good discriminators are not stored. This is very important for inverted indexes since each query term involves a relatively expensive index lookup. The stop-words in text indexes are noted in a negative dictionary, and are not stored in the main index, nor are they used in queries. They are both words that are very common - A, OR,

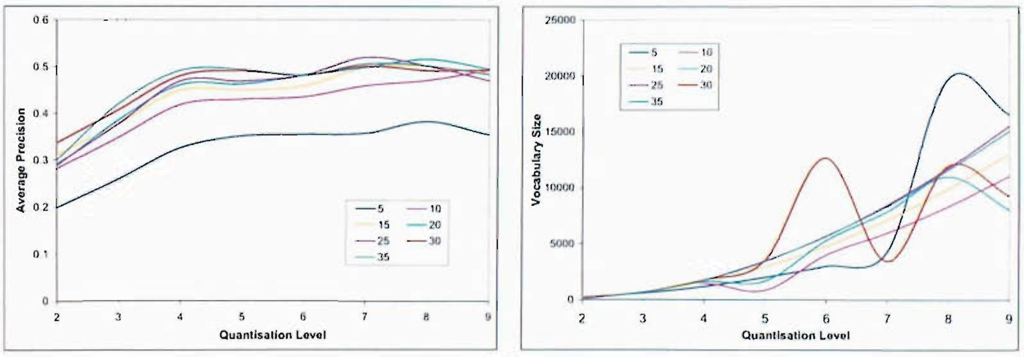

FIGURE 6.3: Bins vs Window Size (series) at Minimum 5 Pixels per Mode

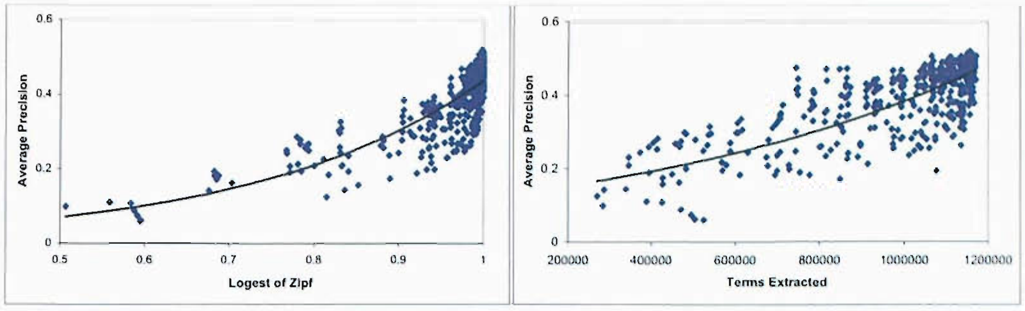

FIGURE 6.4: Logest function vs AP and Total Terms vs AP

IF (function words) - and words that are very rare - PHENOMENOLOGICAL. These words lie at either end of the rank-frequency term graph (see section 2.7.1 for a discussion on Zipf and rank-frequency graphs) and have feature based counterparts - for example very bright (near white) and very dark (near black) pixel colours tend to occur most frequently and in all images, yet colours such as *spring green* (in RGB values this is R000G255B127) occur very rarely.

Words in natural language conform to the Zipfian distribution, which means that the stop-word list can be generated automatically by looking at the term distribution of a corpus. The frequency distribution of the terms (bins) of a some histogram features may not be Zipfian, which makes this procedure more difficult. Histogram features have a raw count of features in feature space. For the RGB histogram this is the count for each of the different 16,581,375 R,G,B values from each image. For QMNS this is a combination of uni-modal, bi-modal and tri-modal patch counts, resulting in a discrete feature space of approximately $4.55 \cdot 10^{21}$ possible points. These feature spaces are quantised into bins, resulting in a set of feature terms representing an image, or an entire corpus. Changing the method of quantising the feature space will result in a different set of feature terms, and a different distribution.

Throughout this discussion three simple image features shall be used - the RGB colour histogram, the CCV colour histogram and the QMNS algorithm. All three of these algorithms have simple feature spaces where the effects of changes can be seen easily in

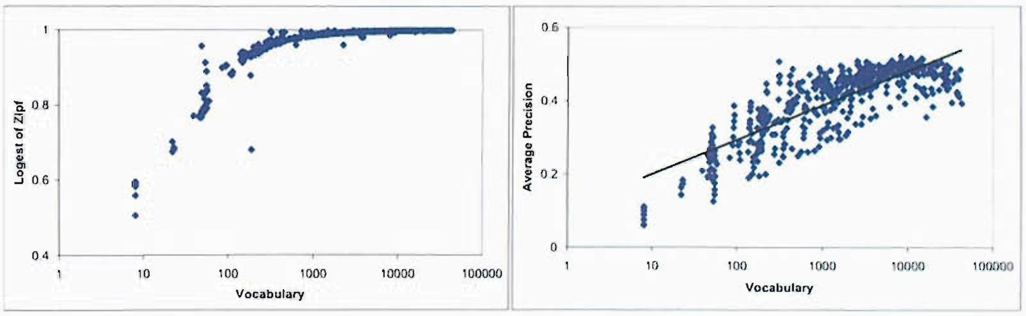

FIGURE 6.5: Vocabulary vs logest function and Vocabulary vs AP

the distributions. For each algorithm a set of indexes were created, with each sub-index differing in vocabulary size. For each index rank-frequency data was generated, and all category tests were run, resulting in the data analysed in the following section.

6.3.1 Effect of the Vocabulary Size on Feature Term Distributions

When the number of divisions along each quantisable axis is increased, the term vocabulary - the number of different, unique, feature terms - naturally increases. The limit to the number of divisions that may be made is set only by the representation (the datatype) of the feature space. As such, the integer RGB colour space used in the CCV and RGB Histogram is limited to 255 divisions, providing over 16 million different terms. QMNS also uses RGB colour space, however, the integer values have been clustered and averaged using the mean shift algorithm, and each point in this space is recorded as a triple of doubles. Having 8 bytes, a double provides a resolution of $1.8 \cdot 10^{19}$[1] and a potential vocabulary of approximately $1.0 \cdot 10^{65}$! In reality there is no need to go to this resolution, or anywhere near it, and the useful range is actually below 10 divisions, providing a more realistic vocabulary size of 1 billion terms or less.

This peak was determined by adjusting the divisions parameter for QMNS, ranging from 2 divisions up to 10 in steps of 1 and from 10 divisions to 100 in steps of 5, as well as an additional division at the limit - 255. Both of the RGB features were also tested, with the RGB histogram ranging from 3 to 200 (in uneven steps) divisions and the CCV ranging from 1 to 8 'divisions'. The use of quotes here is due to the notion of a division in the CCV algorithm: Rather than being defined as the number of partitions created along each axis, the CCV 'divisions' define the number of bits that each colour channel will be reduced into. 3 bits would result in $2^3 = 8$ divisions per axis, resulting in a total vocabulary of $(2^3)^3 \cdot 2 = 128$. The factor of 2 at the end is because the CCV contains *two* histograms - the *coherent* and the *incoherent* components. The set of values assigned to the division parameter for the histogram and the CCV is limited by each algorithm's implementations, and in these tests each feature space was quantised beyond useful limits in order to gather data about the nature of the term distributions. In some cases the size of the image signature was bigger than the image itself, and a single index query took minutes.

Table 6.1 gives outline data from the tests, showing the changes in each indexes term distribution according to the vocabulary size. For clarity the data in this table has been kept brief, refer to table E.1 for the complete data. Note that the average precision value for the histogram at 255 divisions is absent: The histogram comparison function was unable to cope with the number of bins. The columns should be read as follows:

[1]Since floating point arithmetic is performed using a 48bit mantissa and a 16bit exponent the actual accuracy of the number is less than this value suggests.

	Divisions	Vocabulary	Used Vocabulary	Vocabulary Usage	Total Postings	Average Terms	Curve Slope	Average Precision
Histogram	4	64	56	0.875	17062	22.2	0.8080	0.237
	10	1000	689	0.689	93486	121.6	0.9847	0.441
	60	216000	102567	0.475	3694780	4804.7	0.9994	0.538
	127	2048383	754151	0.368	12818258	16683.4	1.0000	0.543
	255	16581375	1317793	0.079	18033673	23450.8	1.0000	
CCV	1	16	16	1.000	7431	9.66	0.7408	0.137
	2	128	100	0.781	21066	27.39	0.9051	0.331
	4	8192	3151	0.385	248145	322.74	0.9954	0.485
	6	524288	120182	0.229	3303475	5406.67	0.9980	0.584
	8	33554432	1365400	0.041	18069469	23497.36	0.9976	0.574
QMNS	5	1.97E+06	3173	1.61E-03	65881	87.39	0.9952	0.471
	10	1.00E+09	19762	1.97E-05	149916	194.95	0.9991	0.513
	20	5.12E+11	72771	1.42E-07	279745	363.78	0.9966	0.521
	100	1.00E+18	371008	3.71E-13	748441	973.27	1.0000	0.359
	255	4.56E+21	716195	1.57E-16	998235	1301.48	1.0000	0.216

TABLE 6.1: Feature Vocabulary Sizes.

- **Divisions** The value of the parameter that directly determines the quantisation of the algorithm's feature space.
- **Vocabulary** The total number of terms possible given the number of divisions. The total number of bins that feature space is partitioned into.
- **Used Vocabulary** The number of unique terms actually present in the collection.
- **Vocabulary Usage** The ratio of used to total vocabulary.
- **Total Postings** The number of index entries - a posting for each unique term that occurred in each image. A direct indicator of index size.
- **Average Terms** The average number of terms per image. An indicator of retrieval time and index size.
- **Curve Slope** The estimated slope of the rank-frequency curve, calculated by logarithmic regression (the Logest function in the MS Excel package.
- **Average Precision** The average precision achieved by the index for the 'standard' test.

This table is accompanied by the rank-frequency data presented in graphs 6.7 to 6.9, and a comparison of some of the metrics from the table above, shown in graphs 6.10 and 6.11.

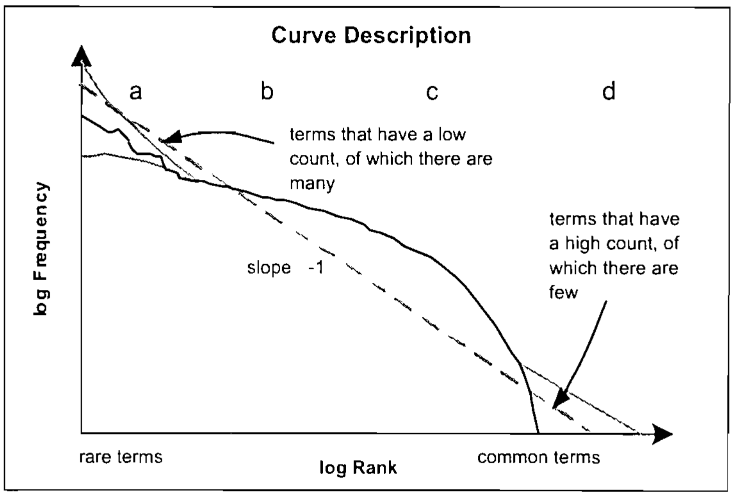
Curve Description
a
b
c
d
terms that have a low count, of which there are many
terms that have a high count, of which there are few
slope -1
log Frequency
log Rank
rare terms
common terms

FIGURE 6.6: Term Distribution Graph Sections

The rank-frequency curves for each algorithm can be broken down into four distinct regions, illustrated in figure 6.6. These are related to the regions identified by Chen in [27], although in these curves there is an additional region, c. Region a (corresponding to Chen's region I), where the rank and density of data points are low, is characterised by a jagged appearance, and was identified by Chen as either concavely decreasing, linearly decreasing, or convexly decreasing. In the case of both the histogram and CCV there is a slight concave curve, whereas the curves for QMNS appear linear. Regions b and d (corresponding to Chen's regions II and III) are linear. However, between these two lies region c which is convexly decreasing. The histogram and the CCV share curves that are of a similar shape, and also undergo a similar transformation as the feature vocabulary is increased.

When the number of divisions along each feature axis is increased the term vocabulary - the number of different, unique, feature terms - increases. Table 6.1 shows the changes in term distribution according to the vocabulary size, showing the usage of the vocabulary, averages for indexing time and number of feature terms per image and the overall average precision for the complete category test.

For each of the features as term vocabulary is increased by changing the number of divisions in feature space the rank-frequency term curve moves towards an approximate slope of 1.0 in the log-log domain (the curve slope column). The distribution also becomes more linear, and as such becomes Zipfian. Both the histogram and the CCV show a continual increase in the retrieval accuracy, as shown by the average precision, whilst the QMNS algorithm peaks at 8 divisions (refer back to figure 6.1). This peak is the point at which the feature terms start to become too rare, each occurring in fewer documents, and hence becoming too specific for general retrieval purposes.

A measure of specificity in the terms is the *vocabulary usage*. This is defined as the ratio of distinct terms present in a corpus to the total possible vocabulary. This is hard to define in text since the vocabularies of natural languages are not strictly defined and continually change as new words are constructed. With feature terms the vocabulary is defined by the extent of feature space and how it is discretised. Consider a set of documents of an arbitrary medium. A feature space for those documents that contains few, sparse, features is highly specific, and if that feature is unable to extract salient features that are common to the documents then it will be a poor feature for retrieval. On the other hand a feature space that is highly populated and contains many clusters that are common to all documents will also be a poor feature for retrieval, since it is too general.

In all three feature types the average precision is higher when the vocabulary usage is low, and both the CCV and QMNS peak at a certain value (0.229 and $9 * 10^{-5}$), the limit of the feature's specificity. The RGB histogram data doesn't show this peak since the algorithm is limited to the number of divisions that can be made, however the curve shown in figure 6.10 indicates that the precision will start to drop at a higher vocabulary size.

As the vocabulary size increases the average number of different terms in documents will naturally increase as the term specificity rises. At the same time the average number of documents per term, that is the number of documents that contain at least one occurrence of a term, will decrease. These metrics are the most important for optimising an inverted index for use with features. The total postings indicates the size of the database - the number of records stored in it. The larger it is, the longer it will take an index algorithm to find the location of all the appropriate postings for a term. Once a term is found it is the average number of documents per term - the postings length - that becomes important.

The RGB and CCV histogram features are less specific than those of the QMNS algorithm, which is shown by QMNS' very low vocabulary usage. This means that whilst the histograms may out perform QMNS in retrieval accuracy, their retrieval speeds rapidly degrade. At its best average precision, QMNS is retrieving an average of 132 terms for 12 documents for a query, resulting in 1,584 index postings retrieved. For the same precision, the histograms are retrieving 173,052 and 75,295 postings for RGB and CCV respectively.

The RGB Histogram

Eleven values of RGB space quantisation were tried for the RGB Histogram, ranging from 3 divisions up to 255 divisions. The lowest value creates a very small vocabulary of only 27 terms, which is only slightly less than the typical configuration of a histogram where 4 divisions are used providing 64 terms. Both of these configurations have a very high vocabulary usage, showing that most of the bins in feature space contained an RGB feature point from an image. Both of these curves do not reach the bottom of the rank-frequency graph (where the frequency is equal to 1) highlighting this further by

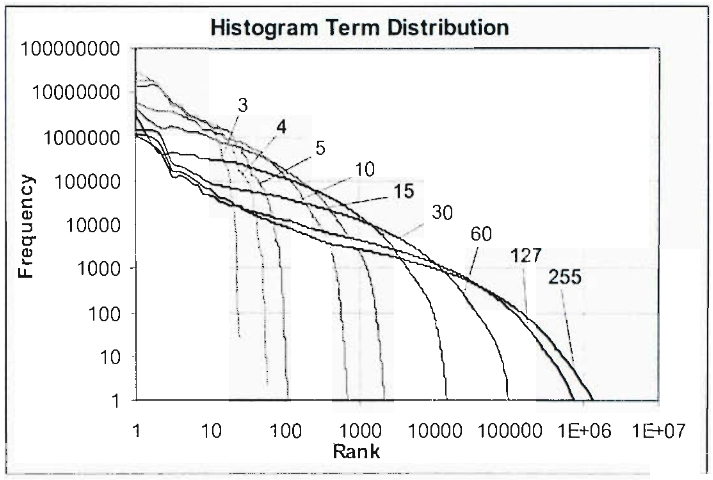

FIGURE 6.7: Term Distribution for the RGB Histogram

showing that all the bins contained more than one point, and resulting in the absence of region b, and a near vertical region d. The low number of terms causes the total number of postings in the collection and the average number of terms per image to be low. Because the feature points are evenly spread between the terms the graph has an approximate slope that is not near -1. The average precision for these two configurations is very low, a clear indication that the terms generated are not very good for using in retrieval.

As the vocabulary size is increased the average gradient of the slope becomes shallower and the different regions become more apparent. Region a tends towards a concave curve, but only marginally. Region b extends, and its gradient moves away from -1, becoming shallower, before entering into region c, which becomes less convex as the vocabulary increases. Region d, initially almost vertical, becomes more shallow, and the whole curve begins to approximate linearity. At the limit, of 255 divisions (where the vocabulary contains a term for every possible 24bit RGB combination), the curve is flatter still, but region c is still convex.

The CCV

The curves belonging to the CCV appear very similar to the RGB histogram's. Like the RGB histogram the lowest number of divisions leads to curves that do not reach the rank axis. When compared with an RGB histogram index that has approximately the same vocabulary usage, the CCV has both a lower average number of terms per image, and images per term. A key reason for this lies in the colour reduction used in the segmentation algorithm, which creates the coherent and non-coherent segments. Rather than counting the raw RGB value for the pixel the colour value that the whole segment was merged to is counted.

QMNS

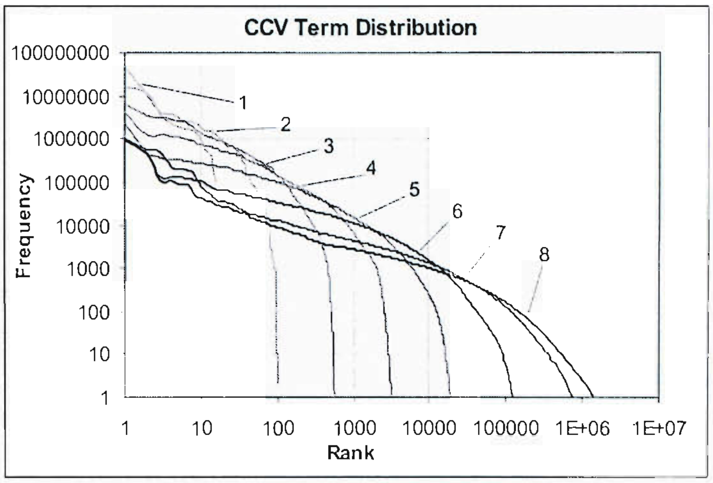

FIGURE 6.8: Term Distribution for the CCV

The curves for QMNS are somewhat different to both the histogram and the CCV, staying closely packed even though the associated vocabulary size is up to 10^{12} times greater than the used vocabulary, and never passing beyond 1,000,000 in either the rank or frequency. Graph 6.9 shows the curves for the QMNS indexes generated from 10 - 100 divisions (in steps of 10) and at the limit of 255.

At 10 divisions the distribution starts in a smooth convex curve, before moving into a very linear section that has a slope close to -1. The final section, d, of this distribution does not become steeper as the other two algorithms showed, but becomes shallower, remaining linear. As the divisions are increased section b becomes apparent, and extends to cover more of the distribution. The curved section moves diagonally down and to the right of the graph, and the shallow 'tail' at the end becomes more prominent. At

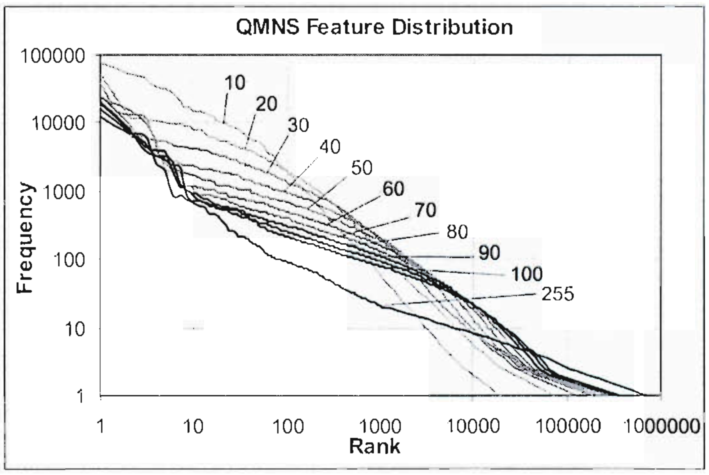

FIGURE 6.9: Term Distribution for QMNS

the limit, the curve of section c has disappeared completely and the whole distribution adopts a concave appearance.

6.3.1.1 Analysis of Distributions

Section a in all three of the distribution graphs appears as quite ragged and uneven due to the low density of data points and greater separation of units at the low end of the log scale. For most of the indexes it appears as approximately linear. This shows that the number of terms which are rare decreases at a regular exponential rate. When it is apparent section b is also nearly linear, and at low vocabulary sizes (low divisions) is of approximately the same slope as section a. When the vocabulary size is high it tends to be shallower than when it is lower.

The curved section c first appears towards the high rank (rare) end of the distribution, obscuring section b, and moving towards the low rank(common) end as the vocabulary size increases. Always a convex curve, it shows a gradual change in the rate of change of the frequency of terms of a particular count. This change shows that the number of terms which occur very frequently begins to drop - that is to say that the common terms become more common, with the number of terms at each ranking becoming rapidly fewer. This section is followed by another linear section, showing that this rate of change has ended and the frequency returns to changing exponentially.

The exception to this is QMNS, where section d does not continue on from section c smoothly, but instead contains a kink. Whilst this artefact does result in the complete loss of the curved section c at the limit of the vocabulary size, it displays a sudden change in the terms. From the gradual decrease in the rate of change of term frequencies we see a sudden increase in the count of the most common terms. Looking at the data points for the distribution it can be seen that this is caused by one value - the most common term. This term occurs, in most cases, far more times throughout the collection than any other. It is a unimodal patch which is a very light grey, but contains slightly more red than blue or green.

Quite naturally the distribution will show changes according to the colours that are in the collection. Since the algorithms that have been used are global colour features the distributions have similar shapes, with QMNS being the least similar due to the inclusion of higher order global colour features (the bi- and tri-modal patches). The curve in the distributions shows that there tend to be more common terms than rare, and this moves and becomes less marked as the vocabulary is increased due to the increased quantisation of feature space.

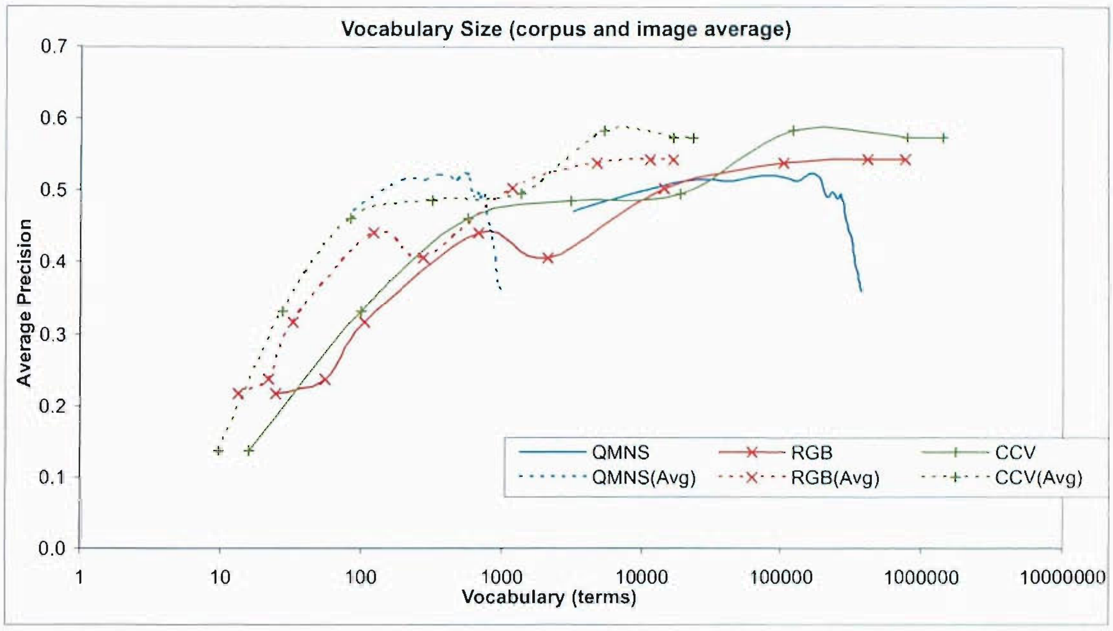

FIGURE 6.10: Vocabulary Size and Usage versus Average Precision

6.3.1.2 The Distribution and Retrieval Performance

As the number of divisions and hence vocabulary size is increased, the vocabulary usage increases - but not at the same rate. This is due to the fixed number of feature points and the increased partitioning of feature space. This leads to a more regular distribution of the feature points between the available terms, which is displayed by the slope of the curve, which approaches -1 as the vocabulary increases. If the original feature space contained uniquely random points, of a continuous, rather than discrete, nature, and the resolution of the quantisation was smaller than the smallest distance between two points, the distribution of the terms would reduce to a single point. This would happen because every single occurrence of a feature point would be binned and would become a term in its own right. Since every term would have an occurrence frequency (count) of 1, the single point on the rank-frequency graph would be located at rank 1[2], and at a frequency equal to the total number of feature points in the entire collection. This would be a highly undesirable situation, since a query term could only ever match one image at most. It is also a situation that would be very unlikely to occur, since feature space is not typically continuous[3], and so any quantisation finer than the original resolution of the discrete data will have no effect.

[2]Rank 1 here does not refer to the frequency of each term, the point is the only point and so must be ranked first. The frequency on these curves indicates the number of terms which occur with a particular count, i.e. the far left of the graph shows terms that have a low count, yet there are many of them, and vice-versa on the far right.

[3]Again the digital nature of floating point numbers is apparent, but can be ignored due to the precision of double accuracy floating point number.

Graph 6.10 shows how the vocabulary size of the algorithms compared to the resulting average precision. Average precision is a meaningful metric to use for comparing the different feature's distributions because of its common grounding - the image collection and categories used in the test. In the diagram there are two sets of curves - those that are solid, which show the corpus vocabulary size (the number of unique terms in the collection) and those that are dashed, which show the average number of unique terms per image.

Both the histogram and the CCV exhibit a high vocabulary usage, shown on this graph by the proximity of solid and dashed curves. QMNS has a significantly lower usage, especially when the vocabulary is at its largest. Whilst both the CCV and the colour histogram surpass QMNS in greatest average precision, QMNS does achieve a peak precision of 0.52 at a significantly lower vocabulary usage. More significantly, it is for QMNS that the average number of terms per document attains a higher precision at lower values. It is this metric that is the most important for considering the effectiveness of a particular combination of a feature extraction and quantisation algorithm. The position of the curve will determine the overall worth of this strategy - a curve that fits high into the top left will have a very good retrieval rate with a very low number of terms - a good combination of accuracy and speed.

The three curves shown for each of the algorithms approximately follow a log-linear curve, before peaking at a high vocabulary size. In the case of both the RGB histogram and the CCV the end of the graph indicates the end of the workable limit of both algorithms, since the quantisation parameters cannot be adjusted to increase the vocabulary size further. QMNS, however, contains a clear peak indicating its useful limit, after which it begins to drop down. All three of the curves contain undulations, resulting in some local maxima and minima, the cause of which has not been ascertained. Graph 6.11 shows the average terms graph for just QMNS, where the undulations are more apparent. The solid section of the graph contains data points taken at a resolution of 1 quantisation division, and the dashed section contains points taken at a resolution of 5 divisions. Since the resolution of the curves is determined by the quantisation parameter (which for the histogram and CCV is integer) little more information about these peaks

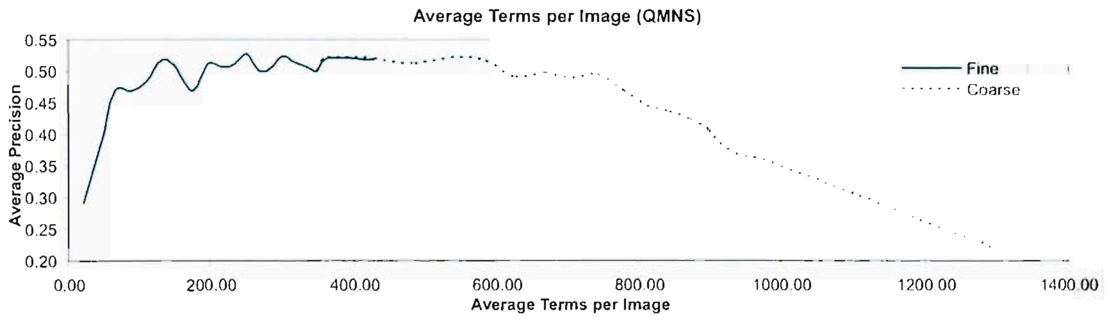

FIGURE 6.11: Average QMNS Terms per Image versus Average Precision

can be determined. A possible explanation for these undulations lies in the quantisation procedure itself, which may be subject to a form of *moiré* as the already discrete features are discretised further.

The results presented in this section demonstrate the affect on the index of a change in vocabulary size. The size of the vocabulary directly affects index size and retrieval speed and also changes the quality of retrieval. By generating the distribution data (which fits, in general, a Zipfian curve) for different vocabulary sizes, the size of the index can be configured to provide a suitable balance between retrieval time and quality.

6.4 Specificity, Exhaustivity and Discrimination

So far in this chapter we have looked at how an index transforms information from a document into a form that may be stored and queried rapidly. These notions of information are however, not suitable for measuring the utility of an indexed feature - in high level, semantic, terms. We would like to know how good a particular term, or set of terms are at discriminating relevant documents from non-relevant. We would like to be able to choose between a specific index on the one hand, which would allow us to focus on a particular topic, and an exhaustive index, where we can search for many different topics.

A highly specific feature is a one that fits into a narrow information domain, one the is not very specific into the broad domain. Of the metrics shown so far there are a some which have an obvious correlation with specificity, and some with discrimination.

6.4.1 Vocabulary Usage

A measure of specificity of terms is the vocabulary usage. This is defined as the ratio of distinct terms present in a corpus to the total possible vocabulary. This is hard to define in text since the vocabularies of natural languages are not strictly defined and continually change as new words are constructed. With feature terms the vocabulary is defined by the extent of feature space and how it is discretised. Consider a set of documents of an arbitrary medium. A feature space for those documents that contains few, sparse, features is highly specific, and if that feature is unable to extract salient features that are common to the documents then it will be a poor feature for retrieval. On the other hand a feature space that is highly populated but contains many clusters that are common to all documents will also be a poor feature for retrieval, since it is too general.

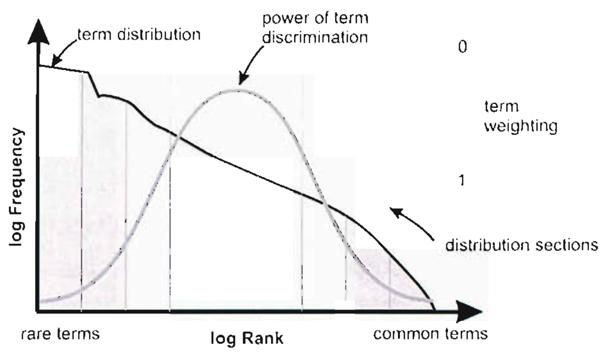

FIGURE 6.13: Term Discrimination

The term distribution shows a lot of information about the power of discrimination of terms for the purpose of retrieval. Terms that occur at the far ends of the distribution are either too rare, or too common[4].

Typically common words are removed from many text indexes by means of a predetermined stop-list, chosen because they are functional words. Removing these terms which don't contribute much to (typical) queries reduces the size of the index and the retrieval time.

Most feature terms, however, cannot have any form of importance (in semantic terms) assigned to them - the relative worth of them is determined by the number of times they occur in a corpus of documents. Figure 6.13 shows a rank-frequency graph, annotated by a curve that denotes the power of discrimination (or resolving power).

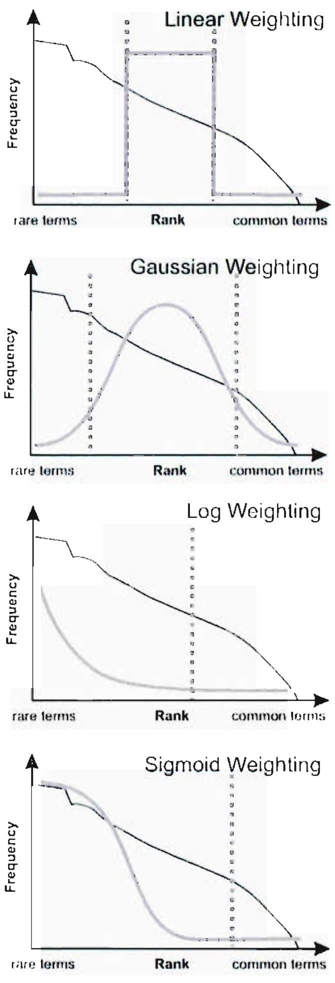

FIGURE 6.12: Term Weighting Curves

To date there is no analytical method for determining this curve, and therefore the levels at which to remove from the index terms that are too common or too rare. Removing terms from the index is often complemented by weighting the terms left in the index - again based on this notion of discrimination. Some of these algorithms, such as TF*IDF, opt for modelling only one side of this curve - by adversely weighting terms which are very common. Again a tradeoff has occurred - removing terms makes the index faster, but less precise.

Depending on the application of the index we can make assumptions about the shape of this curve, and the location

[4]Quite often the literature gets the ends mixed - the order is determined by whether the graph is rank-frequency or rank-count

of potential cut off levels. The diagrams in figure 6.12 present some of the potential weightings and cut off points:

The linear weighting exclusively concentrates on the central area of the graph - terms that are neither too rare nor too common. Such a weighting would benefit a broad category retrieval application that maintains a large number of documents, since the cut-off points are at the end of the high linear section. The location of the cut-off points minimises the number of terms in the index, allowing for rapid retrieval.

The gaussian weighting is again targeted at the broad category, however it makes use of more terms and weights them rather than simply dropping them from the index.. The cut-off points remove some of the rare and common terms, those that are left in are weighted appropriately. Such a scheme could provide slower retrieval than the linear weighting due to the additional effort of weighting the terms, but provides better precision.

The log weighting, used commonly in text retrieval, favours rare terms for retrieval. This scheme is used in conjunction with the document frequency of the term - that is the number of documents in which the term occurs - such that a rare term that occurs many times in one document but very little throughout the entire corpus is weighted favourably.

The sigmoid weighting adopts a similar approach, only altering the weighting so that the balance is shifted back towards the centre of the distribution.

The approach used will depends on the terms that are important to the application, and the number of terms that can realistically be dropped from the index without affecting retrieval quality too much.

6.4.2 Index Pruning

In order to determine the amount that terms contribute to the effectiveness of an index different sections of the distribution may be extracted, and inserted into a separate index, then tested. Since each term contributes towards the query similarity score independently, the sections tested do not need any overlap. Rather than dividing the distribution equally from the log perspective, each section may contain the same number of unique terms. This will result in sections like in figure 6.14. If the graph was of a rank-count distribution, as opposed to the rank-frequency distributions used here, or if the distribution were divided so that each section contained the same number of features (occurrences of terms), then the divisions would appear largest on the left of the log-log scale.

The manner in which the sections are divided has implications for how each section will perform. Using the equal term approach will create sections that contain very few

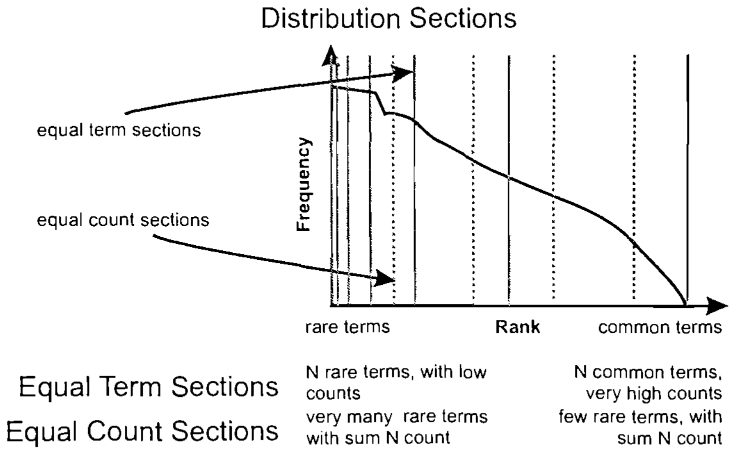

FIGURE 6.14: Sections of Equal Term Count in the Distribution

features at the rare end (low ranking) of the distribution (since each individual term is rare and has a low count there are many of them, giving it a low rank), and sections that contain the same number of terms yet each at a very high count. The equal feature approach will mean that sections contain approximately the same number of features (equal to the sum of the count of each term).

The sub-indexes were generated by listing all terms in the index in ascending frequency order and then splitting and extracting all the terms for each section and inserting them directly into a the new index. For the equal term approach the last section will be smaller than the others if the number of sections doesn't divide exactly into the number of terms. For the equal count approach terms are added to the section until the sum of the term counts exceeds the maximum count size of the section. This means that in sections at the common (high ranking) end of the distribution may exceed this value quite considerably, due to the very large term count of common terms, and also that additional sections may be required.

Section	Equal Terms			Equal Count		
	Terms	Count	Hi	Terms	Count	Hi
0	1977	1977	1	19164	117966	102
1	1977	1977	1	407	118242	397
2	1977	1977	1	98	118836	512
3	1977	1977	1	39	122089	1141
4	1977	1977	1	21	119478	812
5	1977	3913	2	14	121880	895
6	1977	5158	3	10	130761	654
7	1977	9034	6	6	151257	1033
8	1977	21805	15	3	177928	1365
9	1969	1128642	1365	0	0	

TABLE 6.2: Size of the Sub-Indexes in Both Approaches

The indexes from which the sub-indexes were generated were for the QMNS algorithm, using the same parameters as the distribution tests in the previous section. Retrieval was performed using the CMR4 algorithm. Four indexes were used, with quantisation divisions of 5, 10, 15 and 20. Each equal term index was split exactly into 10 sub-indexes, and each equal count index was measured to split into 10 sub-indexes, but did in fact split into 9, 14, 15 and 10 sub-indexes respective to the quantisation divisions. Table 6.2 shows the sections for the two types of sub-index when the QMNS feature space is quantised by 5 divisions. The **terms** column shows the number of terms in the section, the **count** column shows how many individual features (the sum of the counts for all the terms) are in the section and the **hi** column shows the highest count of any term in the section. Note how the terms value stays constant in the equal terms index, and how the count stays approximately the same in the equal count index.

The results for the two different approaches, shown in graphs a and b of figure 6.15 appear to be at opposites, yet with further analysis they concur. The first section (section 0) for the equal terms approach shows a clear separation for each of the four sub-indexes, with the sub-index with the largest vocabulary (the largest quantisation value) performing better than the others, which come below it and in descending order of quantisation value. Since the overall vocabulary is bigger the number of terms in each section will be bigger, which provides better retrieval performance (at least for the first two sections). From sections 0 until section 3 there is an overall downward trend in the average precision, which without further analysis cannot easily be explained. From the third section onwards the average precision continually increases and as the terms become more common.

Graphs a and b appear to indicate that rare QMNS terms are not useful to retrieval, and that the more common terms are. Closer inspection reveals that sections 0-4 contain almost exclusively rare terms which each occur only once throughout the corpus. Such terms are not ever useful for retrieval, and can be safely ignored. As each term's individual count increases, so does the number of documents it is likely to occur in. At the common end of the graph the terms are occurring in many more documents, and so the probability that a query and a relevant document will share common terms is high.

The second graph, in figure 6.15.b for the *equal count* approach shows a different result. In the graph the left hand graph shows the equal term approach and the right the equal count approach. Here it is the sections at the rare end of the graph that provide better retrieval performance. Table 6.2 shows this is because the first section of the graph contains over 96% of the terms in the index. The remaining sections are effectively sub-indexes of the *last* sub-index of the equal term approach, and show a decline as the number of terms in the index drops to just a few terms.

From these two views of the distribution it could be concluded that for QMNS it is the *common* terms that are the most important to retrieval. The tradeoff between retrieval

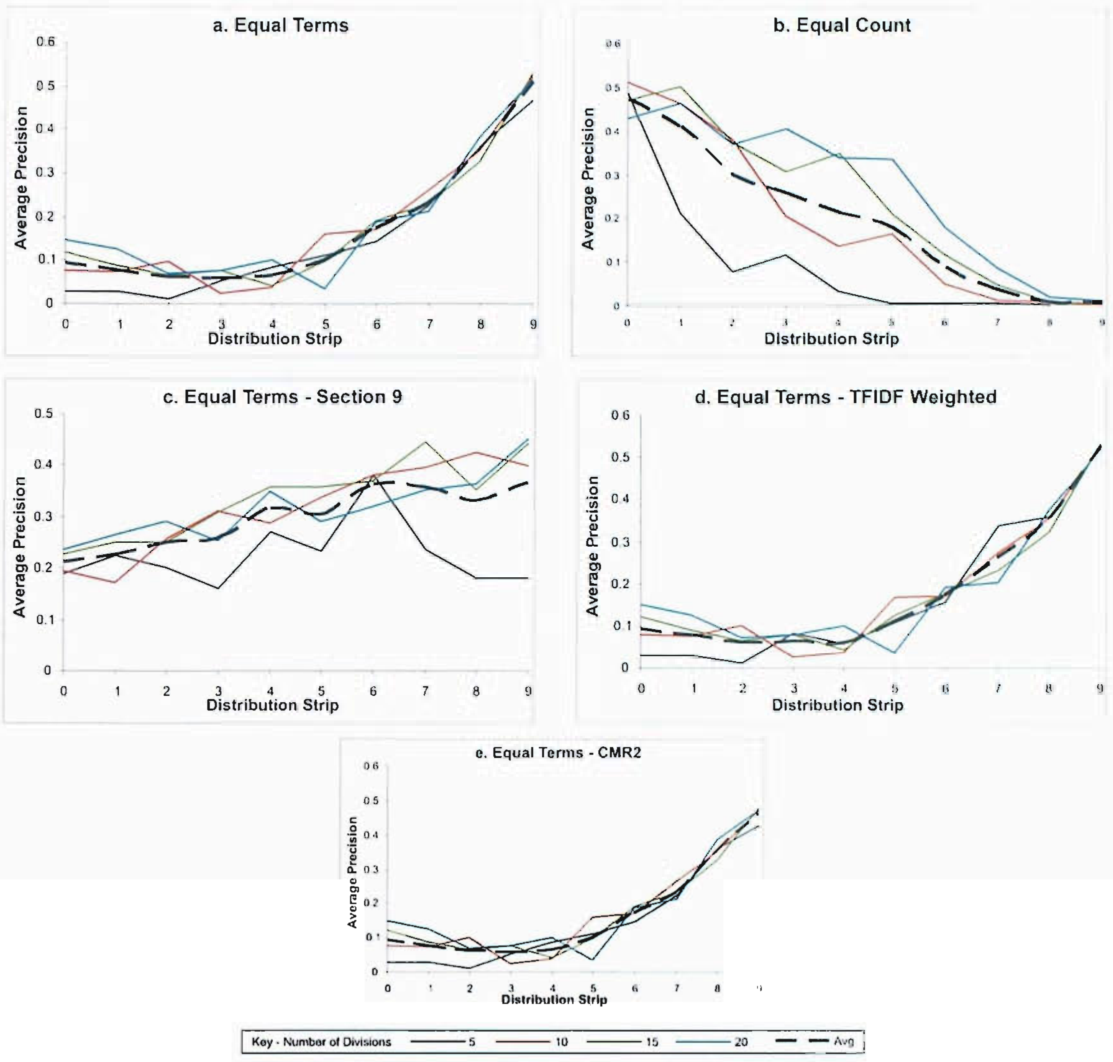

FIGURE 6.15: Retrieval Performance of Sections Within the Distribution.

time and performance for the image collection tested, and the ground truth known about it, has a balance that would enable just section 9 from the equal terms group to be used. This would provide retrieval accuracy that was just slightly less than using the entire set of terms, but only using a tenth of the terms.

Looking at the distribution of terms as a 'bag of features' is useful in providing a lot of information about the algorithm and the image collection it is applied to, but we lose a lot of information regarding the documents from which the terms are derived. A term that occurs regularly (has an average count in the distribution) may be spread evenly throughout the documents of the collection, or it may be concentrated in a few. In 2.4.3.2 we saw that a good weighting system for text is TF*IDF, a weight that acknowledges terms which occur infrequently throughout most of the collection, yet are frequent in some documents, must be important to those documents. We can use this as another approach to analysing the distribution, and split it up according to the *regularity* of the terms.

When the terms are ranked according to the TF*IDF[5] weight and divided into equal term sections the results are only slightly different from the first equal term test. A quick comparison of graphs 6.15.a and .d makes this apparent. This demonstrates that the document frequency is highly correlated with the corpus frequency, and that the same sections in the two equal term approaches contained nearly identical sets of terms.

The equal term graphs show that the commonest 10% of the terms in the index may be used reliably for full image, general, retrieval. Graph 6.15.c shows that within this 10% it is still possible to create even smaller sub-indexes (containing 1% of the terms of the original index) which still achieve an average precision of over 0.4.

The tests presented in this section have demonstrated that the size of a QMNS index can reduced by removing terms which are not good discriminators. It has been shown that the global QMNS colour algorithm is, for the selected image collection and ground truth images, effective in the 'common' part of the index only, and that a significant proportion of the other 'rarer' terms can be removed without degrading retrieval quality. Since the quality of retrieval is measured by the ground truth another set of categories in the collection might provide different results. A category of images that contains colours not present often in the other images in the collection would provide the best retrieval results when used with a sub-index containing rarer terms. Such a specific index would then only work well a small number of images, and would provide good recall and poor precision.

Alternatively, an exhaustive index can be generated that contains all the terms, but which may be weighted to favour some part of the distribution (such as the common terms). The index would be better at retrieving different types of images, and would have a poor recall due to the and high precision due to the extra features being retrieved.

6.5 Summary

This chapter introduces techniques which can be used to optimise a feature term for inclusion in an inverted index. An extensive set of indexes was generated for QMNS by modifying signature generation parameters, providing an excellent comparison of term distributions. The feature term distributions of QMNS, and two other features, an RGB and a CCV histogram, are analysed.

The key findings in this chapter have been:

- QMNS Vocabulary: The clustering performed by the mean shift on neighbourhood pixels can be controlled, which has a great effect on the distribution of terms produced. Testing then shows which type of distribution is the best for retrieval.

[5]Here TF indicates the corpus frequency of the term, whereas it is the query term frequency when used in a query

- Degree of Quantisation: The amount by which the feature space is quantised directly effects the size of a feature's vocabulary. The effect it has on different features will depend on the nature of the feature itself - the average number of terms per image is significantly less in QMNS than in the RGB or CCV histograms.
- Term Distribution: The distribution of terms approaches linear in the log-log domain as the vocabulary is increased - which makes it easier to model as a Zipfian distribution.
- QMNS Term Specificity: It has been found that rare QMNS terms are too specific and are not useful for retrieval, whilst the most common terms are the best.
- Index Optimisation: It is possible to remove a significant number of terms from a QMNS index because they are poor discriminators, as demonstrated by a test of different portions of the index. There are a number of different ways in which the index can be partitioned.

Chapter 7

Conclusion

7.1 Overview of Thesis

7.1.1 Background Material and Literature Review

This thesis begins with an overview of the field of content based retrieval, where it is introduced as a primarily cognitive activity. An important distinction that is made is that content based retrieval is about the retrieval of documents - defined as an ordered, coherent, collection of information with a common concept, that may be composed of multiple forms of media 2.2.

The focus is then moved to the applications of computer vision and CBR 2.2.1, illustrating their role in medical systems, web retrieval, digital archives, manufacturing inspection and autonomous systems. It is shown that there are four key stages in CBR - acquisition, analysis, storage and retrieval. The last stage is that which is exposed to the user (be it a human or machine), for which there are a number of paradigms - query by example, query by sketch, browsing and navigation.

The next section returns to the idea of document retrieval, discussing the structure of media within documents, and then introducing the concept of features 2.3.2. The feature is described broadly as an entity which is a representation of some salient aspect of the document - a statement intended to highlight the potential range and scope of features. The role of a document's signature as a container for features is given, and a series of metrics for comparing feature points in Euclidean space are described - the Minkowsky metrics, correlation metrics, and three vector set comparison metrics.

Since feature comparison is essentially an exercise in classification, sections 2.3.5 to 2.3.6 provide examples of clustering techniques such as the spanning algorithm, K-means and the mean shift (which is used by the QMNS algorithm). Approaches to combining the results from classifiers is given: the use of knowledge rules and fuzzy logic, probabilistic and belief approaches, and neural networks.

Section 2.4 introduces the field of text retrieval, beginning with the descriptions of the inter-document parsing process, which is part of the acquisition and storage stages, and the intra-document parsing which involves the lexical analysis of text corpora. The analysis is formed of a number of stages: text parsing (token extraction), token handling (manipulation and transformation, such as morphology) and the token indexing.

The indexing of text is the area in the field of text IR which this research has drawn from the most. Three types of index are listed: The inverted index, which is the most commonly used, and has excellent retrieval performance at the loss of update speed, the signature file, a system based on hashing to determine the probability that a term exists in a document, and bitmap files which have excellent document-document comparison speeds, but are very large.

Retrieval of terms from the index is presented in section 2.4.3 in the three forms of boolean, weighted, and vector space model retrieval. This is followed by some notes on advanced topics, including relevance feedback, spatial and adaptive retrieval, and content based navigation.

Image features are considered in section 2.5, which briefly discusses types of colour, textural, shape and hybrid features, with citations to relevant research. After this some approaches to the problem of indexing multi-dimensional data, such as that of features, are given. GRID files and R-Trees are two such techniques which may be used in conjunction with dimensionality reduction algorithms like principle component analysis or the K-L transform. The Kohonen self organising neural network offers a useful alternative from the artificial intelligence community.

In section 2.7 the inverted index is examined in more detail, starting with a look at the Zipfian distribution found in the terms of natural language which is provides a useful model for the index. The structure of these indexes is explored, and attention given to optimisations that can speed up the process of retrieving data.

This background chapter concludes with notes on the future of content based image retrieval.

The third chapter explores the applications in which computer vision is employed. The requirements of the multimedia information system are split into two groups: The source of the information that is to be stored, analysed and retrieved, and the target users from whom queries are issued and responses returned. The information source will have certain domain, scope, and volume, which must be considered when designing the architecture of the system and the features which will be required. User considerations include the response time, query scope, and retrieval accuracy.

Five types of MIS architecture are given: The medical PACS systems (which may have sources such as MRI, CAT and PET scanners), digital libraries (archives of existing material), web retrieval, life recording (continuos storage and query requirements), and content delivery systems.

The second section of this chapter is a review of small number of image retrieval systems (section 3.3.1 to 3.3.7). QBIC, the first commercial available content based retrieval system is described in detail. The media analysis subsystem performs histogram extraction, object segmentation, video shot detection and motion analysis. The features are then stored by the modular indexing component, for which an R*-tree experimental index was tested. The user interface offers the ability to query by example or sketch, and provides palettes for the user to select prototype features from.

The academically highly successful MARS project is architecturally and functionally very similar to QBIC, however the scope of work accomplished in this later project is far greater. Extensive work was done on video indexing and multidimensional feature indexing.

The two systems that are described next (VisualSEEk and VideoQ) both offer novel query interfaces: The ability to diagram queries visually associating feature prototypes with objects in the former, and a system for sketching the motion of a desired video object in the latter.

The MAVIS systems include support for multimedia navigation, with the second project of the same name offering generic multimedia links and a concept thesaurus. The Artiste and Sculpteur systems offer image and 3D retrieval capabilities for museum artifacts. The last system described is from the Viper group who developed MRML, the multimedia markup language, and created the GiFT (GNU image finding tool).

7.1.2 Core Research

Chapters 4 to 6 form the core of the thesis. The first of these chapters introduces the architecture and applications used to perform the research. Collectively titled Invistor, the system comprises three main components.

Section 4.3 deals with the CBIR image indexer. This component encapsulates all four stages of CBR (section 2.2.1). A relation database is used to store image metadata, binary signatures and the term features, which are generated using image analysis modules written for the Artiste project. Image indexing can be performed in parallel by running multiple instances of the application. The output of queries is in the form of HTML and a simple hitlist file.

The next component described is the index analysis module. This small Java component will connect to a feature term index and will generate its Zipfian distribution automatically.

The last component is the results analysis module that is designed to run sequences of queries using the CBIR indexer, and produce precision and recall data and the average precision metric for ground truth categories which have been defined. It is also capable of executing sub image queries and calculating the sub image retrieval metric (section 5.4.2).

QMNS, the Quantised Multimodal Neighbourhood Signature, is presented in chapter 5 which is broken down into the following sections: A description of how the original MNS algorithm functions and the extensions written to form QMNS (sections 5.2 and 5.3), the test environment, image collection, and test scripts followed by an analysis of the results (section 5.4 and 5.6), finally presenting the testing of QMNS when it is used to generate multimodal signatures 5.7.

The MNS signature is extracted by dividing the image into regular patches, or neighbourhoods. The pixels from each neighbourhood are clustered using the mean shift algorithm to find the colour modes. Information on bi-modal patches is then clustered using the mean shift to produce the signature points. Comparison is performed using a form of stable marriage matching.

QMNS quantises the bi-modal features that are extracted from each neighbourhood, labelling them as terms. The frequency of each quantised bi-modal term is inserted into the index with the document ID.

After the descriptions of the signature generation the chapter moves into describing the three index operations supported - term insertion, deletion, retrieval - including descriptions of the four retrieval techniques. CPR1 awards 1 to a target image for each term that it shares in common with the query, whilst CPR2 adds the query term frequency in an attempt to return target images where the query is a subimage. CPR3 is intended for full image retrieval and uses weighted normalised frequencies, as does CPR4 but with a TF*IDF scoring method.

Section 5.4 presents the image collection used for testing, and the ground truth categories used for the precision and recall statistics. Section 5.5 details the three tests that are initially executed: Variations of parameters in MNS (designed to find the best configuration for retrieval), QMNS full image testing and QMNS subimage testing.

After this the results are presented, and from the testing of the different configurations of our implementation of MNS, we have found an optimal set of parameters. A key addition to the original MNS algorithm is the reduction in the number of pixels required to find the modes of a neighbourhood. We have shown that by using 30% of the pixels in each neighbourhood we are able to achieve better retrieval results than when using all of them. Reducing the number of points which are clustered using the mean shift also leads to much faster feature generation times. Another parameter which proved to be very influential on retrieval accuracy is the window size of the mean shift. By making this smaller, and therefore increasing the number of clusters per neighbourhood, we have increased the number of different features present in an image. This has in turn led to better results.

The first series of QMNS tests show that the indexing times for QMNS are much slower than the RGB histogram and the CCV. The rapid version of QMNS (using a reduced number of neighbourhood pixels) is shown to be as fast as the multiscale CCV feature generation. The average precision of the different CPR retrieval algorithms is much better than the normal MNS feature, and equivalent to the RGB histogram and the CCV. Improvements are seen in QMNS results when quantising using 5 divisions (bins) per feature axis. Retrieval speed for QMNS is slow, but is 10 times faster when using the rapid index - due to the reduced number of terms which must be retrieved.

Section 5.6.3 presents the subimage retrieval results which show that there is little difference between the CPR algorithms in this test. Again improvements are seen when 5 bins are used. The MCCV proves to be superior to all other tested algorithms at subimage retrieval.

The next section provides an analysis of the findings from the tests described above. It is theorised that the MNS features used suffer when images are scaled, because the neighbourhood sizes stay the same, and that a possible solution would be to use a hierarchical series of different sized neighbourhoods to generate features.

The original version of MNS only uses bi-modal patches, which is addressed by the work in section 5.7. In multimodal retrieval the uni-modal and tri-modal neighbourhoods are kept in the signature, and quantised into RGB and RGB^3 space respectively. Larger sized neighbourhoods of 16x16 pixels are also tested. The indexing times for such patches is naturally longer even when a reduced number of pixels is clustered by the mean shift.

43 test indexes were generated, the parameter changes being the neighbourhood size, minimum mode (cluster) size, mean shift window size and quantisation factor. As the bin size is increased the number of uni and bi-modal terms in the index increases, however the tri-modal terms decreases. Using the indexes a optimum set of parameters for multimodal retrieval are found.

The full image and sub image tests from earlier in the chapter are repeated for the multimodal indexes (section 5.7.3 and 5.7.4). It is found that the use of tri-modal terms alone leads to very poor retrieval, and that the best combination is offered by uni- and bi-modal terms.

The last chapter discusses generalised feature indexing, and introduces indexes containing terms from the RGB histogram and the CCV. The testing of QMNS configurations are extended to cover a far wider range of the parameters, resulting in 504 indexes. The effect on the term vocabulary and the index size is examined in detail when the minimum mode size, the mean shift window and the bin size is changed.

Correlations between the Logest of the index distribution and the vocabulary versus the average precision are seen showing that a larger number of terms offers better retrieval. It is noted, however, that the best configuration will be one which does actually have a small vocabulary, and hence the average number of terms per image will be lower leading to faster retrieval.

Section 6.3 presents results for the indexing of QMNS, the RGB histogram, and the CCV at different levels of quantisation up to the maximum level possible. All three of the algorithms have a peak where retrieval is best. This data is accompanied by graphs of the Zipfian distributions of the indexes, which are seen to become more linear in the log-log domain as the vocabulary increases. Four sections in the distribution are observed and discussed for each of the three feature algorithms.

The terms of QMNS are deemed to be more specific because far fewer of them than for either the RGB histogram or the CCV are required to produce a particular average precision. This is also seen in the average number of postings retrieved, where the CCV must retrieve approximately 50 times more terms than QMNS, and the RGB histogram must retrieve 100 times more. Even though the QMNS is outperformed in most cases by the histogram and the CCV in terms of average precision it is considerably faster to retrieve.

The final section of this chapter looks in further detail at the specificity of the terms in the QMNS indexes. The merits of four different weightings are discussed - linear, guassian, log and sigmoid weighting. The premise behind these weightings is that certain terms in the distribution - those that are very rare, and those that are very common, are not useful for retrieval because they are not good discriminators. This was demonstrated by the poor retrieval ability of the tri-modal patches which were very few in number.

To test whether certain sections of the distribution are indeed good discriminators the index was pruned. Section 6.4.2 describes how these test indexes were formed. Two approaches to dividing the term distribution are given - equal term sections, where each section contains the same number of terms from the vocabulary, and equal count sections, where the sum of the term frequencies in each section are approximately equal.

Four QMNS indexes (with different bin sizes and hence vocabularies) were split using the two techniques and the standard category retrieval tests executed against them. We see that the rare terms are not good for retrieval, and find that it possible to prune 90% of these terms from the index without degrading retrieval performance. It is noted that this is only demonstrated for the ground truth categories and image collection used in the testing and might not be true for other collections.

The chapter is summarised by considering the exhaustivity of an index. One that is exhaustive will contain all possible terms, which means that recall is improved at the cost of the size of the index and hence retrieval speeds. The configuration of the index is very important for retrieval, and the steps taken to optimise it will always result in less information - but it is possible to determine which terms are actually good discriminators for a particular image collection.

7.2 Contribution to the Field

The aims and objectives of this thesis, listed in section 1.2 have been fulfilled, and are presented here with a summary of the work accomplished:

- To demonstrate that sub-second content based image retrieval is possible from massive image collections:
  - **Use available feature extraction algorithms, or develop a novel global colour feature extraction algorithm:** The MNS feature extraction algorithm was adapted and improved, and three histogram algorithms were used for investigating retrieval techniques.
  - **Develop a technique for transforming image features into a form storable in an inverted index:** The QMNS algorithm was designed explicitly to quantise multi-dimensional feature space into labelled bins with their associated frequencies (collectively called feature terms).
- To provide techniques that allow analysis of the distribution of the feature space of different features:
  - **Identify how changing a feature extraction algorithm's parameters change the distribution of points in feature space:** Rigorous work was performed in chapter 5 to find both the most effective parameter set for retrieval (measured using recall and precision data) of four image features, and the distributions analysed.
  - **Investigate the effects of different techniques of feature space quantisation:** The degree of quantisation of feature space was varied and the results presented in chapters 5 and 6.
  - **Present optimisations for indexed feature descriptors:** Using data from the analysis of different parameter sets it was found that a significant proportion of image feature terms could be excluded without reducing retrieval performance.
- To demonstrate whether feature term retrieval is flexible and extensible.

  - **Show the potential for heterogeneous feature storage:** An index was populated with feature terms from three different image features, and retrieval against this index performed successfully.

- **To develop a novel global colour feature:** The Quantised Multimodal Neighbourhood Signature is a novel extension to a global colour algorithm, which is flexible, fast, and has highly specific features.
- **To investigate the distribution of this algorithm's feature space:** Extensive testing was performed on the QMNS algorithm, involving varying feature generation parameters, producing a large number of indexes for which the term distributions were analysed.
- **To develop techniques for optimising image features for use with an inverted index:** The pruning of term distributions has demonstrated that not all feature terms produced by a feature algorithms are useful, resulting in a more compact and concise index.
- **To demonstrate such techniques are applicable to any multimedia feature:** The RGB histogram and CCV were shown to operate well when stored in an inverted index.

7.3 Future Work

There is a significant amount of work which can be done to extend the research presented in this thesis. This can be split into three primary categories: Techniques for feature generation and term extraction, evaluation of retrieval, and indexing technologies.

7.3.1 Term Generation

The work on QMNS itself is quite comprehensive, however there are still certain areas which could be given attention. In terms of implementation there is many improvements which could be made to the algorithm. In particular the use of the mean shift could be considered a disadvantage because of its inherently slow processing. There has been some work on fast mean shift algorithms [14] which could prove useful, or a different clustering algorithm could be used.

A significant modification discussed in section 5.6.4.2 would be to introduce multiple levels of hierarchical neighbourhoods. Cluster the large sized neighbourhoods might prove very time consuming so it might be useful to resample them to the same size. This use of different neighbourhoods would help in making QMNS more scale invariant.

QMNS holds no spatial information, and it could be that the addition of a spatial variable into the feature would help certain types of retrieval. This could be achieved by extending the feature space to include two spatial axes, which would be quantised with the colour data. This variable would need to be quantised, or clustered, very roughly, otherwise the terms generated would be too specific - although there might be certain applications where this was desirable.

Some work was done on creating a colour invariant version of QMNS, which would reflect the work by the original authors of MNS. The 5 dimensional chrominance feature proved to be worse than the RGB feature, and with a higher dimensionality was slower to generate and compare. The use of an HSV colour space could result in feature that was less susceptible to colour change.

Chapter 6 includes some work on the RGB and CCV histograms and the distributions that their term indexes have when different levels of quantisation are used. It would be very worthwhile examining not only these features, but other features in greater detail. Other features were written for the Artiste image processing library, including a PWT (pyramid wavelet transform) [44], could easily be modified by adding a quantisation routine so that they would generate feature terms.

Multimedia features could also be explored, and comparisons between them found. This would help extend the available information on feature term distributions and would help lead to a generalised model of feature terms. Initial investigations into the storage of feature terms was investigated, and this avenue would be an exciting one to continue on. Retrieval of postings from an inverted index does not require a specific comparison algorithm for individual feature types - only a model of the distribution of the terms - which means that this approach could be very flexible and portable. Needing no knowledge of the underlying features makes query processors and clients far simpler, and with appropriate labelling of terms it would be possible to weight the different feature individually.

7.3.2 Evaluation of Retrieval

Without good evaluation techniques it is very difficult to determine the worth of any retrieval algorithm, leaving results susceptible to subjective analysis. Section 3.3.7 looks at the work of the Viper image retrieval group who have had influence in the organisation of a content based image retrieval evaluation group - known as Benchathlon [96]. This aims to provide tools and techniques for assessing retrieval performance.

The use of only one image collection in this research has meant that results are very limited. If the tests were to be executed using a standardised image collection, with accepted ground truth categories then the data produced would help to further establish the theories presented in this research. In particular the scalability of the system was never tested with a large image collection (see forward to the next section).

7.3.3 Indexing Systems

The inverted index is central to the work presented here. The implementation (described in section 4.3) was only intended for proof of concept, and did not have an efficient retrieval engine. Much work could be done on improving the index itself, to provide a better postings structure. The postings could be compressed and maintained in blocks [111] or it could be distributed across multiple machines. The understanding of term distributions would be very valuable in such situations, allowing the most often retrieved terms to be separated.

The CMR algorithms used to rank documents in the index would benefit from a better implementation - in particular the inclusion of a system for dynamically weighting terms would be highly beneficial. Given the knowledge gained from an analysis of the term distributions and their pruning (sections 5.6.4, 6.4.2) a better TF*IDF weighting algorithm could be developed.

Inverted indexes are not the only type index that might be applicable to feature term retrieval. The signature and bitmap techniques (section 2.4.2) could both prove to be suitable. Like text retrieval (except boolean), image feature term retrieval does not have to have all terms present to match documents, making the probabilistic approach of signature files attractive. Bitmap files would be less likely to be worthwhile due to their large size - which would be compounded by the large vocabularies sizes of some feature terms.

Other techniques that have been developed for text retrieval are Latent Semantic Indexing [38] which has already been investigated in the image domain in [109]. Relevance feedback is another area which it would provide worthwhile investigation, although there is a large amount of research which has already been done in this area.

7.4 Summary

This thesis has presented a novel global colour algorithm, QMNS, which is indexed using techniques adopted from the text retrieval community. It is shown that the QMNS algorithm has features which are more specific than those of an RGB histogram, and a CCV, allowing retrieval to be performed with a relatively low number of features. The term distributions of the three algorithms is shown to be Zipfian in nature which provides a good model and allows the index to be optimised. Analysis of the distributions has shown it is possible to optimise the index by pruning of feature terms, making the index more compact and concise.

Appendix A

MNS Configuration Parameters

Technique	Signature Generation											Signature Comparison		
Name	MNS Feature	Neighbourhood Definition						Mean Shift				Technique	Measure	Thresh
		Location			Pixels			Neighbourhood		Signature				
		X	Y	Aliased	Reduction	Factor	% Mode	Size	Thresh	Size	Thresh			
General Database Groups														
RGB MNS Group														
Base MNS	RGB2	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Fixed MNS	RGB2	8	8	FALSE	None	100.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Big MNS	RGB2	16	16	TRUE	None	100.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Bigger MNS	RGB2	32	32	TRUE	None	100.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Rapid MNS 0.5	RGB2	8	8	TRUE	Pseudo	50.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Rapid MNS 0.3	RGB2	8	8	TRUE	Pseudo	30.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Rapid MNS 0.1	RGB2	8	8	TRUE	Pseudo	10.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Rapid MNS Random 0.5	RGB2	8	8	TRUE	Random	50.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Rapid Big MNS 0.3	RGB2	32	32	TRUE	Random	30.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Enhanced MNS	RGB2	8	8	TRUE	None	100.00%	12.50%	11	0.5	22	1	Seq Stable Marriage	Euclidean	20
Reduced MNS	RGB2	8	8	TRUE	None	100.00%	12.50%	44	0.5	88	1	Seq Stable Marriage	Euclidean	20
Large Mode MNS	RGB2	8	8	TRUE	None	100.00%	15.00%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Small Mode MNS	RGB2	8	8	TRUE	None	100.00%	10.00%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
High Thresh MNS	RGB2	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	30
Low Thresh MNS	RGB2	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	10
City MNS	RGB2	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	City	20
Chrominance MNS Group														
Base CMNS	Chrominance	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
Rapid Big CMNS 0.3	Chrominance	32	32	TRUE	Random	30.00%	12.50%	22	0.5	44	1	Seq Stable Marriage	Euclidean	20
QMNS Group														
QMNS CPR1	RGB2 - 4x4x1	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Inverted Index	n/a	n/a
QMNS CPR2	RGB2 - 4x4x4	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Inverted Index	n/a	n/a
QMNS CPR3	RGB2 - 4x4x4	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Inverted Index	n/a	n/a
QMNS CPR4	RGB2 - 4x4x4	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Inverted Index	n/a	n/a
3 Bin QMNS CPR4	RGB2 - 3x3x3	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Inverted Index	n/a	n/a
5 Bin QMNS CPR4	RGB2 - 5x5x5	8	8	TRUE	None	100.00%	12.50%	22	0.5	44	1	Inverted Index	n/a	n/a
Rapid QMNS 0.3 CPR4	RGB2 - 4x4x4	8	8	TRUE	None	30.00%	12.50%	22	0.5	44	1	Inverted Index	n/a	n/a
Enhanced QMNS CPR4	RGB2 - 4x4x4	8	8	TRUE	None	30.00%	12.50%	11	0.5	22	1	Inverted Index	n/a	n/a

TABLE A.1: MNS Configuration Parameters

Variation Name	Description
Base MNS	Baseline parameters, defined by Koubaroulis in [73].
Fixed MNS	As for Base MNS, but each neighbourhood is fixed in position and not randomly shifted.
Big MNS	As for Base MNS, but the neighbourhood size is 16x16 pixels.
Bigger MNS	As for Base MNS, but the neighbourhood size is 32x32 pixels.
Rapid MNS 0.5	As for Base MNS, but a pseudo random grid is used to select only 0.5 of the pixels in the neighbourhood.
Rapid MNS 0.3	As for Base MNS, but a pseudo random grid is used to select only 0.3 of the pixels in the neighbourhood.
Rapid MNS 0.1	As for Base MNS, but a pseudo random grid is used to select only 0.1 of the pixels in the neighbourhood.
Rapid MNS Random 0.3	As for Base MNS, but a only random 0.3 of the pixels are selected in the neighbourhood.
Enhnaced MNS	As for Base MNS, but the radius of the shift window in the mean shift operation at both neighbourhood and signature clustering stages is halved.
Reduced MNS	As for Base MNS, but the radius of the shift window in the mean shift operation at both neighbourhood and signature clustering stages is doubled.
Large Mode MNS	As for Base MNS, but a larger proportion of the pixels in a neighbourhood must be homogenous for modes to to qualify.
Small Mode MNS	As for Base MNS, but a smaller proportion of the pixels in a neighbourhood must be homogenous for modes to to qualify.
High Thresh MNS	Signature generated as for Base MNS, but the threshold distance between two RGB^2 features is increased.
Low Thresh MNS	Signature generated as for Base MNS, but the threshold distance between two RGB^2 features is reduced.
City MNS	Signature generated as for Base MNS, but the distance measure between two RGB^2 features is the L1, or City Block distance.
Base CMNS	As for Base MNS, but the feature used is not RGB^2 but a 5 dimensional chrominance feature.
Rapid Big CMNS 0.3	As for Rapid Big MNS 0.3, but the feature used is not RGB^2 but a 5 dimensional chrominance feature.
QMNS CPR1	As for Base MNS, but the signature is quantised and translated into feature terms. The CPR1 retrieval algorithm is used to calculate image similarity.
QMNS CPR2	As for Base MNS, but the signature is quantised and translated into feature terms. The CPR2 retrieval algorithm is used to calculate image similarity.
QMNS CPR3	As for Base MNS, but the signature is quantised and translated into feature terms. The CPR3 retrieval algorithm is used to calculate image similarity.
QMNS CPR4	As for Base MNS, but the signature is quantised and translated into feature terms. The CPR4 retrieval algorithm is used to calculate image similarity.
3Bin QMNS CPR4	As for QMNS CPR4, but the signature is quantised using 3 partitions per axis (equivalent to $3^6 = 729$ possible terms). The CPR4 retrieval algorithm is used to calculate image similarity.
5Bin QMNS CPR4	As for QMNS CPR4, but the signature is quantised using 3 partitions per axis (equivalent to $5^6 = 15625$ possible terms). The CPR4 retrieval algorithm is used to calculate image similarity.
Rapid QMNS 0.3 CPR4	As for Rapid MNS 0.3, but the signature is quantised and translated into feature terms. The CPR4 retrieval algorithm is used to calculate image similarity.
Enhanced QMNS CPR4	As for Enhanced MNS, but the signature is quantised and translated into feature terms. The CPR4 retrieval algorithm is used to calculate image similarity.

TABLE A.2: MNS Variations Descriptions

Appendix B

Example Query

B.1 Query Image and Image Category

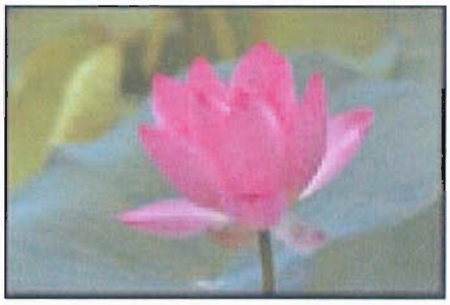

FIGURE B.1: Example Query Image

This query image is taken from the *Pink Flowers* category. Shown below are all the other images in this category:

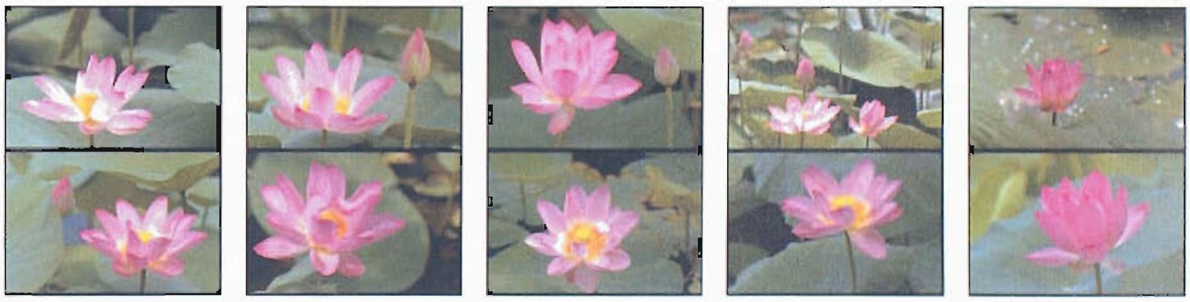

FIGURE B.2: Images in the Pink Flowers category

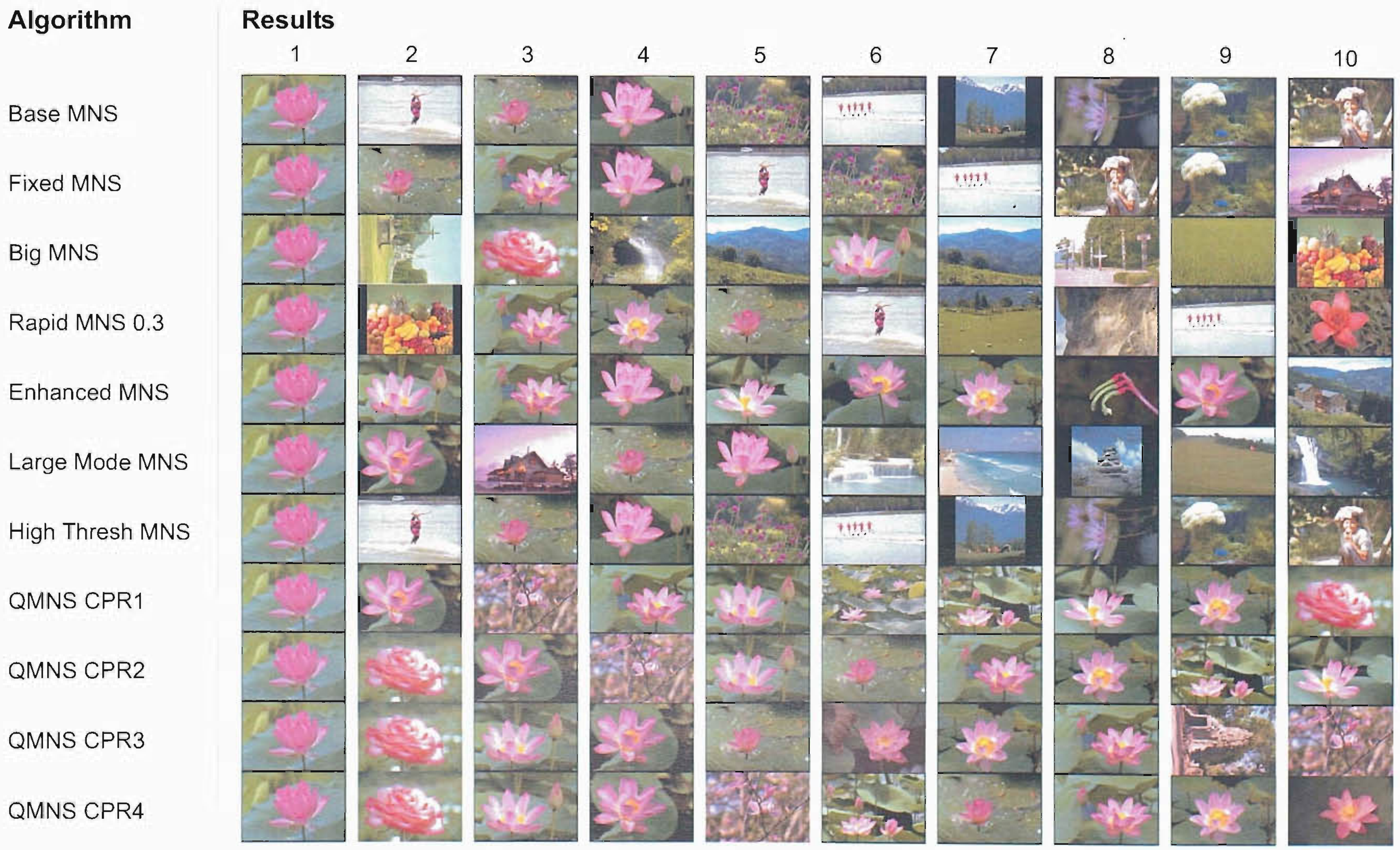

FIGURE B.3: Query Results

Appendix C

Test Results Tables

C.1 Test Group 1 - MNS Variations

Algorithm	Indexing Time (s)								
	Half Scale			Normal Scale			Double Scale		
	min	avg	max	min	avg	max	min	avg	max
Base MNS	1.388	3.092	4.375	5.944	12.395	20.949	19.344	51.025	98.246
Fixed MNS	1.354	3.138	4.561	5.724	12.263	19.130	18.544	50.660	102.884
Big MNS	6.700	15.095	23.673	23.136	55.251	93.896	61.549	146.919	294.051
Bigger MNS	26.420	77.865	156.787	104.155	276.060	521.919	268.298	700.809	1972.139
Rapid MNS (0.5)	0.421	0.795	1.125	1.648	3.488	6.512	5.560	17.845	58.529
Rapid MNS (0.3)	0.132	0.348	0.632	0.704	1.808	4.545	2.394	13.375	55.931
Rapid MNS (0.1)	0.350	0.774	1.225	1.612	3.299	5.603	5.662	16.268	52.661
Rapid MNS (Random 0.5)	1.448	3.159	4.473	5.746	12.303	19.767	19.551	50.793	106.782
Enhanced MNS	1.442	2.773	3.488	5.742	11.091	15.327	19.699	48.820	85.277
Reduced MNS	1.501	2.967	4.649	5.856	11.230	19.798	19.771	42.682	91.930
Large Mode MNS	1.402	3.165	4.509	5.748	12.564	20.124	19.650	52.355	111.912
Small Mode MNS	1.504	3.152	4.491	5.732	12.464	20.546	20.074	49.798	99.033
Base Chrominance MNS	1.481	3.143	4.484	5.752	12.625	19.141	19.689	48.403	83.561
RGB Histogram	0.016	0.019	0.040	0.064	0.072	0.132	0.172	0.311	0.596
CCV	0.219	0.818	1.081	0.629	0.828	0.987	0.448	0.829	1.499
MCCV	0.113	0.265	0.557	0.634	0.986	1.195	2.992	5.085	10.173

TABLE C.1: MNS Variations - Minimum, Average and Maximum Indexing Times

Algorithm	Features								
	Half			Normal			Double		
	min	avg	max	min	avg	max	min	avg	max
Base MNS	1	8.437	34	1	10.808	57	1	10.663	63
Fixed MNS	1	8.363	37	1	10.718	54	1	10.584	55
Big MNS	1	6.316	20	1	9.286	41	1	11.408	65
Bigger MNS	0	3.792	11	1	6.540	22	1	9.514	46
Rapid MNS (0.5)	1	8.137	32	1	10.574	52	1	10.155	54
Rapid MNS (0.3)	1	9.155	33	1	11.646	55	1	10.957	57
Rapid MNS (0.1)	1	6.049	17	1	8.283	34	1	6.618	32
Rapid MNS (Random 0.5)	1	8.481	33	1	10.743	56	1	10.644	55
Enhanced MNS	3	26.814	72	3	45.628	179	5	69.523	333
Reduced MNS	1	2.192	8	1	2.239	11	1	1.817	9
Large Mode MNS	1	9.027	33	1	11.403	64	1	11.752	68
Small Mode MNS	1	7.395	25	1	9.849	49	1	9.299	48
Base Chrominance MNS	2	14.140	46	2	22.917	82	2	26.613	108

TABLE C.2: MNS Variations - Feature Statistics

C.2 Test Group 2 - Full Image Tests

Algorithm	Test Category and Image Size								
	PinkFlowers			BeachScenes			BuildingsAtNight		
	Half	Normal	Double	Half	Normal	Double	Half	Normal	Double
Base MNS	0.194	0.291	0.342	0.204	0.137	0.129	0.349	0.466	0.392
Fixed MNS	0.275	0.354	0.318	0.167	0.159	0.13	0.305	0.423	0.367
Big MNS	0.1	0.167	0.187	0.086	0.175	0.211	0.158	0.219	0.318
Bigger MNS	0.088	0.132	0.118	0.069	0.091	0.084	0.116	0.208	0.252
Rapid MNS (0.5)	0.2	0.382	0.318	0.131	0.15	0.124	0.349	0.417	0.42
Rapid MNS (0.3)	0.262	0.276	0.295	0.167	0.184	0.122	0.425	0.478	0.441
Rapid MNS (0.1)	0.284	0.223	0.257	0.127	0.148	0.132	0.428	0.504	0.393
Rapid MNS (Random 0.5)	0.198	0.24	0.32	0.149	0.177	0.128	0.294	0.384	0.387
Enhanced MNS	0.462	0.666	0.639	0.202	0.243	0.307	0.183	0.457	0.535
Reduced MNS	0.096	0.077	0.107	0.03	0.028	0.043	0.104	0.184	0.127
Large Mode MNS	0.234	0.26	0.325	0.102	0.202	0.162	0.254	0.381	0.373
Small Mode MNS	0.272	0.274	0.291	0.19	0.137	0.121	0.35	0.383	0.341
High Thresh MNS	0.327	0.291	0.523	0.178	0.137	0.108	0.421	0.466	0.551
Low Thesh MNS	0.053	0.291	0.199	0.107	0.137	0.091	0.146	0.466	0.174
City MNS	0.035	0.109	0.074	0.024	0.025	0.009	0.266	0.323	0.278
Base Chrominance MNS	0.308	0.394	0.418	0.043	0.061	0.07	0.067	0.111	0.122

Algorithm	Test Category and Image Size									
	SeaAndSky			SeaWorld			Test Average			
	Half	Normal	Double	Half	Normal	Double	Half	Normal	Double	Overall
Base MNS	0.088	0.103	0.105	0.224	0.227	0.192	0.212	0.245	0.232	**0.229667**
Fixed MNS	0.103	0.118	0.103	0.161	0.179	0.25	0.202	0.247	0.234	**0.227667**
Big MNS	0.099	0.148	0.162	0.146	0.202	0.345	0.118	0.182	0.245	**0.181667**
Bigger MNS	0.061	0.12	0.127	0.163	0.067	0.265	0.099	0.124	0.169	**0.130667**
Rapid MNS (0.5)	0.084	0.103	0.108	0.148	0.129	0.087	0.182	0.236	0.211	**0.209667**
Rapid MNS (0.3)	0.099	0.137	0.089	0.233	0.303	0.209	0.237	0.276	0.231	**0.248**
Rapid MNS (0.1)	0.102	0.107	0.105	0.158	0.132	0.099	0.22	0.223	0.197	**0.213333**
Rapid MNS (Random 0.5)	0.096	0.116	0.094	0.089	0.149	0.124	0.165	0.213	0.2112	**0.1964**
Enhanced MNS	0.075	0.123	0.117	0.491	0.619	0.688	0.283	0.421	0.457	**0.387**
Reduced MNS	0.155	0.203	0.135	0.115	0.182	0.125	0.1	0.135	0.107	**0.114**
Large Mode MNS	0.113	0.143	0.134	0.132	0.382	0.236	0.167	0.274	0.246	**0.229**
Small Mode MNS	0.096	0.113	0.11	0.159	0.086	0.119	0.214	0.199	0.196	**0.203**
High Thresh MNS	0.12	0.103	0.129	0.232	0.227	0.337	0.256	0.245	0.33	**0.277**
Low Thesh MNS	0.0744	0.103	0.08	0.097	0.227	0.145	0.096	0.245	0.138	**0.159667**
City MNS	0.015	0.007	0.009	0.045	0.05	0.064	0.077	0.102	0.087	**0.088667**
Base Chrominance MNS	0.12	0.138	0.152	0.35	0.443	0.429	0.177	0.23	0.238	**0.215**

TABLE C.3: MNS Variations - Average Precision for Categories

Algorithm	Retrieval Time (s)								
	Half Scale			Normal Scale			Double Scale		
	min	avg	max	min	avg	max	min	avg	max
Base MNS	9.516	11.828	14.086	0.222	0.302	0.365	9.558	11.848	14.122
Fixed MNS	9.479	11.772	14.124	0.211	0.280	0.331	9.526	11.808	14.168
Big MNS	40.607	51.219	62.690	0.198	0.234	0.275	40.681	51.285	62.780
Bigger MNS	198.599	253.926	331.614	0.178	0.193	0.229	198.641	253.968	331.669
Rapid MNS (0.5)	2.777	3.527	4.374	0.210	0.270	0.330	2.822	3.557	4.408
Rapid MNS (0.3)	1.487	1.946	2.773	0.230	0.353	0.543	1.536	1.978	2.808
Rapid MNS (0.1)	2.588	3.273	3.870	0.179	0.204	0.230	2.593	3.279	3.875
Rapid MNS (Random 0.5)	9.496	11.777	14.077	0.215	0.293	0.369	9.548	11.814	14.125
Enhanced MNS	10.073	12.276	14.064	1.459	2.891	3.957	13.180	15.255	18.288
Reduced MNS	9.180	10.762	13.007	0.132	0.135	0.138	9.178	10.760	13.006
Large Mode MNS	9.554	11.955	14.530	0.227	0.309	0.375	9.618	12.001	14.591
Small Mode MNS	9.392	11.694	13.970	0.201	0.249	0.285	9.431	11.720	14.003
Base Chrominance MNS	9.727	11.909	13.899	0.596	0.734	0.877	10.198	12.285	14.177
High Thresh MNS	9.483	11.766	14.067	0.214	0.285	0.351	9.536	11.804	14.116
Low Thesh MNS	9.498	11.783	14.084	0.214	0.285	0.351	9.552	11.821	14.131
City MNS	9.479	11.759	14.056	0.197	0.268	0.333	9.529	11.798	14.103

TABLE C.4: MNS Variations - Minimum, Average and Maximum Retrieval Times

Database	Indexing Time (s)									Feature Statistics (Normal)					
	Half Scale			Normal Scale			Double Scale			Features			Unique Terms		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
QMNS	1.39	3.07	4.35	5.96	13.16	20.86	18.90	43.47	64.01	34	320.2	681	5	38.4	115
Enhanced QMNS	1.40	2.70	3.34	6.19	11.97	18.55	18.79	41.94	54.09	16	373.9	641	5	42.7	105
3 Bin QMNS	1.47	3.17	4.92	5.70	12.37	19.10	19.68	44.99	66.80	36	319.8	662	3	24.0	65
5 Bin QMNS	1.45	3.18	4.50	5.85	14.12	33.40	18.87	44.12	64.72	36	320.2	681	4	53.6	178
Rapid QMNS	0.17	0.29	0.36	0.66	1.11	1.40	2.36	4.20	5.30	27	377.5	692	6	41.9	123
Base MNS	1.38	3.09	4.37	5.94	12.39	20.94	19.34	51.02	98.24						
RGB Histogram	0.01	0.01	0.04	0.06	0.07	0.13	0.17	0.31	0.59						
CCV	0.21	0.81	1.08	0.62	0.82	0.98	0.44	0.82	1.49						
MCCV	0.11	0.26	0.55	0.63	0.98	1.19	2.99	5.08	10.17						

TABLE C.5: QMNS Variations - Minimum, Average and Maximum Indexing Times

Algorithm	Database	Test Category and Image Size								
		PinkFlowers			BeachScenes			BuildingsAtNight		
		Half	Normal	Double	Half	Normal	Double	Half	Normal	Double
CPR1	QMNS	0.457	0.407	0.333	0.092	0.137	0.149	0.178	0.289	0.438
CPR2	QMNS	0.529	0.488	0.409	0.096	0.196	0.207	0.239	0.307	0.437
CPR3	QMNS	0.288	0.345	0.156	0.131	0.208	0.211	0.283	0.345	0.366
CPR4	QMNS	0.482	0.543	0.500	0.119	0.189	0.166	0.304	0.393	0.555
CPR4	Enhanced QMNS	0.492	0.637	0.253	0.136	0.150	0.111	0.154	0.283	0.430
CPR4	3 Bin QMNS	0.416	0.632	0.513	0.074	0.073	0.079	0.307	0.390	0.462
CPR4	5 Bin QMNS	0.437	0.526	0.467	0.209	0.224	0.208	0.369	0.427	0.568
CPR4	Rapid QMNS	0.510	0.585	0.477	0.150	0.181	0.117	0.384	0.439	0.600
Base MNS	Base MNS	0.194	0.291	0.342	0.204	0.137	0.129	0.349	0.466	0.392
RGB Histogram	RGB Histogram	0.393	0.390	0.394	0.154	0.138	0.143	0.632	0.633	0.632
CCV	CCV	0.415	0.421	0.430	0.164	0.158	0.149	0.643	0.646	0.649
MCCV	MCCV		0.274	0.088		0.159	0.166		0.459	0.536

Algorithm	Database	Test Category and Image Size									
		SeaAndSky			SeaWorld			Test Average			
		Half	Normal	Double	Half	Normal	Double	Half	Normal	Double	Overall
CPR1	QMNS	0.144	0.107	0.096	0.558	0.660	0.723	0.286	0.320	0.348	0.318
CPR2	QMNS	0.184	0.137	0.127	0.377	0.590	0.667	0.285	0.344	0.369	0.333
CPR3	QMNS	0.178	0.168	0.154	0.513	0.582	0.617	0.279	0.330	0.301	0.303
CPR4	QMNS	0.231	0.176	0.142	0.461	0.682	0.696	0.319	0.397	0.412	0.376
CPR4	Enhanced QMNS	0.227	0.228	0.152	0.512	0.693	0.757	0.304	0.398	0.341	0.348
CPR4	3 Bin QMNS	0.231	0.137	0.114	0.695	0.611	0.580	0.344	0.369	0.350	0.354
CPR4	5 Bin QMNS	0.207	0.172	0.149	0.625	0.689	0.760	0.369	0.408	0.431	0.403
CPR4	Rapid QMNS	0.276	0.213	0.134	0.645	0.647	0.585	0.393	0.413	0.382	0.396
Base MNS	Base MNS	0.088	0.103	0.105	0.224	0.227	0.192	0.212	0.245	0.232	0.230
RGB Histogram	RGB Histogram	0.322	0.331	0.332	0.322	0.344	0.344	0.365	0.367	0.369	0.367
CCV	CCV	0.309	0.309	0.310	0.520	0.504	0.496	0.410	0.408	0.407	0.408
MCCV	MCCV		0.370	0.351		0.238	0.414		0.300	0.311	0.306

TABLE C.6: QMNS Variations - Average Precision for Categories

Retrieval Algorithm	Database	Retrieval Time (s)								
		Half Scale			Normal Scale			Double Scale		
		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
CPR1	QMNS	9.299	11.463	13.340	9.389	11.557	13.420	9.466	11.649	13.539
CPR2	QMNS	9.327	11.479	13.326	9.380	11.549	13.420	9.466	11.650	13.544
CPR3	QMNS	9.250	11.417	13.271	9.356	11.508	13.355	9.435	11.575	13.423
CPR4	QMNS	9.251	11.418	13.272	9.356	11.509	13.357	9.437	11.575	13.425
CPR4	Enhanced QMNS	8.432	10.589	11.521	8.508	10.647	11.572	8.558	10.624	11.641
CPR4	3 Bin QMNS	9.244	11.387	13.255	9.304	11.452	13.334	9.351	11.490	13.380
CPR4	5 Bin QMNS	9.180	11.392	13.256	9.353	11.541	13.356	9.369	11.533	13.394
CPR4	Rapid QMNS	1.124	1.370	1.557	1.270	1.480	1.702	1.299	1.522	1.764
Base MNS	Base MNS	9.516	14.086	11.828	0.222	0.365	0.302	9.558	14.122	11.848
RGB Histogram	RGB Histogram	0.195	0.197	0.199	0.120	0.122	0.129	0.196	0.198	0.201
CCV	CCV	0.463	0.467	0.475	0.107	0.110	0.118	0.462	0.463	0.466
MCCV	MCCV				0.449	0.465	0.493	11.214	12.526	14.111

TABLE C.7: QMNS Variations - Minimum, Average and Maximum Retrieval Times

Appendix D

Categorisation of Features

The QMNS algorithm, as a general purpose global colour feature, produces three modes of RGB colour feature, corresponding to patches in the image that contain one, two or three dominant colours. These result in feature tuples that contain three, six and nine floating point values. Both QMNS and the Candida cell feature have variable sized signatures that are composed of multiple features, whereas other features define only one fixed length feature. The RGB histogram defines a vector in which the value of each component is the number of pixels that occur in one particular region of RGB space. Typically these regions are created by evenly dividing the space into regular cubes - as in figure D. This process is known by a number of different names - quantisation, discretisation and binning.

The CCV histogram contains two vectors, which count the *coherent* and the *incoherentpixels*, by quantising both RGB feature spaces according to the same parameters. All three algorithms can be classed as global colour features. They are global in the sense that the resultant signature contains no reference to the location within the image of the features. Whilst QMNS divides the image into patches whilst processing and so may seem to perform localisation functions this is analagous to the histogram operation of a histogram as it scans across the image. CCV also segements the image, but again the spatial information recorded is lost when the two parts of the image are histogrammed.

The degree to which a feature is narrow, or broad, is important when generating a suitable signature for storage, and when retrieving documents from a collection, however all features carry a certain amount of information and have a certain size. This needs to be considered when determining the size of the index for a collection and the speed in which documents can be indexed and retrieved.

FIGURE D.1: Simple quantisation of RGB space

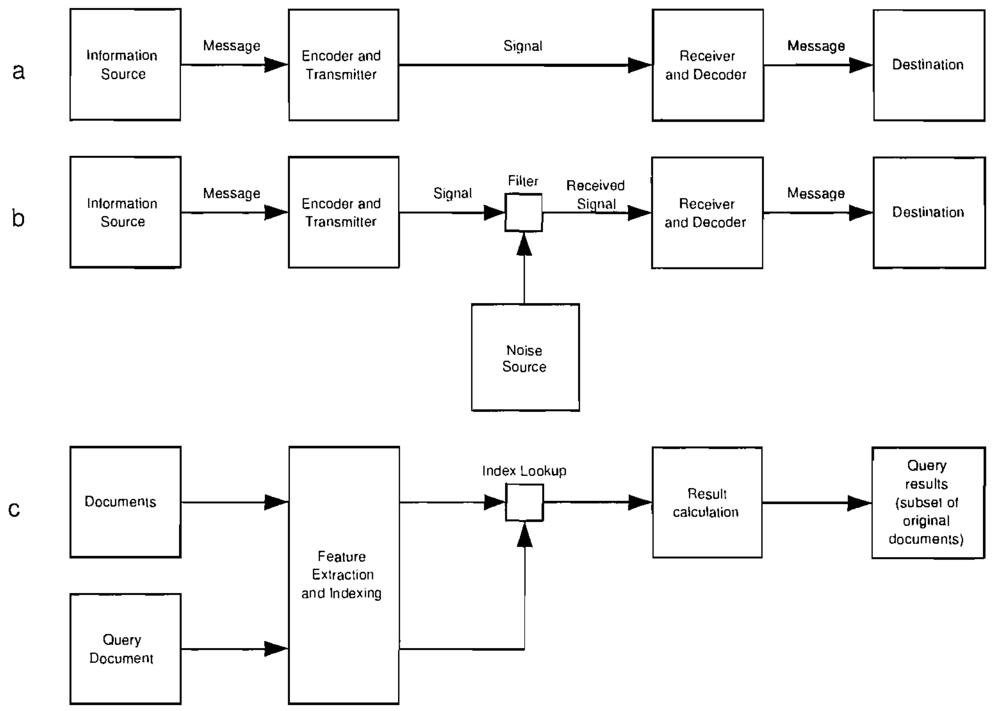

FIGURE D.2: Information Coding: a. A discrete, noiseless, channel. b. A discrete channel with noise. c. A document retrieval system as a filtered information channel. *a and b adapted from [123]*

D.1 The Information Transformation of Features

We have determined that of all the different types of features (be they extracted from images or other media) some provide very specific information that can be used only in a certain context, or domain, and others provide very general information that can be used in very broad domains. This leads us to examine the information transformations that occur from document to feature, from an information theoretic point of view. Chapter 3.1 introduced the concepts of multimedia information systems as a system for storing and retrieving documents according to a particular information requirement. The architecture of a typical MIS was presented in section 4.2, figure 4.2, showing the flow of information through the system. This high level semantic description of information flow can be reduced into a very simple description by using information theory.

Shannon's seminal work on the mathematical theory of communication, [123], defines the simplest form of discrete communication as a system where information is encoded, transmitted and decoded. Figure D.1.a and b shows this diagrammatically. In the channel, over which the information is transmitted, an additional factor is taken into account - noise. Shannon's work was written in the 1940s, when digital electronics were just starting to appear, and transmission over noisy communication media was an important factor of their work. The noise corrupts the information so that when recoded at the end of the channel it is incorrect in some way. The noise effectively filters the information randomly [1] transforming the information.

When we wish to find information in the form of a document from a collection of documents we are doing two things (phrase this better). Firstly we are looking at each document to determine its information content, and then we are evaluating, according to some query criterion, whether that document satisfies our information needs. We are encoding each document in some manner[2] - perhaps simply reading the title, or another proxysuch as the abstract, to determine the significant concepts contained within, and then filtering it according to its relation with the query concepts. Figre D.1.c enhances Shannon's notion of information travelling across a physical medium to that of information retrieval. The information source of the original schematic is now a corpus of documents, which are encoded (and stored) by feature extraction and indexing algorithms. A query document is also encoded in the same manner, and acts as a filter on the indexed documents, selecting those that are desirable. This may seem to oppose the notion of noise since that is considered to be *a bad thing*, yet in reality noise is merely * the entropy of the channel, and other external systems, transforming the signal, and can be managed and dealt with effectively. The filtered, encoded, document information is then decoded to form complete documents again. Whilst this may be as simple as using each document identifier to retrieve the document from a database, conceptually there is a form of decoding occuring. The resultant information then is the set of documents that, hopefully, will be semantically coherent with the query.

The questions that Shannon was asking about communication are also perfectly applicable in this context (this excerpt taken from Weaver's section in [123]:

- How does one measure an *amount of information*?
- How does one measure the *capacity* of a communication channel?
- The action of the transmitter in changing the message into the signal often involves a *coding process*. What are the characteristics of an efficient coding process. And when the coding is as efficient as possible, at what rate can the channel convey information?

D.1.1 The Size of a Document

In discrete information theory the base unit is the bit, able only to indicate the presense or absense of one symbol. In communications one symbol is useless, however when

[1]This is not entirely true, since much of the noise is in a pseudorandom form, created by the entropy of the channel and from other transmissions that are picked up in the electrical (or whichever medium) characteristics. Much of this noise was shown to be in a fractal form by Mandelbrot and later (?) Cantor.

[2]There is also an inherent form of decoding - that of transforming the symbols that lie on the page into words and in turn into concepts within our brain

bits are combined together to form a sequence that represents other symbols they have meaning. The internal organisation and operation of all modern computer and communication systems can be reduced down to bits, which allows us to calculate the amount of information stored in documents.

In order to determine the size of a particular document in a particular medium we must not look at the size of the file on the computer - this is already an encoded form of the information, possibly already compressed, possibly containing much redundant information. Instead we must consider the different combinations of symbols available in the document. A text document contains many symbols - the alphanumerics, punctuation, parentheses, currency symbols and many more. The alphabet used creates a particular problem since we must consider whether or not we are counting upper and lower case as different symbols. This will depend on whether or not the the recipient of the document at the destination needs to have the symbols or not. Old teletype and telegraph systems encoded one alphabet and used a shift symbol to indicate that the following character would be in upper case. In the ASCII symbol set both upper and lower case characters are included, as well as various punctuation, international and mathematical symbols. ASCII also includes 32 symbols to control the channel and the cursor, although many of these are now redundant.

A typical document retrieval system will not have to ever try to decode the encoded information held in the index because it will always have direct access to the original document, and so we only need to encode. This has a distinct advantage in that we can afford to lose more information than would be possible across a traditional channel. Section 2.4.2 describes in detail how the indexing process from text document to complete index works, and so the final representation of individual words stored in the index is the only important factor. It should be noted that this discussion considers only the simplest text index - at the expense of those that index all combinations of symbols.

The raw size of a document in bits will be equivalent to the number of symbols in the document multiplied by the number of bits required to encode those symbols.

$$d_{bits} = d_{symb}\dot{l}og_2 10\dot{A} \tag{D.1}$$

An ASCII text document with 6000 symbols will therefore have a raw size of $6000\dot{l}og_2 10\dot{1}28 = 768,000 bits$.

Whilst images may not seem to be composed of symbols there is a discrete set of the different possible colours that any pixel may assume, and it is these that form the information stream*. An image of dimensions 1024 by 768 pixels, in 24bit RGB (8 bits for each colour channel) would have a raw size of $1024\dot{7}68\dot{2}4 = 18,874,368 bits$.

D.1.2 The Encoding of Documents

The process of extracting features from a document, whether they be text based, or of some other form, is a process of encoding. It is a form of lossy encoding designed not to allow transmission of the document over a channel, but to provide a succint version of the information in the document for retrieval purposes.

Like calculating the amount of information in an a document, measuring the amount of information in a signature is a matter of looking at the different possible combinations of data that can be stored. In a vector based signature there must be limits to the range of values that each component may assume, otherwise the number of choices is effectively unlimited. If the signature contains integer values then its size in bits is equal to the product of the difference of each component's range:

$$s_{bits} = \prod_{i=1..i} componentupper_i - componentlower_i \tag{D.2}$$

If the signature contains continuous values, or near continuous values such as floating point representations, the size can not be calculated in terms of *bits* without first some form of discretisation. It might seem tempting to simply use the bitwise size of a *float* value, which is typically Xbits, however a float is already in an encoded format and wouldn't represent the raw data properly. By quantising each component at the highest resolution that will ever be required then the amount of information in the signature can be calculated. Alternatively this value could be calculated using Shannon's continuous measurements of information[123], which is beyond the scope of this thesis. These two expressions allow us to calculate the amount by which a document is compressed by the indexing process. Equation D.3 shows this ratio in the case where multiple features (denoted by the subscript i) are used:

$$d_{comp}\% = \frac{\sum_{i=1..i} s_{bits i}}{d_{bits}} \tag{D.3}$$

D.1.3 The Capacity of a Document Retrieval System

Communication theory defines capacity as *the amount of information transmitted per second, in bits per second, along a communication channel.* Capacity is important since it allows us to measure the number of symbols (not just bits) that can be transmitted per second. Taken from Weaver's contribution to [123], the following paragraph describes a fundamental theorem for a noiseless channel transmitting discrete symbols:

> This theorem relates to a communication channel which has a capacity of C bits per second, accepting signals from a source of entropy (or information)

> of H bits per second. The theorem states that by devising a proper coding procedures for the transmitter it is possible to transmit symbols over the channel at an average rate [3] which is nearly C/H, but which, no matter how clever the coding, can never be made to exceed C/H.

The capacity of a document retrieval system is therefore equivalent to the average retrieval time for a query, which concerns the number of documents that can be retrieved from the index per second. The relevancy of retrieved documents is not an issue here since we are not measuring the indexes ability to discriminate between documents based on the query.

[3]Footnote from [123]: We remember that the capacity C involves the idea of information transmitted per second, and is thus measured in bits per second. The entropy H here measures information per symbol, so that the ratio of C to H measures symbols per second.

Appendix E

QMNS, CCV and RGB Algorithm Data

	Divisions	Vocabulary	Used Vocabulary	Vocabulary Usage	Total Postings	Average Terms/Image	Index Ratio	Average Docs per Term	Average Index Time	Curve Slope	Average Precision
Histogram	3	27	25	0.926	10435	13.6	54.28%	416.64	0.113	0.6298	0.216
	4	64	56	0.875	17062	22.2	39.63%	303.5	0.126	0.8080	0.237
	5	125	105	0.840	25300	32.9	31.33%	240.19	0.119	0.8907	0.317
	10	1000	689	0.689	93486	121.6	17.64%	208.52	0.245	0.9847	0.441
	15	3375	2116	0.627	214469	278.9	13.18%	135.46	0.440	0.9944	0.407
	30	27000	14739	0.546	918331	1194.2	8.10%	62.59	2.091	0.9985	0.503
	60	216000	102567	0.475	3694780	4804.7	4.68%	36.02	10.259	0.9994	0.538
	100	1000000	407407	0.407	8889984	11560.44	2.84%	21.82	26.380	1.0000	0.543
	127	2048383	754151	0.368	12818258	16683.4	2.21%	16.99	47.978	1.0000	0.543
	200	8000000	1166126	0.146	16790377	21845.01	1.87%	14.398	42.780	1.0000	0.539
	255	16581375	1317793	0.079	18033673	23450.8	1.78%	13.6848		1.0000	
CCV	1	16	16	1.000	7431	9.66	60.40%	463.31	1.42	0.7408	0.137
	2	128	100	0.781	21066	27.39	27.39%	210.37	0.82	0.9051	0.331
	3	1024	569	0.556	65487	85.16	14.97%	114.92	1.37	0.9839	0.460
	4	8192	3151	0.385	248145	322.74	10.24%	78.68	1.58	0.9954	0.485
	5	65536	18932	0.289	1053326	1369.73	7.24%	55.61	4.42	0.9969	0.496
	6	524288	120182	0.229	3303475	5406.67	4.50%	27.83	25.18	0.9980	0.584
	7	4194304	768653	0.183	12980550	16879.78	2.20%	16.89	72.52	0.9978	0.573
	8	33554432	1365400	0.041	18069469	23497.36	1.72%	13.25	182.48	0.9976	0.574
QMNS	5	1.97E+06	3173	1.61E-03	65881	87.39	2.75%	19.469		0.9952	0.471
	10	1.00E+09	19762	1.97E-05	149916	194.95	0.99%	7.586		0.9991	0.513
	15	3.85E+10	44439	1.16E-06	219043	284.84	0.64%	4.929		0.9973	0.513
	20	5.12E+11	72771	1.42E-07	279745	363.78	0.50%	3.844		0.9966	0.521
	25	3.81E+12	102454	2.69E-08	329348	428.28	0.42%	3.214		0.9972	0.518
	30	1.97E+13	130410	6.63E-09	372561	484.47	0.37%	2.857		0.9977	0.513
	35	7.88E+13	156893	1.99E-09	411580	535.21	0.34%	2.623		0.9982	0.522
	40	2.62E+14	182865	6.98E-10	448042	582.63	0.32%	2.45		0.9989	0.517
	45	7.57E+14	205302	2.71E-10	481342	625.93	0.30%	2.345		0.9994	0.491
	50	1.95E+15	226090	1.16E-10	512653	666.65	0.29%	2.268		0.9999	0.498
	55	4.61E+15	245529	5.33E-11	542492	705.45	0.29%	2.209		1.0000	0.489
	60	1.01E+16	263031	2.61E-11	570531	741.91	0.28%	2.169		1.0000	0.496
	65	2.07E+16	278696	1.35E-11	596991	776.32	0.28%	2.142		1.0000	0.470
	70	4.04E+16	293432	7.27E-12	622592	809.61	0.28%	2.122		1.0000	0.446
	75	7.51E+16	307399	4.09E-12	646882	841.20	0.27%	2.104		1.0000	0.435
	80	1.34E+17	320614	2.39E-12	669412	870.50	0.27%	2.088		1.0000	0.424
	85	2.32E+17	333407	1.44E-12	689965	897.22	0.27%	2.069		1.0000	0.407
	90	3.87E+17	346128	8.93E-13	710932	922.49	0.27%	2.054		1.0000	0.376
	95	6.30E+17	358412	5.69E-13	730133	949.46	0.26%	2.0371		1.0000	0.365
	100	1.00E+18	371008	3.71E-13	748441	973.27	0.26%	2.0173		1.0000	0.359
	255	4.56E+21	716195	1.57E-16	998235	1301.48	0.18%	1.3938		1.0000	0.216

TABLE E.1: Complete Feature Vocabulary Data - RGB Histogram, CCV Histogram and QMNS

Glossary

A

ACI	**acronym** Autonomous Citation Indexing. The extraction of citation (bibliographic) information from academic documents.
agent	An instance of the agent computing software paradigm. An agent is typified by its ability to communicate and exist in heterogenous environments, its social behaviour and its artificial intelligence, with which it seeks to achieve certain preordained goals.
Artiste	**project** A european project to develop a digital library for archiving, analysis and retrieval of paintings.
ASCII	**acronym** American Standard Code for Information Interchange. A 7 bit code.

B

bin	A single, contiguous, volume in feature space partitioned by some manner of quantisation.
bio-metric	Some element of people which can be used to uniquely identify them: Fingerprints, irises, gait.
bit	1. *Information Theory* A **bi**_nary digi_**t** can indicate the presence or absence of just one symbol. 2. *Electronics* An electric signal that is typically either a high voltage - 1, or a low voltage - 0.
bitvector	.
blog	**abbr.** Weblog. An online, public, diary onto which is published the inconsequencies of the lives of some individuals.

C

capacity (bps) The amount of information transmitted per second by a communications channel.

CBR **acronym** Content Based Retrieval. The process of retrieving documents based on the informational content that they contain.

centroid The centre of a cluster of points in n-space, often defined as the point obtained by calculating the average value for each component.

cue Of the senses: a sparticular sign. Often accompanied by the sense - e.g. visual cues.

D

Dewey decimal A system invented by X Dewey in the latter 19th century for manual cataloging of books.s.

document A ordered, coherent, collection of information with a common concept, most often considered as being in writing, but may be in any form of media, and is most often a combination of media.

document frequency The number of documents in a corpus in which a particular term occurs.

E

exhaustivity The degree to which a term, document, or index is able to include a wide range of topics.

F

feature 1. A high level element of an image, as in *global colour feature.*
2. The data produced by a feature extraction algorithm, as in *feature indexing.*

finite state machine A mathematical model, where different states of execution can be entered and left according to predfined rules.

G

generic link A type of hypermedia link which provides as targets a list of documents returned by a CBR search engine using the link anchor as a query.

granularity A measure of the amount of detail stored in an index. A coarse index will merge terms and synonyms together using morphology, helping to reduce storage. A fine index will store all terms.

H

hitlist The ranked results of a query, normally ordered by similarity to the query item. A hitlist will typically include a numerical similarity score.

homogenous In computer vision, a property of a collection of .

L

linkbase A database containing links from document to document, which avoids direct references in the documents themselves, helping to avoid broken links,.

M

MIS **Acronym** Multimedia Information System. A complete system for multimedia document management and retrieval. Will typically contain integrated CBR components.

morphology . See 2.4.1.3.

N

negative dictionary A list of noise, or stop-words. Also known as a stop-list.

P

PACS **Acronym** Picture Archiving and Communication System. MIS used in the medical industry.

pixels **acronym** A **Pict**ure **el**ement. The smallest individual component of an image.

power of discrimination How well a term, or query, is able to discriminate between documents of different topics.

proxy A succint description of a document that provides a good description of the entire content. The terms in a text document proxy are typically weighted more than others in the document.

Q

query In information retrieval, a question provided to a search engine that should return a hitlist of similar documents.

query by humming A audio query paradigm that allows a user to hum the pitch changes of a desired tune. Analogous to query by sketch in audio.

query by sketch A image query paradigm that allows a user to sketch the outline of an object which is required from the database. Analogous to query by humming in audio.

R

registration The process of fitting a model to an acquired image source, used, for example, when overlaying a 3D model of a scene of a video of parts of the scene.

RTF **acronym** **R**ich **T**ext **F**ormat. A simple text markup language that allows limited control over fonts and layout.

S

Sculpteur **acronym** **S**emantic and **c**ontent-based m**ul**timedia ex**p**loi**t**ation for **Eur**opean benefit. A three year European project to research and develop a system for multimedia information organisation, storage, and retrieval of items and artifacts held in museums. www.sculpteurweb.org.

segment The process of grouping homogenous regions in an image that are deemed to represent part, or the whole, of an object in the depicted scene.

snake A visual segmentation device that uses a combination of heterogeneity constrains on pixels and energy functions to iteratively adapt itself to an objects outline.

specificity The degree to which a term, document or index is able to focus? on a particular topic.

stemming A particular application of morphology where syntactic pre- and post-fixes are removed to reveal the underlying root meaning of a word.

stop-list See stop-words.

stop-words Terms in natural language that are functional words (i.e. A, THE, IT, OF ETC) with a correspondingly high frequency count, and as such have a low power of discrimination. Also *stop-list.*

T

template matching A simple, but useful, technique for identifying objects in images by moving a template of pixels over an image. At each location in the image the number of pixels matching the template are counted, and where the count exceeds a threshold the object is deemed to exist.

TFxIDF **Term Frequency times Inverse Document Frequency** A document vector similarity heuristic.

V

vocabulary usage The ratio of distinct terms present in a corpus to the total possible vocabulary. In natural language the total vocabulary is taken from a fixed source, eg a dictionary contemporary to the corpus.

Z

Zipfian A rank-frequency distribution that when plotted on a log-log scale has a log-linear slope that is near to -1.0.

Mathematical Definitions

d_j	Document j from the set of all documents, D in a corpus.
df_i	The document frequency - the number of documents in which term t_i occurs at least once.
qr_i	The relative term frequency given to term t_i in the query.
qw_i	The weight given to term t_i in the query.
r_{ij}	The relative term frequency given to term t_i in document d_j.
t_i	Term i from the set of all terms, T that are present in a corpus, or index.
tf_{ij}	The frequency of t_i in document d_j.
tn_j	The number of terms in document j.

V	The set of all possible terms.
w_{ij}	The weight given to term t_i in document d_j.

Bibliography

[1] The equator project. http://www.equator.ac.uk/.

[2] M. Addis, M. Boniface, S. Goodall, P. Grimwood, S. Kim, P. Lewis, K. Martinez, and A. Stevenson. Sculpteur: Towards a new paradigm for multimedia museum information handling. 2nd international semantic web conference. In *Proceedings of the 2nd International Semantic Web Conference*, pages 582–596, 2003.

[3] P. Allen, M. Boniface, P. Lewis, and K. Martinez. Interoperability between multimedia collections for content and metadata-based searching. In *Proceedings of the World Wide Web conference. W3C. WWW2002*, 2002.

[4] Paul Allen. Artiste image processing API (IPA). Technical report, University of Southampton, 2000.

[5] Shih-Fu Chang amd William Chen, Horace J.Meng, Hari Sundaram, and Di Zhong. Videoq: an automated content based video search system using visual cues. In *ACM Multimedia 97*, pages 313–324, 1997.

[6] Unknown author. Bringing user satisfaction to media access networks. http://busman.elec.qmul.ac.uk/.

[7] Unknown author. Trial final report - spt expedited arrivals trial. Technical report, EyeTicket Corporation, 2002.

[8] Unknown author. Nokia lifeblog, 2004. http://www.nokia.com/nokia/0,1522,,00.html?orig=/lifeblog.

[9] Luiz Andre Barroso, Jeffrey Dean, and Urs Holzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, pages 22–28, 2003.

[10] John Barton and Tim Kindberg. The cooltown user experience. Technical report, 2001.

[11] R. Bayer and C. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, pages 173–189, 1972.

[12] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. R* tree: An efficient and robust access method for points and rectangles. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 322–331, 1990.

[13] R.K. Belew. *Finding Out About*. Cambridge Univ. Press, 2000.

[14] C. Beleznai, B. Frstck, H. Bischof, and W.G. Kropatsch. Detecting humans in groups using a fast mean shift procedure. In *Proceedings of the 28th AAPR Workshop*, pages 71–78, 2004.

[15] Ana B. Benitez, Mandis Beigi, and Shih-Fu Chang. A content-based image meta-search engine using relevance feedback. *IEEE Internet Computing*, pages 59–69, 1998.

[16] Jon Louis Bentley and Jerome H. Friedman. Fast algorithms for constructing minimal spanning trees in coordinate spaces. *IEEE Transactions on Computers*, pages 97–105, 1978.

[17] Stefan Berchtold, Bernhard Ertl, Daniel A. Keim, Hans-Peter Kriegel, and Thomas Seidl. Fast nearest neighbor search in high-dimensional space. In *Proc. 14th IEEE Conf. Data Engineering, ICDE*, pages 23–27, 1998.

[18] Steven Blackburn and David DeRoure. A tool for content-based navigation of music. In *Proceedings of ACM International Multimedia Conference*, pages 361–368, 1998.

[19] M. Bowmans, K. Hohne, U. Tiede, and M. Riemer. Segmentation of MR images of the head for 3-D display. In *IEEE Transactions on Medical Imaging*, pages 177–183, 1990.

[20] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks an ISDN Systems*, pages 107–117, 1998.

[21] Chris Buckley and A.F. Lewit. Optimization of inverted vector searches. In *Proceedings of SIGIR-85*, pages 97–110, 1985.

[22] Kaushik Chakrabarti and Sharad Mehrotra. Dynamic granular locking approach to phantom protection in r-trees. In *Proceedings of the International Conference on Data Engineering*, pages 446–454, 1998.

[23] Kaushik Chakrabarti and Sharad Mehrotra. The hybrid tree: An index structure for high dimensional feature spaces. In *Proceedings of the International Conference on Data Engineering*, pages 440–447, 1999.

[24] Kaushik Chakrabarti and Sharad Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. *The International Journal on Very Large Databases*, 2000.

[25] Stephen Chan, Kirk Martinez, Paul Lewis, C. Lahanier, and J. Stevenson. Handling sub-image queries in content-based retrieval of high resolution art images. In *Proceedings of the International Cultural Heritage Informatics Meeting 2*, pages 157–163, 2001.

[26] Shih-Fu Chang, John R.Smith, Mandis Beigi, and Ana Benitez. Visual information retrieval from large distributed on-line repositories. *Communications of the ACM*, pages 63–71, 1997.

[27] Ye-Sho Chen and Ferdinand F Leimkuhler. Analysis of zipf's law: An index approach. *Information Processing and Management*, pages 171–182, 1987.

[28] Yizonq Cheng. Mean shift, mode seekinq, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[29] Paolo Ciacca, Pavel Zezula, and Fausto Rabitti. A data structure for similarity search in multimedia databases. In *9th ERCIM Database Research Group Workshop on Multimedia Database Systems*, 1996.

[30] Tzi cker Chiueh. Content-based image indexing. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, pages 582–593, 1994.

[31] Charles L.A. Clarke and Gordon V. Cormack. Dynamic inverted indexes for a distributed full-text retrieval system. Technical report, Department of Computer Science, University of Waterloo, 1995.

[32] Scott D. Cohen. Measuring point set similarity with the hausdorff distance: Theory and applications. 1995.

[33] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of Conference on Computer Vision and Pattern Recognition 2000*, pages 142–151, 2000.

[34] Doug Cutting and Jan Pedersen. Optimizations for dynamic inverted index maintenance. In *Proceedings of the 13th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 405–411, 1990.

[35] S. Dasmahapatra, D. Dupplaw, B. Hu, H. Lewis, P. Lewis, and N. Shadbolt. Facilitating multi-disciplinary knowledge-based support for breast cancer screening. *Special issue of the International Journal of Healthcare Technology and Management (to appear)*, 2004.

[36] H. Davis, W. Hall, I. Heath, G. Hill, and R. Wilkins. Microcosm: An open hypermedia environment for information integration. Technical report, 1992.

[37] Hugh C. Davis, Wendy Hall, Ian Heath, Gary J. Hill, and Robert J. Wilkins. Towards an integrated information environment with open hypermedia systems.

In *In ECHT '92: Proceedings of the Fourth ACM Conference on Hypertext, Milan, Italy.*, pages 181–190, 1992.

[38] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, pages 391–407, 1990.

[39] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, pages 325–339, 1967.

[40] J. Derganc and F. Pernus. Color image segmentation by nonparametric analysis of feature space. In *Zbornik devete Elektrotehnike in racunalnike konference ERK 2000*, pages 197–200, 2000.

[41] Peter Meer Dorin Comaniciu. Mean shift analysis and applications. In *Proceedings of the International Conference on Computer Vision*, pages 1197–1203, 1999.

[42] Mark S. Drew, Jie Wei, and Ze-Nian Li. On illumination invariance in color object recognition. *Pattern Recognition*, pages 1077–1087, 1998.

[43] William Equitz. Retrieving imagees from a database using texture algorithms from the QBIC system. Technical report, IBM Research Division, 1993.

[44] M. Fauzi and P. Lewis. Query by fax for content based image retrieval. In *Challenge of Image and Video Retrieval*, 2002.

[45] C. Fellbaum. *WordNet An Electronic Lexical Database*. The MIT Press, 1998.

[46] M. Flickner, H. Sawhney, W. Niblack, et al. Query by image and video content: The QBIC system. *Computer*, pages 23–32, 1995.

[47] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 2001.

[48] Christopher Fox. A stop list for general text. *ACM SIGIR Forum*, pages 19–21, 1990.

[49] Ophir Frieder, David A. Grossman, Abdur Chowdhury, and Gideon Frieder. Efficiency considerations for scalable information retrieval servers. *Journal of Digital information*, 2000.

[50] Xiaodong Fu, Dingxing Wang, Weimin Zheng, and Meiming Sheng. Gpr-tree: A global parallel index for multiattribute declustering on cluster of workstations. In *Proceedings of the 1997 Advances in Parallel and Distributed Computing Conference (APDC '97)*, pages 300–306, 1997.

[51] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function. *IEEE Transactions in Information Theory*, pages 32–40, 1975.

[52] Brian V. Funt and Graham D. Finlayson. Color constant color indexing. *Pattern Analysis and Machine Intelligence*, pages 522–529, 1995.

[53] Volker Gaede and Oliver Gunther. Multidimensional access methods. *ACM Computing Surveys*, 1998.

[54] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, pages 9–14, 1962.

[55] Alexander Gelbukh and Grigori Sidorov. Zipf and heaps laws' coefficients depend on language. In *Proceedings of the Conference on Intelligent Text Processing and Computational Linguistics*, 2001.

[56] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: Musical information retrieval in an audio database. *ACM Multimedia*, pages 231–236, 1995.

[57] Giunti. Artisteweb, 2002. `http://www.artisteweb.org`.

[58] S. Goodall, P. Lewis, K. Martinez, P. Sinclair, F. Giorgini, M. Addis, M. Boniface, C. Lahanier, and J. Stevenson. Sculpteur: Multimedia retrieval for museums. In *Proceedings of the Third CIVR Conference*, pages 638–646, 2004.

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

[60] Ian Hacking. *Logic of Statistical Inference*. Cambridge University Press, 1965.

[61] J. Hey and A. Wissenburg. Modelling the hybrid library: Project malibu. *The New Review of Information and Library Research*, 1999.

[62] K. Hinrichs. Implementation of the grid file:design concepts and experiences. In *BIT 25*, pages 569–592, 1985.

[63] B. Hu, S. Dasmahapatra, P. Lewis, and N. Shadbolt. Ontology-based medical image annotation with description logics. In *Proceedings of The 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 77–82, 2003.

[64] HK Huang and RK Taira. Infrastructure design of a picture archiving and communication system. *American Journal of Roentgenology*, pages 743–749, 1992.

[65] Tom Huang, Sharad Mehrotra, and Kannan Ramchandran. Multimedia analysis and retrieval system (mars) project. In *Proceedings of the 33rd Annual Clinic on Library Application of Data Processing - Digital Image Access and Retrieval*, 1996.

[66] Andreas Hutflesz, Hans-Werner Six, and Peter Widmayer. Twin grid files: space optimizing access schemes. In *International Conference on Management of Data and Symposium on Principles of Database Systems*, pages 183–190, 1988.

[67] D. Huttenlocher, D. Klanderman, and A. Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 850–863, 1993.

[68] E. Izquierdo, A. Pearmain, P. Villegas, and S. Herrmann L.-Q. Xu. Busman: A system for digital video content management and delivery. In *Proceedings of the International Broadcasting Convention*, 2004.

[69] Ibrahim Kamel and Christos Faloutsos. Hilbert r-tree: an improved r-tree using fractals. In *Proc. of VLDB Conference*, pages 500–509, 1994.

[70] M. Kirby, F. Weisser, and G. Dangelmayr. A model problem in the representation of digital image sequences. *Pattern Recognition*, 1993.

[71] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 226–239, 1998.

[72] Teuvo Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences, 2nd Edition, 1997.

[73] D. Koubaroulis, J. Matas, and J. Kittler. Illumination invariant object recognition using the mns method. In *Proceedings of the 10th European Signal Processing Conference*, pages 2173–2176, 2000.

[74] D. Koubaroulis, J. Matas, and J. Kittler. The multimodal signature method: An efficiency and sensitivity study. In *Proceedings of the 15th ICPR*, pages 379–382, 2000.

[75] Computational Vision Lab. Data for computer vision and computational colour science, 2000. `http://www.cs.sfu.ca/~colour/data/`.

[76] Ting-Sheng Lai, John Tait, and Sharon McDonald. Image browsing and navigation usinq hierarchical classification. In *The Challenge of Image Retrieval - Second UK Conference on Image Retrieval*, 1999.

[77] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, pages 67–71, 1999.

[78] Joon Ho Lee, Won Yong Kin, Myoung Ho Kim, and Yoon Joon Lee. On the evaluation of boolean operators in the extended boolean retrieval framework. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–297, 1993.

[79] P. Lewis, H. Davis, S. Griffiths, W. Hall, and R. Wilkins. Media-based navigation with generic links. In *The Seventh ACM conference on Hypertext*, pages 215–223, 1996.

[80] Paul H. Lewis, Hugh C. Davis, Mark R. Dobie, and Wendy Hall. Towards multimedia thesaurus support for media-based navigation. *Image Databases and Multimedia Search*, pages 111–118, 1996.

[81] Shih-Hao Li and Peter B. Danzig. Precision and recall of ranking information-filtering systems. *Journal of Intelligent Information Systems*, pages 287–306, 1996.

[82] H.P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1957.

[83] Peter Lyman and Hal R. Varian. How much information. Technical report, School of Information Management and Systems, University of California at Berkeley, 2003.

[84] J MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistical Problems*, pages 281–297, 1967.

[85] Kavi Mahesh. Text retrieval quality: A primer. Technical report, Oracle Corporation, 2001.

[86] B. Mamalis, P.Spirakis, and B. Tampakas. High performance parallel text retrieval over large scale document collections: The pfire system. *International Journal on Computers and their Applications (IJCA)*, 1999.

[87] B. S. Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley, 2002.

[88] David F. Manlove. Stable marriage with ties and unacceptable partners. Technical report, Computing Science Department of Glasgow University, 1999.

[89] Patrick Martin, Ian A. Macleod, and Brent Nordin. A design of a distributed full text retrieval system. In *Proceedings of the 1986-ACM Conference on Research and Development in Information Retrieval*, pages 131–137, 1986.

[90] J. Matas, D. Koubaroulis, and J. Kittler. Colour-based image retrieval from video sequences. In *Proceedings of the CIR2000, Third UK Conference on Image Retrieval*, pages 1–12, 2000.

[91] J. Matas, D. Koubaroulis, and J. Kittler. Performance evaluation of the multimodal neighbourhood signature method for colour object recognition. In *Proceedings of the Czech Pattern Recognition Workshop*, pages 27–34, 2000.

[92] Sergey Melnik, Sriram Raghaven, Beverly Yang, and Hector Garcia-Molina. Building a distributed full-text index for the web. In *WWW10*, pages 396–406, 2001.

[93] Makoto Miyahara and Yasuhiro Yoshida. Mathematical transform of (r,g,b) color data to munsell (h,v,c) color data. *Visual Communication and Image Processing*, pages 650–657, 1988.

[94] Alistair Moffat and Justin Zobel. Compression and fast indexing for multi-gigabyte text databases. *Australian Computer Journal*, pages 1–9, 1994.

[95] A. Mojsilovic, J. Kovacevic, D. Kall, R. Safranek, and K. Ganapathy. The vocabulary and grammar of color patterns. Technical report, IBM Research, 2000.

[96] H. Muller, W. Muller, and D. M. Squire. Performance evaluation in content-based image retrieval: Overview and proposals.

[97] Henning Müller. *User interaction and evaluation in content-based visual information retrieval*. PhD thesis, Computer Vision and Multimedia Laboratory, University of Geneva, 2002.

[98] Henning Müller, David McG. Squire, Wolfgang Müller, and Thierry Pun. Efficient access methods for content-based image retrieval with inverted files. In *Proceedings of Multimedia Storage and Archiving Systems IV (VV02)*., 1999.

[99] Wolfgang Müller. *Design and implementation of a flexible Content-Based Image Retrieval Framework - The GNU Image Finding Tool*. PhD thesis, Computer Vision and Multimedia Laboratory, University of Geneva, 2001.

[100] Wolfgang Müller, Henning Müller, Stéphane Marchand-Maillet, Thierry Pun, David McG. Squire, Zoran Pecenovic, Christof Giess, and Arjen P. de Vries. Mrml: A communication protocol for content-based multimedia retrieval. Technical report, Computer Vision Group, Computing Centre, University of Geneva, 2000.

[101] W. Niblack, R. Barber, W. Equitz, et al. The QBIC project: Querying images by content using color, texture, and shape. *Computer Science*, 1993.

[102] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: An adaptable symmetric multikey file structure. *ACM Transactions on Database Systems, : ACM CR 8411-0931*, pages 38–71, 1984.

[103] Michael Ortega, Kaushik Chakrabarti, Kriengkrai Porkaew, and Sharad Mehrota. Similarity search using multiple examples in MARS. In *International Conf. on Visual Information Systems (Visual'99)*, pages 68–75, 1999.

[104] Michael Ortega, Yong Rui, Kaushik Chakrabarti, Kriengkrai Porkaew, Thomas S. Huang, and Sharad Mehrota. Supporting ranked boolean similarity queries in mars. *IEEE Transaction on Knowledge and Data Engineering,*, pages 905–925, 1998.

[105] Michael Ortega-Binderberger, Sharad Mehrotra, Kaushik Chakrabarti, and Kriengkrai Porkaew. WebMARS: A multimedia search engine for full document retrieval and cross media browsing. In *Multimedia Information Systems Workshop 2000 (MIS2000)*, 2000.

[106] S. Paek, C. Sable, V. Hatzivassiloglou, A. Jaimes, B. Schiffman, S. Chang, and K. McKeown. Integration of visual and text based approaches for the content labeling and classification of photographs. In *ACM SIGIR'99 Workshop on Multimedia Indexing and Retrieval*, 1999.

[107] Greg Pass, Ramin Zabih, and Justin Miller. Comparing images using color coherence vectors. In *ACM Multimedia*, pages 65–73, 1996.

[108] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[109] Z. Pecenovic, M. Do, S. Ayer, and M. Vetterli. New methods for image retrieval. In *Proceedings of the International Congress on Imaging Science*, pages 242–246, 1998.

[110] Zoran Pecenovic, Serge Ayer, and Martin Vetterli. Image retrieval using latent semantic indexing. 1997.

[111] Steve Putz. Using a relational database for an inverted text index. Technical report, Xerox Palo Alto Research Center, 1991.

[112] B. Ribeiro-Neto, E.S.Moira, M.S.Neubert, and N.Ziviani. Efficient distributed algorithms to build inverted files. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 105–112, 1999.

[113] Frank Rosenblatt. The perceptron: a perceiving and recognizing automaton. 1957.

[114] Y. Rubner, C. Tomassi, and L. Guibas. A metric for distributions with applications to image databases. In *In Proc. of the IEEE Int. Conf. on Comp. Vision, Bombay, India*, pages 59–66, 1998.

[115] Yossi Rubner, Jan Puzicha, Carlo Tomasi, and Joachim M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Computer Vision and Image Understanding*, pages 25–43, 2001.

[116] Y. Rui, T. Huang, and S. Mehrotra. Browsing and retrieving video content in a unified framework. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 237–240, 1998.

[117] Y. Rui, A. She, and T. Huang. Automated region segmentation using attraction based grouping in spatial-colortexture space. In *Proceedings of the International Conference on Image Processing*, 1996.

[118] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Constructing table-of-content for videos. *Multimedia Systems*, pages 369–368, 1999.

[119] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[120] H.S. Sawhney, S. Ayer, and M. Gorkani. Model-based 2d and 3d dominant motion estimation for mosaicking and video representation. In *Proceedings of the 5th International Conference on Computer Vision*, pages 583–590, 1995.

[121] Robert Schalkoff. *Pattern Recognition*. Wiley, 1992.

[122] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[123] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. The University of Illinois Press: Urbana, 1949.

[124] Philppe Smets. *What is Dempster-Shafer's Model*. Wiley, 1994.

[125] Arnold W.M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval - at the end of the early years. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, pages 1349–1380, 2000.

[126] John R. Smith. Quantitative assessment of image retrieval evaluation. *Journal of the American Society for Information Science*, pages 969–979, 2001.

[127] John R. Smith and Shih-Fu Chang. Tools and techniques for color image retrieval. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 426–437, 1996.

[128] D. Squire, W. Muller, H. Muller, and J. Raki. Content-based query of image databases, inspirations from text retrieval:inverted files, frequency-based weights and relevance feedback. In *Proceedings of The 10th Scandinavian Conference on Image Analysis (SCIA'99), (Kangerlussuaq, Greenland)*, 1999.

[129] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, pages 11–32, 1991.

[130] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. *IEEE Trans. Sys. Man, and Cybernetics*, pages 460–473, 1978.

[131] John Tait Ting-Sheng Lai. General photographic image retrieval simulating human visual perception. In *ACM SIGIR98 Post-Conference Workshop on Multimedia Indexing and Retrieval*, pages 17–28, 1998.

[132] Anthony Tomasic and Hector Garcia-Molina. Performance of inverted indices in shared-nothing distributed text document information retrieval systems. In *PDIS '93*, 1993.

[133] M. Turk and A. Pentland. Eigen faces for recognition. *Journal of Cognitive Neuroscience*, 1991.

[134] M. Uenohara and T. Kanade. Use of fourier and karhuene-loeve decomposition for fast pattern matching with a large set of templates. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, pages 891–898, 1997.

[135] Scott E. Umbaugh. *Computer Vision and Image Processing.* Prentice Hall 1998, 1998.

[136] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.

[137] W3C. W3c semantic web, 2004. `http://www.w3.org/2001/sw/`.

[138] James Z. Wang and Yanping Du. RF*IPF : A weighting scheme for multimedia information retrieval. In *Proc. IEEE International Conference on Image Analysis and Processing (ICIAP)*, pages 380–385, 2001.

[139] S. Watanabe. Karhunen-loeve expansion and factor analysis, theoretical remarks and applications. In *Transactions of the 4th Prauge Conference on Information Theory, Statistical Decision Functions and Random Processes*, pages 645–660, 1965.

[140] W. M. Wells III, W. E. L. Grimson, R. Kikinis, and F. A. Jolesz. Adaptive segmentation of MRI data. In *Int. Conf on Computer Vision, Virtual Reality and Robotics in Medicine*, pages 59–69, 1995.

[141] Mike Westmacott and Paul Lewis. An inverted index for image retrieval using colour pair feature terms. In *Proceedings of the Image and Video Communications and Processing Conference 2003*, pages 881–889, 2003.

[142] Mike Westmacott, Paul Lewis, and Kirk Martinez. Using colour pair patches for image retrieval. In *Proceedings of the First European Conference on Colour in Graphics, Imaging and Vision*, pages 245–248, 2002.

[143] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes - Compressing and Indexing Documents and Images.* Morgan Kaufmann, 1999.

[144] L.-Q. Xu, P. Villegas, M. Diez, E. Izquierdo, S. Herrmann, V. Bottreau, I. Damnjanovic, and D. Papworth. A user-centred system for end-to-end secure multimedia content delivery: from content annotation to consumer consumption. In *Proceedings of CIVR*, 2004.

[145] D. Young, C.A. Glasbey, A.J. Gray, and N.J. Martin. Towards automatic cell identification in DIC microscopy. *Journal of Microscopy*, pages 186–193, 1998.

[146] Y. Zheng, C. Tanner, D.L.G. Hill, D.J. Hawkes, M.J. White, M. Khazen, and M.O. Leach. Alignment of dynamic contrast enhanced MR volumes of the breast for a multicenter trial: an exemplar grid application. In *Medical Imaging 2004, San Diego, CA, USA*, 2004.

[147] Y. Zhuang, Y. Rui, T. Huang, and S. Mehrotra. Applying semantic association to support content-based video retrieval. In *Proceedings of the International workshop on Very Low Bitrate Video Coding*, 1998.

[148] George Kingsley Zipf. *Human Behaviour and The Principle of Least Effort*. Addison-Wesley Press, 1949.

[149] J. Zobel, A. Moffat, and K.R. Rao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, pages 453–490, 1998.