

G60/120

MONASH UNIVERSITY
THESIS ACCEPTED IN SATISFACTION OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

ON..... 11 November 2003

.....
Sec. Research Graduate School Committee

Under the Copyright Act 1968, this thesis must be used only under the normal conditions of scholarly fair dealing for the purposes of research, criticism or review. In particular no results or conclusions should be extracted from it, nor should it be copied or closely paraphrased in whole or in part without the written consent of the author. Proper written acknowledgement should be made for any assistance obtained from this thesis.

ERRATA

1. Pg 5, line 7: "This is when" should be read as **"This is true when"**.
 2. Pg 17, line 2: "where n is the revised number of bits to represent each colour channel" should be read as **"where n is the revised number of bits to represent each colour channel and || is the absolute operator"**.
 3. Pg 17, line 3: "quantised into 9 levels" should be read as **"quantised into 8 levels"**.
 4. Pg 18, line -7: "improve on" should be read as **"improve"**.
 5. Pg 19, line -4: "Bin by bin similarity histogram Z is computed" should be read as **"Bin by bin similarity histogram Z is computed using the Manhattan distance metric"**.
 6. Pg 35, line 10: "Sections 2.2, 2.3, 2.4 and 2.5" should be read as **"Sections 2.2-2.5"**.
 7. Pg 46, line 8: "of your choice" should be read as **"of one's choice"**.
 8. Pg 67, line -8: "So, since having a codebook that needs such a huge amount of storage space is not feasible, we propose to use a codebook size which is enough to capture the main features of the database images." should be read as **"A codebook that needs such a huge amount of storage space is not feasible. Thus, we propose to use a codebook size which is just enough to capture the main features of the database images."**
 9. Pg 82, lines -10 and -11: " ε " should be read as **" \in "**.
 10. Pg 142 line -9, insert the following after "for database CCD.":
"A reason for choosing the above codebook configuration is because it will result in similar retrieval efficiency to the autocorrelogram technique, which has the best retrieval effectiveness among the three existing techniques."
 11. Pg 161, line -5: "(9)," should be **"(9)"**.
-

**IMAGE INDEXING AND RETRIEVAL
BASED ON
VECTOR QUANTIZATION**

SHYH WEI TENG

**A DISSERTATION SUBMITTED IN FULFILMENT OF THE REQUIREMENT
FOR THE**

DEGREE OF DOCTOR OF PHILOSOPHY

**GIPPSLAND SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY
MONASH UNIVERSITY (GIPPSLAND CAMPUS), VICTORIA, AUSTRALIA**

JULY 2003

Supervisor : A. Prof Guojun LU

Associate Supervisor : Dr Kai Ming TING

ABSTRACT

As more and more images are being captured and stored in computers, an efficient and effective image retrieval system is required to make use of the information stored in images. In traditional databases, the approach to indexing and retrieval of images is based on simple attributes such as image number and text description. The performance of this text-based indexing and retrieval technique is low because the simple text attributes are not able to fully describe every detail of the images' appearance accurately. Besides, text-based retrieval systems also cannot accept content-based queries. To overcome these limitations of the text-based indexing and retrieval techniques, content-based indexing and retrieval (CBIR) techniques are pursued. In CBIR, low level information, like colour, shape and texture, in the images is used for indexing and retrieval. This thesis focuses on colour-based CBIR techniques.

Colour-based CBIR techniques are popular because colour information can be easily extracted from images. Besides that, compared with other low level information, it is easier for humans to identify and describe the colour information in images. Due to these reasons, colour-based techniques are widely implemented in CBIR systems. However, existing colour-based techniques have many limitations. Firstly, many colour-based techniques do not consider the spatial relationships among the colour pixels in the images. Secondly, for techniques that attempt to consider the spatial relationships among the colour pixels in the images, only very limited amount of such information are considered. In some of these techniques, the computation time required to build their features is very expensive due to the large dimension size of their features. Thus these techniques are not ideal to be implemented in many CBIR systems. In this thesis, we propose a technique that can overcome the limitations in the existing techniques.

The CBIR technique we propose uses a compression technique called vector quantization (VQ). VQ compresses an image by encoding each pixel block in the image with the index of the best match pixel block in a codebook. The pixel blocks (also called codewords) in the codebook are representative of the variety of pixel blocks which can be found in the database images. To build the image feature in our proposed technique, a histogram, which represents the frequency of each codeword being used to encode the image, is constructed. By using such a histogram for image retrieval, the spatial relationships among the colour pixels in the codewords used to encode the image are considered in our proposed technique. To ensure that our proposed technique can be performed efficiently, we have also studied several multidimensional search methods, which can be used to improve the efficiency of the image encoding process and the image retrieval process. From these studies, promising methods are identified. Based on the characteristics of these methods, new methods are proposed for the encoding process and the retrieval process. In this thesis, a tree-based search method is proposed to improve the image encoding efficiency. To improve the efficiency of the retrieval process, we have proposed two search methods that are based on the inverted file structure. For this thesis, we have implemented the VQ-based technique, the tree-based search method for image encoding and the two inverted file-based search methods for image retrieval. Our experimental results show that (i) the proposed VQ-based technique has better retrieval effectiveness compared with three commonly used colour-based techniques; (ii) the proposed tree-based search method has improved the efficiency of the image encoding process greatly; (iii) the two proposed inverted-file-based search methods have improved the efficiency of the retrieval process greatly.

DECLARATION

I, Shyhwei Teng, affirm that this thesis contains no material which has been accepted for award of any other degree or diploma in any university or other institution and affirm that to the best of the candidate's knowledge, the thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.


Signature of the Author

Date: 15/7/2003

ACKNOWLEDGMENTS

Writing the acknowledgements is one of the nicest moments I had throughout the entire journey of completing this thesis. This is because I get to show my appreciation towards all the people who have given me their support in one way or another during my PhD candidature.

Firstly, I would like to thank my supervisor, A. Prof. Guojun Lu for his constant guidance, support and suggestions throughout the whole course, in which I have gained valuable knowledge in the field of multimedia information indexing and retrieval.

I would also like to thank my associate supervisor, Dr Kai Ming Ting for reading through my thesis and making valuable suggestions of ways I can further improve the writing of the thesis.

I would also like to extend my gratitude to Gippsland School of Computing and Information Technology for providing such a fantastic research environment. The staffs here are excellent in their assistance in all administrative and research matters. Special thanks to other research students in my research groups for their friendship.

Next, are my family: dad, mum, sister, brother-in-law, brother and also my little niece. I give my deepest gratitude for their constant support and encouragement. Thanks for their unwavering confidence in me.

Finally, I would also like to thank my girlfriend, Wendy for all the care, support and encouragements in my many moments of self-doubt, agony and frustration. Thanks for being so patient with me and for brightening up my day whenever my morale is low.

This thesis will not become a reality without all of you. Thank you.

PUBLICATIONS

Teng, S. and G. Lu. *Codebook generation in vector quantization used for image retrieval*. in *International Symposium on Intelligent Multimedia and Distance Education*. Aug 2-7, 1999. Baden-Baden, Germany.

Teng, S. and G. Lu. *Performance study of image retrieval based on vector quantization*. in *ICCIMADE'01: International Conference on Intelligent Multimedia and Distance Education Conference*. June 1-3, 2001. Fargo, ND, USA. p. 76-80.

Teng, S. and G. Lu. *An evaluation of the robustness of image retrieval based on vector quantization*. in *IEEE Pacific-Rim Conference on Multimedia*. Oct 24-26, 2001. Beijing, China. p. 474-481.

Teng, S. and G. Lu. *Effects of Codebook Sizes, Codeword Dimensions, and Colour Spaces on Retrieval Performance of Image Retrieval using Vector Quantization*. in *The 3rd IEEE Pacific-Rim Conference on Multimedia*. Dec 16-18, 2002. Hsinchu Taiwan: Springer. p. 217-228

Lu, G. and S. Teng. *A Novel Image Retrieval Technique based on Vector Quantization*. in *Computational S. Intelligence for Modeling Control and Automation*. 1999. Viana, Austria. p. 160-165.

He, Y., G. Lu, and S. Teng. *An Investigate of using K-D Tree to Improve Image Retrieval Efficiency*. in *Digital Image Computing-Techniques and Applications*. Jan 21-22, 2002. Melbourne, Australia. p. 128-133.

TABLE OF CONTENTS

Title	i
Abstract	ii
Declaration	iv
Acknowledgments	v
Publications	vi
Table of Contents	vii
1. INTRODUCTION	1
1.1 Background	1
1.2 Content-based Image Retrieval	2
1.3 Scope of this Research	4
1.4 Introduction to our Approach	5
1.4.1 <i>Introduction to Vector Quantization</i>	5
1.4.2 <i>Introduction to VQ-Based Histogram Indexing and Retrieval Technique</i>	6
1.4.3 <i>Introduction to Improved Tree-Structured Image Encoding Method</i>	6
1.4.4 <i>Introduction to Image Retrieval using the Inverted File Structure</i>	7
1.5 Objectives of the Thesis	8
1.6 Summary of Contributions of our Research	9
1.7 Organization of the Thesis	10
2. CONTENT-BASED IMAGE INDEXING AND RETRIEVAL	13
2.1 Introduction	13
2.2 Image Retrieval based on Text	14

2.3	Image Retrieval based on Colour	16
2.3.1	<i>Basic Colour-based histogram Image Indexing and Retrieval Technique</i>	16
2.3.2	<i>QBIC Colour Histogram Technique</i>	19
2.3.3	<i>QBIC Colour Layout Technique</i>	20
2.3.4	<i>Image Retrieval based on Colour Coherence Vector</i>	20
2.3.5	<i>Image Retrieval based on Colour Correlogram</i>	22
2.3.6	<i>Image Retrieval based on Colour Moment</i>	24
2.3.7	<i>Effects of Colour Spaces on Retrieval</i>	25
2.4	Image Retrieval based on Shape	26
2.4.1	<i>Simple Shape Descriptors</i>	26
2.4.2	<i>Spectral Transform Descriptors</i>	27
2.4.3	<i>Invariant Moments</i>	28
2.4.4	<i>Grid Shape Descriptors</i>	30
2.4.5	<i>MPEG-7 Region-based Shape Descriptor</i>	30
2.4.6	<i>MPEG-7 Contour-Shape Descriptor</i>	31
2.5	Image Retrieval based on Texture	32
2.6	Integrated CBIR Techniques	34
2.7	Discussions on CBIR Techniques	35
2.8	Summary	38
3.	FUNDAMENTALS OF VECTOR QUANTIZATION FOR IMAGE COMPRESSION	40
3.1	Introduction	40
3.2	Basic Concepts of Vector Quantization	41
3.3	Concepts of Codebook Generation	45
3.3.1	<i>Linde-Buzo-Gray Algorithm</i>	45
3.3.2	<i>Example to Illustrate the LBG Algorithm</i>	47
3.3.3	<i>The Splitting Technique</i>	51

3.3.4	<i>Example to Illustrate the Full Codebook Generation Process</i>	53
3.4	Summary	59
4.	USING VECTOR QUANTIZATION FOR IMAGE INDEXING AND RETRIEVAL	60
4.1	Introduction	60
4.2	VQ -- A Good Candidate for Image Indexing and Retrieval	61
4.3	Image Indexing and Retrieval Technique based on VQ	62
4.4	Related Work	63
4.5	Codebook Generation in VQ used for Image Retrieval	64
4.5.1	<i>Choice of Training Images</i>	64
4.5.2	<i>Codebook Structure Suitable for Image Retrieval</i>	65
4.5.3	<i>Codebook Generation Algorithm Suitable for Image Retrieval</i>	68
4.5.4	<i>Effects of Codebook Sizes, Codeword Block-sizes and Colour Spaces on Retrieval Performance of the Proposed Technique</i>	70
4.6	Limitations Overcome by the Proposed VQ-based Indexing and Retrieval Technique	73
4.6.1	Basic and QBIC Colour-based Histogram Techniques	73
4.6.2	<i>QBIC Colour Layout Technique</i>	75
4.6.3	Colour Coherent Vector	76
4.6.4	<i>Colour Autocorrelogram</i>	76
4.7	Summary	77
5.	REVIEW OF EFFICIENT SEARCH METHODS FOR ENCODING AND RETRIEVAL PROCESSES	79
5.1	Introduction	79
5.2	Efficiency Issues of the Proposed Technique	80

5.3	Notations Definition	82
5.4	Partial Distance Search	83
5.5	Triangle Inequality based Search	84
5.6	Tree-Structured Search	86
5.7	K-D Tree Search	88
5.8	Methods Specific to the Euclidean Distance	90
5.9	Summary	94
6.	EXPERIMENTAL RESULTS TO SHOW THE EFFECTIVENESS OF VQ BASED INDEXING AND RETRIEVAL	97
6.1	Introduction	97
6.2	Recall and Precision Graphs	100
6.3	Experiments on Small Database	101
6.3.1	<i>Comparison of Retrieval Performance of VQ Scheme to QBIC Colour Histogram Technique using Small Database</i>	102
6.3.2	<i>Studies on Sensitivity to Translation</i>	103
6.3.2.1	<i>Potential Robustness Problem in Proposed Technique</i>	103
6.3.2.2	<i>Histogram Studies</i>	105
6.3.2.3	<i>Recall and Precision Studies</i>	108
6.3.2.4	<i>Summary of Experimental Results on the Proposed Technique's Sensitivity to Translation</i>	109
6.4	Experiments on Large Databases	110
6.4.1	<i>Database SCD</i>	110
6.4.1.1	<i>Experiment A on Database SCD: Effects of Codebook Sizes, Codeword Block-sizes and Colour Spaces on Retrieval Performance of the VQ Scheme</i>	113
6.4.1.2	<i>Results of Experiment A on Database SCD</i>	114
6.4.1.3	<i>Experiment B on SCD : Comparison of Retrieval Performance of VQ Scheme to Existing Techniques</i>	122

6.4.1.4	<i>Results of Experiment B on Database SCD</i>	122
6.4.2	<i>Database CCD</i>	126
6.4.2.1	<i>Experiment Descriptions for Database CCD</i>	126
6.4.2.2	<i>Results of Experiment A on Database CCD</i>	127
6.4.2.3	<i>Experiment B on CCD : Comparison of Retrieval Performance of YQ Scheme to Existing Techniques</i>	133
6.4.2.4	<i>Results of Experiment B on Database CCD</i>	134
6.5	Discussions	140
6.6	Conclusions	145
7.	IMPLEMENTATION & EXPERIMENTAL RESULTS OF EFFICIENT SEARCH METHOD FOR IMAGE ENCODING	146
7.1	Introduction	146
7.2	Space Partitioning by Codewords	148
7.3	Construction of the Tree Structure	153
7.4	Encoding Algorithm based on Tree	155
7.5	Accuracy of the Tree-structured Algorithm	156
7.6	Further Improvement to Encoding Algorithm	157
7.7	Effects of Colour Spaces on Encoding Efficiency	161
7.8	Experimental Results	163
7.8.1	<i>Algorithm Performance - Encoding Efficiency</i>	163
7.8.2	<i>Algorithm Performance - Encoding Accuracy</i>	169
7.8.3	<i>Algorithm Performance - Effects on Retrieval</i>	171
7.9	Conclusions	174
8.	IMPLEMENTATION & EXPERIMENTAL RESULTS OF EFFICIENT SEARCH METHODS FOR IMAGE RETRIEVAL	175
8.1	Introduction	175
8.2	Efficient Text Retrieval using Inverted File	176

8.3	Features in VQ-based Image Indices Suitable for the Implementation of the Inverted File Structure	180
8.4	Structure of the Inverted File in our Application	181
8.5	Methods to Reduce Size of the Search Space	183
8.6	Experimental Results	187
	8.6.1 Retrieval Performance of IF_M1	188
	8.6.2 Retrieval Performance of IF_M2	191
8.7	Conclusions	194
9.	CONCLUSIONS AND FUTURE DEVELOPMENTS	195
9.1	Conclusions	195
9.2	Future Developments	198
	BIBLIOGRAPHY	200
	APPENDIX	
A	Survey Forms for identifying ground truth images for each query image of Database SCD.	A
B	Average recall and precision graphs for Database SCD	B
<i>B1</i>	<i>Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD</i>	<i>B1-1</i>
<i>B2</i>	<i>Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD</i>	<i>B2-1</i>
<i>B3</i>	<i>Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD</i>	<i>B3-1</i>

C	Average recall and precision graphs for Database CCD	C
C1	<i>Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database CCD</i>	C1-1
C2	<i>Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database CCD</i>	C2-1
C3	<i>Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database CCD</i>	C3-1

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

For many years, images have been playing many important roles in our lives. While educators use images to illustrate their ideas, doctors, on the other hand, use radiological images to diagnose the medical conditions of their patients. Historians use images to provide a clear documentation of important events in history. Crime fighters also use images of crime scenes and criminals to assist them in solving crimes.

From the examples given above, we can see how diversely images are used in many areas. Very often in these areas, after the images are used, they are stored for documentation purposes. Another reason for storing the images is to ensure reusability of these images in the future. As new images are archived regularly, the number of archived images can easily accumulate to thousands and millions.

In recent years, the job of archiving the images is made easier with the rapid advancements of modern computer technology. With cheaper and larger amount of disk space, large number of images are digitised and stored. With workstations that possess greater processing power, these digital images can also be easily displayed to the users for browsing. However, to retrieve the relevant images by

browsing linearly through all the images is an extremely slow and tedious process. Thus, effective and efficient techniques to retrieve images from the database are required.

To allow effective and efficient searching and retrieval of relevant images from the image database, earlier systems used the traditional database management approach to index and retrieve images. The indexing and retrieval of images are based on simple attributes such as the image number and text description [14, 42]. Such systems have a number of limitations. Firstly, as these systems cannot accept image content-based queries, query types are limited. Secondly, the retrieval effectiveness is low because the simple attributes cannot fully describe every detail of the images' appearance accurately. To overcome these limitations, content-based image retrieval techniques have been actively pursued.

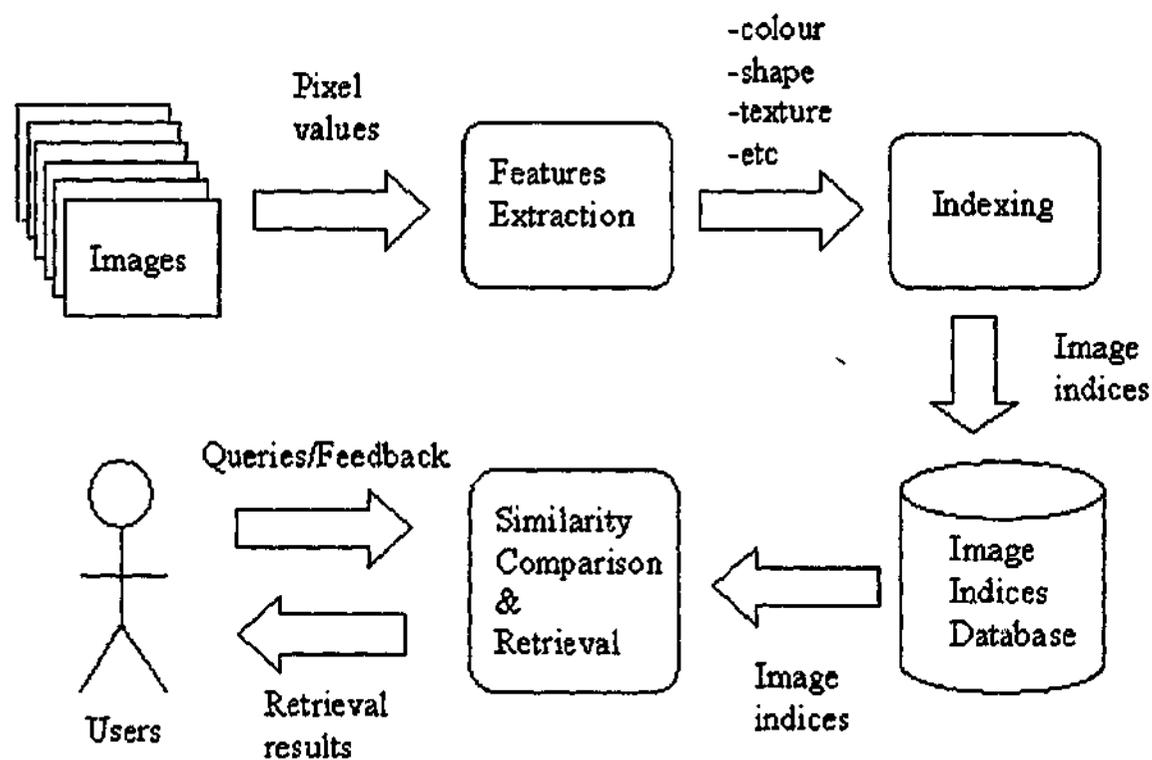


Figure 1.1 Components of a typical CBIR system

1.2 CONTENT-BASED IMAGE RETRIEVAL

Content-based image retrieval (CBIR) is a terminology used to describe the

process of utilising the low-level features in the query image to retrieve similar images from the database. Most of the current CBIR techniques make use of low-level features such as the colour histogram, object shape and texture [27, 43, 58, 64, 71] in the images. A CBIR system may employ one or several indexing and retrieval techniques. Regardless of the number of techniques employed, a typical CBIR system would normally consist of several main components illustrated in Figure 1.1. For a database in the CBIR system, each image will firstly go through a process where the features required are extracted. These features, which can be colour, shape, texture, etc, are normally extracted from the raw image pixels' values. Next, using the features extracted, a unique descriptor (also called image index) is built to index the image. The image index is normally a multi-dimensional vector and has a size that is normally much smaller than the image itself. The image indices are then stored in the image indices database. When a user wants to retrieve images from the database, he/she is required to submit a query image through the system's query interface. By randomly browsing through the images that are already in the database, an image can be selected as the query image. The user can also use a query image which is not part of the database. If the query image is part of the database, the image index is already available in the CBIR system. If not, the query image index must be built online. The query image index is compared with the image indices in the database using some similarity metrics and the retrieval results are then presented to the user. The retrieval results are normally displayed to the user as a ranked list of images starting from the most similar image to the least similar image. Some CBIR systems also allow the user to supply some feedback information so as to refine the query.

Examples of existing CBIR systems are QBIC (Query by Image Content) system that was developed by IBM researchers [1], VisualSEEK system that was developed in Columbia University [2, 80], Virage Image Search Engine [3],

Multimedia Analysis and Retrieval System (MARS) [61] and the ImageRover WWW Search Engine [90].

1.3 SCOPE OF THIS RESEARCH

To date, colour-based image retrieval techniques are the most popular and are commonly implemented in many content-based image retrieval systems [39]. Its popularity is mainly due to two reasons. Firstly, compared with shape and texture, it is normally much easier to remember and describe the colour elements in the images. Secondly, colour-based image retrieval techniques are relatively easier to implement and are more effective. This is because implementation of shape and texture based techniques often require images to be correctly segmented into individual objects. Unfortunately, segmentation itself is a difficult problem and there is no effective automatic segmentation method. This has resulted in the implementation of shape-based and texture-based techniques being relatively more difficult.

Although colour-based retrieval techniques have been implemented in many existing CBIR systems, many of these techniques only consider the colour distribution of the pixels, ignoring spatial relationships among the colour pixels. This leads to low retrieval performance since very different images can have similar colour histograms.

The research we have carried out in this thesis can be classified into three areas:

1. Investigate a way to incorporate information on the spatial relationship among the image pixels into a colour-based image indexing and retrieval technique. The proposed image indexing and retrieval technique should be more effective than the current techniques. The proposed CBIR technique is based on a compression technique called vector quantization (VQ).

2. For images to be retrieved from a database, the images must first be indexed. In CBIR systems, the indexing of the database images is usually carried out offline. Although the indexing of the images is carried out offline, an efficient way to perform the indexing process of our proposed technique is still desirable since there are normally thousands of images in a database. If image indexing is not efficient, it will take a long time to index these huge number of database images. At times, we may also need to index images online. This is when the user uses query images that are not from the database. In such situation, the indexing of the query images must be carried out efficiently before similar images can be retrieved from the database. As such, it is necessary to explore a way to ensure that the image indexing process of our proposed technique is carried out efficiently.
3. Besides being effective, a good retrieval technique must also be efficient. So, our research includes finding a way to ensure that the image retrieval process can be carried out efficiently.

1.4 INTRODUCTION TO OUR APPROACH

In this section, we will briefly introduce some techniques and show how they are used in our proposed image indexing and retrieval technique.

1.4.1 INTRODUCTION TO VECTOR QUANTIZATION

Vector quantization has been used successfully for image compression [26, 29, 41, 75]. In VQ, there is a codebook consisting of a number of codevectors (codewords), each of which is identified by a unique index number. During compression, an image is divided into blocks (vectors) of the same size of the codeword. For each image block, a best match codeword is determined and the

image block is represented by the unique number of the codeword. Since the number of bytes required to represent the unique number is normally much smaller than that of the image vector itself, significant data compression is achieved.

1.4.2 INTRODUCTION TO VQ-BASED HISTOGRAM INDEXING AND RETRIEVAL TECHNIQUE

We propose to carry out image indexing and retrieval directly from VQ compressed image data. Since the compressed data is just a sequence of unique numbers, we can build a histogram from these numbers to indicate how many image blocks are the same or similar to each codevector. This histogram then represents the content of the image and is used as the index of the image. The similarity between images can be calculated from the similarity between their histograms.

1.4.3 INTRODUCTION TO IMPROVED TREE-STRUCTURED IMAGE ENCODING METHOD

In our proposed CBIR technique, to index an image, the image must first be compressed using VQ. The image compression using VQ is also commonly known as image encoding. As described in Section 1.4.1, the VQ image encoding process involves searching for the best match codeword in the codebook for each image vector. This process of encoding an image, if done using a full-search method, is very computationally intensive and time consuming. To improve the efficiency of the image encoding process, we have organised the codewords in a tree-structure. To find the best match codeword in the codebook for a query image vector, we will need to traverse through the tree. Efficiency is gained as

only a small number of codewords in the codebook is compared with the query image vector when traversing through the tree. To further improve the encoding efficiency of the tree-structured method, we also precalculate and store information of the codewords required for comparing their similarity. By using these information, the number of operations performed online to compare their similarity is greatly reduced. Thus, more efficiency is achieved.

1.4.4 INTRODUCTION TO IMAGE RETRIEVAL USING THE INVERTED FILE STRUCTURE

To retrieve images from the database efficiently, we have adopted the inverted file structure in our retrieval process. An inverted file is a structure commonly used in text retrieval system to achieve efficient text document retrieval. For a text retrieval system, the inverted file contains every distinct word that can be found in the database documents. Along with each distinct word in the inverted file, a collection of document names, which the word has appeared in, is listed. During retrieval, by first identifying the distinct words in the query document and then looking into the inverted file constructed, a set of database documents, which have word(s) common to the ones in the query document, can be compiled. Next, using a similarity metric, the similarity between each document in the list compiled and the query document is calculated. The set of documents that are most similar to the query document is retrieved. Retrieval efficiency is achieved since the time saved on calculating the similarity between the relatively smaller list of likely documents is much greater than the extra time needed to compile the list of likely documents.

In order to speed up our image retrieval process using the inverted file structure, each of our encoded images is treated like a text document. This way, the codewords in the codebook can be treated like the set of distinct words compiled

from the database documents. The inverted file for our image retrieval system can therefore be built in a similar way to the one for text retrieval system.

1.5 OBJECTIVES OF THE THESIS

The main objective of this thesis is to develop a colour-based image indexing and retrieval technique, which captures the spatial information of the colour pixels in the images. Compared with the existing techniques, the technique we proposed should have better retrieval effectiveness. In order to implement the proposed technique in an interactive CBIR system, it should also have the ability to index images and retrieve images from the database efficiently.

To achieve our objectives, our investigation involves:

- Studying the concepts of compression and decompression of digital still images using VQ.
- Implementing VQ encoding and decoding in software.
- Studying the feasibility of indexing and retrieval of images based on VQ compressed image data.
- Identifying which colour space, codebook size and codeword block-size are suitable for our proposed technique to perform effectively and efficiently.
- Implementing indexing and retrieval of images based on VQ compressed image data.
- Evaluating the effectiveness of indexing and retrieval of images based on VQ.
- Studying efficient search methods for multidimensional data.
- Using knowledge acquired from the studies of multidimensional search methods, develop efficient methods which are suitable for the image encoding and retrieval processes in our proposed scheme.

- Implementing the efficient search methods for the encoding and retrieval processes.
- Evaluating the efficiency of the search methods developed.

1.6 SUMMARY OF CONTRIBUTIONS OF OUR RESEARCH

The main contributions of our research are illustrated as follows:

- Proposed a new CBIR technique that captures spatial information of the colour pixels in the images.
- Investigated and identified a codebook generation algorithm which is suitable for such application.
- Investigated the impacts of different codebook sizes, codeword block-sizes and colour spaces on the indexing and retrieval performances of the proposed technique.
- Compared the retrieval effectiveness of the proposed VQ-based technique with that of the existing techniques on four image databases of various sizes. We have found that our proposed technique performs better than the existing techniques.
- Investigated and developed strategies that allow query images to be encoded at a response time that is acceptable for an online application.
- Investigated and developed efficient image retrieval strategies for the VQ-based image features. This is to ensure that the response time of the image retrieval is acceptable for an online application.

1.7 ORGANIZATION OF THE THESIS

Chapter 2 : *Content-based Image Retrieval*

This chapter provides a comprehensive introduction of various content-based image retrieval techniques. Discussions of the strengths and weaknesses of the various techniques are also presented.

Chapter 3 : *Fundamentals of Vector Quantization for Image Compression*

This chapter describes various important concepts and techniques in image compression using vector quantization. This chapter also provides readers with the fundamental knowledge in VQ which are required in our discussion in the next chapter.

Chapter 4 : *Using Vector Quantization for Image Indexing and Retrieval*

This chapter describes the VQ-based image indexing and retrieval technique we proposed. We discuss the properties of VQ that are suitable for our purposes. We also discuss how our proposed technique is able to overcome limitations faced by the existing techniques. We have also highlighted some issues to be considered during the implementation of our proposed technique.

Chapter 5 : *Review of Efficient Search Methods for Encoding and Retrieval processes*

This chapter reviews various search methods for multi-dimensional data. The study of these methods allows us to design search methods which we can implement to improve the efficiency of the encoding and retrieval processes in our proposed VQ-based indexing and retrieval technique.

Chapter 6 : *Experimental Results to show the Effectiveness of VQ-based Indexing and Retrieval*

This chapter presents and analyses the results of experiments which are conducted to evaluate the features and performances of indexing and retrieval based on vector quantization. The experiments are conducted using the implemented vector quantization software.

Chapter 7 : *Implementation & Experimental Results of Efficient Search Method for Image Encoding*

This chapter describes the search method we adopted in the image encoding process. The experimental results are also presented to illustrate the efficiency gained by the adopted method as compared with the full-search method. The findings of the effects on retrieval effectiveness of our proposed VQ-based CBIR technique using this search method is also illustrated.

Chapter 8 : *Implementation & Experimental Results of Efficient Search Methods for Image Retrieval*

This chapter describes two search methods we adopted in the image retrieval process. The experimental results are also presented to illustrate the efficiency gained by the adopted methods as compared with the full-search method. The findings of the effects on retrieval effectiveness of our proposed VQ-based CBIR technique using these search methods is also illustrated.

Chapter 9 : *Conclusions and Future Developments*

This chapter summarizes the concepts and findings in our research. In addition, the possible future developments in this area are highlighted.

Appendix A

Appendix A shows the survey forms used for conducting subject test to identify ground truth images for each query image of a test image database, called SCD. This image database is compiled by our research group.

Appendixes B1 to B3

Appendix B1 to B3 show the complete set of average recall and precision graphs used to evaluate the effects of different codebook sizes, codeword block-sizes and different colour spaces on the retrieval performance on our proposed CBIR technique. The test image database used for generating the retrieval results is Database SCD.

Appendixes C1 to C3

Appendix C1 to C3 show the complete set of average recall and precision graphs used to evaluate the effects of different codebook sizes, codeword block-sizes and different colour spaces on the retrieval performance on our proposed CBIR technique. The test image database used for generating the retrieval results is Database CCD. This database is adopted by MPEG-7 as the standard database for testing the retrieval performance of different colour-based descriptors.

CHAPTER 2

CONTENT-BASED IMAGE INDEXING AND RETRIEVAL

2.1 INTRODUCTION

Images have always played an important role in many areas of our daily lives. One such area is in human communication. For example, whenever we are trying to convey a concept or an idea, it is always easier to do it with the aid of images. Due to such importance of images, many digital images are kept in computer databases for future use. Currently, with the rapid growth of the Internet, it is also possible to access large number of such image databases all over the world. In order to access these information efficiently and effectively, good image indexing and retrieval techniques are required.

In this chapter, we will review some commonly used image indexing and retrieval techniques. The first technique is based on text (Section 2.2), in which images are indexed and retrieved based on textual descriptions given to the images, normally by the database administrator. Here, we will also discuss about the limitations of this technique, which lead to a new concept of indexing and retrieving images using the image content or the low-level features in the images. The three main low-level features used in content-based image indexing and retrieval techniques

are colour, shape and texture. We will discuss about these techniques in Section 2.3, 2.4 and 2.5 respectively. Next, in Section 2.6, an overall discussion on the techniques is presented. Finally, Section 2.7 summaries the chapter.

2.2 IMAGE RETRIEVAL BASED ON TEXT

The technique of retrieving images based on text is evolved from techniques used in automated information retrieval (IR) systems [42, 44]. IR systems are basically developed to manage the huge amount of scientific literature more effectively and efficiently [23, 73]. In text-based image retrieval, to make use of the techniques used in IR systems, we need to describe the image with uncontrolled or free text. These descriptions can be in terms of textual information, captions and annotations. From the text descriptions, a set of keywords is extracted to characterise the image. This is done by filtering out the stop words. Examples of common stop words are shown in Table 2.1. During retrieval, each database image is determined if it is relevant to the query image by matching their sets of keywords. The matching process can be done using Boolean operators. As this technique of identifying images is similar to traditional ways of selecting information in databases, it is used in many image retrieval applications. Examples of such applications are Picture Cardbox [28] and Center of Excellence for Document Analysis and Recognition's (CEDAR) Piction [85].

A	An	Be
About	Again	Begin
All	Are	Beginning
Also	At	Between
Around	Among	Become
Always	Although	Becoming

Table 2.1 Examples of common stop words

Text-based image retrieval can be very effective if each image is very well described. With the text annotation, we are able to capture high-level abstractions and concepts like human expressions, such as "surprise" and "angry", in the contents of an image. However, text-based image retrieval techniques also have many limitations.

Firstly, the annotation aspect in this technique is a difficult task. Text descriptions for each image have to be done manually because computers are still incapable of understanding the high-level abstractions and concepts in the images. It is also difficult to describe the images completely and consistently. The following are some examples regarding these problems. Words like "angry", "furious" and "mad" have similar meaning. So when describing such a human expression, which word should we use? These words can also be classified as a subclass of a more general term called "human expressions". Do we add this term to the description while describing the images?

Secondly, text description may be subjective. Figure 2.1 is an example of such an image. While some people think it is an image of an old woman, others may think it is an image of a young lady.



Figure 2.1 Old woman or young lady?

Thirdly, it is also difficult to describe the low-level features, like texture and irregular shapes, in an image using text.

Finally, text-based image retrieval techniques do not support query-by-example, which the application can retrieve similar database images based on the query image provided by the user.

To complement and partially overcome the limitations of text-based retrieval techniques, content-based image retrieval (CBIR) techniques based on colour, shape and texture are used.

2.3 IMAGE RETRIEVAL BASED ON COLOUR

Colour features are widely used in CBIR systems for two main reasons:

1. Colour information can be easily extracted from the image regardless of the image file format.
2. It is a relatively effective way to represent the content of the images.

In this section, we will describe some commonly used CBIR techniques which are based on colour.

2.3.1 BASIC COLOUR-BASED HISTOGRAM IMAGE INDEXING AND RETRIEVAL TECHNIQUE

In the basic colour-based histogram image indexing and retrieval technique, each image in the database is represented using three primaries of the colour space chosen, for example RGB, HSV or LUV. Each colour channel is quantised into m intervals. So the total number of discrete colour combinations (called bins) n is equal to m^3 . For example, if each colour channel of a real image with colour depth of 24 bits is quantised into 8 intervals, we have 512 bins in total. To calculate which bin number the pixel belongs to, a formula can be used. Assuming the RGB colour space is used, the formula is illustrated as follows:

$$\text{Bin num} = \left(\left\lfloor \frac{\text{Red channel}}{2^{(8-n)}} \right\rfloor \times 2^n \right) + \left(\left\lfloor \frac{\text{Green channel}}{2^{(8-n)}} \right\rfloor \times 2^n \right) + \left(\left\lfloor \frac{\text{Blue channel}}{2^{(8-n)}} \right\rfloor \right)$$

where n is the revised number of bits to represent each colour channel. For example, if a 256-levelled colour channel in a real image is quantised into 9 levels, then the revised number of bits, n , is 3 bits.

A colour histogram $H(M)$ is a vector (h_1, h_2, \dots, h_n) , where each element h_j represents the number of pixels falling in bin j in image M . This histogram is the feature vector (index) to be stored as the index of the image in the database.

During image retrieval, a histogram is found for the query image or estimated from the user's query. A metric is used to measure the distance between the histograms of the query image and images in the database. If images are of different size, their histograms are normalized. Images with a distance smaller than a pre-defined threshold are retrieved from the database and presented to the user. Alternatively, the first k images with smallest distances are retrieved. The distance metric between images I and H is defined as

$$d(I, H) = \sum_{l=1}^n |i_l - h_l| \quad \text{where } i_l \text{ and } h_l \text{ is the number of pixels falling in bin } l \text{ in}$$

images I and H respectively.

This basic technique has a number of limitations. The first limitation is that this technique only considers the colour distribution of the pixels and ignores the spatial relationships among pixels. This means that as long as the number of pixels falling into bin l for images I and H are equal, the two images will be considered to be exactly the same. Let us look at the following example. Image  has exactly the same number of pixels in two colour bins as image . Thus, using the basic colour-based histogram technique, the two images are considered exactly the same, but perceptually, they are actually very different.

The next limitation of this technique examined is related to the quantization of each colour channel into a number of intervals. This quantization of channels may cause the retrieval performance of this technique to be less effective or efficient.

Let us look at the following example. If each colour channel of images with colour depth of 24 bits is quantised into 8 intervals, some colours which are quantised into different bins may be very similar perceptually.



Red, Green, Blue: (63, 64, 64) (64, 64, 64)

Figure 2.2 Two Colours from the different Bins

For example, Figure 2.2 shows two perceptually very similar colours which belong to two different bins. If the majority area of one image consists of one of these colours while the majority area of another image consists of another one of these colours, we should consider the two images as very similar. However, since the two colours are from different bins, the histograms of the two images will be very different. This causes the retrieval to be less effective. It is also difficult to improve on the effectiveness of the techniques by adjusting the number of quantization intervals for each colour channel because:

- The number of similar colours is not evenly distributed. Thus it is difficult to decide on the number of quantization intervals [58, 87].
- The perceptual similarity between colours is very subjective. So even if we quantised the colours manually, it is still difficult to decide which colours should belong to which bins.

2.3.2 QBIC COLOUR HISTOGRAM TECHNIQUE

The colour histogram technique implemented in QBIC is the one proposed by Ioka [36]. As in the basic colour histogram technique, the colour space is quantised. This is done by firstly dividing each of the Red, Blue and Green channel into 16 levels, thus giving a colour space of 4096 cells. The MTM (mathematical transform to Munsell) [51] coordinates of the center of each cell are then computed. Next, by clustering the derived MTM coordinates according to their colour similarity, the 4096 cells are grouped into K clusters, called "supercells" [19]. Each supercell is represented by the center colour of the range of colours in it. The center colours also represent the partitions of the quantised colour space. K (user-settable) is also the number of bins that the colour histogram has. As part of the clustering, a table that maps any RGB triple to its corresponding bin number is also constructed. To build the colour histogram for an image, the RGB value of each pixel is firstly mapped to the supercell number. Then, the value in the corresponding histogram bin number is incremented. Finally, the histogram is normalized so that the sum is unity.

To partially solve the quantization related problem in the basic colour histogram technique stated above, contributions of perceptually similar colours are taken into account in the distance calculation. To do this, the technique accounts both for the perceptual distance between different pairs of colour (for example, orange and red are less different than orange and green) and for the difference in the amounts of a given colour (for, example, a particular shade of red). The similarity metric used in QBIC [1, 58] is as follows. Let X be the query histogram and Y the histogram of an image in the database, both normalized. Bin by bin similarity histogram Z is computed. Then the similarity between X and Y is given by $\|Z\| = Z^T A Z$, where A is a symmetric colour similarity matrix with

$$a(i, j) = 1 - d(c_i, c_j)/d_{\max}$$

where c_i and c_j are the i th and j th colour bins in the colour histogram, and $d(c_i, c_j)$ is the colour distance in MTM colour space, and d_{max} is the maximum distance between any two colours. The similarity matrix A accounts for the perceptual distance between different pairs of colours.

2.3.3 QBIC COLOUR LAYOUT TECHNIQUE

The second colour-based technique in QBIC uses colour layout [1, 58]. In this technique, an image is divided into 64 (8×8) blocks of equal size. The average colour for each colour channel is found for each block. Therefore, for each colour channel, the image is represented by 64 average colours. The 2-D Discrete Cosine Transform (DCT) is applied to these 64 average colours to obtain 64 coefficients. The difference between two images is calculated as the sum of Euclidean distance between corresponding coefficients of the colour channels of the two images. In practice, only first few DCT coefficients are used in distance calculation.

In image retrieval, users normally want the system to retrieve not only images from the database which are the exact matched (in terms of object location) to their query images, but also the ones which are the rotated or skewed versions of their queries. From the way this technique is designed, the positions of the colours in the images have great influence to their indices. Thus, the rotated and skewed versions of the user's query images are unlikely to be retrieved using this technique.

2.3.4 IMAGE RETRIEVAL BASED ON COLOUR COHERENCE VECTOR

As explained in Section 2.3.1, one of the limitations in colour histogram techniques is that they fail to take into consideration the spatial relationships among the neighbouring image pixels. To overcome this limitation, Pass and Zabih [63] refine the colour histogram technique by classifying the pixels in each histogram bin into coherent pixels and non-coherent pixels. The refined colour

histogram is called a colour coherent vector (CCV). A pixel is classified as coherent if it belongs to a large uniformly coloured region, e.g., a region with area larger than 1% of the image. The process of building the CCV is as follows:

1. The original image is smoothed using a lowpass filter. The purpose of this operation is to reduce the noise in the image, especially in regions of the image that are of similar colours. Thus, the classification of pixels into coherent or non-coherent pixels can be carried out more effectively.
2. The chosen colour space is quantised to acquire N colour bins.
3. To determine if a pixel is coherent or non-coherent, connected components are computed by grouping each image pixel with their neighbouring pixels according to their colour. A connected component P is a maximal set of pixels such that for any two pixels $p_1, p_2 \in P$, there is a 4-connected or 8-connected path in P between p_1 and p_2 . A pixel is coherent if the size of its connected component is larger than 1% of the image.

Thus, if α_i represents the number of coherent pixels of the i -th colour bin of an image and β_i represents the number of non-coherent pixels, then the CCV of the image is defined as $\{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \dots, (\alpha_N, \beta_N)\}$ and the colour histogram of the same image is defined as $\{(\alpha_1 + \beta_1), (\alpha_2 + \beta_2), \dots, (\alpha_N + \beta_N)\}$.

With this refinement, the distance metric between images I and H is defined as

$$d(I, H) = \sum_{i=1}^N (|I(\alpha_i) - H(\alpha_i)| + |I(\beta_i) - H(\beta_i)|).$$

Though CCV technique is reported to be a better retrieval technique compared with the basic colour histogram technique, it has its limitations. Firstly, it is difficult to carry out the lowpass filtering process automatically. This is because different images require different levels of lowpass filtering to achieve an ideal smoothing effect. If an arbitrary filter level is picked for all images, then some images will be blurred, destroying the important colour information in them, while some images will remain too sharp. Secondly, images that are different

perceptually can still have the same CCV. The following are examples of such images:



The two images in the example are made up of the same number of white pixels and blue pixels. Although the white and blue pixels are in their respective uniformly coloured regions, they are different perceptually. However, their CCVs are likely to be the same since the technique only classify if the image pixels belong to a non-uniformly or uniformly coloured regions in the image using a threshold (e.g. 1% of the image). Information about how pixels are positioned in their respective regions is not captured in their CCVs.

2.3.5 IMAGE RETRIEVAL BASED ON COLOUR CORRELOGRAM

Another colour-based technique that captures spatial relationships among pixels uses colour correlogram [34]. Let I and $I_{c(i)}$ represent the entire image pixels and the set of pixels whose colours are $c(i)$ respectively. Then the colour correlogram is defined as

$$\gamma_{i,j}^{(k)} = \Pr_{p_1 \in I_{c(i)}, p_2 \in I} [p_2 \in I_{c(j)} \mid |p_1 - p_2| = k],$$

where $i, j \in \{1, 2, \dots, N\}$, $k \in \{1, 2, \dots, d\}$, and $|p_1 - p_2|$ is the distance between pixels p_1 and p_2 .

Basically, a colour correlogram represents a table indexed by colour pairs, where the k -th entry for $\langle i, j \rangle$ specifies the probability of finding a pixel of colour j at a distance k from a pixel of colour i in the image. In this feature, besides capturing the colour distributions of the pixels in terms of percentage of a given colour, the spatial correlation of a pair of colours in the image are also captured. It is reported that this retrieval technique outperforms the colour histogram and CCV techniques [34, 48]. However, the dimension size of the colour correlogram is very large ($O(N^2d)$) if all the possible combinations of the colour pairs are considered.

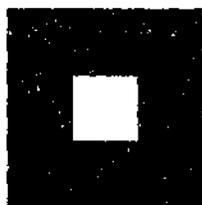
Image feature of such dimension size is not ideal for many applications because it is very computationally expensive to build such a large feature. To reduce the feature dimension size, a simplified version called the colour autocorrelogram is proposed [34]. This feature only captures spatial correlation between identical colours, thus its dimension is reduced to $O(Nd)$.

In terms of the retrieval performance, the autocorrelogram technique also has some limitations. Firstly, to keep the dimension size of the autocorrelogram to a size that is reasonable for processing in applications, only spatial correlation between identical colours is captured in this feature. Thus, by not capturing spatial correlation between different colours, its retrieval performance is likely to be less effective compared with other techniques that do consider such information in the images.



6 x 6 pixels

$$\gamma_{i,j}^{(k)}(I) = [0.38, 0, 0, 0, 0.66, 0.26, 0.088, 0]$$



36 x 36 pixels

$$\gamma_{i,j}^{(k)}(I') = [0.88, 0.55, 0.37, 0.23, 0.94, 0.68, 0.57, 0.48]$$

Figure 2.3 Autocorrelograms of 2 similar images of different scales

Secondly, the retrieval effectiveness of the technique is image size variant. Figure 2.3 is an example to demonstrate this limitation. In this figure, there are two perceptually similar images which are of different scales. However, if normalisation of the images' sizes is not carried out first, their autocorrelograms built can be very different. There is no normalisation method to make the autocorrelograms to be image size invariant, unlike many other existing techniques which have normalisation method to make their features built to be image size invariant. Thus the autocorrelogram technique can only be effective if

all the database images are similar in size. Such a requirement of the technique is difficult to achieve as most databases contain images of a variety of sizes. If the database consists of images of a variety of sizes, normalisation of image sizes before building the autocorrelogram also creates many other issues. This is because it is difficult to resize images of different sizes to a single size without distorting the colour contents in the images. Besides, it is difficult to decide what is the most ideal image size to normalise all the images' sizes. Due to these limitations, the robustness of the technique is questionable.

2.3.6 IMAGE RETRIEVAL BASED ON COLOUR MOMENT

Images can be characterised by the moments of their colour distribution. This is a technique derived from probability theory. In this theory, the first three moments of an image colour distribution are defined as:

$$\text{mean (first order moment)} \quad \mu_i = \frac{1}{N} \sum_{j=1}^N p_{ij}$$

$$\text{variance (second order moment)} \quad \sigma_i = \left(\frac{1}{N} \sum_{j=1}^N (p_{ij} - \mu_i)^2 \right)^{1/2}$$

$$\text{skewness (third order moment)} \quad s_i = \left(\frac{1}{N} \sum_{j=1}^N (p_{ij} - \mu_i)^3 \right)^{1/3}$$

where N is the number of pixels in the image, p_{ij} is the value of the i -th colour channel of the j -th image pixels.

There are two advantages of using colour moments to index images. Firstly, no colour quantization is needed. Secondly, the image feature is usually a lot more compact compared with other methods. For example, only $3 \times 3 = 9$ (three moments of each colour channel) values are needed to represent the colour content of each image. However, the compact image feature also has lower ability to discriminate the differences between images. Thus, the colour moment technique is normally used as the first pass to narrow down the search space before using other more effective colour methods for retrieval.

2.3.7 EFFECTS OF COLOUR SPACES ON RETRIEVAL

In any colour-based indexing and retrieval techniques, before any further image processing can be done, the first step is to represent the pixels values in a common colour space. A colour space is a three dimensional space which defines how colours are represented. To date, there are many colour spaces available. Examples of colour spaces are RGB, HSV and LUV. They are designed with different sets of characteristics so that they can be effectively used in their respective industry areas. Since each colour space possesses different characteristics, they may influence the effectiveness of the kind of processing carried out in a colour-based image retrieval technique. Thus, the choice of colour space may affect the retrieval performance of a colour-based technique.

Generally, a colour space which is suitable for image indexing and retrieval should be perceptually uniform. This means that if the distance between colours c_1 and c_2 is d , and the distance between colours c_3 and c_4 is also d , then c_1 and c_2 should look as different as c_3 and c_4 . This characteristic is important in colour-based techniques because of two reasons. Firstly, colour spaces, that are more capable of reflecting the perceptual differences between colours through their distances calculated, are more likely to produce better retrieval performance. This is because the colour-based techniques would usually make use of distances between colours in the process of building their features to index their images. By utilising a colour space whose distances between colours can more accurately reflect the perceptual differences between colours, the features built are more likely to be capable of differentiating the differences between images. Secondly, in many colour-based techniques where colour quantization is required, such characteristic enables more colours that are similar to be accurately classified in the same bins. Thus better colour quantization is achieved. Another characteristic that make a colour space suitable for image indexing and retrieval is the ease of converting the pixel values in the original colour space (normally in RGB) to that colour space. This characteristic allows efficient indexing of images.

To date, many colour spaces have been used for colour-based image retrieval. However, it is still not clear which colour space is best suited for such applications as there has been no comprehensive testing being carried out using all of these colour spaces on a common large image dataset.

2.4 IMAGE RETRIEVAL BASED ON SHAPE

Shape is another important low-level feature which is commonly used in CBIR. To index an image, it must first be segmented into individual objects using some segmentation algorithms. After the segmentation process, shape analysis techniques are used to find a descriptor to represent the object boundary. The shape descriptor, which is also called a shape feature vector, must be robust to noise. It is also often required to be scale, rotation or translation invariant.

In the past ten years, there are many shape analysis techniques being published in literature. We can classify them into two main categories. They are boundary-based and region-based. Boundary-based shape analysis techniques use only the object boundary information to build the shape descriptor. Region-based shape analysis techniques use all the pixels within the object boundary to derive the shape descriptors. In the following sub-sections, we will describe a few commonly used techniques.

2.4.1 SIMPLE SHAPE DESCRIPTORS

Area, circularity ($\text{perimeter}^2/\text{area}$), eccentricity (length of major axis/length of minor axis), major axis orientation and bending energy [17] are some simple shape descriptors that are commonly used in commercial CBIR systems like QBIC [1, 58] and Photobook [64]. Each of these shape descriptors can only capture a rough representation of the object shape, thus can only differentiate shape with large dissimilarities. For better retrieval performance, they are usually used together with other shape representations.

2.4.2 SPECTRAL TRANSFORM DESCRIPTORS

Spectral transform descriptors are also commonly used in many shape analysis techniques. Spectral transform descriptors, like Fourier descriptors (FD) and wavelet descriptors (WD), are derived by applying spectral transformation on 1-D shape signatures. Examples of 1-D shape signatures are complex coordinates, polar coordinates, central distance, tangent angle, cumulative angle, curvature, area and chord-length [17, 62].

In the FD-based technique [19, 37, 65, 102], a discrete Fourier transform is applied to the signature to obtain the FD. The discrete Fourier transformation of a shape signature $f(i)$ is defined as

$$F_u = \frac{1}{N} \sum_{i=0}^{N-1} f(i) \cdot \exp \left[-\frac{j2\pi ui}{N} \right]$$

for $u = 0$ to $N-1$, N is the number of samples in $f(i)$.

The FDs derived are used to index the object and Euclidean distance can be used to calculate the similarity between objects.

The use of WD [95, 60, 100] in shape-based image retrieval techniques is relatively new compared with FD. In this technique, each shape is represented by a set of wavelet coefficients. To derive the wavelet coefficient, a mother wavelet function is firstly applied to a set of sample points on the shape boundary. Then the translated and dilated wavelets, which are required in the next step of the process, are derived from the mother wavelet. Finally, after applying wavelet decomposition to each points of the shape boundary, the set of wavelet coefficients is derived. These coefficients are used to represent the shape.

The main advantage of using WD is the descriptor has multi-resolution on spatial space. However, the complicated matching scheme of wavelet representation makes it impractical for online shape retrieval.

2.4.3 INVARIANT MOMENTS

From the region-based moments of the objects, we can derive the invariant moments [33, 52]. For a digital image $f(x,y)$, the moment of order $(p + q)$ is defined as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y)$$

where x, y is the pixel position of the image, and $f(x,y)$ is the pixel intensity.

If \bar{x} and \bar{y} are defined as

$$\bar{x} = m_{10} / m_{00}$$

$$\bar{y} = m_{01} / m_{00}$$

then the central moments are expressed as

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x,y)$$

The central moments of up to order 3 are defined as follows:

$$\mu_{00} = m_{00}$$

$$\mu_{10} = 0$$

$$\mu_{01} = 0$$

$$\mu_{20} = m_{20} - \bar{x} m_{10}$$

$$\mu_{02} = m_{02} - \bar{y} m_{01}$$

$$\mu_{11} = m_{11} - \bar{y} m_{10}$$

$$\mu_{30} = m_{30} - 3\bar{x} m_{20} \bar{y} + 2m_{10} \bar{x}^2$$

$$\mu_{30} = m_{30} - 3\bar{x} m_{20} \bar{y} + 2m_{10} \bar{x}^2$$

$$\mu_{30} = m_{30} - 3\bar{x}m_{20} + 2m_{10}\bar{x}^2$$

$$\mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2m_{10}$$

$$\mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2m_{01}$$

$$\mu_{03} = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2m_{01}$$

The normalised central moment of order $(p + q)$, denoted by η_{pq} , is defined as

$$\eta_{pq} = \mu_{pq} / \mu_{00}^{\gamma}$$

$$\text{where } \gamma = \frac{(p+q)}{2} + 1 \quad \text{for } p+q=2, 3, \dots$$

It has been shown that the following seven moments are invariant to translation, rotation and scale change [33]:

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} + 3\eta_{12})^2 + (3\eta_{21} + \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (\eta_{30} + 3\eta_{12})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ + (3\eta_{21} + \eta_{03})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\phi_6 = (\eta_{20} + \eta_{02}) [(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} + 3\eta_{30})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ + (3\eta_{12} + \eta_{30})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

The above seven moments are used as the object's shape descriptor. With relatively lower dimensional shape descriptor, the similar computation of this technique is very efficient. However, it is also not sufficient to accurately describe the object shape.

2.4.4 GRID SHAPE DESCRIPTORS

In some shape-based retrieval techniques [8, 45, 70], a grid space with cells of equal size is first overlaid over the shape. Next, 1s are assigned to those cells that are fully or partially covered (more than 15% of the cell) by the shape and 0s to the rest of the cells. By scanning from top to bottom and left to right, a binary sequence is obtained for the shape. This binary feature vector is used to index the shape. The shape must be normalised before the scanning process to accommodate translation, rotation and scaling of the shape. To measure the similarity between two shapes, the binary Hamming distance is used.

2.4.5 MPEG-7 REGION-BASED SHAPE DESCRIPTOR

The region-based shape descriptor defined in MPEG-7 is based on Angular Radial Transform (ART) [49]. Using ART, a shape can be transformed into a number of orthogonal 2-D basis functions (complex-valued). The normalised and quantised magnitudes of the coefficients are used as the descriptor for the shape.

Basically, ART is the orthogonal unitary transform defined on a unit disk that consists of the complete orthonormal sinusoidal basis functions in polar coordinates, and the coefficients are defined as:

$$F_{nm} = \langle V_{nm}(\rho, \theta), f(\rho, \theta) \rangle = \int_0^{2\pi} \int_0^1 V_{nm}^*(\rho, \theta) f(\rho, \theta) \rho d\rho d\theta$$

where F_{nm} is an ART coefficient of order n and m ; $f(\rho, \theta)$ is an image function in polar coordinates and $V_{nm}(\rho, \theta)$ is the ART basis function that are separable along the angular and radial directions, that is,

$$V_{nm}(\rho, \theta) = A_m(\theta) R_n(\rho)$$

To achieve rotation invariance, the angular basis function used is

$$A_m(\theta) = 1/2\pi \exp(jm\theta)$$

The radial basis function is defined as

$$R_n(\rho) = \begin{cases} 1 & n=0 \\ 2 \cos(\pi n \rho) & n \neq 0 \end{cases}$$

To compare the dissimilarity between two shape descriptors, the L-1 norm distance matrix is used.

2.4.6 MPEG-7 CONTOUR-SHAPE DESCRIPTOR

The contour-shape descriptor defined in MPEG-7 is mainly based on the Curvature Scale-Space (CSS) representation of a contour [53, 54]. Thus before we describe the components in the descriptor, we will first briefly illustrate the CSS representation.

CSS REPRESENTATION

The CSS representation is derived based on the observation that when we are comparing the similarity between shapes, we would firstly divide each shape contours into concave and convex sections. Then the similarities between contours in the corresponding sections of the shapes are compared.

The CSS representation of a shape is captured in the form of a CSS map. To build a CSS map, we will first represent the shape boundary as a 1-D signal. Next, the zero crossings of the curvature at different scale are examined to extract information on the concavities and convexities of the shape. The algorithm of constructing the CSS map is as follows [49]:

1. Standardise the number of boundary points to represent each shape;
2. Record the curvature zero crossing points at each scale in an array A[];
3. Compute the curvatures of each position t at current scale δ ;
4. Record curvature zero crossing points at current scale δ to A[z_c(t, δ)];
5. Smooth the boundary;
6. Repeat step 3 to 5 until no zero crossing points are found;
7. Plot the zero crossing points in A[] onto Cartesian space to create CSS map.

DESCRIPTOR REPRESENTATION

As mentioned above, the contour-based descriptor defined by MPEG-7 consists of several components. They are:

1. Eccentricity and circularity values of the original and filtered contour (each quantized to 6 bits).
2. The index indicating the number of peaks [49] in CSS map (6 bits).
3. The magnitude (height) of the largest peaks in the CSS map (7 bits).
4. The x- and y-positions on the remaining peaks (9 bits per peak).

In [49], it is reported that this descriptor is very efficient in applications where high variability in the shape (such as deformations in objects) is expected.

2.5 IMAGE RETRIEVAL BASED ON TEXTURE

Another commonly used low-level feature for CBIR systems is texture. Among past computer vision literature, the one which best define texture specifications is proposed by Tamura et al. [89]. In his work, texture is described by six features:

- **Coarseness.** Coarse is the opposite of fine. Coarseness measures the scale of texture. The larger the distinctive image elements, the coarser the image. So, an enlarged image is coarser than the original one.
- **Contrast.** Contrast measures the vividness of the texture. It is measured using four parameters: (1) dynamic range of gray level of the image, (2) ratio of the black and white areas, (3) sharpness of the edges, (4) period of the repeating patterns.
- **Directionality.** Directionality measures the “peakiness” of the distribution of gradient directions in the image. It measures both element shape and placement.

- **Line-likeness.** This parameter measures the shape of a texture element. Line-like and blob-like are two common types of shapes.
- **Regularity.** This measures the variation of an element placement rule. It is concerned with whether the texture is regular or irregular. Different element shape reduces regularity. A fine texture is normally perceived as regular.
- **Roughness.** Roughness measures how rough or smooth the texture is. It is related to coarseness and contrast.

Many of the texture-based image retrieval techniques proposed make use of some of the six features described above. For example, QBIC uses the modified versions of coarseness, contrast and directionality features for its texture-based image retrieval [58].

Many texture-based image retrieval techniques also make use of Gabor filter. Gabor filter is used to extract texture feature from the images for image retrieval [18, 79], and has been shown to be very efficient. Basically, Gabor filters are a group of wavelets, with each wavelet capturing a specific frequency at specific direction. Expanding a signal using this basis provides a localized frequency description, therefore local features of the signal are captured. By applying Gabor filters to an image, energy distributions on different spatial frequencies at different orientations of the image are found. Texture features can then be extracted from this group of energy distributions.

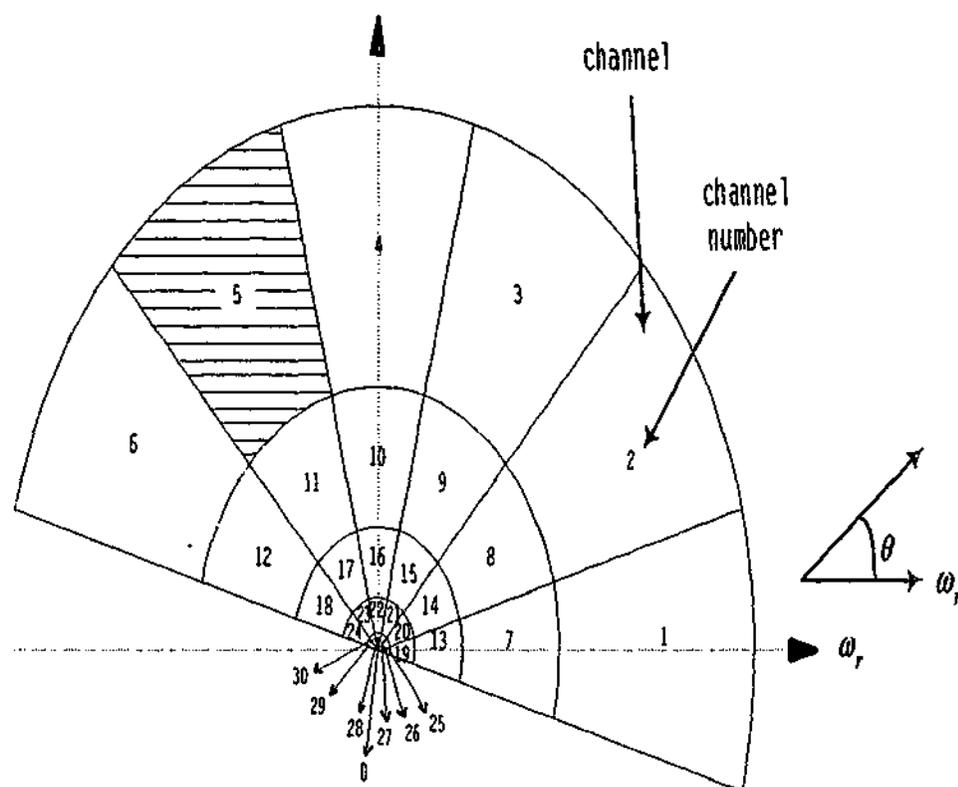


Figure 2.4 Frequency layout for feature extraction [33]

Figure 2.4 shows a set of Gabor filters with 6 orientations and 5 scales. Every filter can extract one energy feature and one energy deviation feature from the image, and altogether 60 features are obtained for image representation.

Other popular texture representations are fractal dimension [10], Fourier coefficients [16] and colour distribution statistics [88].

2.6 INTEGRATED CBIR TECHNIQUES

Some researchers have also developed CBIR systems based on more than one categories of CBIR techniques discussed above. One well-known CBIR system is the QBIC (Query by Image Content) system developed by IBM researchers [1]. This system offers CBIR techniques which are text-based, colour-based, texture-

based and shaped-based. Besides using these techniques separately on their own, the system also offers the users the option to combine different techniques when carrying out a query. Another CBIR system that falls into this category is the VisualSEEK system developed in Columbia University [2, 80]. Besides using keywords for querying images, this system also enable users to query images using image region, colour, shape and spatial locations. Other examples of such CBIR systems are the Virage Image Search Engine [3], Multimedia Analysis and Retrieval System (MARS) [61] and the ImageRover WWW Search Engine [90].

2.7 DISCUSSIONS ON CBIR TECHNIQUES

In Sections 2.2, 2.3, 2.4 and 2.5, we have discussed different techniques that are used for image retrieval. We have discussed that although text-based image retrieval techniques are popular and are commonly used in many image retrieval systems, there are still many limitations linked to the text-annotation aspects of the techniques. These limitations have driven researchers to make use of low-level features in the content of the images for indexing and retrieval. Colour, shape and texture are the commonly used low-level features that are used in CBIR systems.

Among the three low-level features, the colour feature is the easiest to extract. This is because basic colour information of the image is always present no matter which file format the image is in. To extract the shape and texture features from an image, a good segmentation algorithm must first be applied to it. For shape, the segmentation algorithm is needed to detect individual object boundary in the image. For texture, the segmentation algorithm is needed to detect regions with the same texture in the image. Unfortunately, with current automated segmentation techniques, robust and accurate image segmentation is still difficult to achieve. Thus, manual segmentation is still often used for these techniques. Due to this major drawback of using these features, shape and texture image retrieval techniques are only used in very specific areas. For shape, such areas are

those where object shapes are already available. For texture, example of such areas is the textile industry where each image has uniform texture throughout. Due to the above factors, CBIR techniques using colour are still the most effective and efficient for indexing and retrieving general images.

To assess which of the colour-based techniques discussed above has the best performance, we present the findings in [48], where a database of 20000 colour images is used for the experiment. The images are obtained from Corel photo galleries and Brodatz album. To evaluate the retrieval performance of the colour autocorrelagram, CCV and the basic colour histogram techniques, a graph of the number of relevant images against the number of retrieved images for each technique is plotted (Figure 2.5). Although colour moments technique is not shown in Figure 2.5, it is stated that its retrieval performance is similar to the basic colour histogram technique [48]. The colour correlogram technique is also not shown as it was discussed in Section 2.3.5 that, it is not ideal for many applications due to the high dimensionality of this feature.

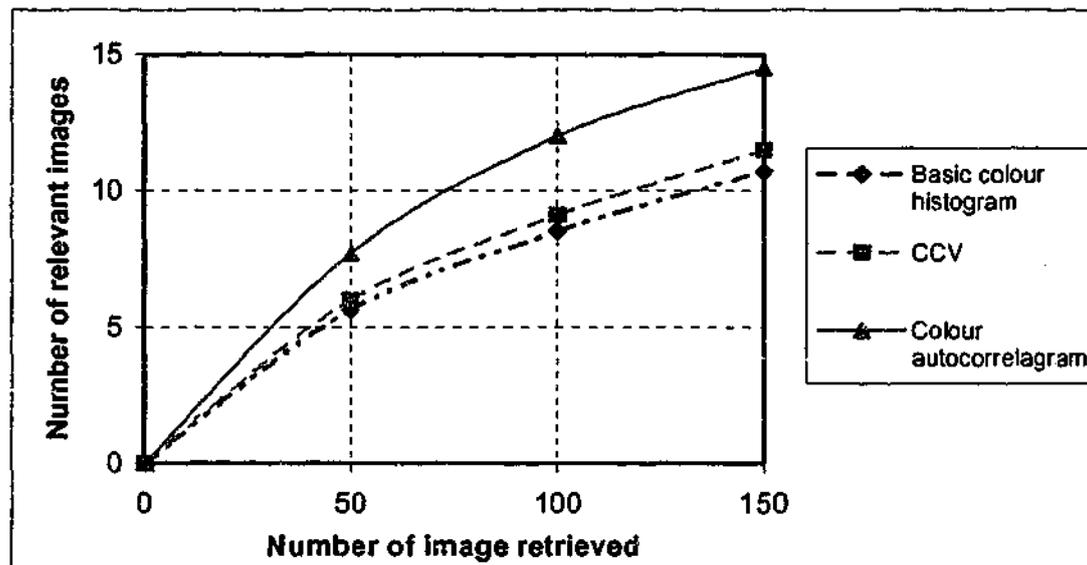


Figure 2.5 Comparison of colour autocorelogram, CCV and basic colour histogram features based on 64 LUV colours. The graph shows the average number of relevant images retrieved against number of image retrieved [48]

Figure 2.5 shows that the colour autocorrelagram technique has the best retrieval performance, whereas the CCV technique performs slightly better than the basic

colour histogram technique. This is because colour autocorrelogram takes into account the most amount of information on pixel correlation in the image compared with the other two techniques. In [48], it is also found that although CCV in theory does take into account the pixel correlation in the image, its effectiveness is limited. If we have a database which consists of images with similar colour composition, CCV is only effective in differentiating images which have their similar colours arranged close together, from those images which do not have their similar colours arranged close together. CCV is not effective in differentiating the perceptual differences between the images in each group of images (see Section 2.3.4 for an example).

Feature	Computation time	Dimension
Basic colour histogram	0.13 sec	64
Colour moments	0.55 sec	6
CCV	0.41 sec	128
Colour autocorrelogram	46.0 sec	256

Table 2.2 Feature Extraction Time and Feature Vector Length of Various Colour Features Based on 64-Colour Quantization [48]

Next, the time required to extract each colour features is taken into account. Although the colour autocorrelogram technique has superior retrieval performance compared with the other techniques, Table 2.2 shows that the time required to build the feature is on the average 183 times greater than the time required to build the features of the other three techniques. Colour autocorrelogram is the most computationally expensive colour feature due to its feature extraction complexity and its high dimensionality. Such characteristics of the feature would likely result in high feature extraction time and low image retrieval efficiency. Such feature extraction time and retrieval efficiency are unlikely efficient enough to meet user satisfaction in a CBIR system. This is because firstly, though image features are

normally built off-line, the comparison between image features during retrieval normally requires online processing. Thus higher dimensionality of the feature will normally result in higher computation time. Secondly, if the user supplies a query image which does not belong to the database, then online feature extraction of query images is required. Thus, high feature extraction time is not favourable for CBIR systems.

Although QBIC colour histogram and colour layout techniques are not shown in the findings above, their retrieval performance is better than the basic colour histogram technique [58]. However, their retrieval performance is unlikely to be better than the colour autocorrelogram technique. This is because like the basic colour histogram technique, information of pixel correlation in the image is not reflected in these features.

2.8 SUMMARY

From the above discussion, we can first conclude that colour is the most suitable low-level feature for indexing and retrieving general images compared with shape and texture. Next, we can also conclude that colour-based techniques that take into consideration the pixel correlation in the images have better retrieval performance compared with those that do not. We have also found that current colour-based techniques, which do take into consideration the pixel correlation in the images, are either slightly more effective than those that do not, or are too computationally expensive to be implemented especially for online systems. Thus, there is a need for a more effective and efficient colour-based image indexing and retrieval. For the basis of comparison for the proposed indexing and retrieval technique in Chapter 6, we will use the two QBIC colour-based techniques and the colour autocorrelogram technique. The two QBIC colour-based techniques are used since they are relatively effective and computationally inexpensive. Besides, they are commercially available. Colour autocorrelogram

technique is used as it is widely reported as one of the most effective existing colour-based retrieval techniques [34, 48].

CHAPTER 3

FUNDAMENTALS OF VECTOR QUANTIZATION FOR IMAGE COMPRESSION

3.1 INTRODUCTION

Non-compressed digital images take up large memory space. This leads to greater storage and data processing costs. Although hardware prices have decreased remarkably in recent years, the processing of a huge amount of image data is still not cost effective or efficient. If non-compressed images are transferred through the Internet, not only will they take up lots of time, they will also incur huge transmission costs. This is even more undesirable when dealing with online processing. Thus, compression techniques are required.

A digital image is actually made up of many small dots called pixels placed in rows and columns. When these pixels are examined closely, we can see that the neighbouring pixels on the same row and the adjacent rows are normally similar. These similarities are called spatial redundancies. The end users of images are normally human beings, who can tolerate some information error or loss without affecting the communication effectiveness. To achieve high compression ratio,

lossy compression techniques can be applied to compress images by exploiting the data statistics (data redundancy) and human perception properties [74]. Thus after lossy compression, besides removing the spatial redundancies, some information loss may occur. Lossy compression is the process that produces a compressed file which usually cannot be decompressed to the original file exactly. In the case of image compression, this information loss will usually not cause the reconstructed image to look perceptually very different from the original one.

Lossy compression techniques normally involve a process of representing a large – possibility infinite – set of values with a much smaller set. This process, called quantization, is the one that causes information loss to the compressed data. The values may be scalars or vectors. If the values are scalars, this process is called scalar-based quantization, otherwise, it is called vector-based quantization.

Presently, the lossy compression technique is most widely used to compress still digital images is JPEG [75]. JPEG is a scalar lossy compression technique because it works on individual image pixels and involves a scalar-based quantization process when compressing or decompressing the image. Though scalar lossy compression has established itself through the years, researchers have discovered that taking multiple samples of source data and compressing them as blocks rather than individual samples can produce better compression results [75]. We call this technique vector lossy compression. The vector lossy compression technique most widely used is vector quantization (VQ).

In this chapter, we describe the fundamentals of how VQ is used for image compression. Understanding the materials in this chapter is required to understand Chapter 4 where we will present how VQ is used for image indexing and retrieval.

3.2 BASIC CONCEPTS OF VECTOR QUANTIZATION

VQ has been successfully used for image and audio compression. A vector quantizer can be defined as a mapping Q of K -dimensional Euclidean space R^K

into a finite subset Y of R^K , that is:

$$Q: R^K \rightarrow Y,$$

where $Y = \{x'_i; i = 1, 2, \dots, N\}$, and x'_i is the i th vector in Y .

Y is the set of reproduction vectors and is called a VQ codebook or VQ table. N is the number of vectors in Y . At the encoder, each data vector x belonging to R^K is matched or approximated with a codeword in the codebook and the address or index of that codeword is transmitted instead of the data vector itself. At the decoder, the index is mapped back to the codeword and the codeword is used to represent the original data vector. In the transmitter and receiver, an identical codebook exists whose entries contain combinations of pixels in a block. Assuming the image block-size is $(n \times n)$ pixels and each pixel is represented by m bits, theoretically, $(2^m)^{n \times n}$ types of blocks are possible. In practice, however, there are only a limited number of combinations that occur most often, which reduces the size of the codebook considerably. This is the basis of VQ. If properties of the human visual system are used, the size of the codebook can be reduced further and fewer bits can be used to represent the index of codebook entries.

We can see that the concept of VQ is to use an index or token to represent a block of data. We use an example to illustrate how it works. Assume that we want to encode a grayscale image of 512×512 pixels with 8 bits per pixel. If we choose block-size as 8×8 pixels, there is a total of 4096 blocks. In a typical image, there are uniform coloured areas and some part of the image may be the same as or similar to another part of the image. Thus, there are many blocks having the same pixel values and patterns among the 4096 blocks. Since our visual system can tolerate some small errors, if the difference between two blocks is below a certain preset threshold, they are deemed as the same. In this case, the number of distinct blocks will be much smaller than 4096. Let us assume that there are 1000 distinct blocks. Then, 10 bits are required to represent each distinct block. These 1000 distinct blocks are the codewords for this image. So if we transmit the block

indices only, a total of 40960 bits are required instead of the original 2097152 (512 x 512 x 8) bits, resulting in a compression ratio of 51.2. In practice, however, we have to send the codebook to the receiver so that it can decode the encoded image. The codebook, in this case, requires 512000 (1000 x 8 x 8 x 8) bits. So the actual compression ratio achieved is 3.8. In practical VQ systems, the number of distinct blocks will be reduced further, since the codebook is somehow compressed, and a number of similar types of images share the same codebook. Thus we achieve a higher compression ratio. To apply VQ to colour image compression, first we need to transform the image into its three colour channels, eg. R, G and B planes for RGB colour space. Then, a codebook is created for each of the three channels. Finally, the encoding process described above is carried out for each of the colour channels. The purpose of encoding colour image in their individual channels rather than working in a single pixels channel is to achieve a higher compression ratio. This is further discussed in the next chapter.

Two major issues of VQ are how to design a good codebook that is representative of all the possible occurrences of pixel combinations in a block, and how to find a best match efficiently in the codebook during the coding process. The codebook generation process is discussed in the next section, while the issues on finding the best match codeword in the codebook are discussed in Chapter 4. To summarize, image compression based on VQ involves the following steps (Figure 3.1 and Figure 3.2):

1. A codebook that contains representative pixel patterns is found for the image to be coded. The entries in the codebook are called codewords. Each codeword of the codebook is identified by its index number. Similar types of image may share the same codebook to achieve a high compression ratio.
2. The image is divided into fixed-size pixel blocks called vectors.
3. For each vector, the codebook is searched to find a best match codeword. The index number of the entry is transmitted or stored instead of the vector. Thus

the compressed bitstream is a sequence of index numbers.

4. The codebook together with the compressed bitstream is passed to the decoder. During decoding, codewords in the codebook corresponding to the index numbers are retrieved and they replace the original vectors. If the match in step 3 is the exact match, the vectors are reconstructed exactly. In general cases, the match is just an approximation. Thus the reconstructed vectors are an approximation of the original vectors. Since index numbers are very compact compared with the vectors, we achieve a high compression ratio.

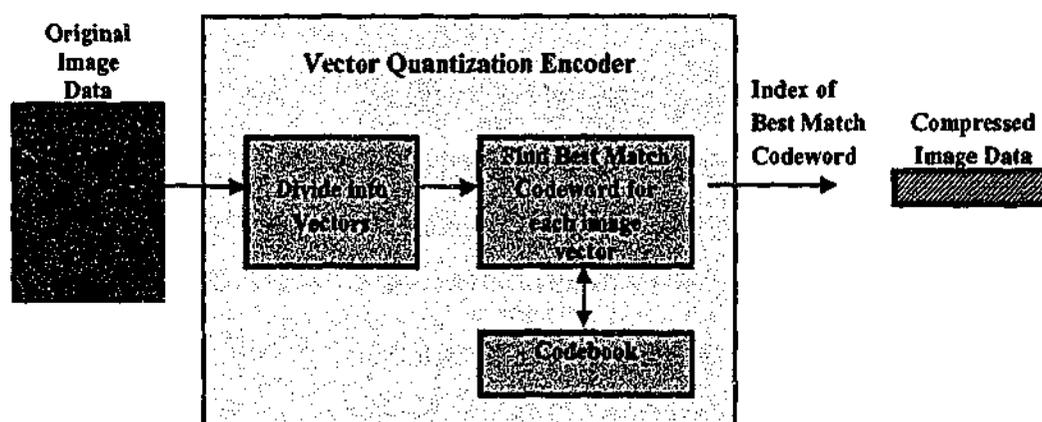


Figure 3.1 VQ Encoding Process

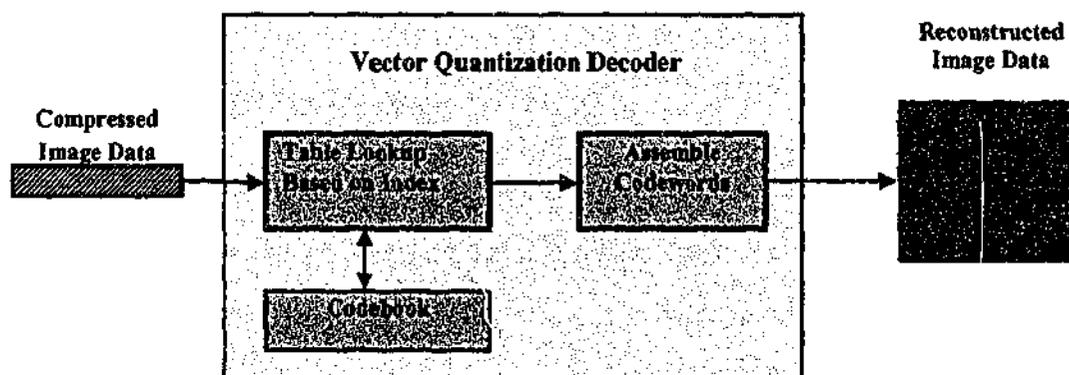


Figure 3.2 VQ Decoding Process

3.3 CONCEPTS OF CODEBOOK GENERATION

The codebook generation is a very important process in the concept of VQ. This is because with a well-designed codebook, we will be able to achieve maximum compression for the images by keeping the number of codewords at a minimum, while the quality of their reconstructed images remains reasonably good.

An image or a set of images, which best represent all the images using the generated codebook, are normally chosen to be used for generating a well-designed codebook. The chosen images best represent all the images because they have many features, eg. colours, which appear in all or most of the images. The chosen images are called training images. The vectors in the chosen images are called training vectors.

In a well-designed codebook, the set of codewords in it must be a set of vectors which all the training vectors would most likely to congregate onto. This process of choosing the right vectors is called codebook design. Since the vector is normally of multiple dimensions, to implement this compression technique effectively, it is impossible for the codebook design to be done by visually selecting the best vectors. Thus, an automatic procedure is needed for locating where the vectors are clustered.

Although many clustering algorithms are proposed for codebook generation in VQ, the main algorithm used is the Linde-Buzo-Gray (LBG) algorithm. The following subsection describes the LBG algorithm.

3.3.1 LINDE-BUZO-GRAY ALGORITHM

The LBG algorithm is an iterative process to find a set of vectors that the training vectors are most likely to congregate onto. Each of these vectors is the centroid of all the training vectors in each cluster. This set of vectors found make up the codebook. The following are the iteration steps of the LBG algorithm:

Initialisation:

1. Decide the size of the codeword which is normally a square block of $n \times n$ pixels. Divide the training image into non-overlapping training vectors ($T = \{t_1, t_2, \dots, t_j\}$) whose size is the same as the codeword size decided. Set the initial codebook $C = \{c_1, c_2, \dots, c_k\}$.

Clustering:

2. Let the codewords act as the initial vectors (IV).
3. Find the closest IV for each training vector using a distance function of your choice. Thus, compute $d(t_h, c_i)$, for $h = 1, 2, \dots, j$ and $i = 1, 2, \dots, k$.

If Euclidean distance is used,

$$d(t_h, c_i) = \sum_{w=1}^x \sqrt{(t_h^w - c_i^w)^2}, \text{ where } x = n \times n \text{ is the codeword block-size.}$$

4. Training vectors closest to the same IV are classified into the same group or cluster. So we have one cluster corresponding to each IV.
5. If a cluster does not have any training vector, the corresponding IV is replaced by a randomly chosen training vector from the cluster with the largest number of training vectors and new clusters have to be formulated again as in steps 3 and 4. Otherwise, continue.
6. Determine the centroid vector for each cluster by finding the average of all the training vectors in the same cluster. The centroid vector (CV) replaces the IV of the cluster to represent the cluster.

Termination Checking:

7. Recalculate the distance between each training vector and each CV to determine if any training vector should be grouped into another training cluster

instead of the one it previously belongs to. If there is any chance of cluster grouping for any training vector, go to Step 2 with CVs as new IVs. If there is no cluster change, the process is terminated and the CVs are the codewords for codebook.

Using the LBG algorithm, we can ensure that the final average distance from each training vectors to their respectively final centroids is definitely equal or lower than the average distance at the beginning of the iteration process. Average distance at the stage of cluster set $R = \{r_1, r_2, \dots, r_k\}$ is defined as

$$D = \frac{1}{j} \sum_{a=1}^k \sum_{b=1}^{\text{card}(r_k)} d(t_b^k, c_i), \text{ where } t_b^k \in r_k.$$

In the following subsection, we use an example to show the working process of the LBG algorithm.

3.3.2 EXAMPLE TO ILLUSTRATE THE LBG ALGORITHM

In the following example, a set of training vectors and a set initial codewords in Table 3.1 and Table 3.2 respectively are used to illustrate the LBG algorithm graphically. The training vectors may shift from one cluster to another when the centroids are determined. To simplify the example, each vector is of only two-dimensions of a grayscale image.

Let the initial codewords act as the initial vectors (IVs). The initial stage in Figure 3.3 is determined, by calculating the distances between the training vectors and each IV. Each training vector will be assigned to the IV where the distance is the closest. Thus, training vector (44,41) is assigned to cluster 1 denoted by IV (45,50), training vectors (59,119), (60,110), (56,91) and (57,88) are assigned to cluster 2 denoted by IV (45, 117), training vectors (62,114) and (65,120) are assigned to cluster 3 denoted by IV (75, 117) and finally, the rest of the training

set is assigned to cluster 4 denoted by IV (80,180). At this stage, the average distance is 118.24.

Dimension 1	Dimension 2
72	180
72	175
65	120
44	41
59	119
62	114
64	150
60	110
65	162
56	91
57	88
70	172

Table 3.1 Training set for designing codebook

Dimension 1	Dimension 2
45	50
45	117
75	117
80	180

Table 3.2 Initial set of codewords for codebook design

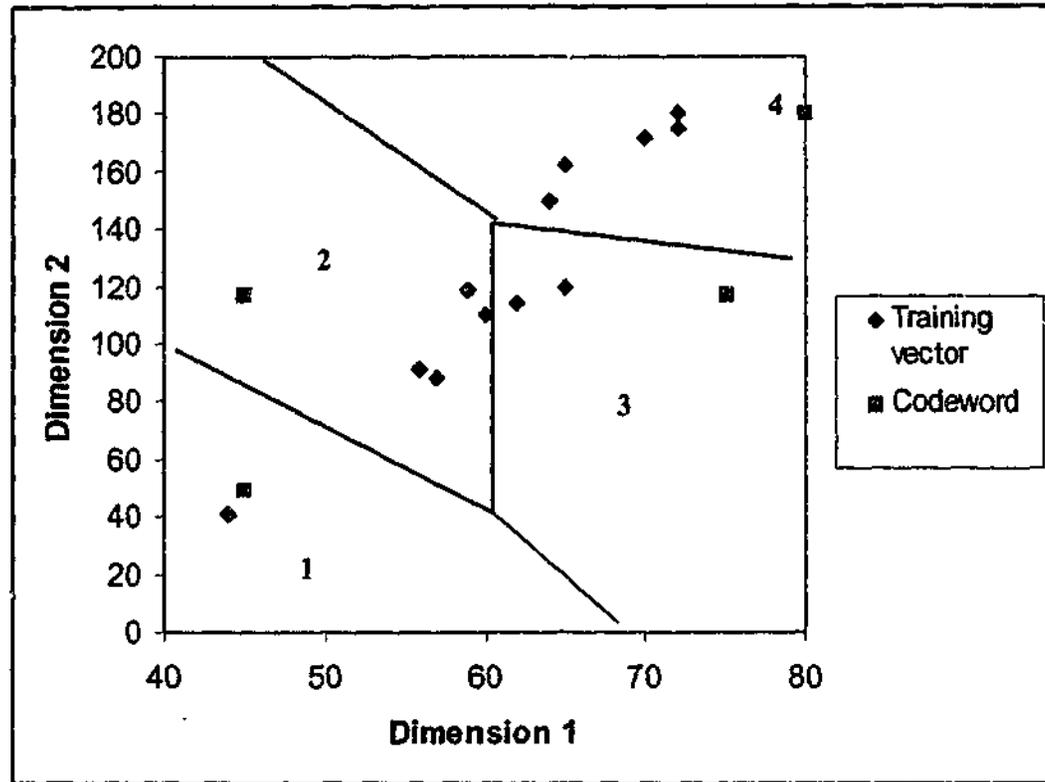


Figure 3.3 Initial state of the codebook

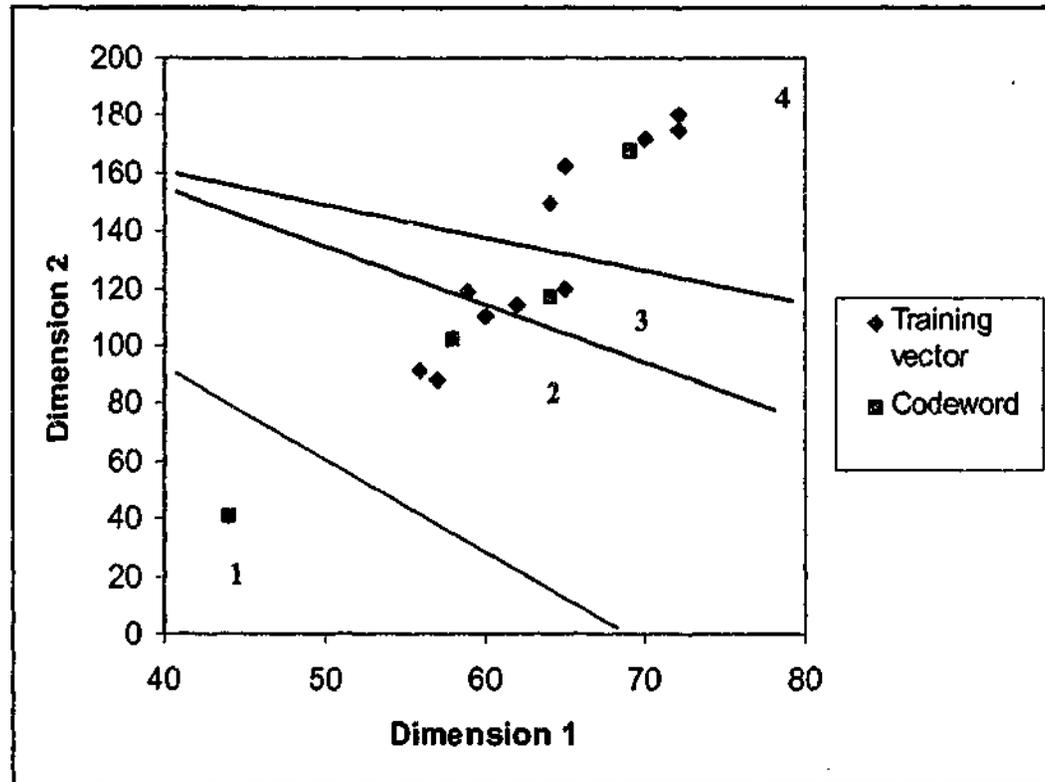


Figure 3.4 The codebook after one iteration

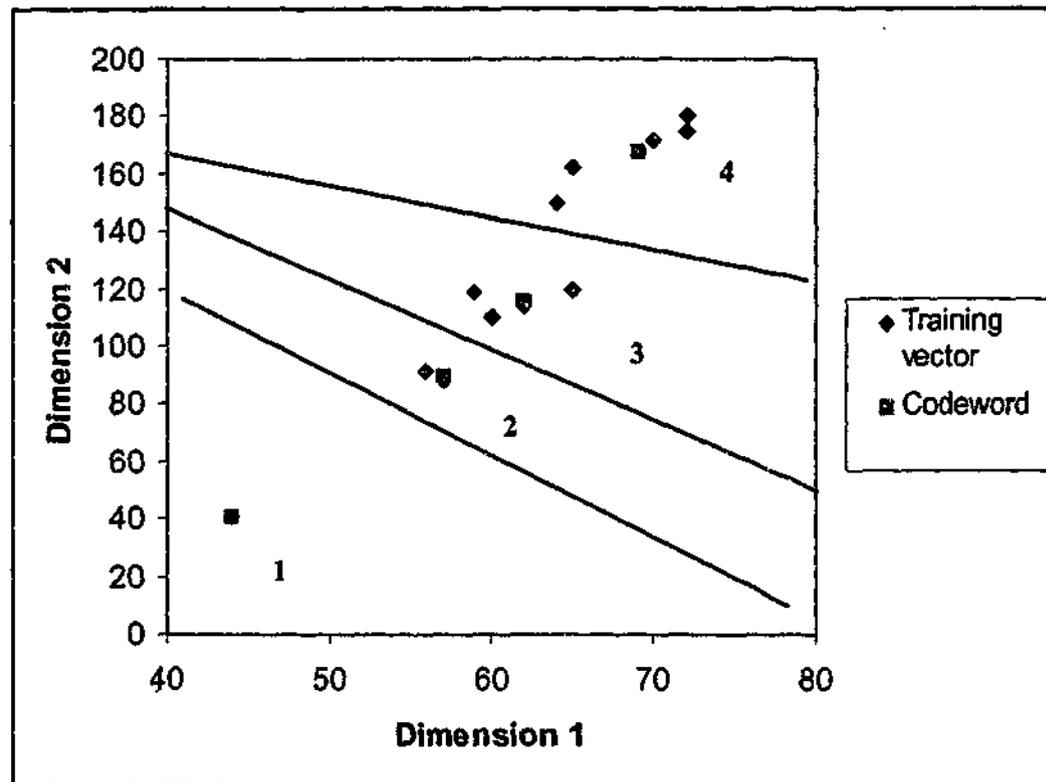


Figure 3.5 Final state of codebook

In Figure 3.4, the centroid of each cluster is determined by taking the average of all the training vectors that exist in the cluster. The centroid vectors (CVs) worked out to be (44,41), (58,102), (64,117) and (69,168) for clusters 1, 2, 3 and 4 respectively (Note that the figures in this example are rounded off to the nearest integer for simplicity). At this stage, the average distance is 82.02, which is much lower than the previous stage. Since the clusters are now represented by the newly determined CVs, distance between each training vector and each CV is recalculated to determine if there is a shift of cluster by any of the training vectors. In this example, since training vectors (59,119) and (60,110) are now closer to the CV of cluster 3 than the CV of cluster 2, they are now assigned to cluster 3 from cluster 2. Clusters 1 and 4 are not affected in this example.

Finally, in Figure 3.5, the centroids of the affected clusters are redetermined. They worked out to be (57,90) and (62, 116) for clusters 2 and 3 respectively. The centroids of clusters 1 and 4 remain the same. The average distance at this stage has further been reduced to 59.77. This average distance is the lowest among all previous stages, which is what LBG algorithm is trying to achieve. Since now all

the training vectors are closest to their respective CVs, the LBG algorithm has successfully determined the 4 codewords for the codebook.

3.3.3 THE SPLITTING TECHNIQUE

In the LBG algorithm, the selection of the initial codewords for clustering is crucial. This is because the algorithm itself can only guarantee that the average distance of the training vectors to the centroid in each cluster will not increase through each iteration. However, it cannot guarantee that the procedure will converge to the global optimal solution. With different initial codewords, the final codewords generated will be different [29, 41, 75]. This is because the choice of the initial codewords will determine how the training vectors will be clustered. Thus, the technique used to select the initial codewords is very important to achieve a good codebook.

One of the best techniques used for such purpose in LBG algorithm is called the splitting technique [29, 41, 75]. In this technique, the codebook is built progressively with the training vectors from the lowest level till one of the following two termination conditions (1. when the average distance is below a chosen threshold; 2. when we obtain the codebook size set.) is attained. The lowest level is a codebook with one codeword. The codebook size of each level is double the codebook size of the previous level. The initial codewords of each level of codebook consist of the codewords in the final codebook of the previous level together with the set of vectors generated by splitting each codeword in the final codebook of the previous level. The final codebook of each level consists of the final centroids of all the clusters after the LBG algorithm iteration.

To obtain the lowest level codebook which consists of one codeword, we will treat all the training vectors to be in the same cluster. Thus the first initial codeword is obtained by getting the average of all the training vectors. Since there is only one cluster, the initial output vector is also the final centroid for the cluster. Hence, the codebook of size 1 is generated.

To obtain the initial codebook of the next size, each codeword in the final codebook of the previous level is split. This is achieved by adding a randomly generated perturbation vector β . β can be different for each element in the codebook. For example, let the perturbation vector β randomly generated be (10, 20) if the codeword is of two-dimensions. The new initial codeword is (72, 62) if codeword used from the codebook is (62, 42). The original codewords together with the set of vectors generated form the initial codebook for the next level. The purpose of keeping the final codebook of the previous level as part of the initial codeword for the next size is to guarantee that the codebook after splitting will be at least as good as the codebook prior to splitting.

The following are the iteration steps of the LBG algorithm using the splitting technique:

Initialisation:

1. Decide the size of the codeword which is normally a square block of $n \times n$ pixels. Divide the training image into non-overlapping training vectors whose block-size is the same as the codeword block-size.
2. A vector which is the average of all the training vectors is calculated. This vector, which is the centroid of all the training vectors, is the codeword of the codebook of size 1.
3. The splitting technique is used to produce two vectors from each codeword to act as the initial vectors (IVs) for generating the codebook of the next level. Hence, if the present codebook size is k , the codebook size of the next level is $2k$. Splitting is done by adding a randomly generated perturbation vector to generate a new vector (NV). The perturbation vector can be different for each pixel of the codeword. So IVs of the next level consist of codewords of the current level and NVs.

Clustering:

4. Find the closest IV for each training vector based on Euclidean distance

between each training vector and each of IVs

5. Training vectors closest to the same IV are classified into the same group or cluster. So we have one cluster corresponding to each IVs.
6. If a cluster does not have any training vector, the corresponding IV is replaced by a randomly chosen training vector from the cluster with the largest number of training vectors and new clusters have to be formulated again as in steps 4 and 5. Otherwise, continue.
7. Determine the centroid vector for each cluster. The centroid vector (CV) replaces the IV of the cluster to represent the cluster.

Termination Checking:

8. Recalculate the distance between each training vector and each CV to determine if any training vector should be grouped into another training cluster instead of the one it previously belongs to. If there is any chance of cluster grouping for any training vector, go to Step 4 with CVs as new IVs. If there is no cluster change, the CVs are the codewords for codebook at this level.
9. If a termination condition for codebook generation is met, then the codebook generation process stops and the above codewords are used as final codebook. Otherwise, go to Step 3. There are two stoppage conditions. Condition one is met when number of codewords reaches the predetermined number. Condition two is met when the average distance between each training vector and its corresponding codeword is below certain threshold.

3.3.4 EXAMPLE TO ILLUSTRATE THE FULL CODEBOOK GENERATION PROCESS

To illustrate the full codebook generation process of generating a codebook with four codewords, we again use the training vectors from Table 3.1. However,

instead of using the chosen four codewords in Table 3.2, we will start generating from one codeword. The progression of the codebook using splitting is also shown in Table 3.3.

Codebook size	Dimension 1	Dimension 2
One	62	127
Initial - two	62	127
	72	137
Final - two	58	98
	69	168
Initial - four	58	98
	68	108
	69	168
	79	178
Final - four	52	73
	62	116
	65	156
	71	176

Table 3.3 Progression of Codebooks Using Splitting

To get the codebook size of 1, we will get the average of all the training vectors and the codeword is (62, 127). This is shown in Figure 3.6.

cluster, centroid (69,168) and (58,98) will be the final codewords in the codebook size 2.

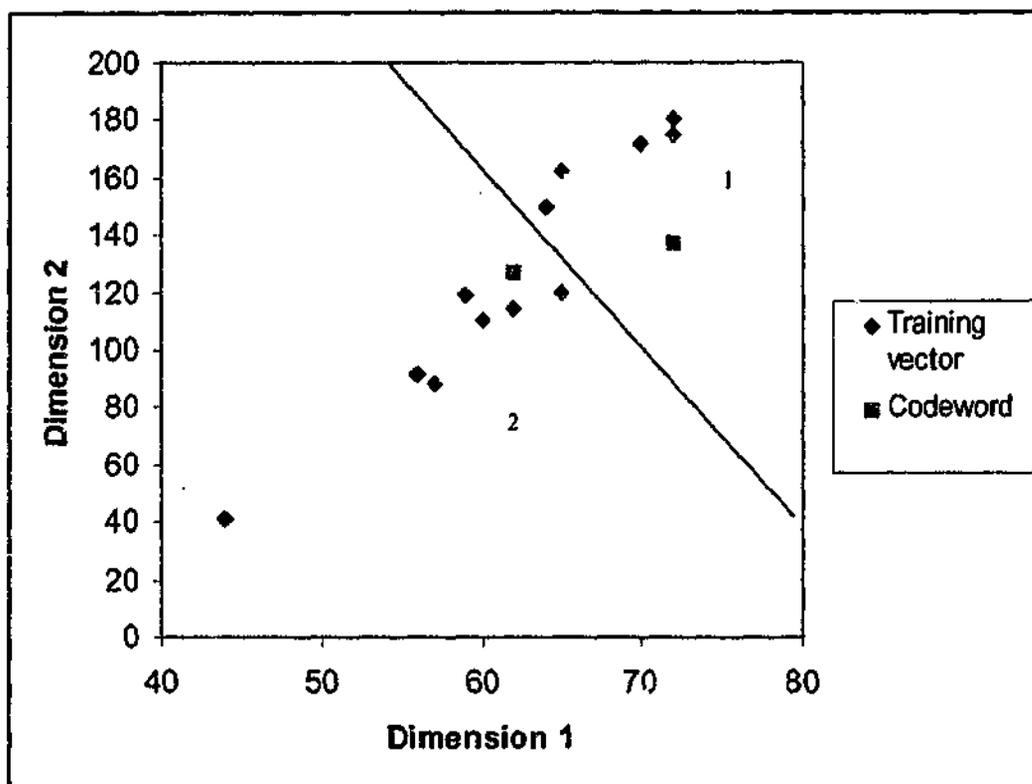


Figure 3.7 Initial Codebook of size 2

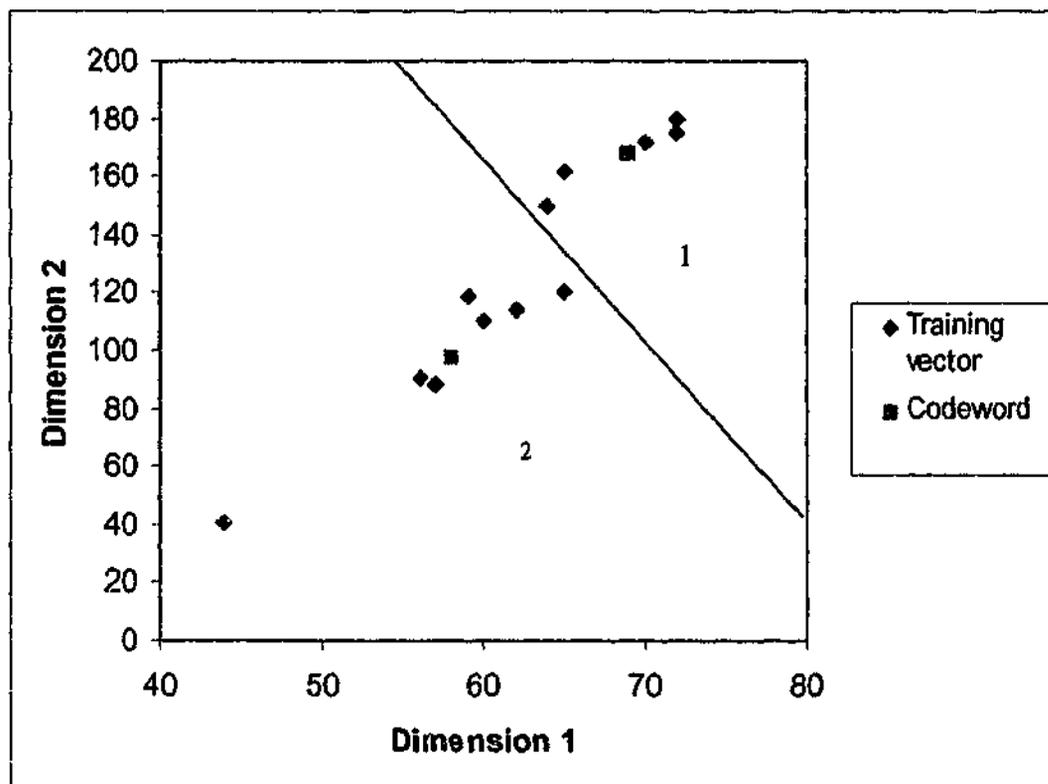


Figure 3.8 Final Codebook of size 2

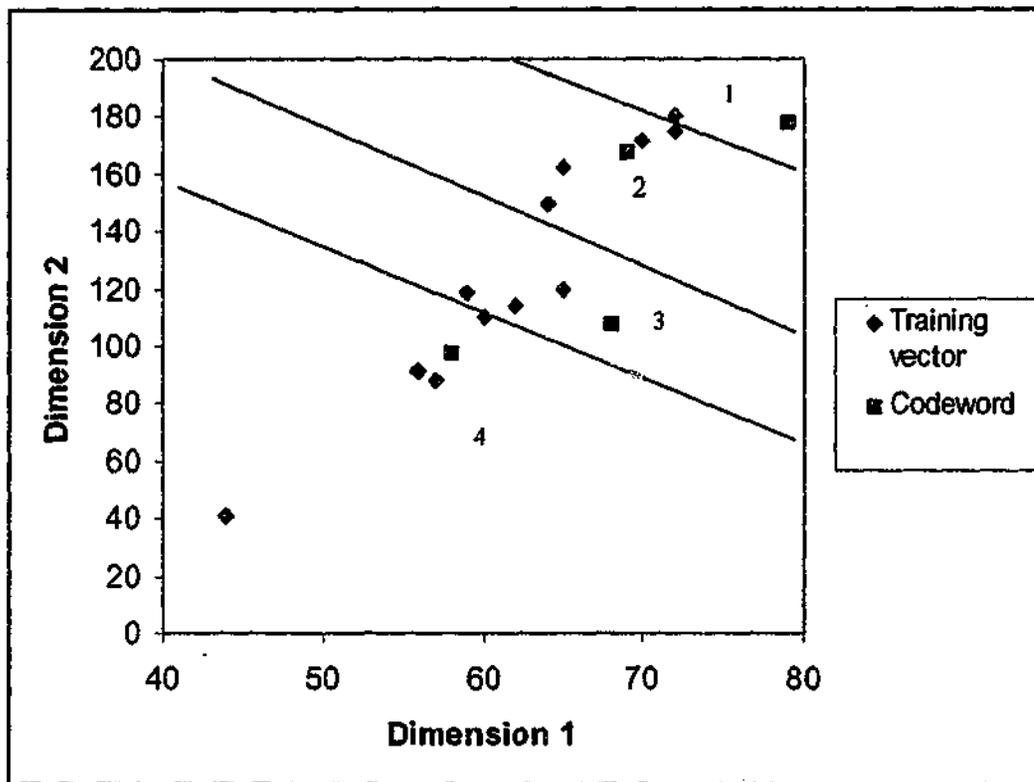


Figure 3.9 Initial Codebook of size 4

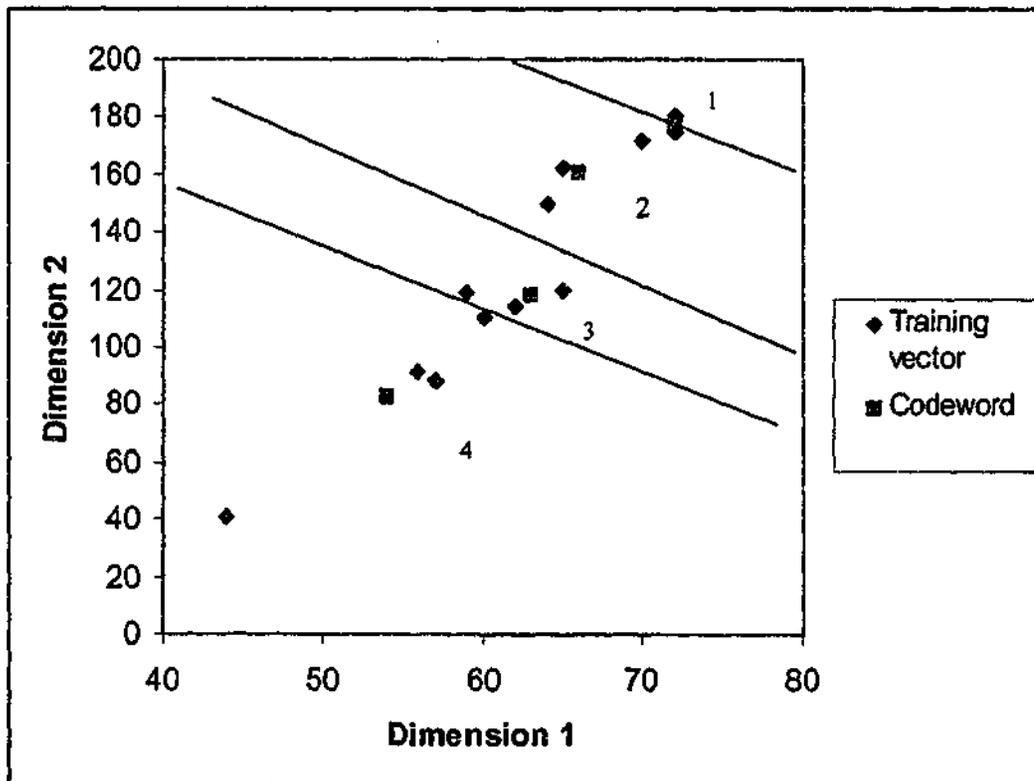


Figure 3.10 First Set of Centroids Determined for Codebook of size 4

To generate the codebook of size 4, the final codewords in the codebook size 2 are used to generate the four initial codewords. They worked out to be (58,98), (68,108), (69, 168) and (79,178). This is shown in Figure 3.9. Again, let the 4 initial codewords act as IVs. Next, by calculating the distances between the training vectors and IVs, the training vectors are grouped into four clusters. Training vector (72,180) belongs to cluster 1 defined by IV (79,178). Training vectors (70,172), (72,175), (65,162) and (64,150) belong to cluster 2 defined by IV (69,168). Training vectors (65,120), (59,119) and (62,114) belong to cluster 3 defined by IV (68,108). Training vectors (60,110), (56,91), (57,88) and (44,41) belong to cluster 4 defined by IV (58,98). The CVs here worked out to be (72,178), (66,161), (63,118) and (54,83) for clusters 1, 2, 3 and 4 respectively. This is shown in Figure 3.10. These determined CVs replace the IVs to define their respective clusters.

Next, we will check if any training vector has shifted to another cluster from its original cluster. We find that training vector (70, 175) has shifted from cluster 2 to 1 and training vector (60,110) has shifted from cluster 4 to 3. The rest of the training vectors remain unchanged. Now, by regrouping the training vectors to their new clusters, the CVs for the clusters are redetermined. The new CVs worked out to be (71,176), (65, 156), (62,116) and (52,73) for cluster 1, 2, 3 and 4 respectively. This is shown in Figure 3.11. The newly determined CVs replace the old centroids to define their respective clusters. Next, we will again check to see if there is a shift of cluster by any training vector. As there is no shift of cluster, centroids (71,176), (65,156), (62,116) and (52,73) are the final codewords in codebook of size 4. Since the codebook size required is 4, the codebook generation process stops.

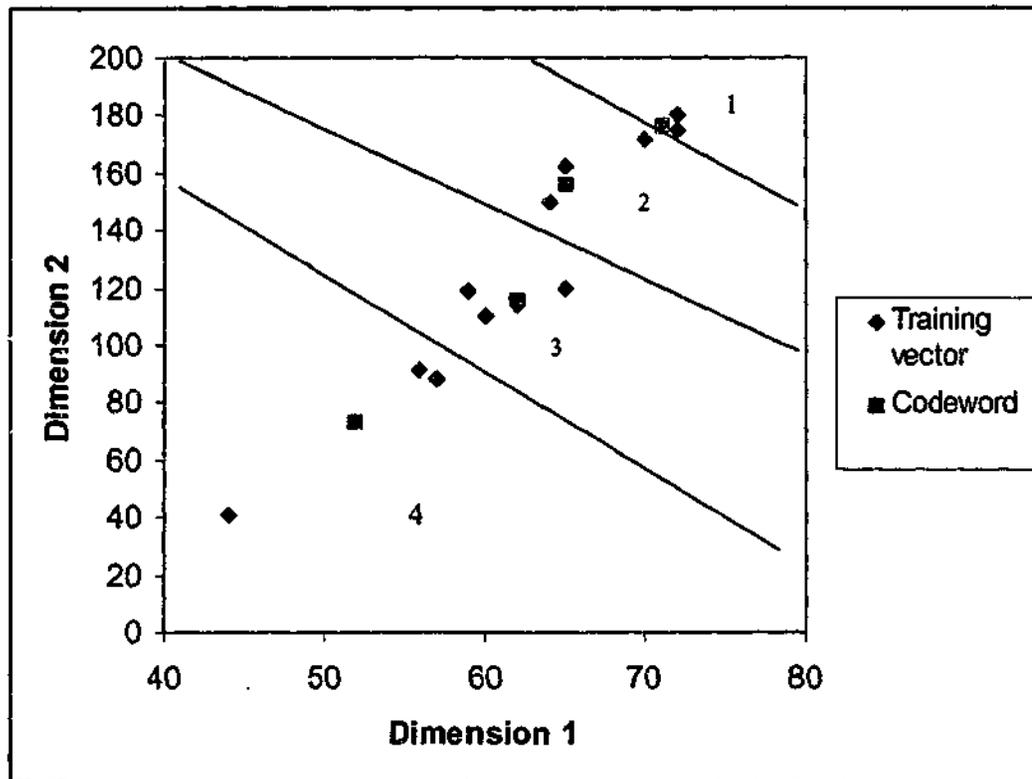


Figure 3.11 Final Codebook of size 4

3.4 SUMMARY

In this chapter, we described several concepts and techniques of vector quantization when it is used for image compression. We described that using VQ, images are compressed by firstly dividing the pixels in images into standard blocks (called vectors) and matching them to the codewords that are in a VQ codebook. Each image vector is then represented by the index of the best-matching codeword. Since the number of bits required to store each index is much smaller compared with the bits required for each image vectors, compression is achieved. As the codebook generation process is an important process in VQ, we have also described in detail the concepts and algorithm for this process. These are the fundamental knowledge necessary for Chapter 4 where we describe how VQ is used in our proposed CBIR technique.

CHAPTER 4

USING VECTOR QUANTIZATION FOR IMAGE INDEXING AND RETRIEVAL

4.1 INTRODUCTION

One of the main aims of our research is to derive a more effective colour-based technique for image indexing and retrieval. In this chapter, we describe the proposed image indexing and retrieval scheme.

The structure of this chapter is as follows. In Section 4.2, we will first look at some properties of VQ which make it a good candidate for image indexing and retrieval. In Section 4.3 we propose the VQ-based image indexing and retrieval technique. In Section 4.4, we briefly describe an independent work which is related to our proposed technique. In Section 4.5, we discuss some implementation issues in the proposed technique. One issue which we will look into is the areas in the traditional VQ codebook generation process where modifications are needed to make the proposed scheme effective. In this section, we will also discuss how the choice of different codebook sizes, codeword block-sizes and colour spaces may affect the retrieval performance of the proposed technique. In Section 4.6, we discuss how the VQ-based technique can overcome

limitations of the current colour-based image retrieval techniques described in Chapter 2. Finally, Section 4.7 summarizes the chapter.

4.2 VQ - A GOOD CANDIDATE FOR IMAGE INDEXING AND RETRIEVAL

A good colour-based image indexing and retrieval technique should firstly be able to retrieve database images with similar colour to that of the query image. Secondly and more importantly, at least to certain extent, the database image retrieved should have similar semantic content to the query image. The following are some VQ properties that make VQ a suitable candidate to be used for image indexing and retrieval:

1. As a compression technique, VQ reduces the number of possible types of pixel-blocks in the compressed database images while maintaining them at an acceptable image quality. The compression process thus allows each set of perceptually similar colours and pixel-blocks in the database images to be represented by a common codeword. Thus if similarity comparison is done using the VQ compressed data, the first capability stated above is achieved.
2. The codebook can be generated automatically using the clustering algorithms. This property is important since it is virtually impossible to manually go through the huge number of images in the database and select the set of image vectors that are suitable for the codebook.
3. The list of codewords generated using the clustering algorithm is the set of image vectors that are representative of the variety of image vectors found in the database images. By using this common set of codewords for encoding and indexing the database images, we can be assured that the feature built is representative of each image.

4. In every image, it is the way the colour pixels are positioned and grouped that produces the objects in the images. By looking at the objects in the images, we can perceive the semantic meaning in the image. Since pixel-blocks rather than individual pixels are used in VQ, some information on the spatial relationship among the neighbouring pixels in the images are retained. Thus, if indexing of the images is based on the VQ compressed data, the second capability stated above can be achieved.

To illustrate how we can take advantage of the VQ properties discussed, we will next describe the proposed technique.

4.3 IMAGE INDEXING AND RETRIEVAL TECHNIQUE BASED ON VQ

The VQ-based image indexing and retrieval technique we proposed is very similar in concept to the basic colour-based histogram image retrieval technique. Instead of quantising each of the three primary colour space channels (e.g. RGB) from the uncompressed image to build a histogram for indexing, this proposed technique uses the codeword indices in the VQ compressed image file to build a histogram for indexing. For a given image, the number of occurrences of each index is calculated to obtain an index histogram $H(v_1, v_2, \dots, v_i, \dots, v_n)$, where v_i is the number of times codeword i is used by the image, and n is the total number of codewords in the codebook. Since each index is unique for each codeword in the codebook and each block of pixels is represented by an index number, this histogram will be able to characterise the major features of the image. To normalise the histogram so that it is invariant to the image scale, the value in each histogram bin is divided by the total number of image blocks in the image. Thus the summation of the values in all bins in a normalised histogram is equal to 1.

During image retrieval, an index histogram $H(q_1, q_2, \dots, q_i, \dots, q_n)$ is calculated for the query image. Then the distance between the query image Q and the target image V is calculated as follows:

$$d(Q, V) = \sum_{i=1}^n |H(q_i) - H(v_i)|$$

Images can be ranked in an ascending order of calculated distances. The larger the calculated distance between the two images, the more different the two images. Finally, the list of top N target images with the smallest distance differences relative to the query image will be retrieved. The list of retrieval images is also ranked from the smallest to the largest, according to their distances.

To ensure effective performances in the proposed image indexing and retrieval scheme, some modifications must be made to some areas of the original codebook generation process. In the Section 4.5, we will highlight these areas and the modifications made.

4.4 RELATED WORK

When our VQ-based CBIR technique was proposed, Indris and Panchanathan have independently proposed another CBIR technique which is also based on VQ compressed data [35]. In their technique, a codeword usage map is used to index an image. The codeword usage map indicates which codewords in the codebook are used by an image. The distance between two images is calculated based on the difference between their usage maps. Compared with the technique we propose, their technique has lower retrieval precision. This is because the usage map does not show the frequency of each codeword being used to encode an image. As a result, the probability of two totally different images having the same usage map is a lot greater compared with the probability of these images having the same VQ histogram.

4.5 CODEBOOK GENERATION IN VQ USED FOR IMAGE RETRIEVAL

The VQ codebook generation process determines if good image compression results can be achieved. The compression ratio is determined by the codebook size. Larger codebook size produces smaller compression ratio and vice versa. When an image is being compressed, each image block is matched with the most similar codeword in the codebook. During decompression, the image is reconstructed using the codewords. Thus the content of codebook is very important. With a larger codebook size, the chances for an image block to find a better match codeword will be greater. Thus, there must be a compromise between the image quality and the compression ratio. These are the main factors to be considered when the codebook is mainly used for compression. However, for image retrieval based on VQ, we must also ensure that the codebook allows the histograms, which are built from the codeword indices, to reflect the similarity between the images. This means that similar images must have similar histograms and vice versa. Before we discuss about the above factors, we will look at the choice of the training images.

4.5.1 CHOICE OF TRAINING IMAGES

The selection of training images for codebook generation for the purpose of image indexing and retrieval is similar to that for compression. Thus, the database administrator needs to group similar images into similar categories and pick one or more images from each category to make up the training image set. Another way of choosing the training image set is to randomly pick a suitable percentage of the database images and use them as the training images. It is important to note that the set of training images should be representative of the entire set of database images because only one codebook is generated for that database. The purpose of

having only one codebook for a database is to facilitate the comparison of image histograms during retrieval.

4.5.2 CODEBOOK STRUCTURE SUITABLE FOR IMAGE RETRIEVAL

In the traditional VQ image compression, the main aim is to achieve highest compression ratio while maintaining acceptable reconstructed image quality. This means the number of codewords generated must be kept at the minimum while they still capture all the main features of the images to be compressed. This is achieved traditionally by generating a codebook separately for each of the red, green and blue colour channels using the training image if we are working in RGB colour space. Each codebook consists of the codewords which best represent that particular colour channel. To compress an image, the image vectors in their red, green and blue colour channels are separately compared with their respective codebooks to find their best match codewords. In this way, if each codebook size is 256, a total of 2^{24} different image blocks can be represented by the combination of the three codebooks. If each codeword is of 16 pixels, the total number of bytes to store the three codebooks is 12288 bytes. Though this scheme allows high compression ratio to be achieved, it is not suitable for image indexing and retrieval. If three codebooks are used to index an image, one histogram is built for each colour channel. Thus, three histograms are needed to index an image. This causes problems in comparing similarity between images [91]. For image retrieval based on VQ to be successful, two conditions must be true:

1. Perceptually similar images must have similar histograms.
2. Images with similar histograms should reflect that the images are more likely to be perceptually similar.

However when the codebook is generated in the traditional technique, these conditions may not be met. Here, although perceptually similar images have similar histograms, similar histograms may not reflect that the images are

perceptually similar. An example of such images is shown in Figure 4.1.

Image vector

Image A		Image B	
0,0,0	1,1,1	0,1,2	1,0,3
2,2,2	3,3,3	2,3,0	3,2,1

Figure 4.1 Two perceptually different images having same histograms. The three numbers in each image vector represent the codeword indices of the red-channel codebook, green-channel codebook and blue-channel codebook respectively.

Image A and image B in Figure 4.1 are each made up of four image vectors. If the traditional codebook generation technique is used, three codebooks will be generated. If the colour space used is RGB, a codebook is generated for each of the red channel, green channel and blue channel of images. To encode images A and B using our proposed technique, each colour channel of the images will be encoded using the codebook of the corresponding colour channel. As a result, each image vector is represented by three indices, where each index represents the best match codeword in their respective colour-channel codebook. For an example, as shown in Figure 4.1, the top-left image vector of image B is represented by codeword index 0 of the red-channel codebook, codeword index 1 of the green-channel codebook and codeword index 2 of the blue-channel codebook. Since each image vector is encoded by three indices of three different codebooks, three histograms are built to index each image. In Figure 4.1, only 1 image vector in image A is encoded by codeword index 0 of the red-channel codebook. The same number of image vector in image A is also encoded codeword index 1, 2 and 3 respectively. Therefore, the red-channel histogram of images A is $H_R(1,1,1,1)$. The green and blue channel histograms of image A are $H_G(1,1,1,1)$ and $H_B(1,1,1,1)$ respectively. As for image B, its red, green and blue channel histograms are $H_R(1,1,1,1)$, $H_G(1,1,1,1)$ and $H_B(1,1,1,1)$ respectively. By

comparing their histograms, we would think that the two images look alike. However, they actually look different perceptually. This is because the occurrences in the bins of the histograms may be contributed from different image blocks of the image. As the colour channels in the two images are combined together differently, they are different perceptually.

To facilitate the similarity comparison between images, we propose to generate one codebook instead of three codebooks of the traditional technique. This means each codeword represents a block of image pixels which can be found perceptually from the training images. Each codeword needs 24-bits to represent, instead of 8-bits. Only one histogram is used to index an image in this way. Thus each bin in the histogram counts the occurrences of each codeword. The distance between the histograms of two images shows the total number of different physical image blocks of pixels (codewords) between the images. Perceptual similarity between the images is reflected in similarity of histograms.

However, in this proposed technique, to achieve similar image quality as the traditional technique, more codebook storage space will be needed. For example, to achieve image quality of the traditional technique in above example, 2^{24} codewords must be generated and the storage space needed is 49152 Kbytes, instead of the 12288 bytes of the traditional technique. That is 4096 times greater. So, since having a codebook that needs such a huge amount of storage space is not feasible, we propose to use a codebook size which is enough to capture the main features of the database images. This is feasible because for image retrieval, we are retrieving relevant images from the database, thus an exact match in the features of the retrieved images to the query image is not needed. As long as the database images have similar features to the query image, they should be retrieved. Experimental results in Chapter 6 will be used to show that the proposed technique is acceptable in both image quality and retrieval effectiveness.

4.5.3 CODEBOOK GENERATION ALGORITHM SUITABLE FOR IMAGE RETRIEVAL

To generate a codebook with the suitable structure for image indexing and retrieval, modifications are made to the traditional LBG algorithm described in the previous chapter. In order to build a codebook where each codeword represents a block of image pixels, the values of all the three colour channels are used for each codeword dimension. As a result, the Euclidean distance metric has to be modified to accommodate the values of all the three colour channels, rather than just one colour channel. The modified Euclidean distance metric is shown in Step 4 of the following modified LBG algorithm:

1. Decide the size of the codeword which is normally a square block of $n \times n$ pixels. Each pixel of the codeword has 24 bits for colour images. Divide the training images into non-overlapping training vectors whose size is the same as the codeword size decided.
2. A vector which is the average of all the training vectors is calculated. This vector, which is the centroid of all the training vectors, is the codeword of the codebook of size 1.
3. The splitting technique is used to produce two vectors from each codeword to act as the initial vectors (IVs) for generating the codebook of the next level. Hence, if the present codebook size is k , the codebook size of the next level is $2k$. Splitting is done by adding a randomly generated perturbation vector to generate a new vector (NV). The perturbation vector can be different for each pixel of the codeword. So IVs of the next level consist of codewords of the current level and NVs. Note that each codeword pixel has 24 bits corresponding to red, green and blue channels of 8 bits each. So when generating the perturbation vector, "randomness" should be included for each of the colour channel.

4. Find the closest IV for each training vector based on Euclidean distance between each training vector and each of IVs, as follows:

$$d(X, Y) = \sum_{i=0}^{H-1} \sqrt{(X_i^R - Y_i^R)^2 + (X_i^G - Y_i^G)^2 + (X_i^B - Y_i^B)^2}, \text{ where } H = nxn \text{ is the}$$

codeword block-size, X is a training vector and Y is an IV, superscripts R, G and B denote the red, green and blue channels of each pixel in the training vector and IV.

5. Training vectors closest to the same IV are classified into the same group or cluster. So we have one cluster corresponding to each IVs.
6. If a cluster does not have any training vector, the corresponding IV is replaced by a randomly chosen training vector from the cluster with the largest number of training vectors and new clusters have to be formulated again as in steps 4 and 5.
7. Determine the centroid vector for each cluster. The centroid vector (CV) replaces the IV of the cluster to represent the cluster.
8. Recalculate the distance between each training vector and each CV to determine if any training vector should be grouped into another training cluster instead of the one it previously belongs to. If there is any chance of cluster grouping for any training vector, go to Step 4 with CVs as new IVs. If there is no cluster change, the CVs are the codewords for the codebook at this level.
9. If a termination condition for codebook generation is met, then the codebook generation process stops and the above codewords are used as final codebook. Otherwise, go to Step 3. There are two termination conditions that we can use to stop this iteration process. Condition one is met when number of codewords reaches the predetermined number. Condition two is met when the average distance between each training vector and its corresponding codeword is below certain threshold.

4.5.4 EFFECTS OF CODEBOOK SIZES, CODEWORD BLOCK-SIZES AND COLOUR SPACES ON RETRIEVAL PERFORMANCE OF THE PROPOSED TECHNIQUE

To effectively implement the proposed technique, there is a need to determine the appropriate codeword block-size, codebook size and colour space. With different codeword block-sizes, codebook sizes and colour spaces, there can be different potential effects on the retrieval performance of the proposed technique as follows:

- **Potential Effects of Different Codebook Sizes on Retrieval performance**

The codebook size determines the distortion rate for the compressed image using VQ. A larger codebook contains more distinct codewords, which increases the possibility for each image block of finding a better match codeword during encoding. Thus, VQ compressed images that use larger codebooks are more likely to have a lower distortion rate. Since the histogram built to index each image is based on its VQ compressed image data, less distortion rate should allow more accurate representation of the images. Therefore, theoretically, larger codebook size should lead to more accurate retrieval. However, using a codebook size that is too large can also pose a few issues. Firstly, larger codebook size results in more number of bins in each histogram. The increase in the histogram bins leads to the increase in similarity comparison time [44]. Thus, larger codebooks decrease the efficiency of the proposed technique. For efficiency purpose, the codebook size should not be greater than 4096. Secondly, the use of very large codebook size may not improve the retrieval effectiveness. This is because the purpose of a CBIR technique is not only to retrieve database images that are exactly the same as the query image, but it should also retrieve images that are similar to the query. By using codebook size that is very large, very fine details

may be captured in the image histograms. Such details may cause the proposed technique to be too discriminative during the similarity comparison, resulting in similar database images not retrieved. Thirdly, the increase in the retrieval effectiveness may not always be proportional to the increase in the codebook size. This is because the codebook sizes within the efficiency range (4096 and smaller) are much smaller compared with the number of the distinct original image vectors. Therefore the possibility of having a same set of original image vectors being indexed by a codeword in smaller size codebook and another codeword in the larger size codebook is rather high. Thus, unless there is a very large difference in the codebook size, there may not be a great difference in their retrieval effectiveness. For these reasons, it is important to select a suitable codebook size.

- **Potential Effects of Different Codeword Block-sizes on Retrieval Performance**

Larger block-size of the codeword allows more information on the spatial relationship among the image neighbouring pixels to be captured when indexing. However, to maintain a constant distortion rate in VQ, the size of a codebook with larger codeword block-size is greater than the size of a codebook with smaller codeword block-size. This is because the possible combinations of colour pixels in a block increase exponentially as the codeword block-size increases. For example, if each pixel consists of 256 colour levels, it is possible to have 256^n combinations of codewords (n is the number of pixels in each codeword). Since it is important to keep the codebook at a suitable size for the proposed technique to be both effective and efficient, the codeword block-size must be kept to a size which good amount of information on the spatial relationship among the image neighbouring pixels are captured while the distortion rate of the compressed images are kept at a suitable level.

- **Potential Effects of Different Colour Spaces on Retrieval Performance**

As the proposed technique mainly uses the image colour information for indexing and retrieval, the effects of different colour spaces on its retrieval performance are investigated next. A colour space is a three dimensional space which defines how colours are represented. There are many colour spaces, which are designed with different sets of characteristics so that they can be effectively used in their respective areas.

Many of these colour spaces have been used for colour-based image retrieval. However, it is still not clear which colour space is best suited for such applications as there has been no comprehensive testing being carried out using all of these colour spaces on a common large image database. Thus to investigate the effects of different colour spaces on the retrieval performance of our proposed technique, we select three commonly used colour spaces with very different characteristics. The three colour spaces are RGB, LUV and HSV.

RGB colour space is one of the most commonly used colour space in image processing. The three colour channels in this colour space are Red, Green and Blue. Many devices like colour cameras, scanners and displays are provided with RGB signal input or output. Many image formats, like Windows Bitmap, also store pixels colour data of images in based on this colour space. Image colour data in this colour space therefore can be easily obtained without any colour space conversion. If RGB is used in image retrieval technique, the ease of obtaining the colour data is an advantage as it allows more efficient processing. However, there are some characteristics of RGB that makes image retrieval relatively less effective compared with some other colour spaces. Among these characteristics, the most critical one is that this colour space is not perceptually uniform. Thus distance calculated between colours in this colour space cannot accurately evaluate the perceived differences between them.

The next colour space we will describe is LUV. LUV is a colour space which is commonly used for industries considering additive mixing such as colour displays,

TV and lighting (where light is emitted). In this colour space, L represents luminance, U represents colours approximately red-green and V represents colours approximately yellow-blue. It is also commonly used in image retrieval because it is a uniform colour space, which allows distance calculation between the colours to be more accurate.

Finally, we will describe HSV. HSV is a colour space which is very prominent in computer graphics literature. The three colour channels of HSV are hue, saturation and value. Hue describes the colours in circular spectrum from red to green to blue and back to red. Saturation describes the vividness or the pureness of the colour and value describes the luminance of the colour. The main characteristic of this colour space is it has good compatibility with human intuition.

To determine the actual effects of the different codebook sizes, codeword block-sizes and colour spaces on the retrieval effectiveness of the proposed technique, experiments are carried out and the experimental results will be presented in Chapter 6.

4.6 LIMITATIONS OVERCOME BY THE PROPOSED VQ-BASED INDEXING AND RETRIEVAL TECHNIQUE

In the final section of this chapter, we will examine in which ways the proposed technique are potentially more effective compared with the five of the colour-based techniques we have discussed in Chapter 2. The colour-based technique using colour-moment is not discussed here, as it is relatively less effective compared with the rest of the techniques.

4.6.1 BASIC AND QBIC COLOUR-BASED HISTOGRAM TECHNIQUES

One of the limitations in the basic colour-based histogram technique and the QBIC colour-based histogram technique is that they ignore the spatial relationships

among pixels during similarity comparison and retrieval. In the proposed technique, the histogram generated to index the image represents frequency of each codeword used to encode the image vectors in the image. Therefore, the distance between two image histograms represents the total number of image vectors in the two images that are encoded by different codewords. This distance is a stronger indication of how similar the two images are, compared with distance representing the number of scalar units, eg. pixels, which are not common in the images. This is because more information about the image is captured in vectors compared with pixels. To understand this, let us review what a vector is. A vector is actually a block of pixels. It not only contains information about the pixels' colour, but also contains information on the colour pattern/combination of the pixels in the block. This means that the vectors are able to capture the spatial relationships among the pixels in them. In pixel-based histogram, information on the spatial relationships among pixels cannot be captured. Since distance of the proposed technique is calculated using vectors, its indexing and retrieval performance will be more effective, compared with the two colour-based histogram techniques. To illustrate what we have just discussed, let us look at an example which compares the proposed technique with the two colour-based histogram techniques. The proposed technique will be able to pick up the difference between block  and block  , but the colour-based histogram techniques will not be able to do so. This is because both blocks contain equal numbers of blue and red pixels, but the pattern of the pixels is different in each. The proposed technique is able to pick up the pattern difference but the colour-based histogram techniques will not be able to do so. In this way, the proposed technique is more effective as it can take into consideration both the pattern and colour distribution differences when comparing the similarity of two images.

The next limitation that the proposed technique has improved on relates to quantization of each colour channel into a number of intervals. In Chapter 2, we have explained that colours which are perceptually similar may be quantised into

different histogram bins, resulting in the colour-based histogram techniques to be less effective. For our proposed technique, perceptually similar image vectors are less likely to be quantised into different histogram bins because:

- The quantization of vectors, compared with the quantization of the colours, is coarser. For example, in RGB colour space of 24 bits colour depth, the number of possible colours for a pixel is 2^{24} . These colours are normally quantised into 512 groups of colour, which are represented by 512 histogram bins [58, 87]. However, for our proposed technique, if the image vector block-size is 4×4 , then there can be $(2^{24})^{4 \times 4}$ possible combinations for a vector. These combinations are normally quantised into 1024 distinct codewords, which are represented by 1024 histogram bins [91, 94].
- To determine which histogram bin an image vector belongs to, it is dependent on the distance of every pixel in the vector to the corresponding pixel in the codeword the bin represents. Compared with colour quantization, which the quantization is only dependent on one pixel, the quantization in our proposed technique is more robust.

4.6.2 QBIC COLOUR LAYOUT TECHNIQUE

In this colour layout technique, the image is divided into 64 large blocks and information on the dominant/average colours are kept based on the blocks. Thus, like the above colour histogram techniques, very little information about spatial relationship between pixels is kept. Besides, if the database contains images which are similar to the query image but are being rotated or shifted, this method might consider them as images that are different. This is because the dominant colours in the database images may not belong to the same block as the query image. The VQ scheme, however, by comparing relatively much smaller blocks between the images, would enable such database images to be retrieved. This is

because the number of blocks that are similar between the query image and database image are still high.

4.6.3 COLOUR COHERENT VECTOR

In Chapter 2, we have stated that although CCV attempts to capture information about spatial relationships among the pixels in the images, its effectiveness is limited. This technique generally works well in differentiating images with mostly uniform colour or mostly texture regions from other images that have similar colour compositions but the similar colour pixels are sparsely scattered. Otherwise, its retrieval performance is only as effective as the basic colour-based techniques [48]. The reason for this limitation of the technique is because CCV only captures information on the percentage of image pixels that belong to areas in the image that have colours belonging to the same colour histogram bins and those that do not. Information on how the pixels are actually arranged in the uniform coloured areas is not captured in CCV. Similarly, information on how colour pixels are scattered in the non-uniform coloured areas is also ignored in CCV. However, in the proposed VQ scheme, more information on spatial relationships among colour pixels in the image is captured. In our proposed technique, many variations of colour pixel arrangements are captured in the codewords. Since each image is indexed based on the indices of these codewords, each image histogram thus carries more details about the spatial relationships of the pixels in the image. In this way, the image histograms derived from the VQ scheme can better discriminate the dissimilarities in the images compared with CCV.

4.6.4 COLOUR AUTOCORRELOGRAM

In Section 2.3.5, we have stated that a limitation of colour autocorrelogram is that only spatial correlation between identical colours is captured in this feature. As this feature does not capture information on spatial correlation between non-identical colours in the image, the effectiveness of its retrieval performance is

likely to be lower than other features that do. In our proposed technique, information on spatial correlation between both identical and non-identical colours is captured. Such information is captured in the VQ-based histogram which is built by counting the number of image vectors being encoded by each codeword in the codebook. Since the codebook consists of codewords that are made up of pixels with similar colours, and also others that are made up of pixels with non-similar colours, various information on the colour arrangements in the image are captured in the histogram. As a result, compared with the autocorrelogram technique, our proposed technique is likely to be more effective in discriminating the perceptual differences in the images. Thus, our technique is likely to have better retrieval effectiveness.

The next limitation of autocorrelogram technique is that similar images of different scales have autocorrelograms that are very different, if no normalisation of the images size is carried out before building their autocorrelograms. However, our proposed technique does not encounter such a problem, as the size of the images does not need to be normalised before the indexing process. This is because the VQ-based histograms will be normalised. This will ensure that perceptually similar images, which are of different sizes, would have similar VQ-based histograms.

By overcoming such limitations which the colour autocorrelogram technique faces, our proposed technique is likely to have better retrieval performance.

4.7 SUMMARY

In this chapter, we presented a novel image indexing and retrieval technique which uses VQ. In this technique, a histogram is built based on the number of image vectors encoded by each codeword in the codebook. Spatial correlation of colours within each image block is captured in the histogram because:

- Each codeword is a block of pixels.

- The codewords consist of the ones that have pixels of uniform/similar colours and also the others that have colour pixels that are not uniform/similar.

We also described the modifications made to the original codebook generation process in order to facilitate its use in our proposed technique. These modifications are necessary for building a codebook whose structure is suitable for our proposed technique. The potential effects of various parameters, like different codebook sizes, codeword block-sizes and colour spaces, on the retrieval performance of the proposed technique are also discussed. By understanding their potential effects on the retrieval performance of the proposed technique, better combinations of these parameters can be chosen to achieve better retrieval performance. Finally, we have discussed how the VQ-based technique can overcome limitations of the current colour-based CBIR techniques described in Chapter 2. With the ability to overcome the limitations in the current techniques, the proposed technique is likely to have a better retrieval performance. Experimental results obtained by comparing the retrieval performance of the proposed technique to the current techniques will be presented in Chapter 6.

CHAPTER 5

REVIEW OF EFFICIENT SEARCH METHODS FOR ENCODING AND RETRIEVAL PROCESSES

5.1 INTRODUCTION

For an image indexing and retrieval technique to be useful, besides being effective in retrieving the set of relevant images for a query, it should also be efficient in completing the task. This is because even if the retrieval technique is capable of retrieving every relevant image in the database, it is unlikely that the user would be patient enough to wait a long time for a response to a query. So, what response time is considered reasonable for an interactive application? According to Nielsen's work on Usability Engineering [59], a response time of 0.1 second is about the limit for the user to feel that the system has instantaneous response. For the user's flow of thought to stay uninterrupted, the response time of 1.0 second is the limit. If the response time is more than 10 seconds, it will be difficult for the user to stay focused on the dialogue. Based on these findings, a CBIR system should attempt to have the response time as short as possible.

In the situation when a user supplies the query image, the processes that are involved in the proposed technique before a set of results can be presented to the user are encoding, indexing and finally retrieval. Encoding and retrieval are two very computationally intensive processes. Thus, for our proposed scheme to meet the efficiency criterion suggested above, we would need to make the encoding and retrieval processes more efficient.

The purpose of this chapter is to review some existing efficient multidimensional data search scheme. The study of their strengths and weaknesses allows us to design a search method for each process. The search method designed for each process should cater for its needs and also the characteristics of its data. The structure of the chapter is as follows. Section 5.2 further explains the efficiency issues in the proposed CBIR technique. In Section 5.3 to Section 5.7, general efficient search methods which may be used for both encoding and retrieval processes are described. Next, Section 5.8 describes some efficient search methods that are specific to the Euclidean distance. Finally, Section 5.9 summaries the chapter.

5.2 EFFICIENCY ISSUES OF THE PROPOSED TECHNIQUE

As stated above, the encoding and retrieval processes are very computationally intensive. In the encoding process, after the query image is segmented into the required image blocks, a search through the codebook is required to find the best match codeword for each image block. In the retrieval process, a search through the entire set of image indices (histograms) is required to find a set of images that is most similar to the query image. In both processes, the data involved can be represented as multidimensional vectors. For example, when the image block and codeword are of 4x4 pixels in the encoding process, they are represented as 48 (16 pixels block x 3 colour channels) dimensional vectors. In the retrieval process, a 1024-bins histogram is represented as a 1024 dimensional vector. The simplest way to carry out the searches is to linearly calculate the distance between the query

vector and each database vector to find the required vectors. However, this is very computationally intensive and time-consuming. To use this full-search (FS) method in our CBIR system is not practical, since the response time for a query will definitely not meet the user requirement. Thus, other search methods must be used to speed up the searches in the two processes.

In this chapter, we will review multidimensional data search methods which can be used to make the encoding and retrieval processes more efficient. Not all the search methods reviewed may be used for both processes. Although both processes involve multidimensional data search, the distance metrics used to measure the similarity between vectors are different. The encoding process uses the Euclidean distance metric whereas the retrieval process uses the Manhattan distance metric. The choice of distance metric is due to the following reasons:

Euclidean distance metric is a popular metric for VQ encoding, as it is effective for measuring colour similarity [26, 29, 66, 75]. An experiment in [66], where a list of commonly used distance metrics are compared on their effectiveness on measuring colour similarity, shows that Euclidean distance is very effective. However, a drawback of this metric is its computation speed. Euclidean distance is relatively slower than metrics like Manhattan distance metric due to its square and square root operations. However, in our application, the database images are encoded offline. The only situation where online encoding is required is when users introduce query images that are not in the database. Thus, the application is more tolerant in using a relatively slower metric for encoding. Another reason of why the application tolerates using a relatively slower metric is because the vector dimension of the image encoding process is much smaller than the vector dimension of the retrieval process.

Manhattan distance metric is used in the retrieval process mainly because of its computation efficiency. Since the similarity comparisons between the query image histogram and the database image histograms are carried out online, an efficient distance metric is crucial for our application. Besides, the experimental

results in Chapter 6 will show that the VQ histogram should have 1024 bins or above for effective retrieval in our proposed technique. Since the vector dimension in the retrieval process is high, an efficient distance metric is required. Besides being an efficient distance metric for retrieval, experimental results in [103] have also shown that Manhattan distance is also one of the more effective metric in measuring histogram similarity for image retrieval process.

5.3 NOTATIONS DEFINITION

To facilitate the discussion in this chapter, we will first define some notations.

Let Q and T be two subsets of R^k , where $Q = (X_1, X_2, \dots, X_a, \dots, X_M)$ and $T = (C_1, C_2, \dots, C_b, \dots, C_N)$. When used in the encoding process, subset Q is a set of query vectors, representing the image blocks and subset T is the entire set of target vectors (vectors in the search space) representing the codewords in the codebook.

When used in the retrieval process, subset Q is a set of query vectors, representing the query histograms and subset T is the entire set of target vectors representing the database image histograms. Distance metric $d(X_a, C_b)$ is used to measure the similarity (distance) between a query vector $X_a = [x_1, x_2, \dots, x_k]^t \in Q$ and a target vector $C_b = [c_1, c_2, \dots, c_k]^t \in T$. In the encoding process, $d(X_a, C_b)$ is the squared

Euclidean distance, $\sum_{i=1}^k (x_i - c_i)^2$. The squared Euclidean distance instead of the actual Euclidean distance of $[\sum_{i=1}^k (x_i - c_i)^2]^{1/2}$ is used for efficiency reason.

Comparing the two distance metrics, calculating distance using the squared Euclidean distance metric is more efficient since it does not require an extra square root operator. Besides, the distances calculated using any of the two metrics do not compromise the retrieval results since they do not affect the ranking

of the images. In the retrieval process, $d(X_a, C_b)$ is the Manhattan distance, $\sum_{i=1}^k |x_i - c_i|$.

5.4 PARTIAL DISTANCE SEARCH

Partial distance search (PDS) is a simple search method that is widely accepted to be more efficient compared with the traditional FS method [11]. The search result using PDS is also equivalent to that of the FS method.

In the FS method, to compare if a query vector X_a is more similar to a current target vector C_b than all the previous target vectors C_0 to C_{b-1} , the distance calculated between X_a and C_b is compared with the minimum distance calculated and stored between X_a and all the previous target vectors. In this search method, the distance between X_a and C_b is calculated by faithfully using the values in every dimension of the vectors without the consideration that the partial distance calculated may already be greater than the minimum distance stored even before all the dimensions are used for the distance calculation. In PDS, such situations are taken into consideration. In PDS, once the partial distance is greater than the current minimum distance stored, the distance calculation is aborted without calculating the full distance. To allow the distance calculation to be aborted before the full distance is performed, a comparison of the partial distance calculated and the current minimum distance is carried out after each summation of the distance metric ($(x_i - c_i)^2$ in the Euclidean distance metric or $|x_i - c_i|$ in the Manhattan distance metric). With the minor modification to FS method, PDS has the potential to reduce the average distance calculation time between the query vector and the target vectors. The search results using PDS are equivalent to the FS method since every target vector is also compared with the query vector in PDS.

Other versions of PDS are also proposed. Some of these versions attempt to attain better search efficiency by exploiting the variances in different dimensions of the target vectors [50, 78, 101]. One of the better techniques to obtain the variances is by using principal components analysis (PCA). To do this, we will first represent all the target vectors in a matrix. Let us call this matrix A . By using PCA, the singular value decomposition of matrix $A = UDV^T$ is derived. Each

singular value in diagonal matrix D represents the variance of a vector dimension along the principal direction in matrix A . Next, the vectors in the unitary matrix V are arranged in the order of decreasing singular values. Finally, by multiplying matrix V to X_a and C_b and calculating $d(VX_a, VC_b)$, we can increase the probability of calculating the dimensions with greater differences between the query and target vectors first. Thus, on average, the distance summations can be aborted more quickly compared with the original PDS.

Another variation to PDS is to rearrange the target vectors according to both their mean and variances [4, 67]. By using this information, the distances between the query vector and the target vectors that are more similar can be calculated first, allowing more abortion in the distance summation for other target vectors which are more different.

From the description above, we can see that PDS method is simple to implement. Depending on which variations of PDS are used, there may also be the advantage of no extra memory being needed for storing precalculated target vector information. However, the amount of search efficiency gained in this PDS is data dependent. High efficiency can be achieved in situations where the similarity of the target vectors highly varies. If all the target vectors are very similar, the possibility of the partial distance being greater than the current minimum distance long before we reach the last few dimensions of the vectors is small.

5.5 TRIANGLE INEQUALITY BASED SEARCH

Triangle inequality (TIE) is a geometric property that is commonly used in many multidimensional vectors search methods [12, 32, 40, 44]. The TIE states that the distance between two objects cannot be less than the difference in their distances to another object. Thus if $s \in R^k$ is a control vector, then

$$d(X_a, C_b) \geq |d(X_a, s) - d(C_b, s)| \quad (1)$$

By applying the geometric property of TIE, the amount of online distance computation can be reduced, thus making a TIE-based search method more

efficient than the FS method. In a TIE-based search method, using a single control vector s , the distance between each target vector and the control vector s is calculated offline and stored. During online processing, to find the best match target vector for a query vector X , the procedure is as follows:

1. Calculate $d(X, s)$.
2. Calculate $|d(X, s) - d(C, s)|$, where C is a target vector. Since $d(C, s)$ has already been calculated and stored offline, this step only involves one subtraction operation and one modulus operation.
3. If $|d(X, s) - d(C, s)|$ is equal or greater than the current minimum distance, it implies that $d(X, C)$ is definitely equal or greater than the current minimum distance. Thus C cannot be the best match target vector. In this case, go to step 2 with the next target vector. Otherwise, go to step 4.
4. If $|d(X, s) - d(C, s)|$ is smaller than the current minimum distance, calculate $d(X, C)$. If $d(X, C)$ is smaller than current minimum distance, let the minimum distance be $d(X, C)$ and the current best match target vector be C . Then, go to step 2 with the next target vector. End the procedure after the last target vector is reached.

As shown in steps 2 and 3 of the procedure above, the amount of online distance computation can be reduced since the unlikely target vectors can be identified by one subtraction, one modulus operation and one comparison operation, instead of calculating $d(X, C)$ and comparing it with the minimum distance.

In some TIE-based methods [12, 32, 40, 44], multiple control vectors are used to further reduce the online distance computation. If multiple control vectors are used, TIE equation (1) can be expanded as follows:

$$d(X_a, C_b) \geq \max_{1 \leq j \leq L} |d(X_a, s_j) - d(C_b, s_j)|, \quad (2)$$

where (s_1, s_2, \dots, s_L) represents a set of L control vectors.

One method to make use of multiple control vectors in a TIE-based search is as follows [44]:

- Select a set of L control vectors $(s_1, s_2, \dots, s_j, \dots, s_L)$. L should be much smaller compared with the number of target vectors.
- Compute offline $d(C_b, s_j)$ of each target vector to each of the control vectors and store the distances calculated.
- During the search process, the distance $d(X_a, s_j)$ between the query vector X_a and each control vector s_j is calculated.
- Find $l(b) = \max_{1 \leq j \leq L} |d(X_a, s_j) - d(C_b, s_j)|$ for each target vector C_b .
- For target vectors with $l(b)$ less than the current minimum distance, calculate $d(X_a, C_b)$. For target vectors with $l(b)$ greater than the current minimum distance, the calculation of their distances to X_a is not required since their distances are guaranteed to be greater than the current minimum distance.

In the TIE-based search method, the amount of distance computation performed online is likely to be lower if a greater number of control vectors are used. However, it is difficult to automatically determine how many control vectors and which vectors should be used in order to achieve optimal results.

5.6 TREE-STRUCTURED SEARCH

Multidimensional data search can also be carried out efficiently when the data are organised in a tree structure. A tree has a root node (node at the top of the tree), leaf nodes (nodes at the bottom of the tree) and internal nodes (nodes between the root and leaf nodes). In a balanced m -ary tree, each node is linked to a different set of m nodes at the next level. A set of m vectors is also predesignated to each tree node. The vectors at the root and internal nodes are called the test vectors whereas the vectors at the leaf nodes are the target vectors. If the tree structure is used for VQ encoding, the target vectors are the codewords. In the case of image retrieval, the target vectors are the image histograms. Figure 5.1 shows an m -ary tree structure.

To perform a search in a tree, we will start at the root node. The image vector is compared with each of the m test vectors predesignated to the root node. Based on the nearest (minimum distance) test vector, the path to the next internal node at the next level is determined. At the next level, the image vector is again compared with the test vectors predesignated to this internal node to determine the next path. This process will be repeated till we traverse down to the bottom-most layer of the tree. At the leaf node, the K target vectors that are nearest to the query vector are selected.

Form the description above, we can see that the tree-structured search process is efficient because at each level, $1/m$ of the total target vectors will be eliminated from consideration by a relatively small number of operations. However, the target vectors found in this method are usually not the optimal solution [9].

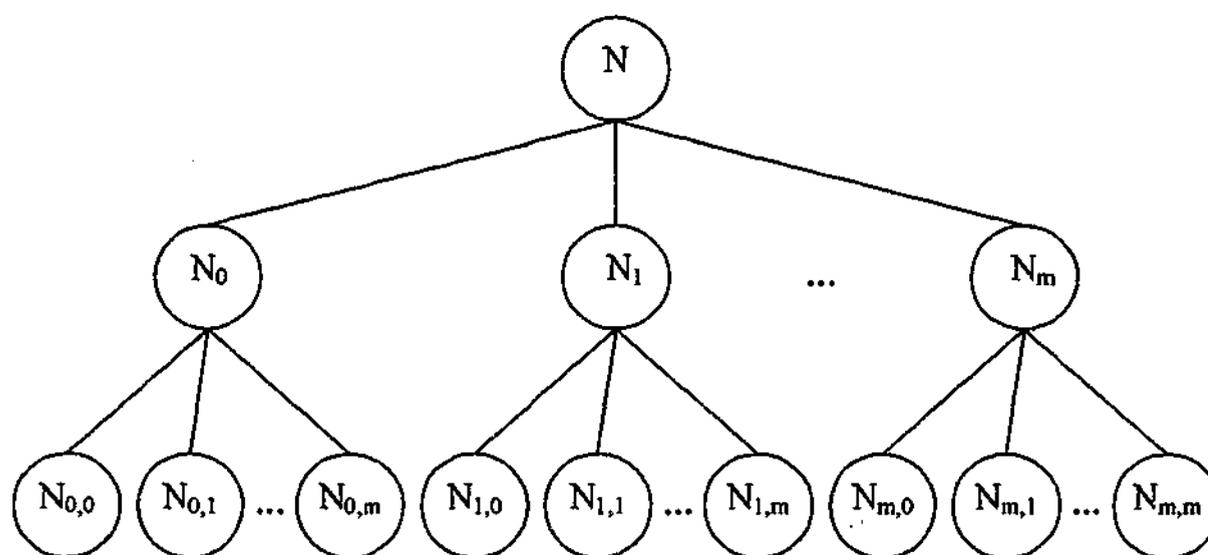


Figure 5.1 Tree structure for multidimensional data

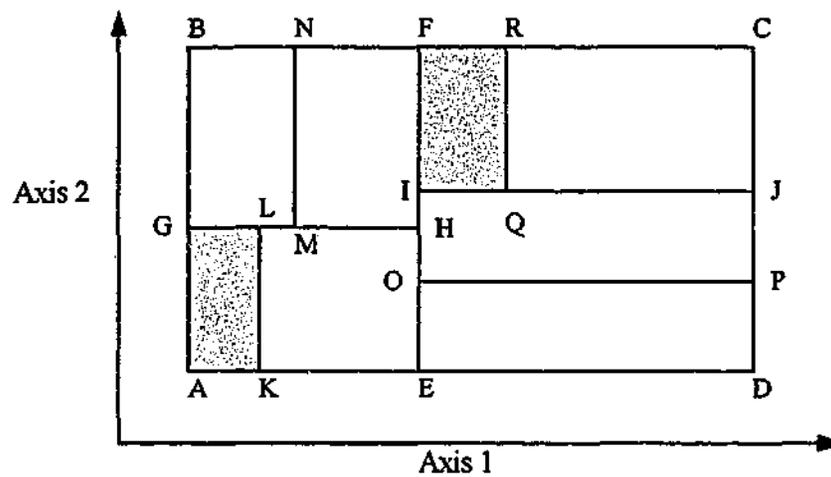
5.7 K-D TREE SEARCH

One tree structure that is frequently used in multidimensional data search processes is K-Dimensional (K-D) tree. K-D tree is a k-dimensional binary tree that recursively split the R^k space into two subspaces by a hyperplane orthogonal to one of the k coordinate axes. The partitioning hyperplane is represented by two scalar numbers. The first scalar number represents the index of the splitting coordinate axis and the second scalar number represents the splitting location of the plane on the axis. The subspace, whose location on the axis is smaller than the splitting location number, is normally called the left node. The subspace, whose location on the axis is equal or larger than the splitting location number, is called the right node. The nodes at the bottom-most of the K-D tree are called the terminating nodes or buckets. Figure 5.2 illustrates how a two-dimensional search space can be partitioned and represented using K-D tree.

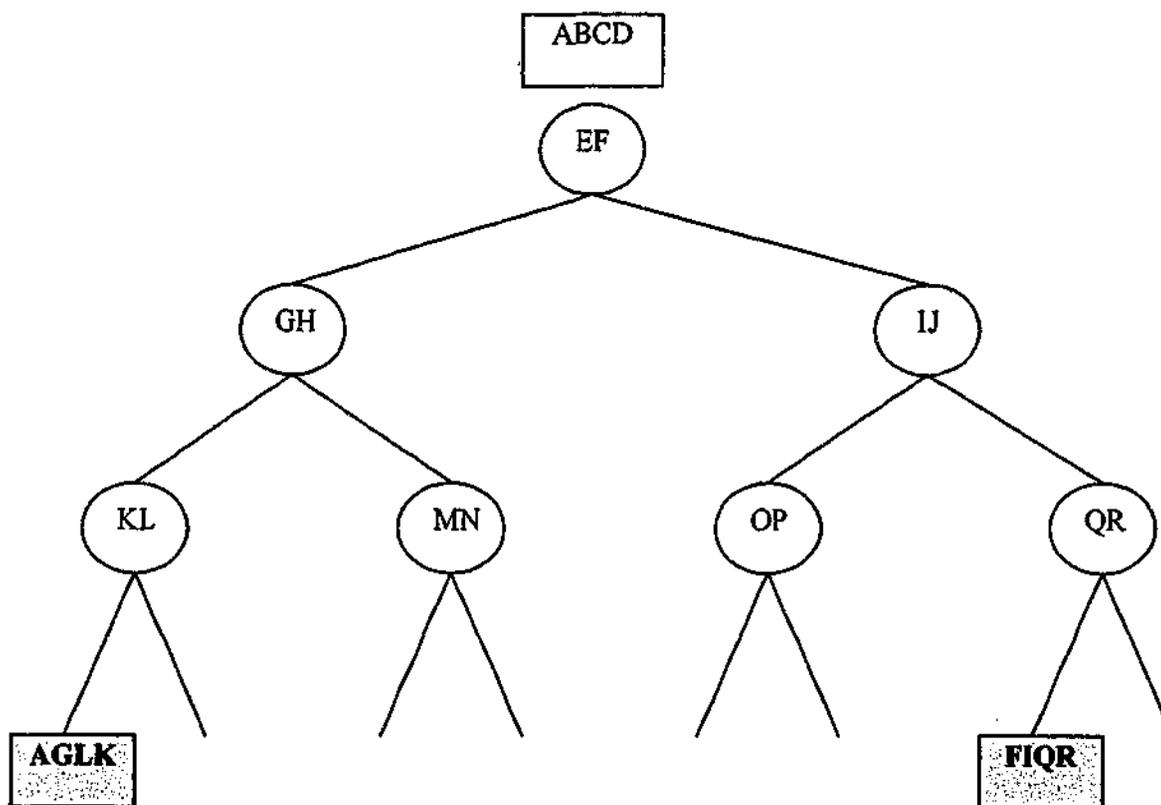
To allow the search in K-D tree to be efficient, we need to build a balanced tree. This means that all the branches of the tree have similar number of levels and each bucket contains similar number of target vectors. To build a balanced K-D tree, we will recursively carry out the following steps:

1. Calculate the variance of the values in each dimension of the target vectors.
2. Find the dimension that has the maximum variance.
3. Use axis corresponding to the dimension found as the splitting axis
4. Use the median of the values in the dimension found as the splitting value.
5. Partition the target vectors using the splitting axis and the splitting value.
6. Using the target vectors in each partition, go to step 1.

The process will be terminated when each bucket contains a suitable number of target vectors.



(a) Space partition



(b) K-D tree built from above space partition

Figure 5.2 Space partition and its corresponding K-D tree

To search for the most similar target vector for a query vector, we will start from the top of the tree. By comparing the two scalar partition numbers of the top-most node to the corresponding query vector component, we will decide if we should next traverse down the left or right node at the next level. This process will be repeated till a bucket is reached. We will then calculate the distance of the query vector to each target vector in that bucket to find the K number of target vectors that are most similar. When the set of most similar target vectors in this bucket is found, we will next use a procedure called backtracking to determine if there are anymore target vectors in the neighbouring buckets that may be closer to the query vector compared with the set that are already found [24]. This backtracking procedure is a very computationally intensive process.

5.8 METHODS SPECIFIC TO THE EUCLIDEAN DISTANCE

In the previous sections, we have discussed some general search methods which are applicable for both the encoding and retrieval search processes. In this section, we will discuss some search methods which are more specific to the encoding process. The reason why these methods are only suitable for the encoding process is because they are based on Euclidean distance. Since the retrieval process uses the Manhattan distance, we cannot implement these methods for retrieval.

In these methods, greater efficiency compared with the FS method is achieved by using some or all the following techniques:

1. Making slight alteration to the Euclidean distance
2. Calculating part of the information needed by the distance calculation offline to reduce the online computation time
3. Setting some constraints and filter out unlikely candidates.

In fact, using the squared Euclidean distance, $\sum_{i=1}^k (x_i - c_i)^2$, instead of the actual Euclidean distance of $[\sum_{i=1}^k (x_i - c_i)^2]^{1/2}$, can be classified under this class of methods.

Torres and Huguet have proposed another encoding algorithm based on this class of methods [96]. In their work, it is shown that the squared Euclidean distance can be expressed as follows:

$$\begin{aligned} d(X_a, C_b) &= \sum_{i=1}^k (x_i - c_i)^2, \\ &= \sum_{i=1}^k (x_i^2 - 2x_i c_i + c_i^2) \\ &= \sum_{i=1}^k x_i^2 + \sum_{i=1}^k c_i^2 - 2 \sum_{i=1}^k x_i c_i \end{aligned}$$

Since the codebook is fixed, $\sum_{i=1}^k c_i^2$ can be calculated offline and stored. It is also unnecessary to calculate $\sum_{i=1}^k x_i^2$ as it does not affect the nearest neighbour search.

Mathematically, this can be proved as follows:

$$\begin{aligned} d(X_a, \hat{C}) &= \min \{ d(X_a, C_b) \}, \text{ where } \hat{C} \text{ is the codeword most similar to } x \\ &= \min \left\{ \sum_{i=1}^k x_i^2 + \sum_{i=1}^k c_i^2 - 2 \sum_{i=1}^k x_i c_i \right\} \\ &= \sum_{i=1}^k x_i^2 + \min \left\{ \sum_{i=1}^k c_i^2 - 2 \sum_{i=1}^k x_i c_i \right\} \end{aligned}$$

Thus the nearest neighbour search is mainly dependent on $\min \left\{ \sum_{i=1}^k c_i^2 - 2 \sum_{i=1}^k x_i c_i \right\}$.

With term $\sum_{i=1}^k x_i c_i$ dependent on both codeword c and image vector x , it cannot be precalculated because image vector x is known only during the online processing.

However, we can easily observe that $x_{\max} \sum_{i=1}^k c_i \geq \sum_{i=1}^k x_i c_i$, if $x_i = 0, c_i = 0$ and

$x_{\max} = \max(x_1, x_2, \dots, x_k)$. Thus, the following expression:

$$d_1(X_a, C_b) = \sum_{i=1}^k x_i^2 + \sum_{i=1}^k c_i^2 - 2 x_{\max} \sum_{i=1}^k c_i \leq d(X_a, C_b) \text{ is always true.}$$

During online processing, distance $d_1(X_a, C_b)$ can be easily obtained by carrying out 1 multiplication and 2 additions since terms $\sum_{i=1}^k c_i^2$ and $2 \sum_{i=1}^k c_i$ can be precalculated and the time to find x_{\max} is negligible.

The proposed algorithm is shown in Figure 5.3.

The algorithm in Figure 5.3 will always proceed to first calculate $d_1(X_a, C_b)$ and to test if it is greater than the current minimum distance (dist_{\min}). If $d_1(X_a, C_b)$ is already greater than dist_{\min} , it is unnecessary to calculate $d(X_a, C_b)$ because $d_1(X_a, C_b) \leq d(X_a, C_b)$ is always true. As the complexity of calculating $d_1(X_a, C_b)$ is much lower compared with $d(X_a, C_b)$, efficiency that is greater than the FS method is achieved.

In [96], it is reported that the average search time of the proposed algorithm is reduced to approximately 31% of the FS method.

Another search method, which is similar to the one we have just discussed, is based on Winograd's identity [15]. We will further discuss this method in the Chapter 7.

Compute and store $\sum_{i=1}^k c_i^2$ of every codeword offline

Compute and store $2 \sum_{i=1}^k c_i$ for every codeword offline

for $a=1$ to M

 Evaluate x_{max}

$dist_{min} = \infty$

 for $b=1$ to N

$dist = d_1(X_a, C_b);$

 if ($dist > dist_{min}$) continue to next iteration of $b;$

$dist = d(X_a, C_b);$

 if ($dist > dist_{min}$) continue to next iteration of $b;$

$index = b;$ (most similar codeword selected)

$dist_{min} = dist;$

 next b

next a

Figure 5.3 Efficient Search Algorithm Proposed by Torres and Huguet [96]

The next method is one that carries out filtering based on the average colour of the images [21, 22] and this method can be applied using any colour space. This method is applicable to the encoding process since we can treat each segmented image block and each codeword in the encoding process as small images. For illustration purposes, we will use RGB colour space for image representation. To find the average colour of an image $\bar{x} = (R_{avg}, G_{avg}, B_{avg})^T$, we can use the following equations:

$$R_{avg} = \frac{\sum_{p=1}^P R(p)}{P}$$

$$G_{avg} = \frac{\sum_{p=1}^P G(p)}{P}$$

$$B_{avg} = \frac{\sum_{p=1}^P B(p)}{P}$$

where P is the number of pixels in the image, and $R(p)$, $G(p)$ and $B(p)$ are red, green, and blue components of the pixel p . Given the average colour vectors x and y of the two images, we define $d^2_{avg}(\cdot)$ as the squared Euclidean distance between these two vectors as follows:

$$d^2_{avg}(\bar{x}, \bar{y}) = \sum_{i=1}^3 (x_i - y_i)^2$$

It has been proven that $d^2_{avg}(\cdot)$ is the lower bound for the actual squared distance calculated using the full histogram. Thus, the full histogram squared distance will be equal or greater than $d^2_{avg}(\cdot)$. Based on this condition, we can select the potential candidates for the full histogram distance calculation.

The methods discussed in this section are simple to implement. Unfortunately, if used alone, they do not achieve very high efficiency.

5.9 SUMMARY

In this chapter, we have described several general search methods which are commonly used to improve the efficiency of multidimensional data search. All the methods described in Sections 5.4 to 5.7 can be used to improve the efficiency of the encoding process and the retrieval process in the proposed technique. This is because these efficient search methods can be used regardless of the metric used to compare the similarity between the multidimensional data in the processes. As for search methods described in Section 5.8, they can be used in the encoding

process. This is because these methods are specific for processes that use Euclidean distance as the metric to compare the similarity between the multidimensional data. Since the encoding process uses Euclidean distance and the retrieval process uses the Manhattan distance as the similarity comparison metric, the search methods in Section 5.8 cannot be used in the retrieval process.

To decide which search method to be used to improve the search efficiency of the encoding and retrieval processes, we will need to assess the difference between the two processes. Besides the difference in the metric used to compare the similarity between the multidimensional data in each process, the size of the data dimension is the main difference between the two processes. As stated in Section 5.2, the experimental results in Chapter 6 will show that the proposed method is more effective when the image vector dimension is not greater than 48 (4x4 pixels block * 3 colour channels). Due to this, the search method chosen for the encoding process must be efficient in searching vectors with low or moderate dimension sizes. The experimental results in Chapter 6 will also show that the proposed method is likely to perform more effectively when the number of histogram bins is 1024 or above. Thus the search method chosen for the retrieval process must be efficient in searching data with high dimensional sizes (i.e. histograms with large number of bins).

In the literature reviewed in this chapter, the experiments conducted to evaluate the efficiency of the search methods were generally performed on data with small and moderate dimensions. Comparing these experimental results, the tree-based methods are the most efficient in searching the data. Since the tree-based methods are capable of achieving good search efficiency on data with small and moderate dimension, these methods can be applied to the encoding process. We also observe that the tree-based methods can be further improved by combining with methods similar to the ones described in Section 5.8. The search method that we have adopted for the encoding process will be described in detail in Chapter 7. Although the tree-based methods are capable of achieving good efficiency for

searching small or moderate dimensional data, studies in [30] reported that these search methods are only capable of achieving low level of efficiency for high dimensional data. For example, [30] reports that when the data is of moderate dimensional size of 64, the tree methods only need approximately 18% of the FS method computation time. However, when the data is of high dimensional size (512), the tree methods require approximately 80% of the FS computation time. The tree-based methods are not efficient in searching high dimensional data because there is high degree of overlapping between the bounding regions. Thus to ensure the search result is accurate, there is a need to check the leaf nodes in the neighbourhood of the leaf node found initially in the tree. As a result, the tree-based methods do not perform well in high dimensional data and they are not adopted for the retrieval process. Since the search methods described in this chapter may not achieve high efficiency when used to search high dimensional data, a new approach is adopted for the retrieval process. This approach is to use a data structure that is widely used in text retrieval. This data structure is called inverted file. In Chapter 8, two search methods based on the inverted file structure will be presented.

CHAPTER 6
EXPERIMENTAL RESULTS
TO SHOW THE
EFFECTIVENESS OF
VQ-BASED INDEXING AND RETRIEVAL

6.1 INTRODUCTION

To investigate the effectiveness of the proposed image indexing and retrieval scheme, a web-based application based on the proposed scheme is developed. Using the developed application, experiments are carried out to obtain retrieval results for evaluating the proposed technique's retrieval performance as compared with that of other current techniques. In this chapter, we will describe the experiments carried out at different stages of our research. In total, three sets of image databases are used in the experiments. At the initial stage, in order to obtain a preliminary evaluation of the strengths of our proposed technique to other existing techniques, experiments are carried out on a small image database. The promising results obtained from the experiments on the small database show that our proposed technique has the potential to be more effective than the existing

techniques. Thus, more comprehensive experiments are conducted on two large image databases at latter stage of our research to test the performance of our proposed technique more thoroughly. Using different databases can also test the robustness of the proposed technique. For a robust technique, the results obtained from experiments performed on different databases should be consistent. In chronological order of our testing, an overview of the experiments conducted on each of the three databases is illustrated in Table 6.1.

Experiments are carried out to compare the retrieval performance of different CBIR techniques using all the three test databases. These experiments follow a standard procedure. The procedure is as follows:

1. Choose the evaluation metric(s).
2. Set up a test image database, $D = (I_1, I_2, \dots, I_k, \dots, I_M)$, where M is the number of images in the database.
3. Set up a query set, $Q = (Q_1, Q_2, \dots, Q_k, \dots, I_N)$, where N is the number of query images.
4. Perform the retrieval using the query set Q in the test image database D with the different retrieval techniques.
5. Use the chosen evaluation metric(s) to evaluate retrieval performance of the different techniques.

The structure of this chapter is as follows. Section 6.2 describes the recall and precision graphs, which are the chosen evaluation metrics used to compare the retrieval effectiveness of different CBIR techniques in all our experiments. Section 6.3 describes the setups and the results of the experiments carried out on the small image database. Section 6.4 describes the setups and the results of the experiments carried out on the two large image databases. Then, in Section 6.5, some issues and observations derived from the experimental results are discussed. Finally, Section 6.6 concludes the chapter.

Database	Experiment Purposes
Small database of 2165 colour images.	1. To evaluate the retrieval performance of the VQ scheme compared with QBIC colour histogram technique. At this early stage of research, the purpose of conducting testing on a small database is to have a preliminary assessment of the potential of VQ scheme.
The small database used for the experiment in the row above plus another 6 sets of 8 translated colour images.	1. A preliminary investigation to evaluate if the VQ scheme faces a potential robustness problem we have identified.
Large database of 10112 colour images compiled by our research group. This test image database is called Shared Colour Database (SCD).	1. To evaluate the effects of the different codebook sizes, codeword block-sizes and colour spaces on the retrieval performance of the VQ scheme. 2. To evaluate the retrieval performance of the VQ scheme compared with the three existing colour-based techniques: colour autocorrelogram, QBIC colour histogram and QBIC colour layout.
Large database of 5466 colour images. This image database is adopted by MPEG-7 as the standard database for evaluating retrieval effectiveness of colour-based descriptors. The database is called Common Colour Database (CCD).	1. To evaluate the effects of the different codebook sizes, codeword block-sizes and colour spaces on the retrieval performance of the VQ scheme. 2. To evaluate the retrieval performance of the VQ scheme compared with the three existing colour-based techniques: colour autocorrelogram technique, QBIC colour histogram and QBIC colour layout.

Table 6.1 Overview of the experiments conducted on each of the 3 databases

6.2 RECALL AND PRECISION GRAPHS

In this chapter, the metrics chosen to evaluate the retrieval effectiveness of the retrieval techniques in all the experiments are the commonly used recall and precision. Recall measures the ability of retrieving relevant images from the database. Mathematically, it is defined as:

$$\text{Recall} = \frac{\text{number of retrieved relevant images}}{\text{total number of relevant images in the database}}$$

The total number of relevant images in the database for each test query during performance testing can be determined by a group of the experts in this domain. Alternatively, the relevant images can also be identified through a subject test. This is carried out prior to the testing. The higher the recall, the better the performance. Precision measures the retrieval accuracy. Mathematically, it is defined as:

$$\text{Precision} = \frac{\text{number of retrieved relevant images}}{\text{total number of retrieved images}}$$

If considered in isolation, the higher the precision, the higher the retrieval performance. However, in practice, when recall and precision are considered together, they have an inverse relationship [42]. So, a good retrieval system should balance its recall and precision.

To compare two techniques based on both their recall and precision, we determine their recall and precision values ranging from 0 to 1 and plot their recall and precision graphs (see Figure 6.1). As more than one test query are normally used to have a good estimate of the performances of the techniques, the average of the queries' recall and precision values are calculated. Then, the average recall and precision graphs are plotted. The technique represented by the graph further from the origin has a higher performance.

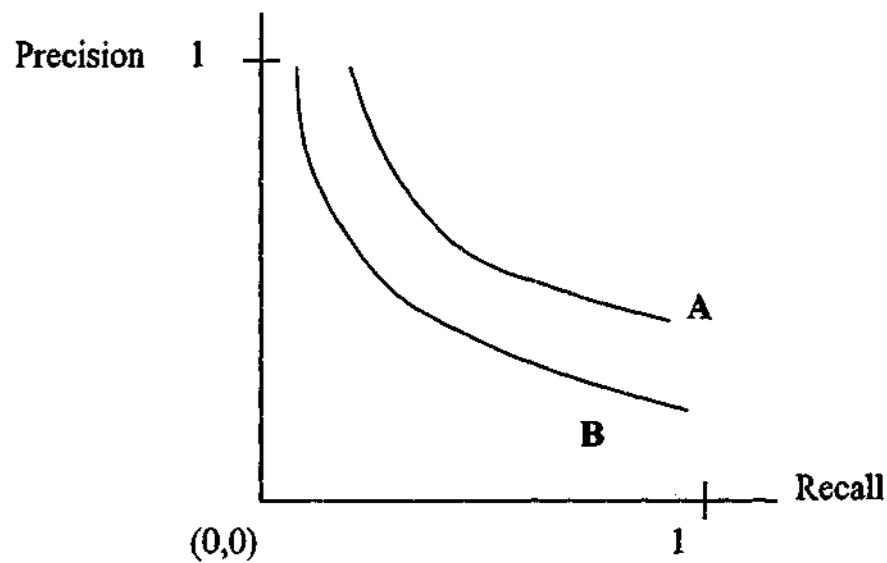


Figure 6.1 Recall-precision graphs. Technique A performs better than technique B since the precision and recall graph of technique A compared with the graph of technique B is further away from the origin

6.3 EXPERIMENTS ON SMALL DATABASE

The experiments conducted on the small database are to investigate two issues.

They are:

1. To investigate if the retrieval performance of our proposed technique is potentially more effective than an existing colour-based technique.
2. To examine the practical implication of a potential robustness problem that we foresee in our proposed technique. Theoretically our proposed technique may have a potential robustness problem when the image database consists of numerous similar images that are slightly translated in different directions by a small number of pixels. This problem will be explained in greater detail in the Section 6.3.2.1.

In the rest of this section, we will illustrate the setup and the findings of the experiments conducted to investigate the above issues.

6.3.1 COMPARISON OF RETRIEVAL PERFORMANCE OF VQ SCHEME TO QBIC COLOUR HISTOGRAM TECHNIQUE USING SMALL DATABASE

In the preliminary testing, we have used a small database to investigate if the retrieval performance of our proposed technique is more effective than an existing colour-based technique. The small database used consists of 2165 randomly chosen general colour images, which are collected from numerous websites. The size of the images is 256x256 pixels and their content ranged from animals to humans to sceneries. Six images are chosen from the database to be the query images. All the query images are very different in their content in order to ensure that the retrieval performance of the techniques are not bias towards a certain colour or image content. The ground truth images for each query image are established independently by five people looking through the database images. The five people are all researchers in working in similar research area. The existing technique that is used for retrieval performance comparison to the VQ scheme is the colour histogram technique used in the commercial product QBIC from IBM [1, 58, 92]. The colour histogram technique used in the QBIC system is an improved version of the traditional colour histogram technique as it takes into account the contributions of perceptually similar colours in the distance calculation [1, 58].

Figure 6.2 shows the retrieval performances of both techniques. We see that the recall and precision graph of the VQ technique is further away from the origin compared with the graph of QBIC's colour histogram technique starting from recall value of about 20%. Thus, VQ technique performs better than the QBIC's colour histogram technique.

During the course of conducting the experiment, we have identified that the VQ-based technique may have a potential robustness problem. This problem which relates to the technique's sensitivity to translation will be illustrated next.

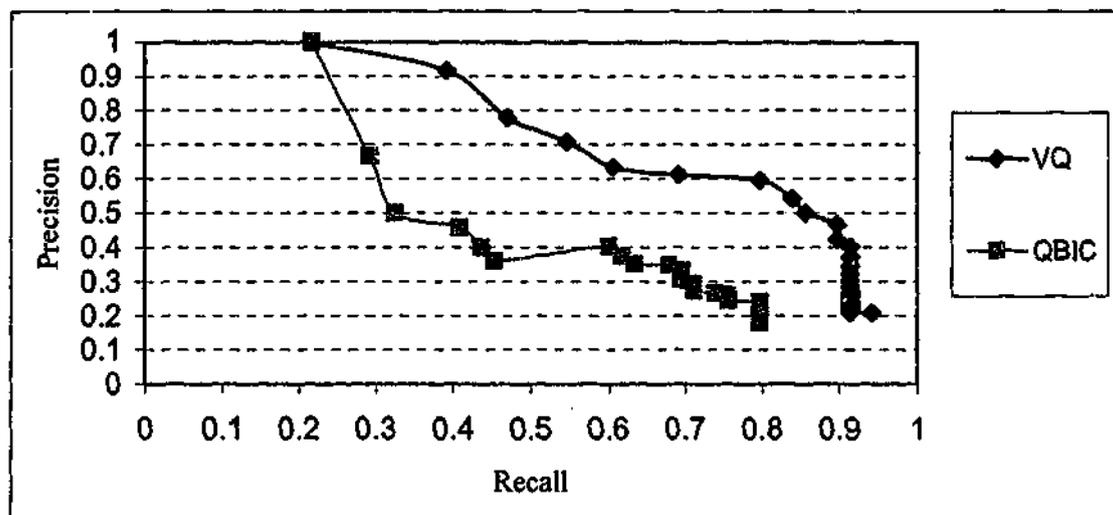


Figure 6.2 Retrieval performance comparison between the VQ-based and the QBIC colour histogram techniques

6.3.2 STUDIES ON SENSITIVITY TO TRANSLATION

When conducting the experiment that is presented in Section 6.3.1, we have identified a potential robustness problem in our proposed technique. In this subsection, we will describe the potential problem and study its effects on the retrieval performance in practice.

6.3.2.1 POTENTIAL ROBUSTNESS PROBLEM IN PROPOSED TECHNIQUE

The potential robustness problem we have identified can be explained as follows. The image encoding in VQ consists of two stages. Firstly, the image is segmented into equal blocks of $M \times N$ pixels. The block-size must be equivalent to the codeword block-size. Secondly, each image block is compared with the codewords to find the best match. The index of the best matching codeword is now used to represent the image block. Due to the manner images are encoded, it is possible that two similar images, where one is a slight translated image of another, have two very different sets of codewords representing their image blocks. Even though majority of the pixels in the two images are the same, the

positions of the pixels in their respective images are different. The pixel position difference might cause the pixels to be grouped differently and coded with different codewords. Since the retrieval of the proposed VQ scheme is based on the codewords, the distance of the two images can be large, though perceptually, they are almost the same. Next, we will now illustrate the potential problem of this technique with an example.

Figure 6.3 shows four images (dimension of 8x4 pixels) where, image B, C and D are image A translated right by 1, 2 and 3 pixels respectively. After translation, perceptually, the four images are still very similar. However, in VQ process, we have eight distinct blocks if the block-size is 4x4 pixels. Each distinct block may be encoded with different codeword, thus these images may have totally different VQ histograms. This is a possibility. We want to determine what is the effect of this potential problem on image retrieval performance in practice.

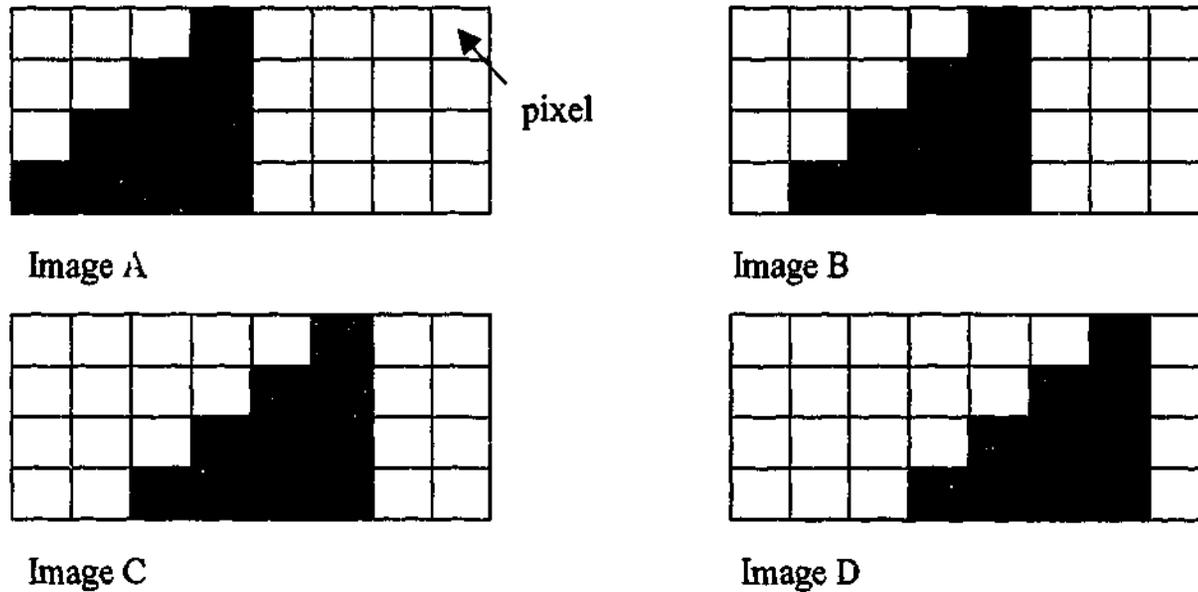


Figure 6.3 Perceptually similar images that differ by a slight pixel shift

To investigate the effects the image translation problem has on retrieval performance of the VQ scheme, two experiments are conducted. The first experiment studies the VQ histograms of the images and their translated images. We investigate if the slight translations in the images will cause great changes in the histograms. The purpose of the second experiment is to compare the retrieval

performance based on the recall and precision graphs of the VQ scheme on databases before and after translated images are added. For both experiments, the codebook size is 1024 and the block-size of the codeword is 4x4 pixels. The dimension of the images used is 256x256 pixels.

6.3.2.2 HISTOGRAM STUDIES

To conduct the first experiment, 200 images are randomly chosen from the database used in the previous experiment (Section 6.3.1). For each image, eight translated images (four translated to the right by 1, 2, 3, 4 pixels; four translated diagonally down by 1, 2, 3, 4 pixels) are produced. Using each of the 200 original images as the query, the Manhattan distance between the VQ histograms of the query and each of its eight translated images is calculated. The largest distance between the histograms of each query and its translated images is identified. In total, 200 largest distances are identified. The average largest distance is then calculated and it is found to be approximately 0.2152. This is 10.76% of the maximum possible distance between two images' histograms. The maximum possible distance between two images' histograms is 2 and this distance can be obtained when there are no common bins that have values greater than zero between the two image histograms. The average largest distance found above means that between the queries and their images that are most affected by the translation, only an average of 10.76% of the blocks are represented by different codewords. Thus, these results show that in practice, the number of image blocks affected by the translation is a lot smaller compared with what is suggested in theory.

To further investigate the results above, we take a closer look into one of the 200 images chosen above. Figure 6.4 shows two images, where MT5D3 is one of the translated (diagonally down by 3 pixels) images of the query image MT5Q. Among the translated images and the query, MT5D3 has the largest distance to MT5Q. However, their histograms are similar and the percentage of difference is

14%. With such percentage of difference, the translated images of MT5Q are ranked among the closest when retrieved from database. Actually, this distance is much smaller than that (53%) between MT5Q and VC14 (Figure 6.5) which is a similar face to MT5Q.



MT5Q



MT5D3

Figure 6.4 MT5Q and its translated image, MT5D3

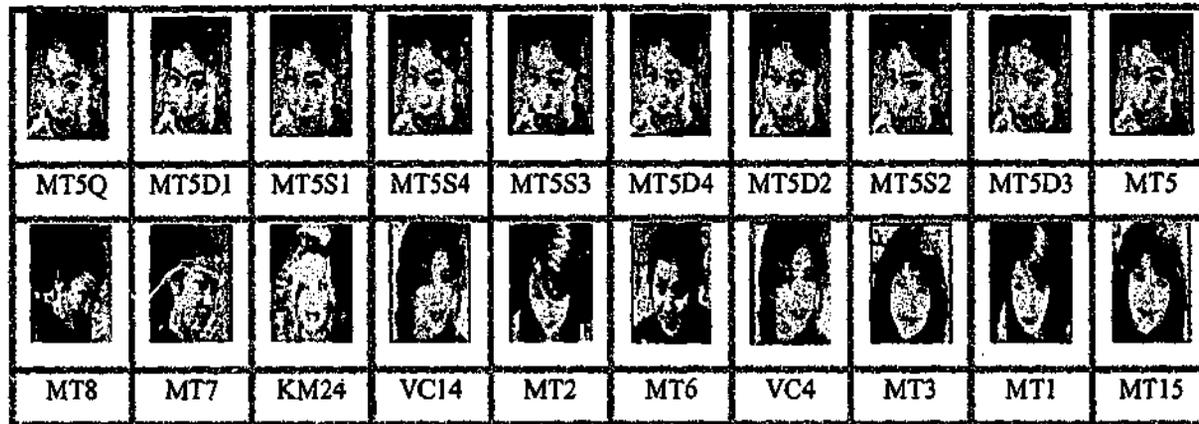


Figure 6.5 An example of VQ based retrieval result of query image "MT5Q"

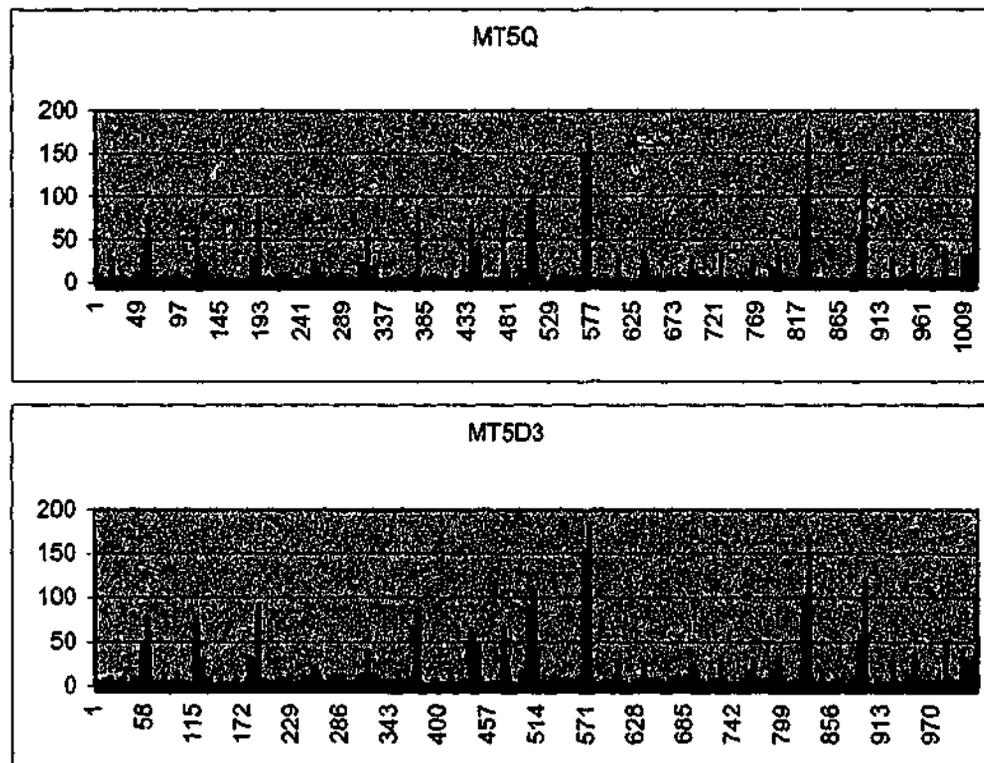


Figure 6.6 VQ-based histograms of "MT5Q" & "MT5D3"

The reasons for the difference between what is suggested in theory and the results obtained in practice are as follows:

- Generally, for an image to be meaningful, majority of the pixels and their neighbours are correlated. For example, in Figure 6.4, large patches of pixels in the areas like hair, face and background have similar colour. When the image is translated by a few pixels, pixels that are shifted out of the VQ segmented blocks and the pixels shifted into them actually have the similar colour. Since there is little pixel colour change in the blocks, the codeword that is closest will likely to stay unchanged. The image blocks that are affected are the ones which contain edges of the picture. The percentage of such blocks in an image is normally small.
- To achieve higher compression rate, the number of codewords in a VQ codebook is usually much smaller compared with the number of all the possible distinct blocks in the database images. Since the number of codewords must be kept small, the codebook generation algorithm groups the similar image blocks in the training set into clusters and calculates a image block to represent each cluster of similar image blocks. This set of the image blocks calculated is the set of codewords. Representing similar image blocks with a codeword is possible because humans are not sensitive to the slight differences in the image blocks. Since majority of the translated blocks are only slightly different from their original ones, the codeword used to represent an original block is likely to be the one used to represent the translated blocks. Thus, the histograms of the translated images are likely to be very similar to the histogram of the original image.

To further demonstrate that many codewords representing the image vectors in image MT5Q are the same as the ones in image MT5D3, their VQ-based histograms are plotted. Figure 6.6 shows that the VQ-based histograms of images MT5Q and MT5D3 are very similar.

In content-based image retrieval technique, the retrieval is not based on exact match and the retrieved database images are normally ranked based on their similarity to the query. Since the difference between the query and its translated images are relatively small, the translated images should still be ranked higher among the retrieved images.

6.3.2.3 RECALL AND PRECISION STUDIES

In the second experiment, we evaluate the retrieval effectiveness of the VQ scheme on the small image database in two cases:

- Before adding 48 translated images.
- After adding 48 translated images.

The 48 translated images (8 from each query) are produced from the six query images used for the experiment in Section 6.3.1.

The retrieval effectiveness of the VQ scheme in both cases is evaluated by comparing its retrieval results to the ones obtained using the QBIC colour histogram technique.

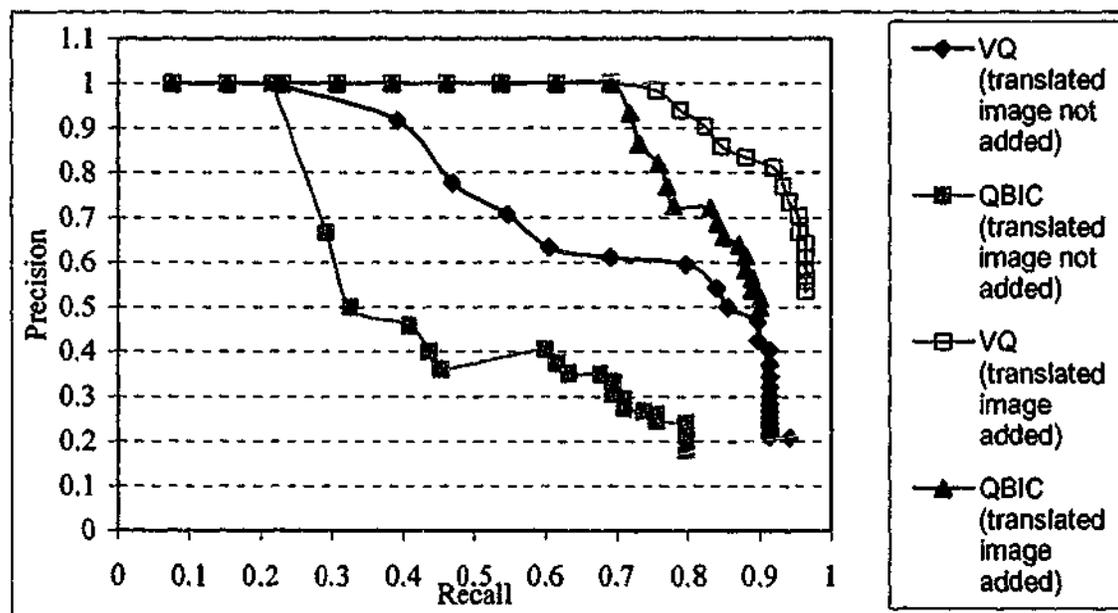


Figure 6.7 Retrieval performance comparison between the VQ-based and the QBIC colour histogram techniques

Figure 6.7 shows the retrieval effectiveness of both techniques on the small database before and after adding the set of translated images. The retrieval results on the small database before the translated images are added have been illustrated in Section 6.3.1. We have illustrated that VQ-based technique performs better than the QBIC's colour histogram technique before the translated images are added. According to the recall and precision graphs plotted from the retrieval results of the two techniques on the small database with the translated images are added, we see that the retrieval performances of both techniques have improved. Between recall values of 0% to about 70%, both techniques' graphs have precision of 1. The reason for the QBIC's colour histogram graph having the precision of 1 is because this technique is invariant to translation [87]. Thus all the added translated images are retrieved in the QBIC system. As for the VQ-based technique, all the added translated images are also retrieved (Figure 6.5) because of the reasons we have discussed in Section 6.3.3. We have observed that due to the similarity in the histograms between the queries and their translated images, the translated images are always ranked right after the original images in the retrieval results. Above the recall value of 70%, the recall and precision graphs show that the VQ-based technique performs better than the colour histogram technique. VQ-based technique performs better because it retrieves not only the added translated images, but also other relevant images that the colour histogram technique has missed out.

6.3.2.4 SUMMARY OF EXPERIMENTAL RESULTS ON THE PROPOSED TECHNIQUE'S SENSITIVITY TO TRANSLATION

The results in Section 6.3.2.2 and 6.3.2.3 have shown that the VQ-based technique is robust as it is not overly sensitive to slight translation of pixels in images. Such robustness of the proposed technique allows it to be more effective than the QBIC colour histogram technique when retrieving images in the small database even after translated images have been added to the database. This is because all the

translated images, which are considered relevant to their respective query images, are retrieved among the top-ranked retrieved images by the proposed technique.

6.4 EXPERIMENTS ON LARGE DATABASES

After obtaining promising results from the experiments on the small database, we proceeded on to conduct more comprehensive experiments on two large general image databases. The first large image database is compiled locally at our research centre. It is shared among all researchers working on image indexing and retrieval at the centre. For this reason, the database is named Shared Colour Database (SCD). The second large image database is the Common Colour Database (CCD): the database adopted as part of MPEG-7 standard for evaluating the retrieval performance of various colour-based descriptors. For our discussions in this thesis, we will refer to the two databases as SCD and CCD respectively. Detailed descriptions of the two databases will be illustrated later in this section. It is important to conduct testing on large databases because with relatively greater number of images in the large databases as compared with the small database, the diversity and complexity in the images' content are increased. The sizes of the two large databases are also better representations of the number of images in databases that require the assistance of an image retrieval system to retrieve images efficiently and effectively. Thus, the performance of the proposed technique can be more thoroughly evaluated.

6.4.1 DATABASE SCD

Database SCD consists of 10112 24-bit full-colour BMP images with dimensions of 256x256 pixels. These images are downloaded from the JPEG Independent Group Site (<ftp://ftp.cs.columbia.edu>) and from commercial photo and image software ("Eurdkasoftware 10000 Photos and Raytraced Images"). The images are categorised into 60 classes ranging from landscapes to animals to human beings.

Query : Image 1



1 []	2 []	3 []	4 []	5 []	6 []	7 []	8 []
9 []	10 []	11 []	12 []	13 []	14 []	15 []	16 []
17 []	18 []	19 []	20 []	21 []	22 []	23 []	24 []
25 []	26 []	27 []	28 []	29 []	30 []	31 []	32 []
33 []	34 []	35 []	36 []	37 []	38 []	39 []	40 []
41 []	42 []	43 []	44 []	45 []	46 []	47 []	48 []
49 []	50 []	51 []	52 []	53 []	54 []	55 []	56 []
57 []	58 []	59 []	60 []	61 []	62 []	63 []	64 []
65 []	66 []	67 []	68 []	69 []	70 []	71 []	72 []
73 []	74 []	75 []	76 []	77 []	78 []	79 []	80 []
81 []	82 []	83 []	84 []	85 []	86 []	87 []	88 []
89 []	90 []	91 []	92 []	93 []	94 []	95 []	96 []

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Figure 6.8 A sample page of pages 3 to 34 in the survey form

To conduct the experiments on this database, a set of 32 images is selected as the queries. Majority of these query images are from different classes and their contents, both in colour and objects, are very diverse. This is to ensure that the retrieval performance is not bias towards any particular colours or classes of images. Thus the final experimental results obtained are more robust.

To establish the ground truth for the set of query images used in the experiments, a subject test was carried out. Of the 35 participants, 20 of them are familiar with techniques used in image retrieval systems while the rest are not. The setup of the subject test is as follows:

Each participant was given a set of survey forms (Appendix A). Page 1 of the survey forms explains the purpose of the test and the activities to be carried out by the participants. Page 2 consists of the consent form and also a portion that allows the participants to fill up their personal information. Pages 3 to 34 consist of the 32 chosen query images. Each of these pages (Figure 6.8) consists of one query image on the top left corner. Below the image is a table with 96 cells numbered 1 to 96. These numbered cells correspond to a web page which displays 96 numbered images. The participants are to compare the query image with the 96 images and tick on the brackets of the numbers that represent the images that they find similar or relevant to the query image. The 96 images for each query image are compiled by looking through the top 96 retrieved images for each retrieval technique and selecting the ones that we think are relevant or similar to the query image. This procedure is to filter out all the irrelevant images and also to make the task more manageable for the participants. Finally, the judging criteria for selecting the similar or relevant images by the participants are also recorded.

By analysing the survey results, we noticed that all participants have treated images that are the zoomed in or out, rotated and skewed versions of the query image as relevant. We also observed that the set of images picked as similar/relevant to its query image by the participants might differ from person to

person. Such subjectivity in judgement makes it difficult to pick only one set of relevant images for each query image. To solve this problem, three sets of relevant images are picked for each query image. The first set comprises of images selected by at least 70% of the participants. The second set comprises of relevant images selected by at least 50% of the participants. The third set comprises of relevant images selected by at least 30% of the participants. We name the relevant image sets GT70, GT50 and GT30 respectively.

6.4.1.1 EXPERIMENT A ON DATABASE SCD : EFFECTS OF CODEBOOK SIZES, CODEWORD BLOCK-SIZES AND COLOUR SPACES ON RETRIEVAL PERFORMANCE OF THE VQ SCHEME

The objective of Experiment A is to investigate the effects of different codebook sizes, codeword block-sizes and colour spaces on the retrieval performance of our proposed technique. This is important as it allows us to determine the appropriate codebook size, codeword block-size and colour space for our proposed technique to achieve good retrieval performance.

To generate the set of codebooks for this experiment on database SCD, 60 images (one from each class) are selected as the training images set. The training images selected are representative of the contents of the images in database SCD. The set of codebooks generated consists of six different codebook sizes - 128, 256, 512, 1024, 2048 and 4096. For each codebook size, four codeword block-sizes 2x2, 4x4, 8x8 and 16x16 are generated. For each combination of codebook size and codeword block-size, three codebooks are also generated in RGB, HSV and LUV colour spaces respectively. In total, 72 codebooks with different combinations of codebook size, codeword block-size and colour space are generated for database SCD. After the codebooks are generated, the images in the database are then encoded correspondingly. The histograms are then built using the VQ compressed image data to index the images.

6.4.1.2 RESULTS OF EXPERIMENT A ON DATABASE SCD

The following procedures are used to compile the experimental results using each of the 72 codebooks in the proposed VQ scheme:

1. Obtain the retrieval results for each of the 32 query images.
2. Using the retrieval results of the 32 query images, plot the average precision and recall graph based on GT70. Also plot the graphs for GT50 and GT30.

In total, there are 216 average recall and precision graphs, 72 each for GT70, GT50 and GT30. We name the three sets of graphs SCD70, SCD50 and SCD30 respectively. Each of the 72 average recall and precision graphs represents the retrieval performance of the proposed technique using a codebook generated from a particular combination of codebook size, codeword block-size and colour space.

EVALUATION OF EXPERIMENTAL RESULTS FOR DIFFERENT CODEBOOK SIZES

First we evaluate the effects of different codebook sizes on retrieval performance when the codebooks have common codeword block-size and colour space. The following procedures are carried out on SCD70, SCD50 and SCD30:

1. Group the graphs according to the codebook size. Since we are considering codebook sizes 128, 256, 512, 1024, 2048 and 4096, six groups (each with 12 graphs) are obtained.
2. Between the six groups, compare graphs of codebooks with the same colour space and codeword block-size. Thus, 12 sets of comparison are made.

The 12 sets of average recall and precision graphs comparison for SCD30, SCD50 and SCD70 are shown in Appendix B1. Since the 12 sets of comparison for SCD70 show similar findings to those in SCD50 and SCD30, only the findings for SCD50 are presented.

All the comparisons in the 12 sets for SCD50 have similar trends. An example of a set of graphs from the 12 sets is shown in Figure 6.9. In each set, the graphs

show that the retrieval performance improves as the codebook size increases. However, the positions from the origin of graphs for codebook sizes 1024, 2056 and 4096 do not differ from each other much as compared with the rest of the graphs. This shows that retrieval performances are not greatly affected when the codebook size is increased from 1024 to 4096. Thus, the retrieval performance is near optimal when the codebook size is 1024 for this database.

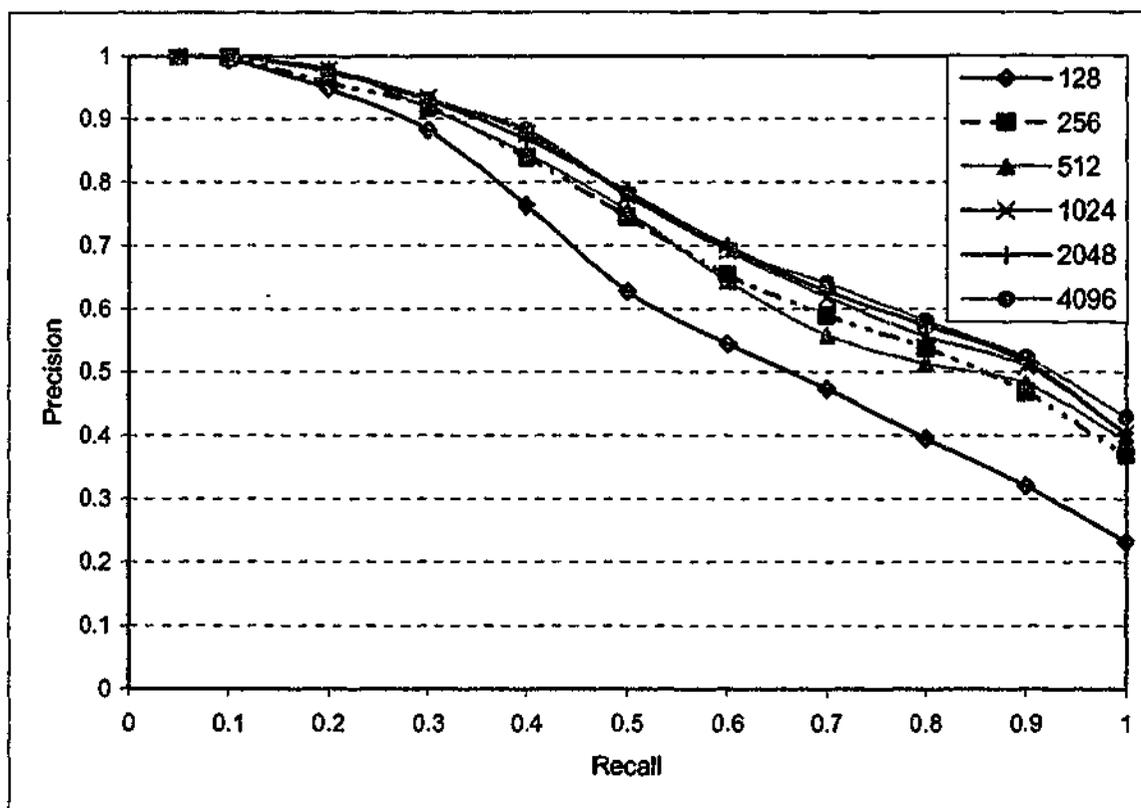


Figure 6.9 Average recall and precision graphs of VQ scheme using different codebooks. The codebooks are generated with common codeword block-size(4x4) and colour space(HSV) but vary in codebook size.

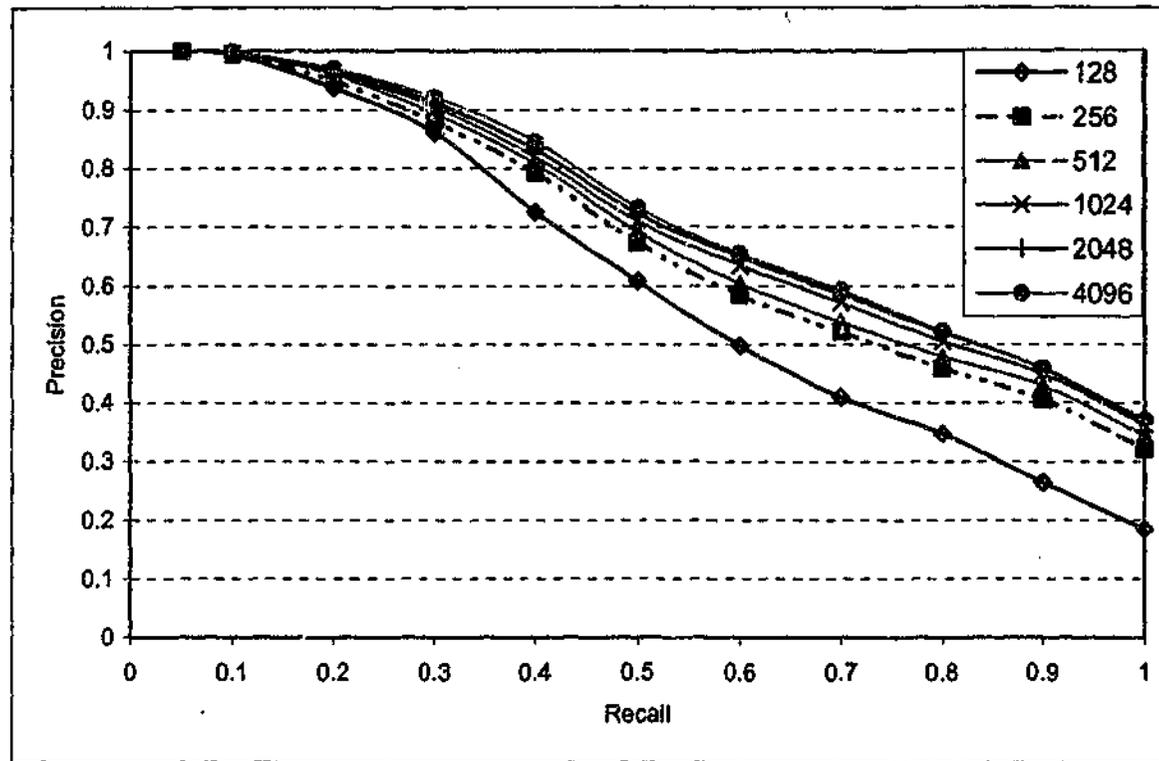


Figure 6.10 Overall performance graphs showing the overall retrieval performance of codebooks with different codebook sizes for SCD50.

Next, to see the overall effects on retrieval performance of different codebook sizes, the average of the graphs in each of the six groups is plotted. The overall performance graphs for SCD50 in Figure 6.10 also show that the overall retrieval effectiveness is optimised when the codebook size is between 1024 and 4096. This characteristic of the proposed technique, where the retrieval performance of codebook with smaller size is similar to codebook larger size, is suitable for retrieval because smaller codebook size allows greater retrieval efficiency.

EVALUATION OF EXPERIMENTAL RESULTS FOR DIFFERENT CODEWORD BLOCK-SIZES

To evaluate the effects of different codeword block-sizes on retrieval performance, the following procedures are carried out on SCD70, SCD50 and SCD30:

1. Group the graphs according to the codeword block-size. Since we are considering codebook block-sizes 2x2, 4x4, 8x8 and 16x16, four groups (each with 18 graphs) are obtained.

2. Between the four groups, compare graphs of codebooks with the same colour space and codebook size. Thus, 18 sets of comparison are made.

The 18 sets of average recall and precision graphs comparison for SCD30, SCD50 and SCD70 are shown in Appendix B2. Since the 18 sets of comparison for SCD70 show similar findings to those in SCD50 and SCD30, only the findings for SCD50 are presented. All the comparisons in the 18 sets for SCD50 show similar trends. In each set (Figure 6.11 shows an example), the graphs plotted from retrieval results of codebooks with codeword block-sizes of 2x2 and 4x4 are furthest from the origin. This shows that for this database, their retrieval performances are better compared with those of block-sizes 8x8 and 16x16.

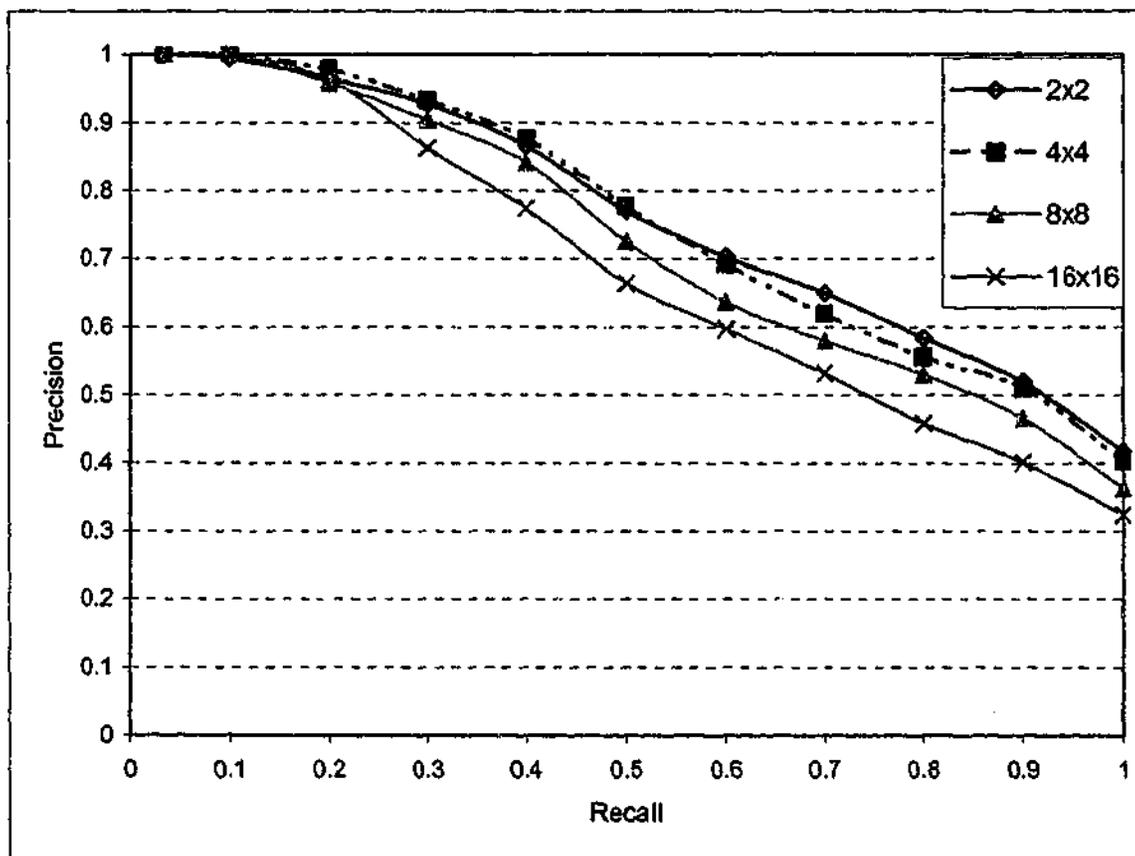


Figure 6.11 Average recall and precision graphs of VQ scheme using different codebooks. The codebooks are generated with common codebook size (1024) and colour space (HSV) but vary in codeword block-sizes.

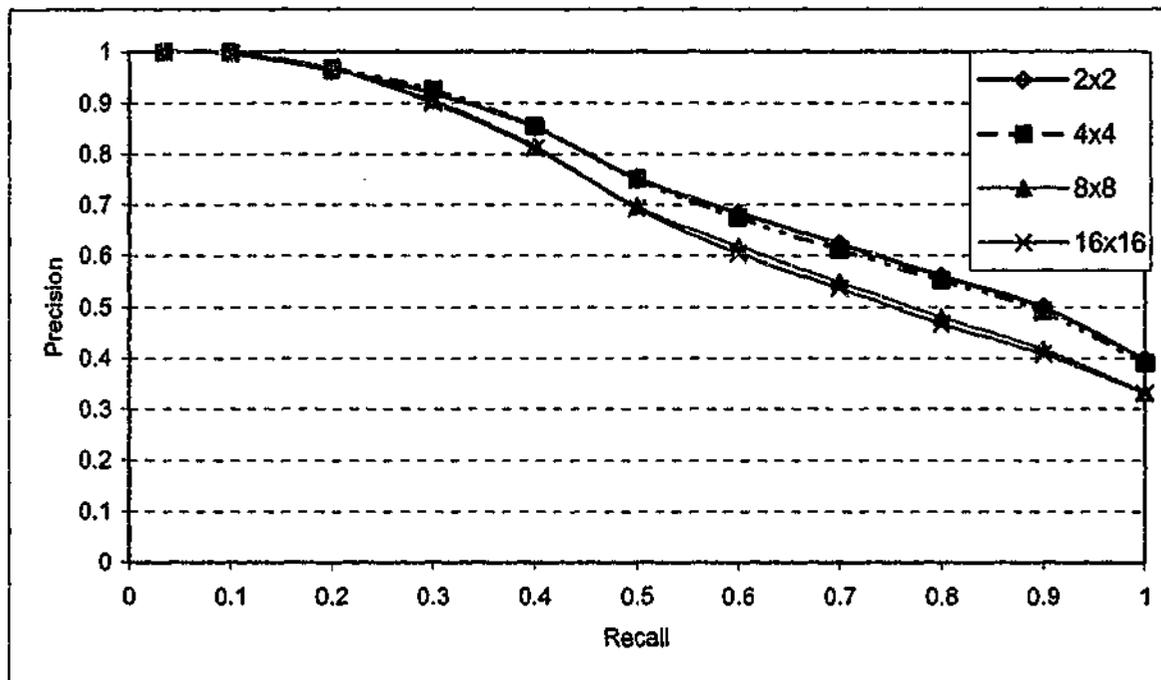


Figure 6.12 Overall performance graphs showing the average retrieval performance of codebooks with different codeword block-sizes for SCD50.

To evaluate the overall retrieval performance, the average of the graphs in each of the three groups is plotted. The newly plotted graphs (Figure 6.12) show that the overall retrieval performances of codebooks with different codeword block-sizes are of similar trend to those shown in Figure 6.11. Thus, 2x2 and 4x4 are the best codeword block-sizes among the four block-sizes evaluated. The proposed technique has better retrieval effectiveness when using block-sizes of 2x2 and 4x4 compared with the larger block-sizes because using 1024 codewords to represent the possible colour pixel combinations in the larger block-sizes is too coarse compared with using 1024 codewords to represent the possible colour pixel combinations in the smaller block-sizes. This is due to the reason that the possible colour pixel combinations in the larger block-sizes are a lot greater than the possible colour pixel combinations in the smaller block-sizes. Since the representation is too coarse for the larger block-sizes, their histograms are less likely to be capable of capturing the colours and the spatial relationships of the pixel colours in the images effectively. Thus, the retrieval effectiveness using the larger block-sizes is worse compared with using the smaller block-sizes.

Although the experimental results show that the proposed technique has similar retrieval effectiveness when block-sizes of 2x2 and 4x4 are used, we would recommend block-size of 4x4 for the proposed technique. This is because it is more efficient generating codebooks and encoding images when the larger codeword block-size is used. Besides, block-size of 4x4 can capture more spatial relationships among the colour pixels in the images compared with block-size of 2x2.

EVALUATION OF EXPERIMENTAL RESULTS FOR DIFFERENT COLOUR SPACES

To evaluate the effects of different colour spaces on retrieval performance, the following procedures are carried out on SCD70, SCD50 and SCD30:

1. Group the graphs according to the colour space. Since we are considering colour spaces RGB, LUV and HSV, three groups (each with 24 graphs) are obtained.
2. Between the three groups, compare graphs of codebooks with the same codebook size and codeword block-size. Thus, 24 sets of comparison are made.

The 24 sets of average recall and precision graphs comparison for SCD30, SCD50 and SCD70 are shown in Appendix B3. Since the 24 sets of comparison for SCD70 show similar findings to those in SCD50 and SCD30. Only the findings for SCD50 are presented.

All the 24 sets of comparison for SCD50 show similar trends. In each set, the graph plotted from retrieval results of codebook with HSV colour space are slightly above the other two graphs. Figure 6.13 shows an example. The graph of codebook with RGB is relatively nearer to the origin compared with the other two graphs. However, the differences between the positions of the three graphs are very little. This shows that for this database, the retrieval performances are not greatly affected by the different colour spaces.

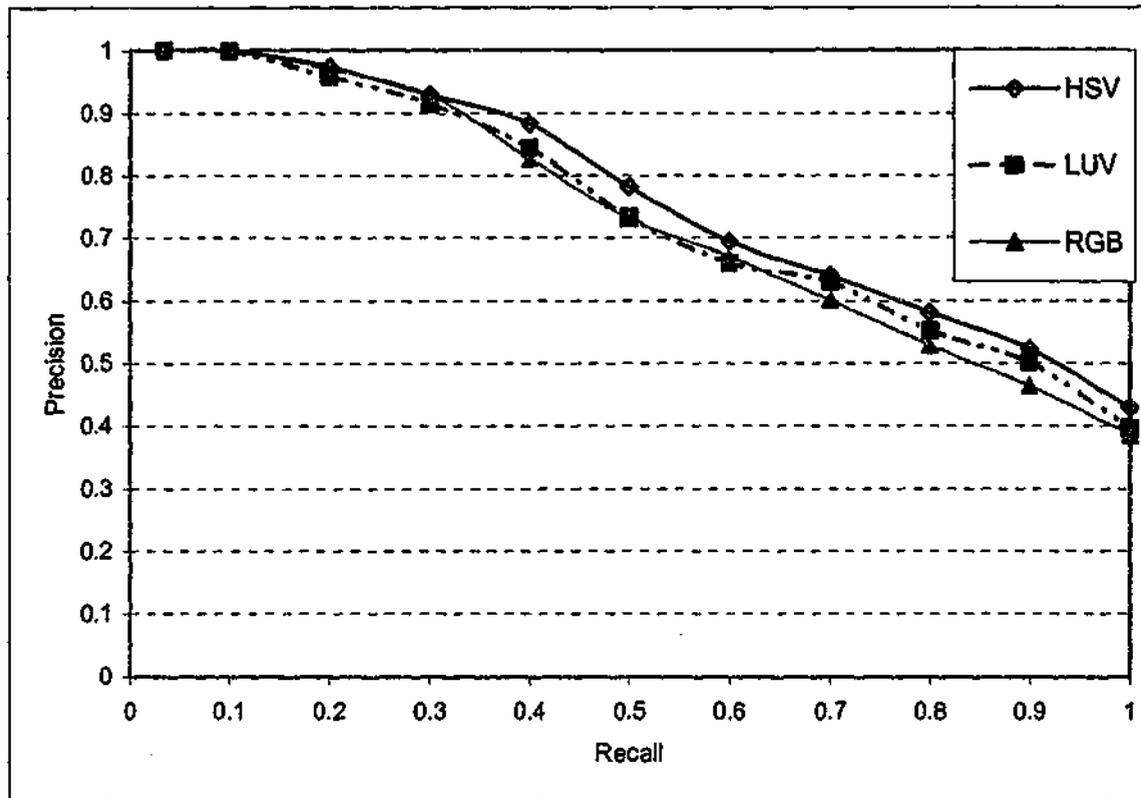


Figure 6.13 Average recall and precision graphs of VQ scheme using different codebooks. The codebooks are generated with common codebook size (4096) and codeword block-size (4x4) but vary in colour space

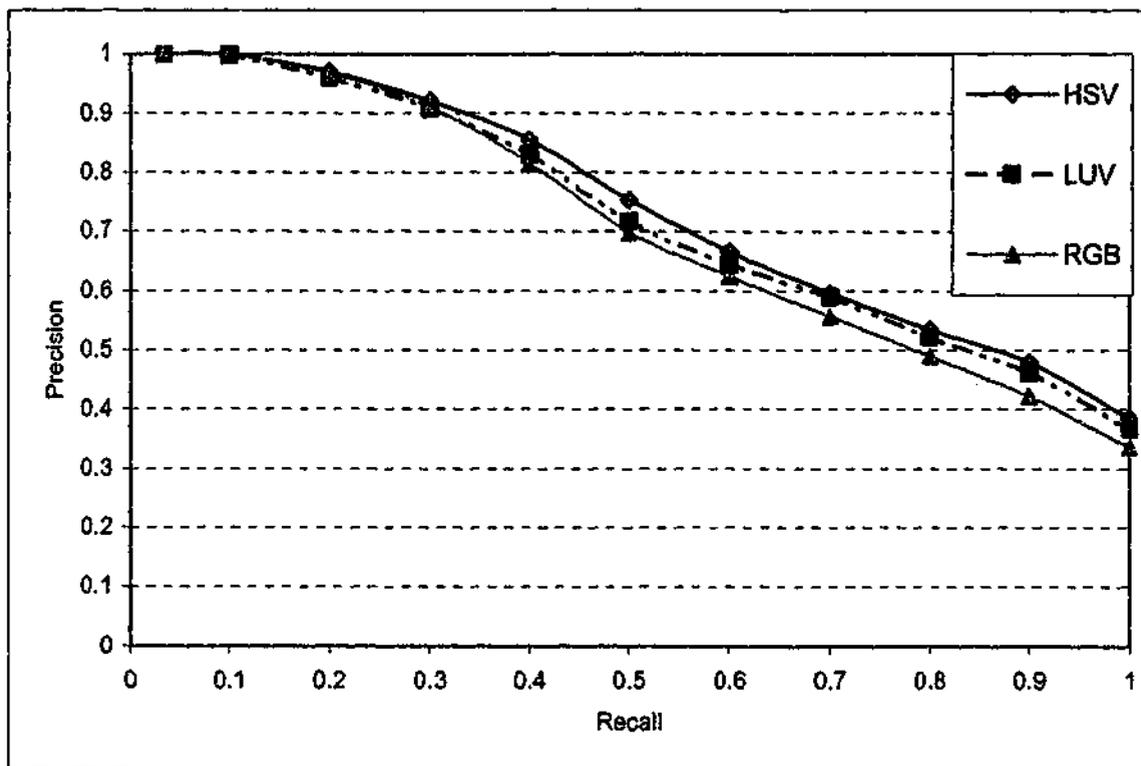


Figure 6.14 Overall performance graphs showing the average retrieval performance of codebooks with different colour spaces for SCD50.

To evaluate the overall retrieval performance, the average of the graphs in each of the three groups is plotted. The newly plotted graphs for SCD50 (Figure 6.14) show that the overall retrieval performances are of similar trend to that in Figure 6.13. For database SCD, the retrieval performance of the proposed techniques is not greatly affected by the three colour spaces.

CONCLUSIONS FOR EXPERIMENT A

Based on the experimental results, we have shown that retrieval performance of our proposed technique are often similar for codebook sizes between 1024 and 4096. Thus if retrieval efficiency is very critical, codebook size of 1024 should be used since using this codebook size allows the proposed technique to attain similar retrieval effectiveness to the retrieval effectiveness when codebook size 4096 is used. The experimental results have also indicated that among codebook size, codeword block-size and colour space, codeword block-size has the most effect on the retrieval performance. Based on the experimental results, codeword block-size of 2x2 and 4x4 pixels produce best retrieval performance when the codebook size is between 128 and 4096. However, codeword block-size of 4x4 pixels is recommended since compared with codeword block-size of 2x2 pixels, it allows higher codebook generation efficiency and shorter image encoding time. Besides, block-size of 4x4 can capture more spatial relationships among the colour pixels in the images compared with block-size of 2x2. Finally, the experimental results show that for the proposed technique, image processing using HSV colour space yields retrieval performance that is marginally better than using RGB and LUV colour spaces.

Although the codebook configuration identified to be suitable for the proposed technique is derived from the testing on database SCD, this codebook configuration is likely to be applicable to other image databases. This is because SCD is built to be as representative as possible of those real-life databases that make use of CBIR systems. Like those real-life databases, database SCD contains

images which have a wide range of colours and semantic contents. In Section 6.4.2.2, to further demonstrate that this codebook configuration is applicable to other image databases, we will show that the codebook configuration identified in this section to be suitable for the proposed technique can also achieve high retrieval performance in a standard MPEG-7 test database.

6.4.1.3 EXPERIMENT B ON SCD : COMPARISON OF RETRIEVAL PERFORMANCE OF VQ SCHEME TO EXISTING TECHNIQUES

The objective of Experiment B is to evaluate the retrieval effectiveness of the proposed technique. This is done by comparing the retrieval performance of the proposed technique to that of three existing techniques. The three existing retrieval techniques are the QBIC colour histogram technique, the QBIC colour layout technique and the colour autocorrelogram technique. The histograms built using the QBIC colour histogram technique are of 512 bins. The colour space used is the Munsell colour system. These are the default bin number and colour space used in the QBIC software. For the colour autocorrelograms built in our experiments, the colour space used is HSV (hue, saturation and value quantised into 21, 3, 3 intervals respectively) and the correlation is measured at $d=1, 3, 5$ and 7. These are the parameters recommended by Huang for good retrieval performance in the colour autocorrelogram technique [34].

6.4.1.4 RESULTS OF EXPERIMENT B ON DATABASE SCD

To obtain the retrieval performance of the QBIC colour histogram technique, each image in database SCD is firstly indexed by building its colour histogram. Then, three sets of the retrieval results for the 32 query images are obtained based on ground truths GT30, GT50 and GT70. Finally, the average recall and precision graph is plotted for each set of retrieval results. The same procedure is used to

plot average recall and precision graphs for GT30, GT50 and GT70 using the QBIC layout and colour autocorrelogram techniques.

To evaluate the retrieval performance of our proposed technique to the three existing techniques, the average precision and recall graphs plotted for the three existing techniques are compared with the 216 average recall and precision graphs (72 each for GT30, GT50 and GT70) plotted from the retrieval results obtained using 72 VQ generated codebooks of different configurations in Experiment A (Section 6.4.1.1). Figures 6.15(a), (b) and (c) show the average recall and precision graphs based on ground truths GT30, GT50 and GT70 respectively of the four different techniques. Since it is recommended that codebook size of 1024, codeword block-size of 4x4 pixels and colour space of HSV be used for our proposed technique implementation, the average recall and precision graphs shown here are plotted from retrieval results using codebook of this configuration. As shown in all the three figures, the VQ-based technique outperforms the three existing techniques.

Besides outperforming the two colour-based techniques used in QBIC when using codebook configuration of codebook size 1024, codeword block-size 4x4 and colour space HSV, the VQ-based technique also has better retrieval performance using any 72 configurations generated. This is consistent when using ground truth images of GT30, GT50 or GT70. Compared with the colour autocorrelogram technique based on GT30, the VQ technique has better retrieval performance for 69 configurations out of the total of 72 configurations (see Table 6.1). When based on GT50, the VQ-based technique has better retrieval performance for 66 out of the 72 configurations. Finally, when based on GT70, the VQ-based technique has better retrieval performance for 70 out of the 72 configurations. This demonstrates the robustness of the VQ-based technique as this technique is not overly sensitive to a particular codebook configuration to perform better than other existing techniques.

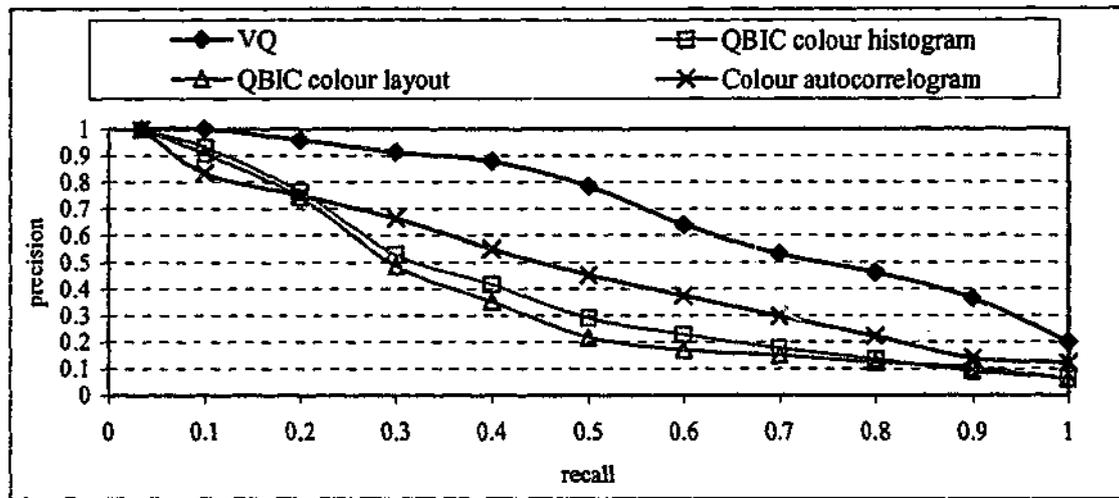


Figure 6.15 (a): Average recall and precision graphs based on ground truth GT30

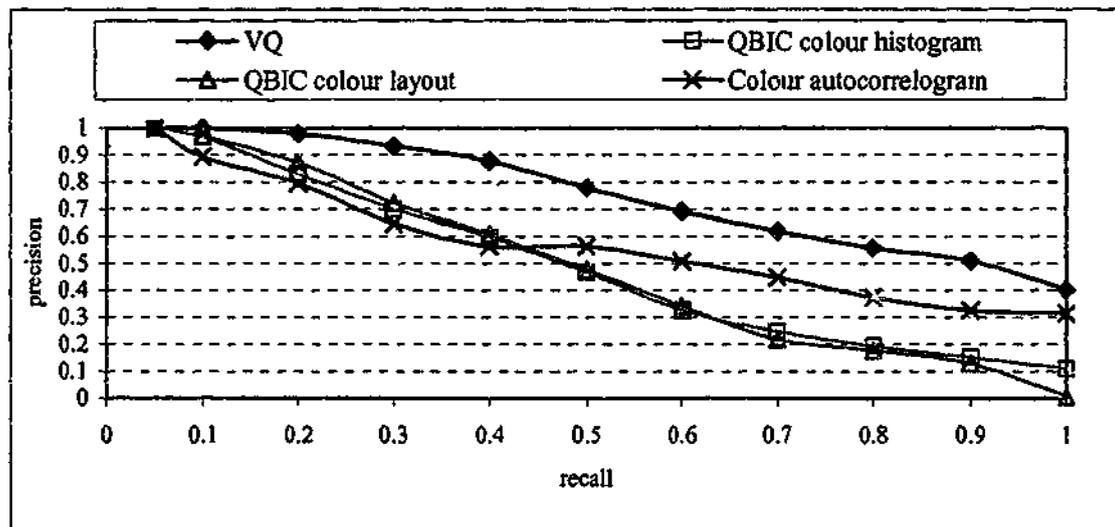


Figure 6.15 (b): Average recall and precision graphs based on ground truth GT50

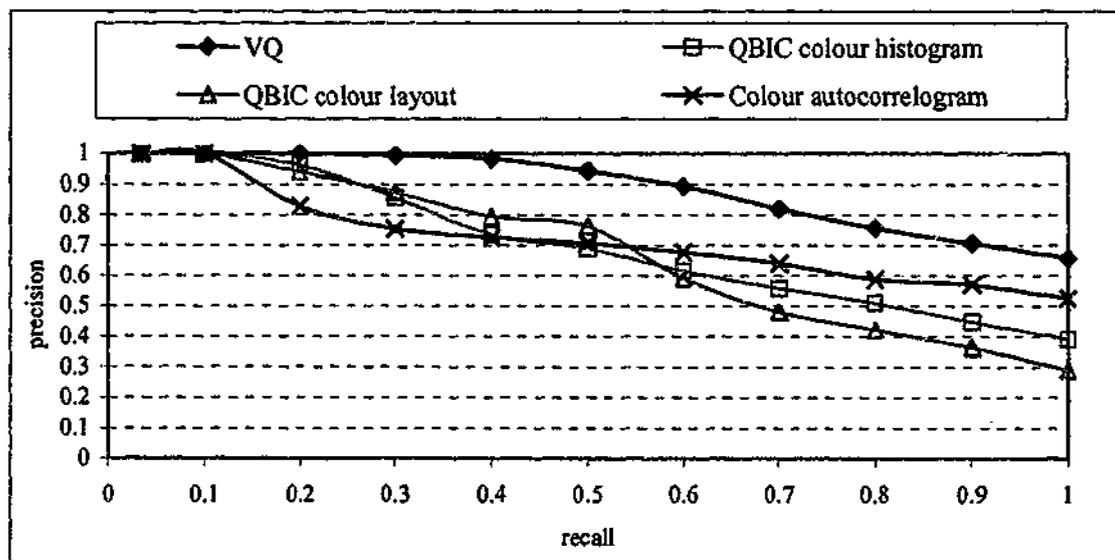


Figure 6.15 (c): Average recall and precision graphs based on ground truth GT70

Table 6.1: Codebook configurations used in VQ-based technique that produce retrieval performance worse than the colour autocorrelogram technique

RGB Colour space

Codebook size 128				Codebook size 256				Codebook size 512				Codebook size 1024				Codebook size 2048				Codebook size 4096			
2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16
x(50)		x(30) x(50)	x(30) x(50) x(70)				x(30) x(50) x(70)																

HSV Colour space

Codebook size 128				Codebook size 256				Codebook size 512				Codebook size 1024				Codebook size 2048				Codebook size 4096			
2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16
			x(50)																				

LUV Colour space

Codebook size 128				Codebook size 256				Codebook size 512				Codebook size 1024				Codebook size 2048				Codebook size 4096			
2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16
			x(50)																				

Legend

- x : Marks the codebook configurations used in the VQ-based technique that produce retrieval performance worse than the colour autocorrelogram technique. The retrieval performance of the two techniques is compared based on their recall and precision graphs.
- (30), (50) and (70) : Represent ground truth image sets GT30, GT50 and GT70 respectively. The retrieval performance of the proposed technique using a particular codebook configuration can be evaluated using each of the 3 sets of ground truth images. Thus, (30), (50) or (70) is shown beside each 'x' in above tables to denote the set of ground truth images we are using to evaluate retrieval performance of the proposed technique. For example, x(50) in the first column of the topmost table means that when GT50 is used as the ground truth images, the retrieval performance of the proposed technique using codebook configuration of codebook size 128, codeword block-size 2x2 and colour space RGB is worse than that of colour autocorrelogram technique. Since x(30) and x(70) do not appear in this column, it also means that when GT30 or GT70 is used as the ground truth images, the proposed technique using the same codebook configuration still outperforms the colour autocorrelogram technique.

6.4.2 DATABASE CCD

During the course of our research, a common colour database is proposed by Heinrich-Hertz Institute (HHI), Berlin, for evaluating the retrieval performance of various colour descriptors. This database has since been adopted as part of MPEG-7 standard. Database CCD consists of 5466 colour images. The images are compiled from sources like stock photo galleries, screen shots of television programs and animations. The images consist of five different sizes: 320 x 240, 352 x 240, 384 x 256, 252 x 288 and 384 x 288 pixels. These images are recommended by various research laboratories in commercial companies like Philips, Sharp, LG CIT and Mitsubishi.

A set of 50 images from this database is also provided to serve as queries in experiments. The documentation of the queries and their corresponding ground truth images can be found in [109]. The queries and their corresponding ground truth images are manually established through a process of visual inspection by different groups of participants. The ground truth images consist of images which are the zoomed in or out, the rotated and the skewed versions of their corresponding query images. From our observation, we can see that the selection of the set of ground truth images for each query is more rigid compared with database SCD. The reason for such observation is because only images belonging to the same video sequence or class to the query are picked as the ground truth images. However, images that have content that are similar to the query but belong to different video sequence or class are not selected as the query's corresponding ground truth images.

6.4.2.1 EXPERIMENT DESCRIPTIONS FOR DATABASE CCD

In order to investigate whether the findings from the experiments conducted on database SCD are heavily dependent on the database or the query image set, the

same set of experiments carried out on database SCD is also carried on database CCD. To generate the codebooks for the experiments, 60 images are also selected from database CCD to form the training images set. In a similar manner as the codebook generation for database SCD, 72 codebooks of different codebook size, codeword block-size and colour space configurations are also generated for the experiments on CCD.

6.4.2.2 RESULTS OF EXPERIMENT A ON DATABASE CCD

As indicated above in Section 6.4.1.1, the purpose of conducting Experiment A is to investigate the effects of different codebook sizes, codeword block-sizes and colour spaces on the retrieval performance of the proposed technique on database CCD. The same procedures described in Section 6.4.1.2 to evaluate the effects of different codebook sizes, codeword block-sizes and colour spaces on retrieval performance of the proposed technique on database SCD are also used here.

EVALUATION OF EXPERIMENTAL RESULTS FOR DIFFERENT CODEBOOK SIZES

The comparisons in the 12 sets of graphs have similar trends and they are presented in Appendix C1. In each set (Figure 6.16 shows an example), the graphs show that the retrieval performance improves as the codebook size increases. However, the positions from the origin of graphs for codebook sizes 1024, 2056 and 4096 do not differ from each other much as compared with the rest of the graphs, i.e., the difference between the precision of graphs for codebook sizes 1024 to 4096 is less than 5% at all levels of recall. This shows that retrieval performance is not greatly affected when the codebook size is increased from 1024 to 4096. Thus, the retrieval performance is near optimal when the codebook size is 1024.

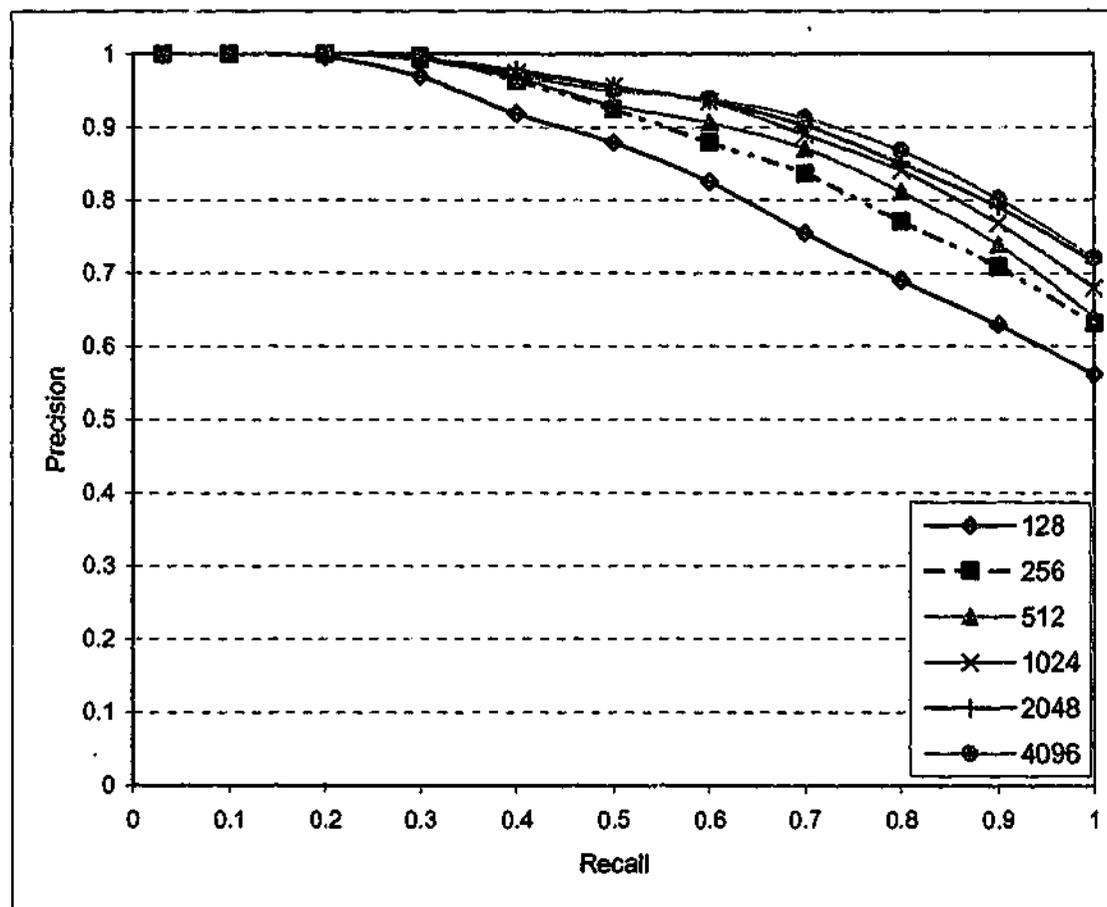


Figure 6.16 Average recall and precision graphs of VQ scheme using different codebooks. The codebooks are generated with common codeword block-size(4x4) and colour space(LUV) but vary in codebook size.

Next, to see the overall effects on retrieval performance of different codebook sizes, the average of the graphs in each of the six groups is plotted. Since the graphs in each group represent average retrieval performances of the codebooks that have the same codebook size, each newly plotted graph represents the overall retrieval performance of a particular codebook size. The overall performance graphs in Figure 6.17 also show that the overall retrieval effectiveness of codebook sizes 1024 and 2048 are similar to that of codebook size 4096.

The findings here are consistent to those found in the experiment performed on database SCD.

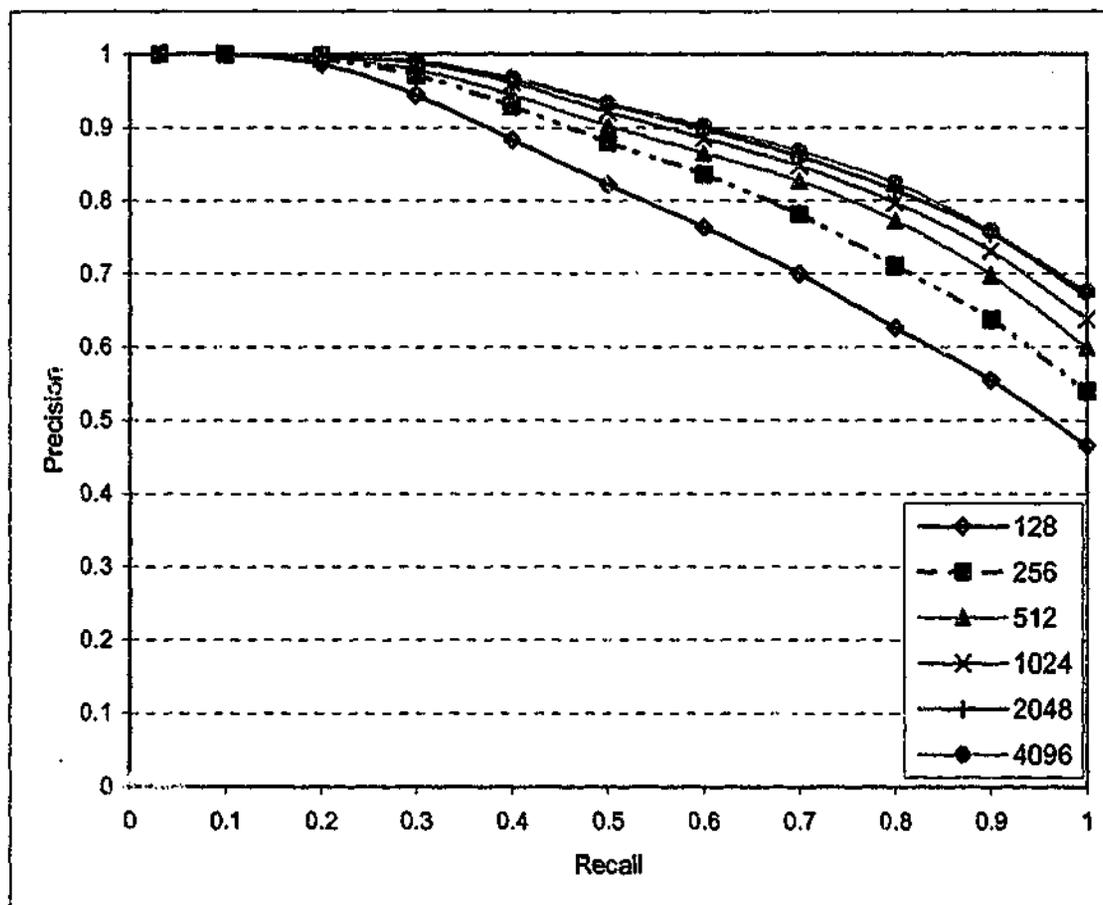


Figure 6.17 Overall performance graphs showing the overall retrieval performance of codebooks with different codebook sizes.

EVALUATION OF EXPERIMENTAL RESULTS FOR DIFFERENT CODEWORD BLOCK-SIZES

The comparisons in the 18 sets of graphs show similar trends and they are presented in Appendix C2. In each set (Figure 6.18 shows an example), the graphs plotted from retrieval results of codebooks with codeword block-sizes of 2x2 and 4x4 are furthest from the origin. This shows that their retrieval performances are better compared with those of block-sizes 8x8 and 16x16.

To evaluate the overall retrieval performance, the average of the graphs in each of the three groups is plotted. Since the graphs in each group represent average retrieval performances of the codebooks that have the same codeword block-size, each newly plotted graph represents the overall retrieval performance of a particular codeword block-size. The newly plotted graphs (Figure 6.19) show that

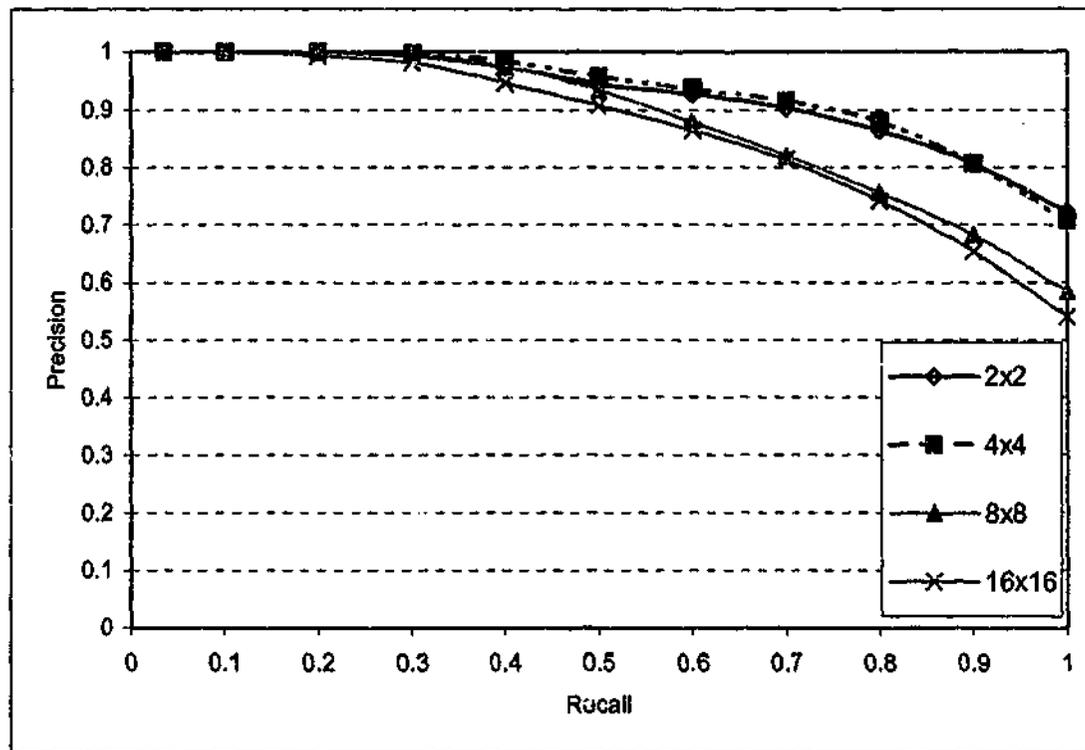


Figure 6.18 Average recall and precision graphs of VQ scheme using different codebooks. The codebooks are generated with common codebook size (1024) and colour space (HSV) but vary in codeword block-sizes.

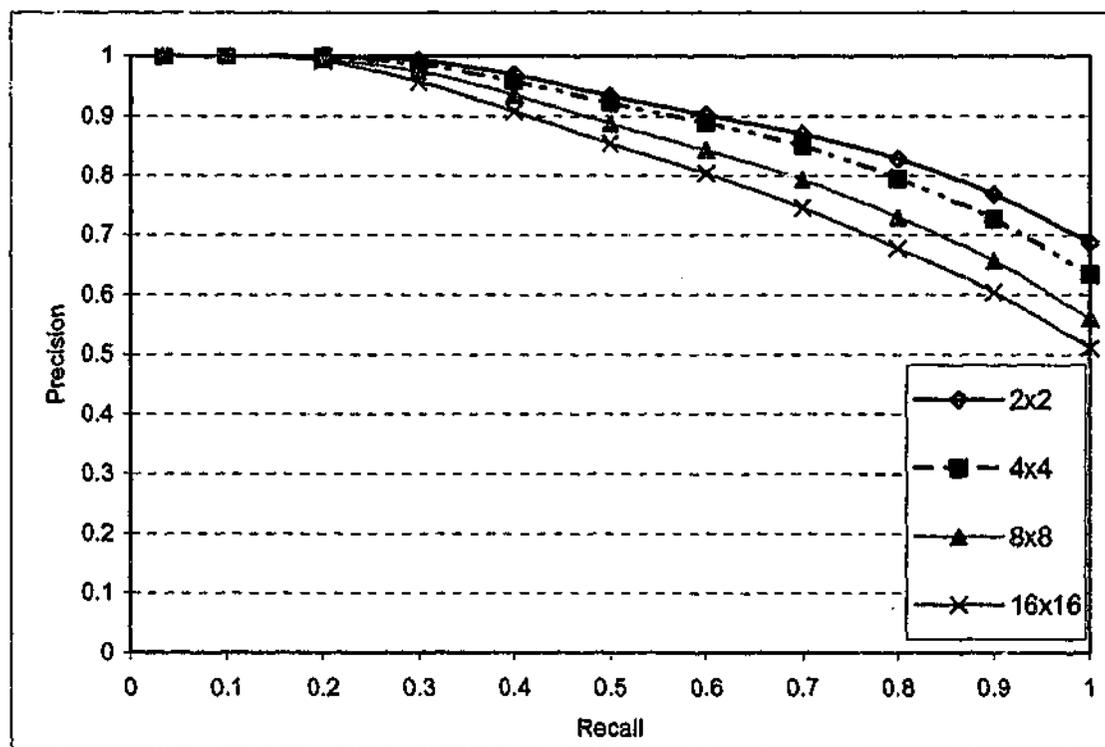


Figure 6.19 Overall performance graphs showing the average retrieval performance of codebooks with different codeword block-sizes.

the overall retrieval performances of codebooks with different codeword block-sizes are of similar trends to those shown in Figure 6.18. Retrieval performances of codeword block-sizes of 2x2 and 4x4 pixels are the better compared with codeword block-sizes of 8x8 and 16x16 pixels. Between codeword block-sizes of 2x2 and 4x4 pixels, the difference between the retrieval performances are more distinct compared with the ones shown in Figure 6.19. However, the precision differences between the codeword block-sizes are still less than 5% at all levels of the recall. Thus, their retrieval effectiveness is still very similar.

The findings here are also consistent to those found in the experiment on database SCD.

EVALUATION OF EXPERIMENTAL RESULTS FOR DIFFERENT COLOUR SPACES

All the 24 sets of graphs comparison show similar trends and they are presented in Appendix C3. In each set (Figure 6.20 shows an example), the graphs plotted from retrieval results of codebooks with HSV and LUV colour space are above the graph plotted for the colour space RGB. This shows that using colour spaces HSV and LUV for the proposed technique produce better retrieval performances compared with using colour space RGB.

To evaluate the overall retrieval performance, the average of the graphs in each of the three groups is plotted. Since the graphs in each group represent average retrieval performances of the codebooks that have the same colour space, each newly plotted graph represents the overall retrieval performance of a particular colour space. The newly plotted graphs (Figure 6.21) show that the overall retrieval performances are of similar trend to that in Figure 6.20.

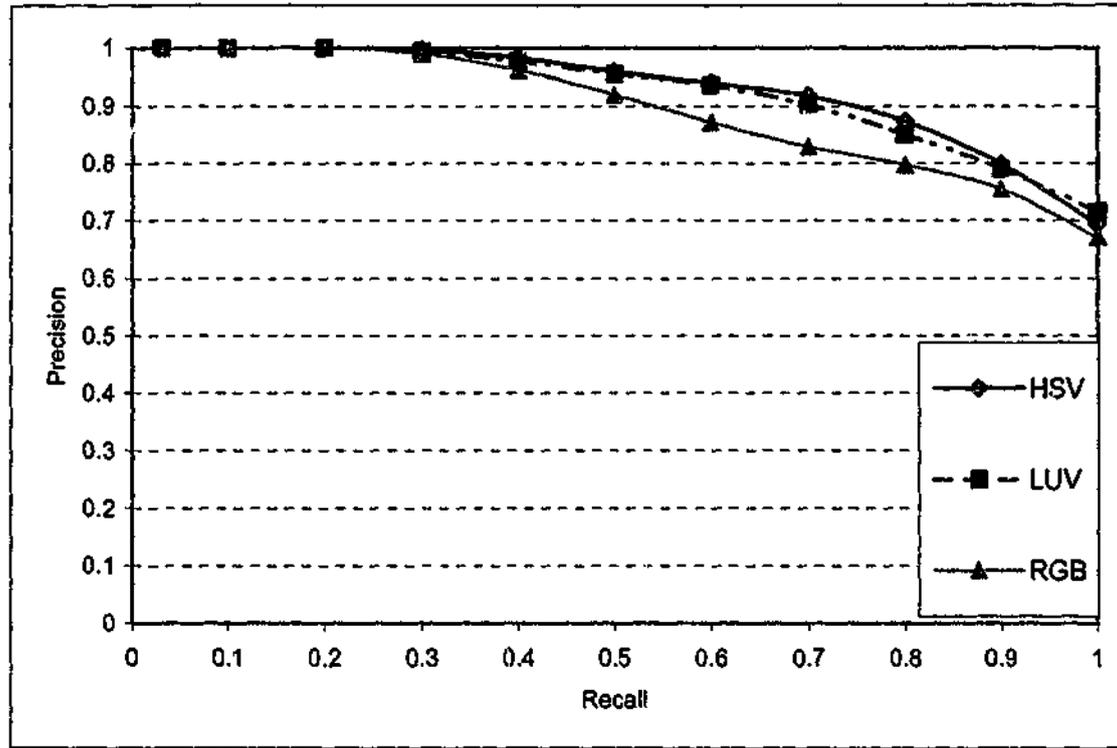


Figure 6.20 Average recall and precision graphs of VQ scheme using different codebooks. The codebooks are generated with common codebook size (2048) and codeword block-size (4x4) but vary in colour space.

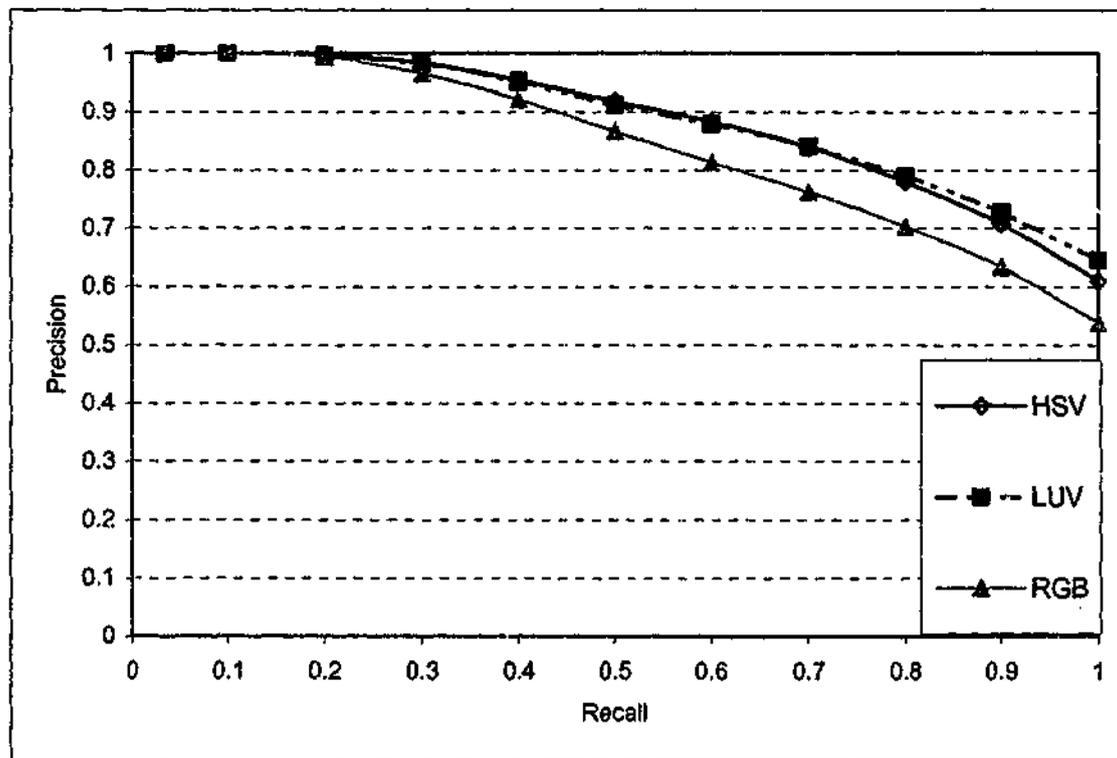


Figure 6.21 Overall performance graphs showing the average retrieval performance of codebooks with different colour spaces.

CONCLUSIONS FOR EXPERIMENT A

Based on the experimental results derived from using database CCD as the test database, we have again shown that retrieval performance of our proposed technique is often similar for codebook sizes between 1024 and 4096. Thus codebook size of 1024 is again recommended to achieve high retrieval efficiency with retrieval effectiveness similar to the codebook size of 4096. In this set of experimental results, codeword block-size again has the most effect on the retrieval performance as compared with codebook size and colour space. Codeword block-sizes of 2x2 and 4x4 pixels blocks produce best retrieval performance when the codebook size is between 128 and 4096. For greater efficiency in codebook generation and image encoding time, codeword block-size of 4x4 is recommended instead of 2x2 pixels blocks. Finally, for the effects of different colour spaces on the retrieval performance of our proposed technique, the experimental results obtained using database CCD has more clearly indicated that the use of HSV and LUV colour spaces produce better retrieval results as compared with RGB. The retrieval effectiveness is similar for both HSV and LUV colour spaces.

The findings we obtained from experiment A on database CCD is similar to the findings obtained using database SCD as the test database. This has reaffirmed our findings on the effects of different codebook sizes, codeword block-sizes and colour spaces have on the retrieval performance of our proposed technique.

6.4.2.3 EXPERIMENT B ON CCD : COMPARISON OF RETRIEVAL PERFORMANCE OF VQ SCHEME TO EXISTING TECHNIQUES

Experiment B is also conducted on database CCD to investigate the retrieval performance of the VQ scheme compared with the three existing colour-based techniques.

6.4.2.4 RESULTS OF EXPERIMENT B ON DATABASE CCD

The same procedure is used to conduct Experiment B on database CCD as the ones used for database SCD in Section 6.4.1.5. Figure 6.22 shows the average recall and precision graphs of the different techniques. The codebook used in the VQ scheme is of codebook size 1024, codeword block-size of 4x4 pixels and colour space HSV. As shown in Figure 6.21, our proposed technique outperforms the other three colour-based techniques.

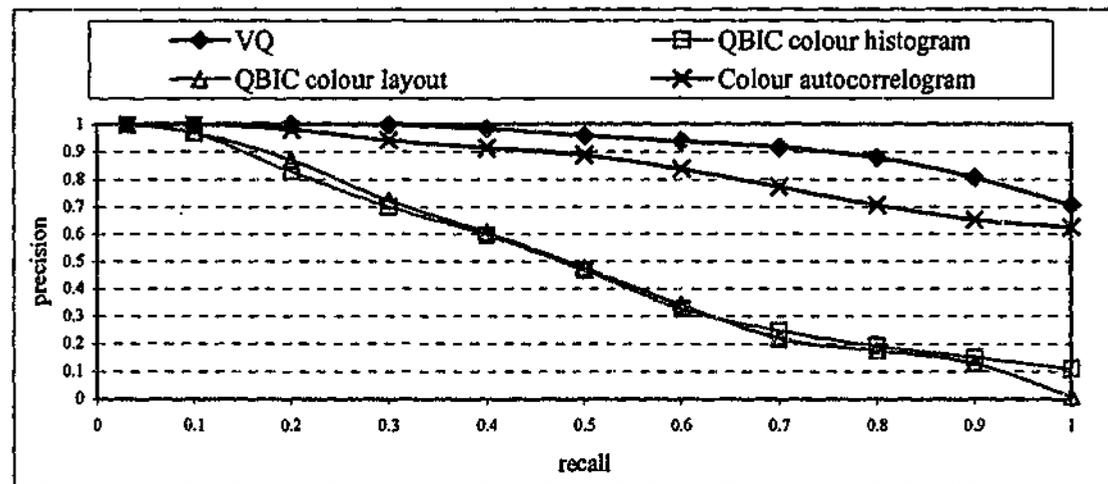
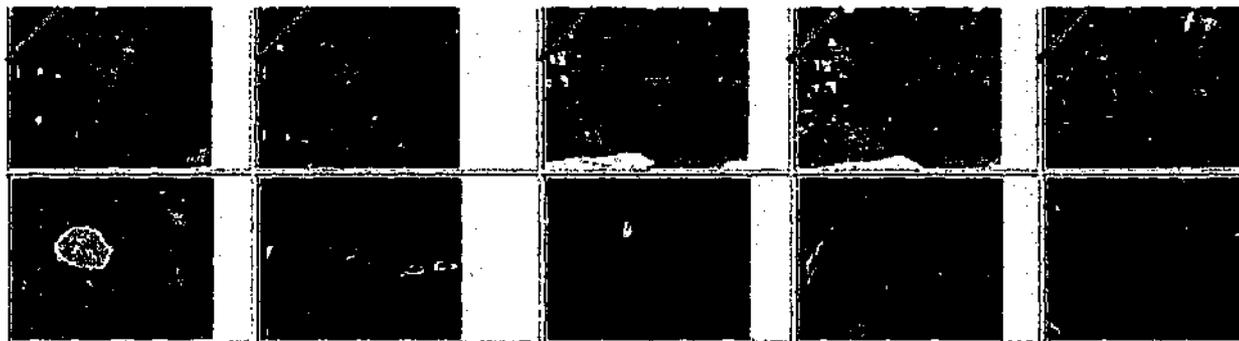


Figure 6.22 Average recall and precision graphs of different retrieval techniques on CCD

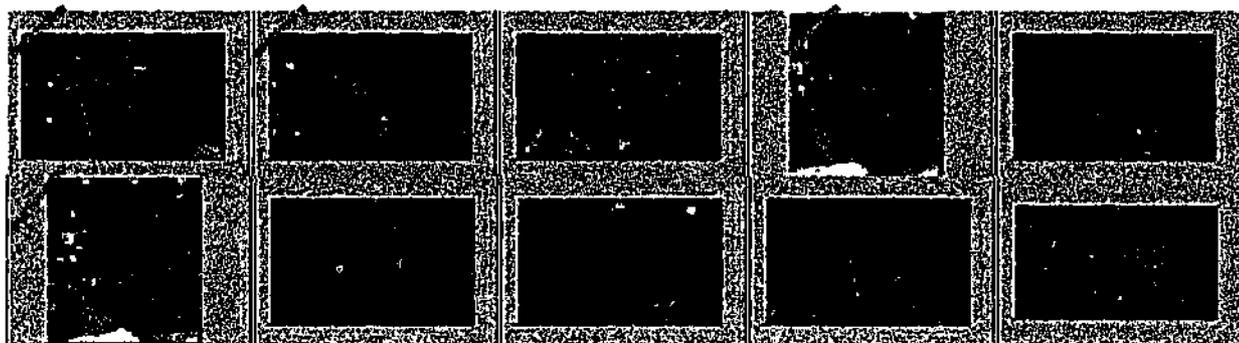
Besides outperforming the two colour-based techniques using in QBIC when using codebook configuration of codebook size 1024, codeword block-size 4x4 and colour space HSV, the VQ-based technique also has better retrieval performance using any 72 codebook configurations generated. Compared with the colour autocorrelogram technique, the VQ technique has better retrieval performance for 52 configurations out of the total of 72 configurations (see Table 6.2). This again demonstrates that the robustness of the VQ-based technique as this technique is not overly sensitive to a particular codebook configuration to perform better than other existing techniques.

Figure 6.23 (a), (b) and (c) show the top 10 retrieved images for three query images using different techniques. The first image in each set of the 12 retrieved

images is the query image. The images with the red dash on the top left corner of the images are the relevant images for the query image. For all the three query images, our proposed technique is capable of retrieving more relevant images compared with the other existing techniques.



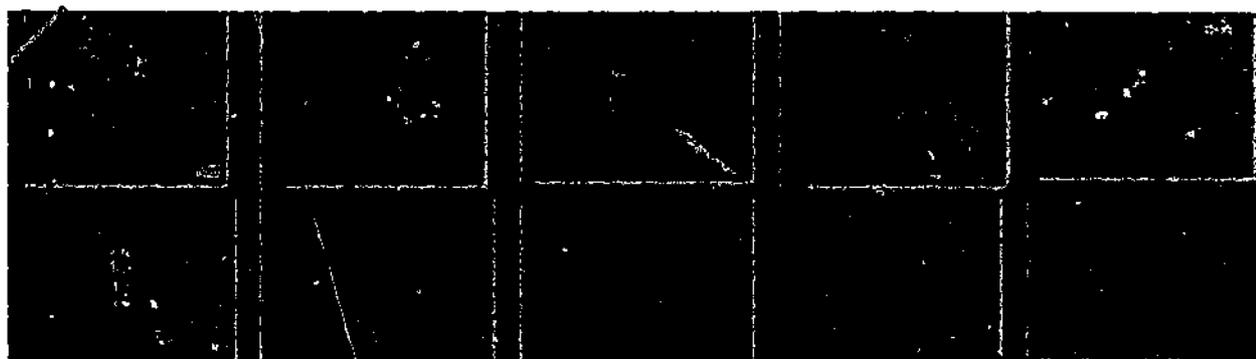
VQ



Colour autocorrelogram

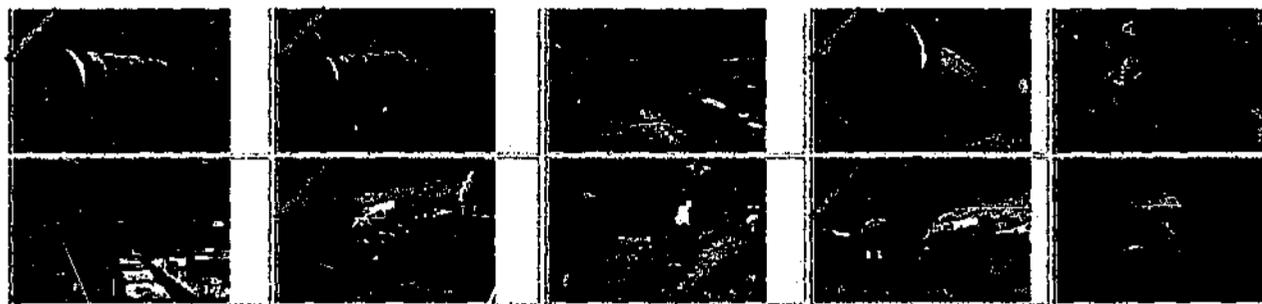


QBIC colour histogram



QBIC colour layout

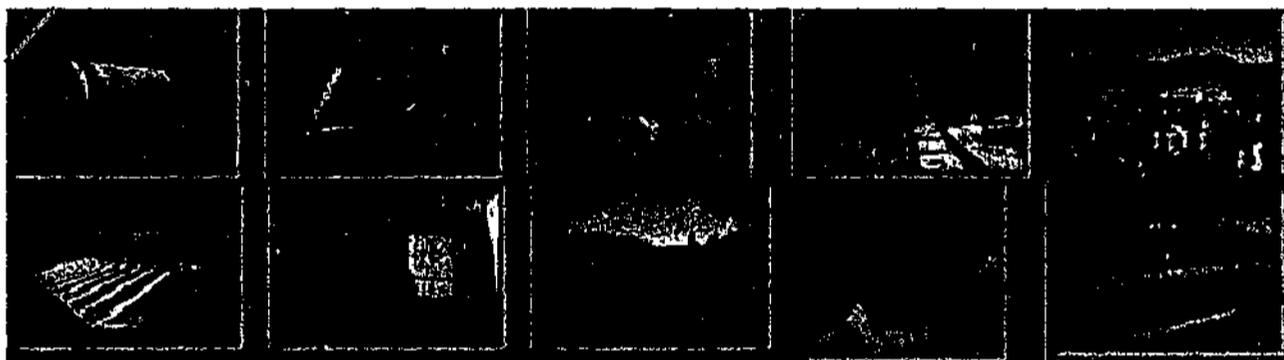
Figure 6.23 (a) Retrieval results of different techniques on database CCD. Each set of images is ranked from top to bottom, left to right according to smallest distance to the largest distance.



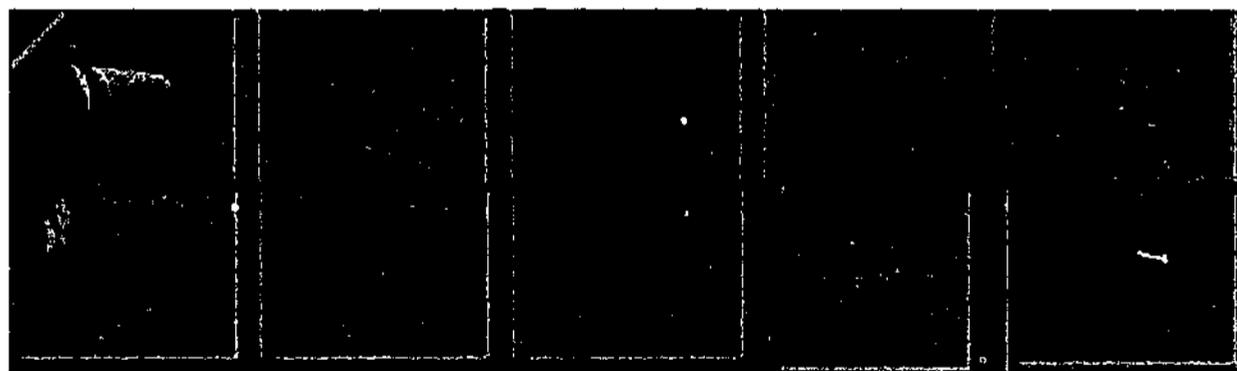
VQ



Colour autocorrelogram



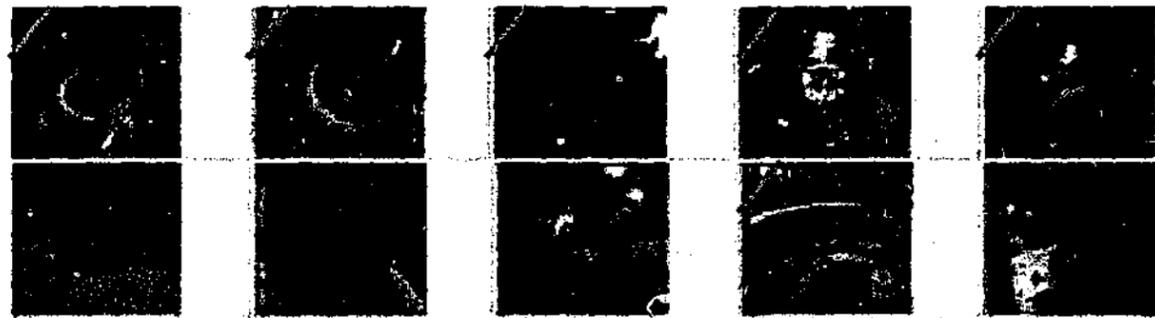
QBIC colour histogram



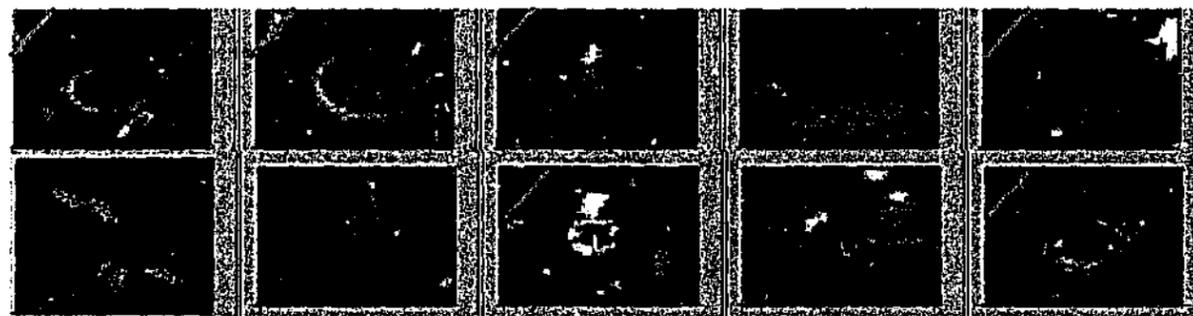
QBIC colour layout

Figure 6.23 (b) Retrieval results of different techniques on database CCD. Each set of images is ranked from top to bottom, left to right according to smallest distance to the largest distance.

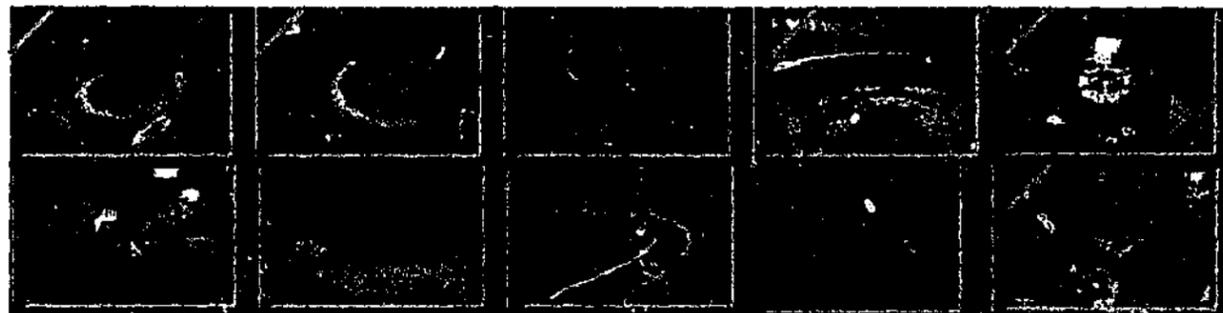
Chapter 6 : Experimental Results to show the effectiveness of VQ based Indexing and Retrieval



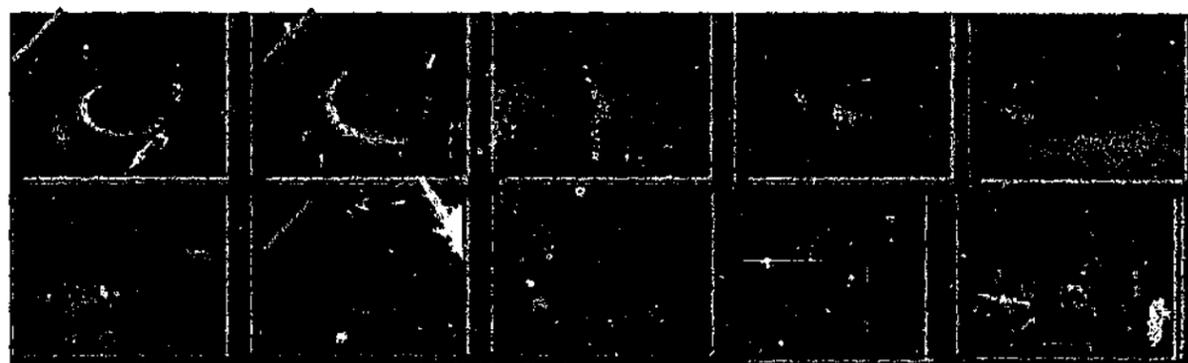
VQ



Colour autocorrelogram



QBIC colour histogram



QBIC colour layout

Figure 6.23 (c) Retrieval results of different techniques on database CCD. Each set of images is ranked from top to bottom, left to right according to smallest distance to the largest distance.

Table 6.2: Codebook configurations used in VQ-based technique that produce retrieval performance worse than the colour autocorrelogram technique

RGB Colour space

Codebook size 128				Codebook size 256				Codebook size 512				Codebook size 1024				Codebook size 2048				Codebook size 4096			
2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16
x	x	x	x		x	x	x		x	x	x												

HSV Colour space

Codebook size 128				Codebook size 256				Codebook size 512				Codebook size 1024				Codebook size 2048				Codebook size 4096			
2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16
		x	x			x	x			x													

LUV Colour space

Codebook size 128				Codebook size 256				Codebook size 512				Codebook size 1024				Codebook size 2048				Codebook size 4096			
2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16	2x2	4x4	8x8	16x16
	x	x	x			x	x																

Legend

x : Marks the codebook configurations used in the VQ-based technique that produce retrieval performance worse than the colour autocorrelogram technique. The retrieval performance of the two techniques is compared based on their recall and precision graphs. For example, x in the first column of the topmost table means that the retrieval performance of the proposed technique using codebook configuration of codebook size 128, codeword block-size 2x2 and colour space RGB is worse than that of colour autocorrelogram technique.

6.5 DISCUSSIONS

In this section, we would like to further discuss some issues or observations, which we have derived by observing and analysing the experimental results presented in the previous sections. The issues are:

- **WHAT ARE THE IMPLICATIONS OF USING THREE DIFFERENT DATABASES TO THE RETRIEVAL PERFORMANCE OF THE VQ SCHEME?**

One of the main objectives in our research is to propose an image indexing and retrieval technique that is more effective than the existing colour-based techniques. The three databases we have used for our testing not only consist of two databases that we have compiled, but also one database, which is a standard MPEG7 test database. From our experimental results compiled from the two different image databases of varying sizes, we have demonstrated that our proposed VQ-based technique has better retrieval performance compared with every one of the existing colour-based techniques we have tested against. Since our proposed technique consistently outperforms the existing colour-based techniques in the three databases, we have also demonstrated that our experimental results are not bias towards a specific test database. Thus, the robustness of our proposed technique is demonstrated.

- **SINCE LUV IS A PERCEPTUAL UNIFORM COLOUR SPACE, WHY DOES THE USE OF THIS COLOUR SPACE ONLY YIELD BETTER RETRIEVAL PERFORMANCE COMPARED WITH THAT OF RGB COLOUR SPACE, BUT NOT HSV COLOUR SPACE?**

By comparing the retrieval performance of the proposed technique using different colour spaces, we observe that the use of LUV colour space has consistently resulted in better retrieval performance compared with RGB colour, but not HSV colour space. The observation of better retrieval

performance when LUV is used compared with RGB is expected since LUV is reportedly built to be a uniform colour space, while RGB is basically built for the ease of displaying graphics onto equipments like computer monitors. The RGB colour space is neither uniform, nor built to represent human intuition. The use of RGB also produces worse retrieval performance compared with HSV since the HSV colour space while not built to be a uniform colour space, it is built to represent human intuition.

After explaining why the use of RGB colour space produces the worst retrieval performance of the three colour spaces, we will next explain why LUV, although being a perceptual uniform colour space, does not yield better retrieval performance compared with HSV. In [74], it is explained that when LUV is used to calculate the distance between two colours, the colour space's property of perceptual uniformity allows the distance calculated to correspond to the human judgement of perceived difference. However, this only applies to colours which are very similar. For example, if the distances calculated in this colour space between colour Green A and Green B is 10; and colour Green A and Green C is also 10, then we can say that Green A and B is as different as Green A and C. However, in another case, if the distances calculated between colour Green A and Blue B is 300; and colour Green A and Red C is 300, we cannot say that Green A and Blue B is as different as Green A and Red C. This is because when comparing colours that are very different, the perceptual uniformity property of the LUV colour space no longer applies. Since our system does not only compares colours that are very similar but also colours that are different, the use of LUV colour space may not produce retrieval performance that are better than the use of HSV. Our experimental results have shown that in many queries, the use of HSV in retrieval can produce performance that as good as or better than the use of LUV colour space.

- **WHAT ARE THE EFFECTS AND IMPLICATIONS OF USING A CODEBOOK GENERATED USING TRAINING IMAGES FROM ANOTHER DATABASE FOR THE IMAGE ENCODING, INDEXING AND RETRIEVAL IN OUR PROPOSED VQ SCHEME?**

In our proposed scheme, it is recommended that the codebook, used for image encoding, indexing and retrieval for a particular image database, should be generated from a good selection of images from the same database. This is to ensure that the database images are encoded and indexed using a codebook that is trained using a set of images that best represents the images in the database. Thus, the best possible retrieval performance can be achieved. To demonstrate the importance of the selection of training images, an experiment is conducted using two codebooks which we have generated for database SCD and CCD. The two codebooks used are of codeword block-size 4x4 pixels blocks, codebook size 1024 and generated in RGB colour space. The difference between them is one codebook is generated using the 60 training images (described in Section 6.4.1.1) for database SCD and the other is generated using the codebook generated using the 60 training images (described in Section 6.4.2.1) for database CCD. Using each of the two codebooks, the following procedure is carried out:

1. Encode the images in database CCD using the codebook.
2. Build VQ histogram for each image.
3. Obtain the retrieval results of the 50 query images with the set of VQ histograms.
4. Plot the average recall and precision graph.

To evaluate how the choice of training images affects the retrieval performance, the two average recall and precision graphs are compared.

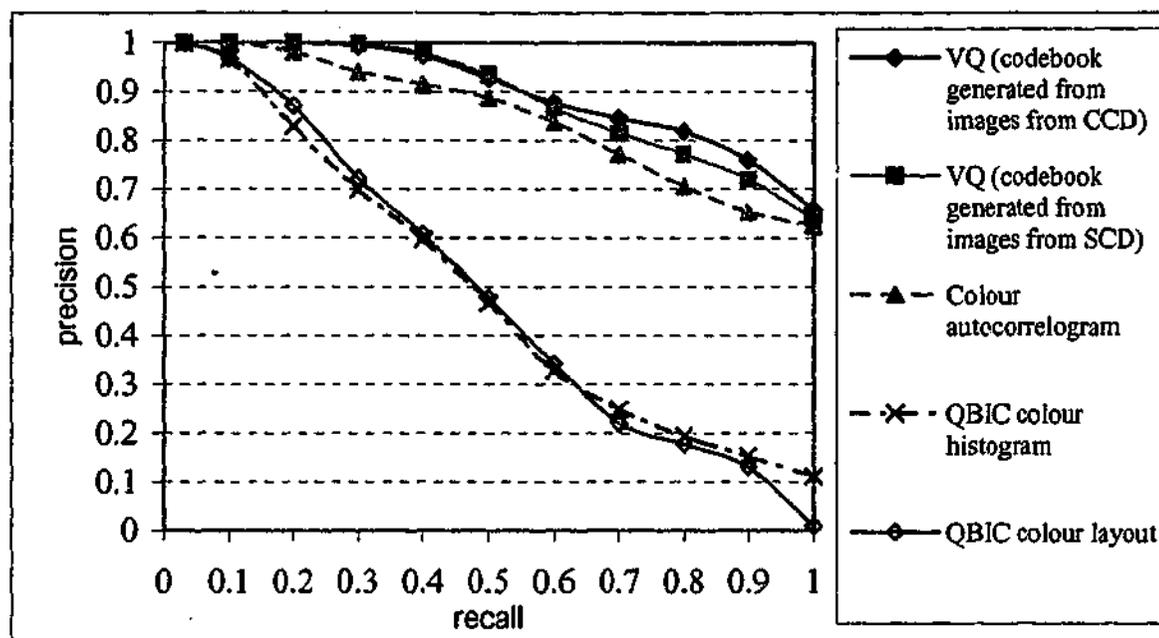


Figure 6.24 Average recall and precision graphs on database CCD. Comparing retrieval performance derived from VQ histograms using codebook generated from database CCD and that derived from VQ histogram using codebook generated from database SCD. Average recall and precision graphs of the three existing techniques are also displayed for comparison purposes.

Figure 6.24 shows the average recall and precision graphs built from the retrieval results of the queries performed on database CCD. The graphs show that the retrieval performance of the VQ technique is better when the codebook is generated from the images in the same database, compared with the case when the codebook is generated from images in database SCD. This shows that the selection of suitable images for codebook generation does affect the retrieval performance of the proposed technique. By comparing the two average recall and precision graphs of the VQ technique to the graphs of the three existing techniques, we observe that even when the images selected for codebook generation are not the ones that best represent the database images, the retrieval performance of the VQ technique can still be better than the retrieval performance of the three existing techniques. This is because the images in databases SCD and CCD have similar colour characteristics. So, since the training images selected to generate the codebook for SCD are

representative of the images in SCD, the codebook generated for SCD can also be used for CCD. Thus databases whose images have similar colour characteristic can share a common codebook when using the proposed technique. These results also imply that when new images (e.g. the images in CCD) are added to a database (e.g. SCD), there is no need to regenerate a new codebook if the new images have similar colour characteristics to the images in the database. This implication is important because it is impractical to use the proposed technique if regeneration of codebook is required whenever some images are added to a database.

- **WHICH CODEBOOK CONFIGURATION SHOULD BE USED IN THE PROPOSED TECHNIQUE FOR THE TECHNIQUE TO WORK EFFECTIVELY AND EFFICIENTLY?**

As stated in Chapter 1, a good CBIR technique should not only be effective in retrieving images, but should also carry out this process efficiently. Therefore both criteria should be taken into account when choosing a suitable codebook configuration for our proposed technique to achieve good retrieval performance. From the observation of the experimental results obtained from experiments performed on Databases SCD and CCD, HSV generally allows our proposed technique to achieve better retrieval effectiveness compared with the other two colour spaces. Therefore, HSV is the most suitable colour space for our proposed technique. For codebook size, we observe that generally, the improvement in retrieval effectiveness of the proposed technique between the codebook sizes 128 to 1024 is significant. However, from 1024 to 4096, the improvement in retrieval effectiveness is very little. In some cases, the retrieval effectiveness using codebook size of 1024 is as good as codebook size of 4096. By considering the effects on retrieval efficiency of the codebook sizes, we recommend using codebook size of 1024 for our proposed technique. This is because codebook size 1024 allows our proposed technique

to be more efficient in retrieval compared with 4096. For codeword block-size, we observe that the retrieval effectiveness of our proposed technique using codeword block-sizes 2x2 and 4x4 pixels is always better than 8x8 and 16x16 pixels. Comparing the retrieval effectiveness of our proposed technique using codeword block-sizes 2x2 and 4x4, they are very similar. Again, considering the effects on the retrieval efficiency of our proposed technique using these two codeword block-sizes, we recommend using codeword block-size of 4x4. This is because codeword block-size of 4x4 allows our proposed technique to be more efficient in retrieval compared with 2x2. Besides, block-size of 4x4 can capture more spatial relationships among the colour pixels in the images compared with block-size of 2x2.

Overall, we recommend codebook configuration of codebook size 1024, codeword block-size 4x4 pixels and colour space HSV to be used for our proposed technique because this configuration is the best compromise between good retrieval effectiveness and retrieval efficiency.

6.6 CONCLUSIONS

In this chapter, we have described the experiments carried out at different stages of our research to investigate how our proposed technique measure-up in terms of retrieval effectiveness to three other existing colour-based image retrieval techniques. Two of the existing techniques used for the comparison are the colour histogram and colour layout techniques. They are adopted in the commercial image indexing and retrieval software, QBIC. The third existing technique used as a comparison is the colour autocorrelogram technique, which is widely reported as one of the more effective colour-based techniques [34]. The experimental results have shown that our proposed technique outperforms all the three existing techniques.

CHAPTER 7

IMPLEMENTATION & EXPERIMENTAL RESULTS OF EFFICIENT SEARCH METHOD FOR IMAGE ENCODING

7.1 INTRODUCTION

In VQ image encoding, the operation to search for the best match codeword is a process that is extremely computationally intensive. In situations where users supply their query images instead of using the ones in the database, the query images must be encoded before they can be indexed and queried. As explained in Chapter 5, the time taken to perform online image encoding using the full-search method will not meet the satisfactory response time for an online CBIR application. Thus to make our proposed VQ-based scheme practical, better searching method in image encoding must be used.

In Chapter 5, we have found that among the different multidimensional data search methods reviewed, the tree-based methods are capable of achieving relatively higher level of efficiency for data of small or moderate dimensional size. From the experimental results in Chapter 6, we have shown that the VQ-based CBIR technique works more effectively when the image vector dimension is of

moderate size. Therefore we have chosen a tree structure to improve the efficiency of our image encoding process.

The tree to be described in this chapter has two distinct characteristics that are different from the m-ary tree structure described in Section 5.6. Firstly, the number of child nodes that each node is linked to is not fixed. For example, one node may be linked to only three nodes at the next level while another node may be linked to six nodes. The second difference is that a node may be linked to more than one parent node. The reason for adopting a tree of such characteristics instead of the m-ary tree structure is because of the distribution of the codewords. For m-ary tree to be efficient, the multidimensional vectors to be searched should be evenly distributed in the search space. However, the distribution of the image vectors for real images are usually not evenly distributed. Studies have shown that vectors in images are likely to be unevenly clustered throughout the search space [30, 31]. Thus the codewords generated for our databases are likely to be unevenly clustered too. Since our codewords are of such nature, it will be difficult to keep the number of links between each node and their nodes in the next level fixed and yet still achieve an efficient and accurate search. To allow the tree structure to be effective for our kind of application, the links between the nodes of two successive levels will not be kept constant.

Besides using a tree structure to organise our codewords, we have also precalculated and stored information in each codeword that is required for the similarity comparison to achieve greater efficiency.

The organisation of this chapter is as follows. Section 7.2 describes how our search space can be partitioned by a set of codewords. Understanding the principle of space partitioning provides the foundation to the understanding of how the tree structure is built. Section 7.3 describes how the tree structure is constructed for our application. Next, the encoding algorithm using the tree structure is presented in Section 7.4. In Section 7.5, we discuss the accuracy of the search using our tree structure. Section 7.6 describes the kind of information

in the codewords that can be precalculated and stored to improve the search algorithm. The way of how this information is used is also described in this section. Section 7.7 discusses the effects of different colour spaces on the efficiency of our search algorithm. Finally, the experimental results and the conclusions for this chapter are presented in Section 7.8 and 7.9 respectively.

7.2 SPACE PARTITIONING BY CODEWORDS

In the encoding process, the codebook is searched to find the best match codeword for each image vectors. If there are N codewords and each codeword is a K -dimensional vector, then these codewords can be perceived as N points lying in a K -dimensional search space. For our application, the dimension K is equal to codeword block-size multiply by 3 colour channels. When the full-search method is used to find the best match codeword for a query image vector, the distance between every codeword point and the query point must be calculated. The need to visit every codeword point in the search space has caused the encoding process to be very computational intensive and time consuming. However, if we can quickly zero-in to the area where the best match codeword point lies in the search space without having to visit every codeword point, the time required to carry out the encoding process can be reduced. A way to do this is to make use of a tree structure where each level of the tree represents the entire search space divided into a different number of partitions. Each leaf-node at the bottom level of the tree represents a partition that a distinct codeword in the codebook resides in. The number of partitions that the search space is divided into decreases as we ascend up the levels of tree. A tree node represents each of these partitions. The links between the nodes in different levels show the overlapping of the partitions in the parent level with the partitions in the child level. By traversing down such a tree, some partitions in the search space, which does not contain the best match codeword to the query vector, may be filtered out. As a result, the encoding process may be carried out more efficiently. A possible way to divide the search

space into partitions that are suitable for such a tree structure is by using the intermediate codebooks generated at the codebook generation process.

In Chapter 3, the algorithm of codebook generation is discussed. The following are the iteration steps of the LBG algorithm using the splitting technique:

Initialisation:

1. Decide the size of the codeword which is normally a square block of $d \times d$. Divide the training image into non-overlapping training vectors whose size is the same as the codewords size decided.
2. A vector which is the average of all the training vectors is calculated. This vector, which is the centroid of all the training vectors, is the codeword of the codebook of size 1.
3. The splitting technique is used to produce two vectors from each codeword to act as the initial vectors (IVs) for generating the codebook of the next level. Hence, if the present codebook size is s , the codebook size of the next level is $2s$. Splitting is done by adding a randomly generated perturbation vector to generate a new vector (NV). The perturbation vector can be different for each pixel of the codeword. So IVs of the next level consist of codewords of the current level and NVs.

Clustering:

4. Find the closest IV for each training vector based on the squared Euclidean distance between each training vector and each of IVs
5. Training vectors closest to the same IV are classified into the same group or cluster. So we have one cluster corresponding to each IVs.
6. If a cluster does not have any training vector, the corresponding IV is replaced by a randomly chosen training vector from the cluster with the largest number of training vectors and new clusters have to be formulated again as in steps 4 and 5. Otherwise, continue.

7. Determine the centroid vector for each cluster. The centroid vector (CV) replaces the IV of the cluster to represent the cluster.

Termination Checking:

8. Recalculate the distance between each training vector and each CV to determine if any training vector should be grouped into another training cluster instead of the one it previously belongs to. If there is any chance of cluster grouping for any training vector, go to Step 4 with CVs as new IVs. If there is no cluster change, the CVs are the codewords for codebook at this level.
9. If a termination condition for codebook generation is met, then the codebook generation process stops and the above codewords are used as final codebook. Otherwise, go to Step 3. There are two stoppage conditions. Condition one is met when number of codewords reaches the pre-determined number. Condition two is met when the average distance between each training vector and its corresponding codeword is below certain threshold.

From the steps above, we can see that to derive at the target codebook size we desire, a set of codebooks with smaller sizes are also generated as part of the process. This set of smaller codebooks are of 2^b sizes, where $b \in 0, 1, \dots, B-1$; and B is the number of bits required to index the number of codewords in the target codebook. For example, if the target codebook size is 256, the bit-number B is 8. To generate the codewords in each codebook, the algorithm divides the entire R^K space into a number of non-overlapping partitions. The number of partitions is equal to the codebook size desired at that iteration and the point in the centre of each partition represents a codeword generated for that codebook. For example, if a 2-bit codebook is desired, the entire space is divided into four partitions. Figure 7.1(a) shows an example, where the black dots represent the positions of the codewords in a R^2 space and the dotted lines show the boundaries of the four partitions. Figure 7.1(b) shows another example of the space partitions

for a 3-bit codebook. In this figure, the space is divided into eight partitions. The white dots represent the positions of the codewords in the space and the lines represent the boundaries of the eight partitions. An important property of the space partitions is any point which resides in a particular partition definitely has a smaller distance to the codeword point in that partition compared with any other codeword point of that codebook. Thus the codeword point can also be used to represent that partition.

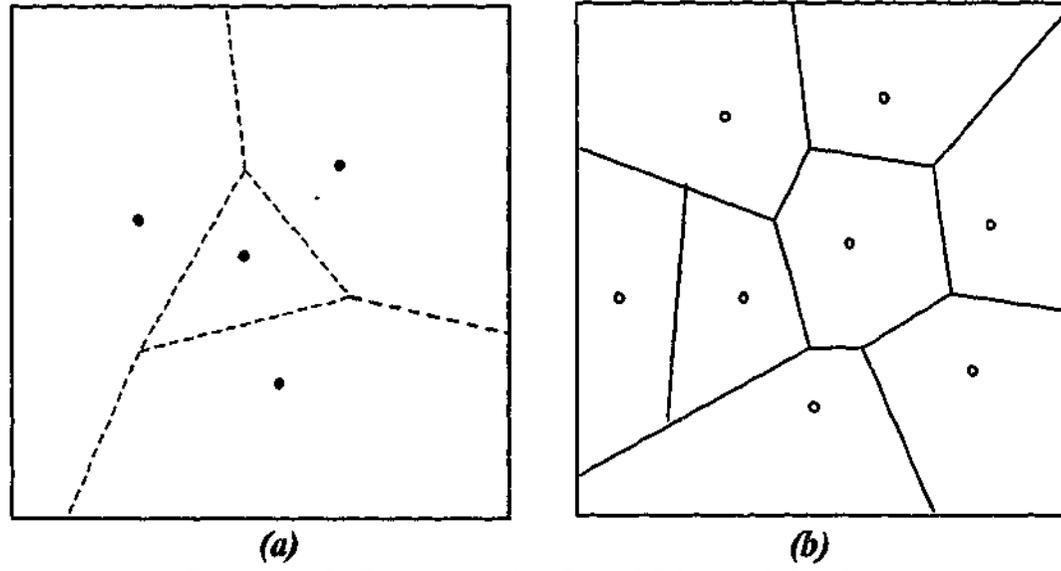


Figure 7.1 Partitions for 2 and 3 bit codebooks

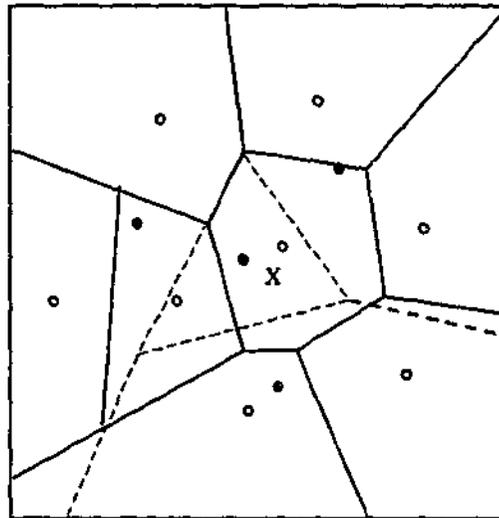


Figure 7.2 Search space reduction using codebook with smaller bit number

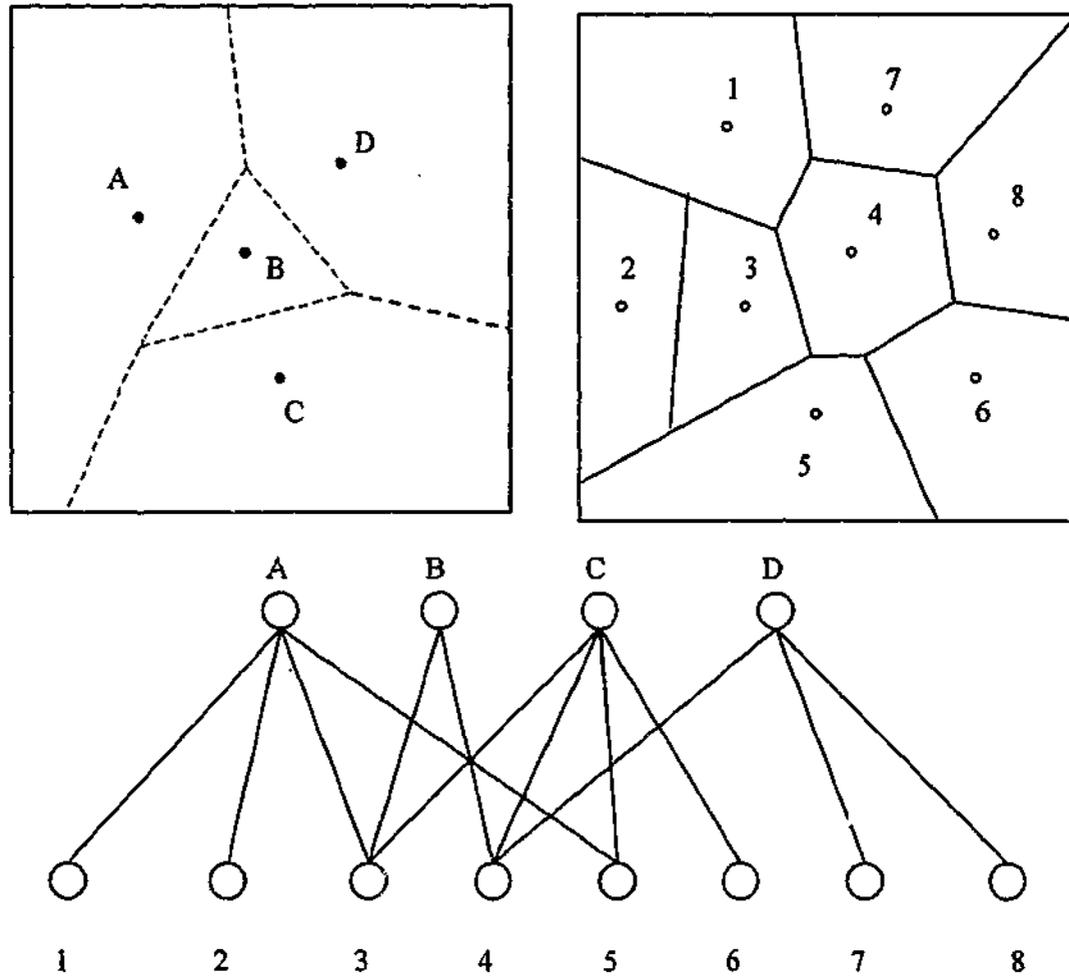


Figure 7.3 Overlapping of partitions shown in a tree

Using this space partition property, some partitions in the search space, which do not contain the codeword that is best match to the query image vector, could be identified by first searching through the codebooks with smaller bit number. We will illustrate how this can be achieved with Figure 7.2. Figure 7.2 shows the overlapping of the partitions derived from the 2-bit and 3-bit codebooks shown in Figure 7.1. To determine if the next codeword is more similar to a query image vector than the current most similar codeword, the similarity operation consists of two parts. First, the distance between the next codeword and the query image vector is calculated using a distance metric. Next, this distance is compared with the current most similar distance. Let this similarity operation be SO. Thus using the full-search method, to find the most similar codeword in 3-bit codebook for the query image vector X, 8 SO's must be carried out. However, if a search is carried out first with the 2-bit codebook, we will be able to identify that the query

image vector is in the centre partition of the 2-bit codebook. Thus, only the 2 partitions in the 3-bit codebook need to be searched. Overall, only 6 SO's, 4 for the 2-bit codebook and 2 for the 3-bit codebook, are carried out. With this concept, we can use one or more codebooks of smaller bit-number to filter out the partitions in the search space where the best match codeword in the target codebook does not lie. Although this example is shown with a 2-dimensional search space, the same concept is applicable to higher dimensional search space. The overlapping between the partitions in Figure 7.2 can also be represented as a tree, as shown in Figure 7.3. In the tree, a node in an upper level is joined to node in the lower level when their partitions overlap. For example, partition 3 of the 3-bit codebook overlaps with partitions A, B and C of the 2-bit codebook. Thus in the tree, there is a link joining nodes 3 and A, 3 and B, and lastly 3 and C. From this example, we can also observe that this tree-structure is different from the traditional tree-structure because a child node can have multiple parent nodes.

7.3 CONSTRUCTION OF THE TREE STRUCTURE

In the previous section, we have stated that multiple codebooks of smaller bit-number may be used to filter out more partitions in the search space where best match codeword in the target codebook does not lie. For our application, all the codebooks, except the codebook with one codeword, generated at the intermediate stages to obtain the target codebook, are used. Thus, if the target codebook is of 8 codewords (3-bit codebook), the 1-bit and 2-bit codebooks are also used to form the tree. The first step to form the tree is to arrange the codebooks into a hierarchical structure according to their codebook sizes. In this structure, the 1-bit codebook occupies the top level. The codebook size increases as we descend down the level of the structure. The target codebook occupies the bottom level of the structure. The top level of structure is denoted as level-1, the level below level-1 as level-2 and so on. At each level, a node is used to represent each codeword in the codebook of that level.

The second step is to establish the links between the nodes in adjacent levels. To build the links, the vectors in the training images are used. The training vectors are suitable vectors for simulating the various paths the query vectors are likely to traverse down the tree because they are representative of the variety of vectors that can be found in the database images. Thus by finding the best match codeword in each level to each training vector, and building a link between the nodes in every parent and child levels, a tree structure, which consists of various paths a query vector may take, is formed. The steps of this procedure are described as follows:

1. Starting at level-1 of the hierarchical structure, calculate the distance between the training vector and each codeword represented on this level. Find the best match codeword.
2. Go to the next level of the hierarchical structure and again, calculate the distance between the training vector and each codeword represented on this level. Find the best match codeword.
3. Construct a link between the two nodes representing the codewords in the parent and the child levels. If this is not the bottom level of the hierarchical structure, go to step 2. Otherwise, with the next training vector in the training set, go to step 1. Terminate the procedure when the links for all the training vectors in the training set are constructed.

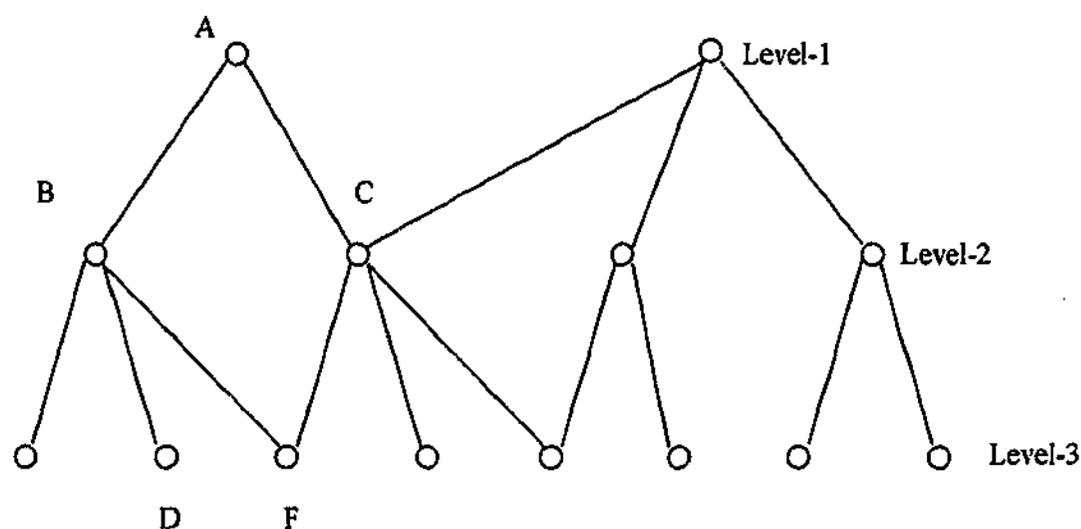


Figure 7.4 Example to illustrate the construction of a 3-level tree

To illustrate the above procedure, let us look at Figure 7.4. Figure 7.4 shows a 3-level hierarchical structure after the links between the nodes have been established. The nodes at level-1, 2 and 3 represent the codewords in codebooks of bit-number 1, 2 and 3. The 3-bit codebook is the target codebook. To build the links A-B-D using the procedure described above, there must be at least one training vector that has codewords A, B and D best match to it, as compared with the codewords in their respective codebook. Similarly, to build the links A-C-F, there must be a training vector that has codewords A, C and F best match to it, as compared with the codewords in their respective codebook.

7.4 ENCODING ALGORITHM BASED ON TREE

With the set of N codebooks in the tree generated and the links between the codewords in different levels established, the fast encoding algorithm can be carried out as follows:

1. Starting at level-1 of a N-level tree, calculate the distance between the query image vector and each codeword at this level. Find the best match codeword.
2. With reference to the links in the tree, find the set of codewords at the next level that are linked to the best match codeword.
3. Go to the next level. Calculate the distance between the query image vector and each codeword in the set found on the previous step. Find the best match codeword.
4. If this is level-N of the tree, encode the query image vector with the index of the best match codeword found. Otherwise, go to step 2.

With the encoding algorithm illustrated above, better image encoding efficiency may be achieved.

7.5 ACCURACY OF THE TREE-STRUCTURED ALGORITHM

After describing the construction of the tree-structure and the encoding algorithm using the structure, the accuracy of the encoding images using this algorithm is discussed in this section. In theory, the encoding of images using this structure is as accurate as the full-search algorithm, which ensures that every image vector is encoded with the codeword that is of best match in the codebook, thus, resulting in an optimal solution. However, in practice, encoding image using the tree-structured algorithm may not always result in an optimal solution. The reason for this is due to the way the tree structure is constructed. As described above, the links between the nodes in every parent and child levels of the tree are established by repeatedly finding the best match node to each training image vector in each level of the tree and linking them together. If all the possible links between the nodes in every parent and child levels of the tree are identified with the training image vectors, then the best match codeword in the codebook for any image vector can be found using the tree-structure. However, it is possible that some links may not be identified with the training image vectors. An example of such a case is shown in Figure 7.5.

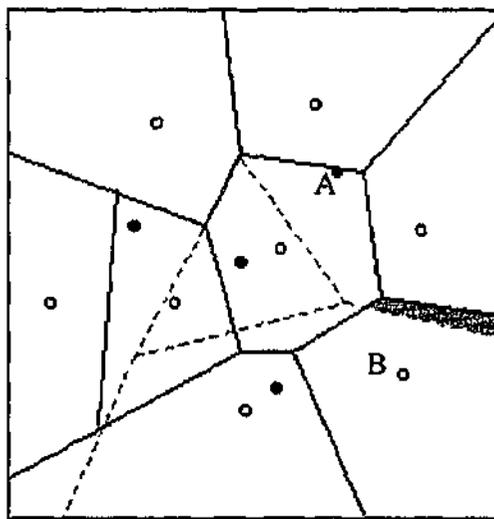


Figure 7.5 An example to illustrate a case where a link between the nodes in parent and child levels of the tree are not established

If no vector in the training images appears in the shaded region of the Figure 7.5, the link between the partition denoted by the filled dot A in the parent level and the partition denoted by the unfilled dot B in the child level will be lost. In such situation, if there are vectors in the image to be encoded that appear in the shaded region of Figure 7.5, the tree-structured encoding algorithm will not consider the codeword denoted by the unfilled dot B as the most similar. The algorithm will instead search from other nodes (codewords) where links have been established. This will lead to a sub-optimal solution. However, if there is no vector in the image to be encoded that are from the shaded region of Figure 7.5, the solution of encoding is still optimal. Besides having an optimal solution for such situation, having smaller number of links between every parent and child levels of the tree also leads to lower encoding time.

7.6 FURTHER IMPROVEMENT TO ENCODING SPEED

In the encoding algorithm presented in Section 7.4, a full-search among all the codewords at level-1 is required to determine the best match to the query image vector. Then at each subsequent level, a full-search among a subset of codewords of that level is again required to determine the best match to the query image vector. Thus numerous computation of the squared Euclidean distance between codewords and the query image vector still exists in the algorithm above. If we can improve the speed of the distance computation, the overall performance of the above algorithm will be enhanced.

We propose to shorten the computation time of each squared Euclidean distance, $d()$, between the query image vector (X) and the codeword (C) by:

1. Identifying data in the distance metric that can be calculated offline.
2. Calculating and storing the data identified in step 1.
3. Simplifying the computation complexity of the distance metric with the pre-calculated data.

The details of the approach are illustrated as follows:

$$\begin{aligned}
 d(X, C) &= \sum_{i=1}^k (x_i - c_i)^2, \\
 &= \sum_{i=1}^k (x_i^2 - 2x_i c_i + c_i^2) \\
 &= \sum_{i=1}^k x_i^2 + \sum_{i=1}^k c_i^2 - 2 \sum_{i=1}^k x_i c_i \quad (1)
 \end{aligned}$$

Employing Winograd's identity [98], the third summation term in expression (1) can be expressed as

$$\begin{aligned}
 \sum_{i=1}^k x_i c_i &= \sum_{i=1}^{k/2} (x_{2i-1} c_{2i-1} + x_{2i} c_{2i}) \\
 &= \sum_{i=1}^{k/2} [(x_{2i-1} + c_{2i})(x_{2i} + c_{2i-1}) - x_{2i-1} x_{2i} - c_{2i-1} c_{2i}] \\
 &= \sum_{i=1}^{k/2} (x_{2i-1} + c_{2i})(x_{2i} + c_{2i-1}) - \sum_{i=1}^{k/2} x_{2i-1} x_{2i} - \sum_{i=1}^{k/2} c_{2i-1} c_{2i} \quad (2)
 \end{aligned}$$

Thus by substituting expression (2), the squared Euclidean distance can be expressed as:

$$\begin{aligned}
 d(X, C) &= \sum_{i=1}^k x_i^2 + \sum_{i=1}^k c_i^2 \\
 &\quad - 2 \left[\sum_{i=1}^{k/2} (x_{2i-1} + c_{2i})(x_{2i} + c_{2i-1}) - \sum_{i=1}^{k/2} x_{2i-1} x_{2i} - \sum_{i=1}^{k/2} c_{2i-1} c_{2i} \right] \\
 &= \left[\sum_{i=1}^k x_i^2 + 2 \sum_{i=1}^{k/2} x_{2i-1} x_{2i} \right] \quad (3)
 \end{aligned}$$

$$+ \left[\sum_{i=1}^k c_i^2 + 2 \sum_{i=1}^{k/2} c_{2i-1} c_{2i} \right] \quad (4)$$

$$- 2 \sum_{i=1}^{k/2} (x_{2i-1} + c_{2i})(x_{2i} + c_{2i-1}). \quad (5)$$

We can express expressions (3) and (4) more generally with function $f()$ and vector $W = (w_1, w_2, \dots, w_k)^t \in R^k$ as:

$$\begin{aligned} f(W) &= \sum_{i=1}^k w_i^2 + 2 \sum_{i=1}^{k/2} w_{2i-1} w_{2i} \\ &= \left[\sum_{i=1}^{k/2} (w_{2i-1}^2 + w_{2i}^2) \right] + 2 \sum_{i=1}^{k/2} w_{2i-1} w_{2i} \\ &= \sum_{i=1}^{k/2} (w_{2i-1} + w_{2i})^2 \end{aligned}$$

We can also express expression (5) more generally with function $g()$, vector W and vector $Z = (z_1, z_2, \dots, z_k)^t \in R^k$ as:

$$g(W, Z) = \sum_{i=1}^{k/2} (w_{2i-1} + z_{2i-1})(w_{2i} + z_{2i}).$$

By substituting functions $f()$ and $g()$,

$$d(X, C) = f(X) + f(C) - 2g(X, C).$$

Thus if C_{\min} is the most similar codeword to image block, it will satisfy the following equation:

$$d(X, C_{\min}) = \min\{f(X) + f(C) - 2g(X, C), C \in \text{codebook}\} \quad (6)$$

Equation (6) can also be defined as follows:

$$\begin{aligned} d(X, C_{\min}) &= \min\{f(X) + f(C) - 2g(X, C), C \in \text{codebook}\} \\ &= f(X) + \min\{f(C) - [g(X, C) + g(X, C)], C \in \text{codebook}\} \quad (7) \end{aligned}$$

From expression (7), we can see that $[g(X, C) + g(X, C)]$ is used instead of $2g(X, C)$. The purpose of using an additional operator rather than a multiplication operator is because it is relatively faster to perform an addition rather than a multiplication in the computer. We also can see that $f(X)$ is a constant which does not affect the search result of the codeword that is the most similar to image block X . Thus, we can use expression

$$f(C) - [g(X, C) + g(X, C)] \quad (8)$$

as the new distance calculation method.

Since $f(C)$ can be calculated offline, the only part of the new calculation method needed to be calculated online is $- [g(X, C) + g(X, C)]$. Comparing the number of arithmetic operations needed in online calculation of distance between an image block and a codeword (see Table 7.1), we can see that the basic method requires more operations than the new calculation method. The difference in the computation time is further amplified by the difference in the number of multiplication operations needed in the basic and the new methods to calculate Euclidean distance. This is because for any modern computer, the time required for a multiplication is much greater compared with an addition or subtraction operation.

Euclidean Distance methods	Subtractions	Additions	Multiplications
Basic	k	k-1	k
New	1	$k+(k/2)$	$k/2$

Table 7.1 Number of arithmetic operations needed for calculating Euclidean distance between an image block and a codeword using the basic and the new methods.

The full proposed encoding algorithm is expressed as follows:

1. Starting at level-1 of a N-level tree, calculate the distance between the query image vector and each codeword at this level **using the new distance calculation method**. Find the best match codeword.
2. With reference to the links in the tree, find the set of codewords at the next level that are linked to the best match codeword.
3. Go to the next level. **Using the new distance calculation method**, compute the distance between the query image vector and each codeword in the set found on the previous step. Find the best match codeword.

4. If this is level-N of the tree, encode the query image vector with the index of the best match codeword found. Otherwise, go to step 2.

7.7 EFFECTS OF COLOUR SPACES ON ENCODING EFFICIENCY

In the implementation of the proposed encoding algorithm, its efficiency is affected by the choice of the colour spaces. This is because the way the colour spaces are modelled may affected the calculation of the Euclidean distance. Since we have concentrated on RGB, LUV and HSV colour spaces in our implementation, we will base our discussion on these three colour spaces.

The algorithm is more efficient if RGB or LUV colour spaces are used for processing the image as compared with when HSV colour space is used. In RGB or LUV colour spaces, the colour differences in each colour channel are arranged in a linear manner. For example, as shown in Figure 7.6, the redness in the R channel increases gradually from 0 to 255. In this colour arrangement, a red of intensity 0 in the R channel is definitely very different from a red of intensity 255.

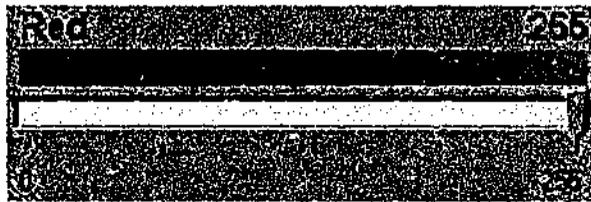


Figure 7.6 The intensity of the redness in the R channel of RGB colour space

With such colour arrangement of the colour channels in the RGB and LUV colour spaces, we can simply calculate the squared Euclidean distance, D , between two image vectors X and C using the metric:

$$D(X, C) = \sum_{i=1}^k [(x_i^{CH1} - c_i^{CH1})^2 + (x_i^{CH2} - c_i^{CH2})^2 + (x_i^{CH3} - c_i^{CH3})^2] \quad (9),$$

where k is the number of vector dimension;

$CH1$, $CH2$ and $CH3$ are R, G and B channels respectively for RGB colour space;

or

$CH1$, $CH2$ and $CH3$ are L, U and V channels respectively for LUV colour space.

For HSV colour space, only S and V channels are arranged in the linear manner described above. For H channel, the values in this channel are arranged in a circular manner as shown in Figure 7.7. A hue with an intensity of 354 is perceptually more similar to a hue of 0 compared with a hue of 90. Due to this, modification to the metric in equation (9) is necessary in order for the distance calculated to accurately reflect the perceptual similarity between the two image vectors.

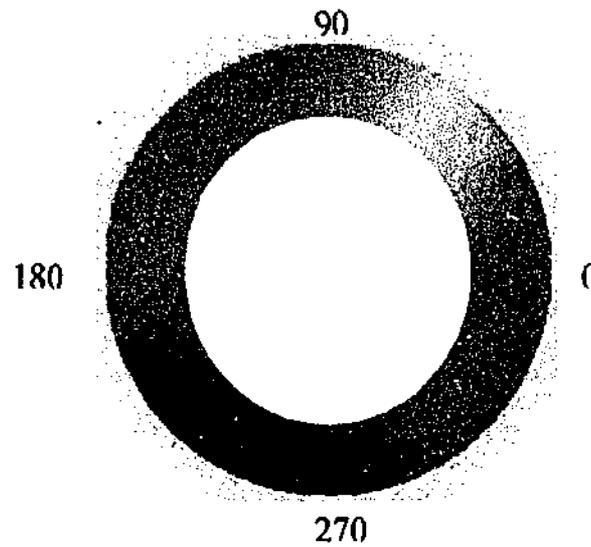


Figure 7.7 Colour arrangement in H channel of HSV

The modified squared Euclidean distance metric [20, 74] is as follows:

$$D^2(X, C) = \sum_{i=1}^k [(\Delta H)^2 + (x_i^S - c_i^S)^2 + (x_i^V - c_i^V)^2] \quad (10)$$

where k is the number of vector dimension, and

$$\Delta H = \min(|x_i^H - c_i^H|, 360 - |x_i^H - c_i^H|).$$

By making the modification of ΔH in equation (10), we can more accurately reflect the perceptual similarity between two hues. For example, using the hue values of 0, 90 and 354 in the previous example, the distance between hue 354 and hue 0 smaller than the distance between hue 90 and hue 0. With the additional $2k$ subtraction operations, $2k$ modulus operators and k min operations in the distance calculation when HSV is used as the colour space, the distance calculation is more

computationally intensive compared with when RGB or LUV colour space is used. Besides that, due to the colour arrangement of the H channel, computation and storing of information offline for the H channel using the new distance calculation method are also not feasible. Both factors increase the amount of online computation of the encoding process when HSV colour space is used. Thus, the overall encoding efficiency for the proposed algorithm is better when RGB or LUV colour space is used compared with when HSV colour space is used.

7.8 EXPERIMENTAL RESULTS

To evaluate the performance of the proposed algorithm for our application, three experiments are carried out. The first experiment is to evaluate the performance of the algorithm in terms of encoding efficiency. The second experiment is to evaluate the performance of the algorithm in terms of encoding accuracy. The third experiment is to investigate the impact of the algorithm on the retrieval effectiveness of our proposed VQ-based technique.

7.8.1 ALGORITHM PERFORMANCE - ENCODING EFFICIENCY

In order to evaluate the encoding efficiency and accuracy of the new algorithm, we have implemented the algorithm. In this experiment, to build the different levels of codebooks for the tree structure, the 60 training images of database SCD selected for generating codebooks in Chapter 6 are used. Since the codebook size and codeword block-size recommended for our proposed VQ-based technique in Chapter 6 are 1024 and 4x4 pixels block respectively, we have used this configuration in this experiment. As the target codebook is 1024, the tree used in the experiment consists of ten levels. Thus the nodes in each level in the tree represent the codewords in its corresponding codebook. For example, the level-1 of the tree represents the codewords in the 1-bit codebook while the level-10 of the tree represents the codewords in the 10-bit codebook. To build the links

between the nodes in every parent and child levels in the tree, the image vectors in the training images are used. To employ the new distance calculation method presented in Section 7.6 in the algorithm, we have also computed and stored the information for the codewords in each level of codebooks that can be calculated offline.

In this experiment, 15 images of size 256x256 pixels are selected as test images. They include images like Lena, Baboon and Pepper which are standard test images in this field of research. The other 12 images are selected from database SCD. A variety of test images are used to test the robustness of the proposed method.

In order to evaluate the performance of the new encoding algorithm (TS_ND), the algorithm is compared with six other algorithms. The six algorithms are:

1. Full-search algorithm using Euclidean distance metric (FS_ED).
2. Full-search algorithm using squared Euclidean distance metric (FS_SqED).
3. Full-search algorithm using new distance calculation method (FS_ND).
4. Partial Distance Search using squared Euclidean distance metric (PDS).
5. Tree-structured algorithm using squared Euclidean distance metric (TS).
6. TS with PDS using squared Euclidean distance metric (TS+PDS).

The machine used for the experiment is an IBM compatible personal computer with 1.60 GHz Pentium IV microprocessor. The encoding program runs on a Linux platform.

The encoding time for each test image with HSV as the colour space is shown in Table 7.2. Tables 7.3 and 7.4 show the encoding time for each test image with LUV and RGB as the colour spaces respectively. In each table, the average encoding time for the 15 test images (in the row "Average") and the percentage of encoding time with respect to the algorithm FS_SqED for algorithms FS_ND, PDS, TS, TS+PDS and TS_ND (in the row "% of FS") are also presented.

Test Images	Algorithms						
	FS_ED	FS_SqED	FS_ND	PDS	TS	TS+PDS	TS_ND
Ando	90.03	79.59	30.82	13.97	4.78	3.62	2.22
Lena	90.13	79.63	30.66	12.66	5.94	4.01	2.81
Ak014	90	79.43	30.99	24.25	12.29	6.73	5.01
Arch	90.4	79.79	31.08	17.59	6.69	4.38	2.98
AV009	89.95	79.36	30.54	17.36	9.9	5.84	4.11
Baboon	90.23	79.7	31.06	23.56	11.26	6.8	5.09
Bee	90.24	79.59	30.76	14	8	5.05	3.56
Bflow	90.19	79.53	30.8	19.54	11.66	6.79	5.77
CB4	90.13	79.62	30.85	12.69	6.24	4.42	2.75
Dallas	90.16	79.55	30.75	21.68	8.6	5.7	3.86
FB018	90.08	79.58	30.67	13.41	7.88	4.92	3.79
FD007	90.08	79.57	30.49	15.82	7.58	4.98	3.72
Pepper	90.02	79.49	30.77	17.38	9.88	5.56	4.37
AlpsA	90.67	79	30.01	12.08	7.12	6.15	5.04
Ani18	90.87	79	30.05	12.02	5.53	4.22	3.25
Average	90.21	79.50	30.69	16.53	8.22	5.28	3.89
% of FS			38.60	20.80	10.34	6.64	4.89

Table 7.2 Encoding time (in seconds) of different algorithms for each test image in HSV colour space. Average encoding time of all images is also shown.

Test Images	Algorithms						
	FS_ED	FS_SqED	FS_ND	PDS	TS	TS+PDS	TS_ND
Ando	63.25	55.73	4.05	15.77	6	4.44	0.48
Lena	63.28	55.69	5.05	9.07	8.31	5.11	0.68
Ak014	63.28	55.75	4.05	18.87	11.08	7.84	0.88
Arch	63.37	55.86	4.05	10.17	8.43	4.89	0.69
AV009	63.25	55.68	4.05	11.42	10.05	6.84	0.79
Baboon	63.27	55.7	4.05	13.3	8.98	6.21	0.73
Bee	63.24	55.7	4.05	10.93	9.42	6.09	0.76
Bflow	63.26	55.69	4.05	7.46	5.73	4.05	0.46
CB4	63.27	55.71	4.05	13.77	7.82	6.5	0.62
Dallas	63.32	55.79	4.07	14.17	7.73	5.61	0.64
FB018	63.32	55.72	4.05	14.96	8.75	6.22	0.69
FD007	63.27	55.75	4.05	10	8.42	5.25	0.69
Pepper	63.26	55.67	5.05	16.22	8.64	6.6	0.71
AlpsA	63.26	55.68	4.05	14	5.9	4.9	0.47
Ani18	63.26	55.68	4.05	13.21	5.8	3.47	0.41
Average	63.28	55.72	4.18	12.89	8.07	5.60	0.65
% of FS			7.51	23.13	14.48	10.05	1.16

Table 7.3 Encoding time (in seconds) of different algorithms for each test image in LUV colour space. Average encoding time of all images is also shown.

Test Images	Algorithms						
	FS_ED	FS_SqED	FS_ND	PDS	TS	TS+PDS	TS_ND
Ando	72.45	59.99	6.78	8.74	2.73	1.77	0.37
Lena	72.41	60.02	6.77	9.35	5.07	2.79	0.69
Ak014	72.43	59.97	6.8	13.04	4.85	3.07	0.66
Arch	72.44	59.99	6.77	13.19	4.4	2.83	0.6
AV009	72.41	59.95	6.78	8.91	4.92	2.65	0.66
Baboon	72.43	60	6.77	13.66	4.9	3.12	0.67
Bee	72.43	59.99	6.78	9.18	4.46	2.55	0.61
Bflow	72.42	59.95	6.76	8.61	3.25	2.05	0.44
CB4	72.45	60	6.79	6.98	3.49	1.97	0.47
Dallas	72.45	59.95	6.79	13.28	3.87	2.45	0.53
FB018	72.44	59.97	6.78	7.72	4.33	2.42	0.59
FD007	72.46	60.02	6.76	10.56	5.5	3.01	0.75
Pepper	72.41	60.06	6.78	10.5	4.49	2.66	0.61
AlpsA	72.46	59.99	6.77	10.17	4.13	2.5	0.56
Ani18	72.45	59.97	6.78	7.06	4.55	2.32	0.63
Average	72.44	59.99	6.78	10.06	4.33	2.54	0.59
% of FS			11.30	16.78	7.22	4.24	0.98

Table 7.4 Encoding time (in seconds) of different algorithms for each test image in RGB colour space. Average encoding time of all images is also shown.

As shown in the three tables, the difference between the average encoding time of the full-search algorithm using the Euclidean distance (FS_ED) and the full-search using squared Euclidean distance (FS_SqED) is about 10 seconds. These results show that the square-root operation in the Euclidean distance metric contributes a percentage of about 12% - 18% of the overall distance calculation time. Thus, if quality of their reconstructed images is not greatly affected by encoding them

using the squared Euclidean distance instead of the Euclidean distance, the squared Euclidean distance should be used to achieve greater efficiency.

Next, we will look at the image encoding time of the full-search algorithm using the new distance method (FS_ND). In all the three tables, the experimental results show that the FS_ND algorithm is more efficient than the FS_SqED algorithm. If HSV is the colour space (Table 7.2), the average encoding time of the images is only 38.6% of that of FS_SqED. If LUV is the colour space (Table 7.3), the average encoding time of the images is 7.51% of that of FS_SqED. If RGB is the colour space, the average encoding time of the images is only 11.3% of that of FS_SqED. As explained in Section 7.6, FS_ND is more efficient as we are able to compute some information required in the squared Euclidean distance calculation offline and store them, resulting in less online computation. From the experimental results, we can also observe that when HSV is used as the colour space, the efficiency gained using FS_ND is lower compared with when LUV or RGB is the colour space. This observation is consistent with our explanation in Section 7.6 that for HSV colour space, the online distance computation is more intensive since we can only apply the new distance calculation method to the S and V channels instead of all the three colour channels like in LUV and RGB colour space. Another factor that contribute to more intensive online distance computation is the modification made to the distance calculation for the hue channel. Besides comparing the encoding efficiency of the FS_ND algorithm to FS_SqED, we have also compared it with the partial distance search algorithm (PDS) described in Chapter 5. The experimental results show that the average encoding time is 20.8% (Table 7.2), 23.13% (Table 7.3) and 16.78% (Table 7.4) of the FS_SqED when HSV, LUV and RGB colour spaces are used respectively. Thus, FS_ND is more efficient than PDS when the colour spaces are LUV and RGB but less efficient when the colour space is HSV. FS_ND is less efficient compared with PDS when the colour space HSV is used because the new distance method is only applicable to the S and V channels and not the H channel.

We will next look at the encoding efficiency of the tree-structured algorithm (TS). The average encoding time of TS is 10.38% of FS_SqED when HSV is the colour space, 14.48% of FS_SqED when LUV is the colour space and 7.22% of FS_SqED when RGB is the colour space. From these results, we observe that TS is the most efficient compared with FS_SqED, FS_ND and PDS when the colour spaces are HSV and RGB. When the colour space is LUV, TS is more efficient than FS_SqED and PDS but less efficient than FS_ND.

Next we will look at the encoding efficiency of the merged algorithms. Our proposed algorithm (TS_ND) combines TS with FS_ND. The other algorithm (TS_PDS) combines TS with PDS. Table 7.2 shows that the average encoding time for TS_PDS is 6.64% of that required by FS_SqED when the colour space used is HSV. It also shows that the average encoding time for TS_ND is 4.89% of that required by FS_SqED when the colour space used is HSV. Table 7.3 shows that the average encoding time for TS_PDS is 10.05% of that required by FS_SqED when the colour space used is LUV. It also shows that the average encoding time for TS_ND is 1.16% of that required by FS_SqED when the colour space used is LUV. Table 7.4 shows that the average time for TS_PDS is 4.24% of that required by FS_SqED when the colour space used is RGB. It also shows that the average encoding time for TS_ND is 0.98% of that required by FS_SqED when the colour space used is RGB. These results show that our proposed algorithm is the most efficient for encoding the images compared with the other six algorithms.

7.8.2 ALGORITHM PERFORMANCE - ENCODING ACCURACY

The performance of an algorithm is judged not only on its encoding efficiency but also on its accuracy. An ideal encoding algorithm should not only be able to encode the images using as little time as possible, but the perceptual quality of its reconstructed images should be as close to the those encoded by the full-search algorithm. In this subsection, we will look at the quality of the reconstructed test

images of the above seven encoding algorithms. The metric we used to assess the quality of the reconstructed test images is the peak signal-to-noise ratio (PSNR) [56, 68]. Based on the PSNR of the reconstructed images of the algorithms, we can divide the seven algorithms in Section 7.7.1 into three groups. The first group only consists of the FS_ED algorithm. The PSNR of the reconstructed images of FS_ED is the perceptual quality we are trying to achieve. This is because we have carried out a full-search on the codebook using the standard Euclidean distance metric while encoding the test images. Thus the perceptual quality of the reconstructed images should be the optimal for that codebook. The second group of algorithms consists of FS_SqED, FS_ND and PDS algorithms. The reconstructed images of a test image using any of these three algorithms have the same PSNR. The PSNR of a reconstructed image of the second group of algorithm is different from the first group because the first group uses the standard Euclidean distance metric while compared the similarity between two image vectors but the second group uses the squared Euclidean distance metric. Finally, the third group of algorithms consists of TS, TS_PDS and TS_ND algorithms. Though this group of algorithms also uses the squared Euclidean distance metric, the PSNR of a reconstructed image of this group of algorithms is different from the second group because the use of the tree structure for encoding images in the third group algorithms may not always result in an optimal solution. In this experiment, to evaluate the reconstructed images' quality of the three groups of algorithms, we have firstly produced the reconstructed images of the 15 test images after they have been encoding using each of the seven algorithms. Then, the PSNR of the reconstructed images are calculated. For each algorithm, the average PSNR of the 15 reconstructed images is calculated. Finally, the difference in average PSNR of the reconstructed images between algorithms in the first group and second group is calculated. The difference in average PSNR of the reconstructed images between algorithms in the first group and third group is

calculated. We have compiled the results for all the three colour spaces. Table 7.5 shows the experiment results.

	Difference in average PSNR of reconstructed images between algorithms in Group 1 and 2	Difference in average PSNR of reconstructed images between algorithms in Group 1 and 3
HSV	0.27	0.50
LUV	0.09	0.07
RGB	0.20	0.42

Table 7.5 Difference in Average PSNR of reconstructed images between algorithms in the three groups

The results in Table 7.5 show that no matter which colour spaces the test images are encoded in, the average PSNR of the reconstructed images between the algorithms in the groups are not more than 0.5 dB. According to the formal threshold set by the MPEG committee, reconstructed images within the PSNR of 0.5 dB have no perceptual difference that is visible to humans [56, 68]. Thus, we can conclude the following from these results:

1. We can use squared Euclidean distance metric instead of the standard Euclidean distance metric to measure the perceptual similarity between image vectors.
2. Reconstructed images of images encoded using the proposed algorithm (TS_ND) have PSNR similar to that of the full-search algorithm (FS_ED). This means that they are perceptually very similar. Therefore, while the proposed algorithm only requires 0.98% to 4.86% of the encoding time of the full-search algorithm, the perceptual quality of the reconstructed images of images using this algorithm is also comparable to the full-search algorithm.

7.8.3 ALGORITHM PERFORMANCE - EFFECTS ON RETRIEVAL

The third experiment conducted is to investigate how the use of our proposed encoding algorithm will affect the retrieval effectiveness of the proposed VQ-

based image indexing and retrieval scheme. Such investigation is necessary because the encoded images using the proposed algorithm is slightly different from the ones encoded using the full-search algorithm, thus the VQ histograms built to index the images may be different for each of the algorithms. The difference in VQ histograms as the result of using the two algorithms thus may affect the retrieval performance of the VQ retrieval scheme.

To evaluate the effects of the proposed encoding algorithm on the retrieval performance of the VQ scheme, the MPEG-7 CCD database, which is also used for the experiments in Section 6.4.2, is used. In this experiment, the images in the database are firstly encoded using the full-search (FS_ED) algorithm. The encoding process is carried out with HSV, LUV and RGB colour spaces. The codebook size used is 1024 and codeword block-size used is 4x4 pixels block. Then the VQ histograms are built using these encoded images. Next, the retrieval results for each of the 50 query images are compiled. Finally, based on the ground truth for the query images, the average recall and precision graphs of the query images for each colour space are built. The same process is also carried out using our proposed image-encoding algorithm. When both sets of average recall and precision graphs are built, the graphs are compared. Figure 7.8 shows the comparisons of the two sets of graphs. Figure 7.8(a) shows the average recall and precision graphs of the 50 query images derived using the two encoding algorithms. The image encoding is done in HSV colour space. As seen in the figure, the two average recall and precision graphs are very close. Thus this shows that using either of the two algorithms encoding the images does not have much effect on retrieval performance of the proposed VQ scheme. Figures 7.8(b) and 7.8(c) show the average recall and precision graphs of the query images derived using the two algorithms when the encoding is done in LUV and RGB colour spaces respectively. Both sets of graphs also show that using either of the two encoding algorithms the images does not have much effect on the retrieval performance of the proposed VQ scheme.

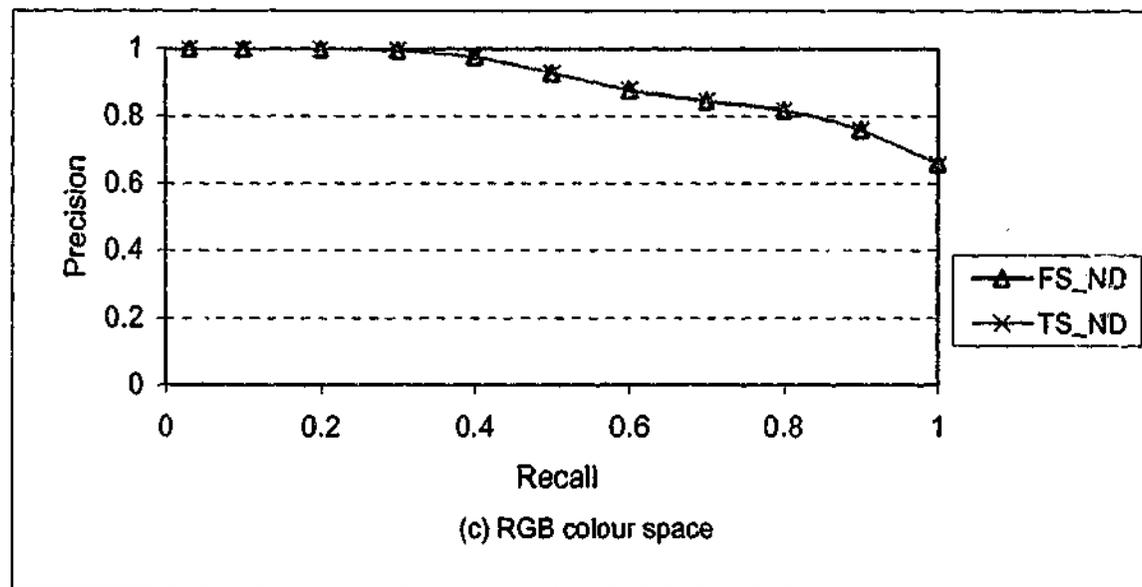
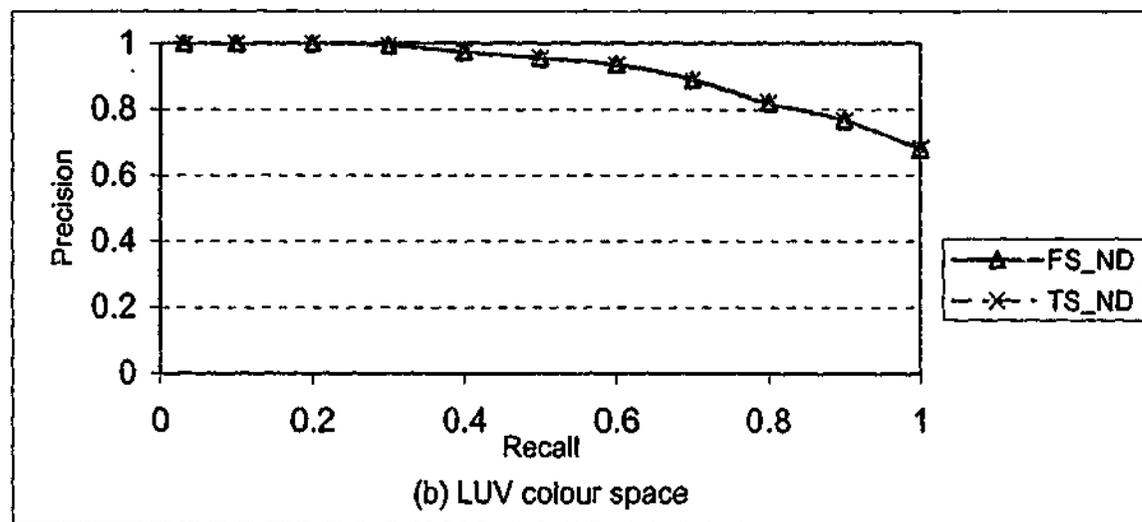
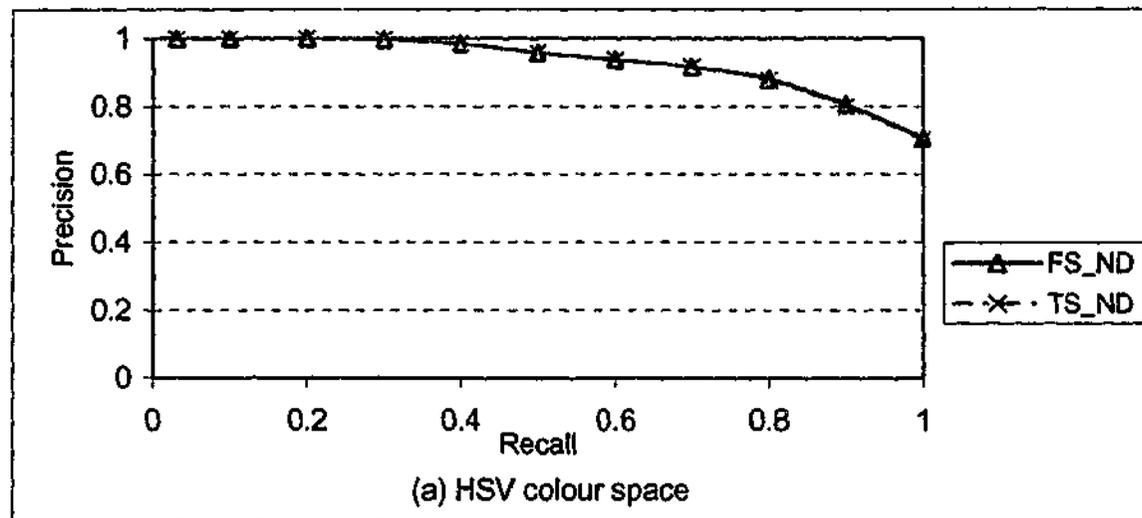


Figure 7.8 Average recall and precision graphs of 50 query images derived using the two encoding algorithms. (a), (b) and (c) shows the graphs derived when HSC, LUV and RGB colour spaces are used respectively.

7.9 CONCLUSIONS

In this chapter, we have proposed a new image-encoding algorithm that we have adopted in our VQ-based image indexing and retrieval system. The experimental results presented in the chapter show that the proposed algorithm requires less than 5% of the time needed by the full-search algorithm to encode the test images. Thus it has made the image encoding more suitable for an online system. The experimental results have also showed that while the image encoding process is more efficient using the proposed algorithm, the retrieval performance of the proposed VQ scheme is hardly affected.

CHAPTER 8

IMPLEMENTATION & EXPERIMENTAL RESULTS OF EFFICIENT SEARCH METHODS FOR IMAGE RETRIEVAL

8.1 INTRODUCTION

The importance of performing the retrieval process efficiently is more critical compared with other processes in our proposed technique. This is because the retrieval process is always performed online and users would expect the system to respond to their query very quickly. As explained in Chapter 5, using the full-search method for the retrieval process is not ideal because it is too time consuming to compare the query image histogram with every database image histogram. In Chapter 5, several efficient search methods are reviewed. Generally, by comparing the experimental results presented in the literature review, the tree-based methods have the greatest potential of achieving high efficiency when they are used to search data of small or moderate dimensional sizes. Unfortunately, studies in [30] reported that when the dimensional size of

the data is high, the efficiency gained by the tree-based methods would deteriorate tremendously. In Chapter 6, we have reported that the proposed VQ-based CBIR technique is likely to perform more effectively when the histogram bin size is 1024 or above, so we are mainly dealing with data with high dimensional sizes in the retrieval process. Thus the tree-based methods are not suitable to improve the efficiency of the retrieval process. This has prompted us to take another approach to improve the efficiency of the retrieval process.

In this chapter, we will describe two efficient search methods which we have adopted in our application. These methods are based on inverted files. An inverted file is a data structure which is widely used in the text retrieval community. Examples of text retrieval applications that use the inverted file structure are the search engines in libraries or in the Internet.

The structure of this chapter is as follows. In Section 8.2, to illustrate how efficient retrieval can be achieved using inverted files, we first describe how the inverted file structure is implemented in text retrieval application to achieve efficient text retrieval. Next in Section 8.3, we establish the similarities between the proposed VQ scheme and the inverted file structure. From the similarities between them, we reason that the inverted file structure can be used to speed up the retrieval process in our application. In Section 8.4, the structure of the inverted file used for our application is illustrated. Section 8.5 presents two methods which use the inverted file to improve the efficiency of the search process in the retrieval process. Next, experimental results are presented in Section 8.6. Finally, we conclude the chapter in Section 8.7.

8.2 EFFICIENT TEXT RETRIEVAL USING INVERTED FILE

An inverted file is one of the most commonly used data structure in a relatively more matured research field of text retrieval [99]. An inverted file consists of two parts: the lexicon and the document lists. In text retrieval, the lexicon contains a collection of all the possible significant words that can be found in the database

documents. Along with every distinct word in the lexicon, a list of the document numbers that contain that word is kept. For our discussion, each list is referred to as a document list. To index each document in the database, a histogram vector, with its number of bins that coincides with the number of distinct words in the lexicon, is used. For each bin in the histogram, if the word it represents appears in the document, it is set to 1. Otherwise, it is set to 0. For retrieval, the set of database documents that have the same word(s) to those in the query document is extracted by using the inverted file. Then using some similarity measures, each database document in the set is compared and ranked according to their similarity with the query document. We will illustrate the above process with the following documents.

Doc 1 : Sally at the seashore

Doc 2 : Sally sells seashells by the seashore

Doc 3 : She sells seashells on the seashells shore

Doc 4 : The seashells she sells are seashore shells

Doc 5 : She sells seashells by the seashore

Doc 6 : She hopes she sells all her seashells soon

From the six documents above, we can identify 16 distinct words. Each distinct word and the document numbers it appears in are shown in Table 8.1.

To reduce the number of words in the lexicon, the stop words (all, are, at, by, her, on, she and the) are removed. Stop words can be removed because they do not carry any significant meaning in the content of the document. The list of words remaining is shown in Table 8.2.

Based on these words and their order in the lexicon, the index for each document is built as in Table 8.3. From Table 8.3, we can see that the histogram vector for document 1 is [0,1,0,1,0,0,0,0].

Lexicon	Document List
all	6
are	4
at	1
by	2,5
her	6
hopes	6
on	3
sally	1,2
seashells	2,3,4,5,6
seashore	1,2,4,5
sells	2,3,4,5,6
she	3,4,5,6
shells	4
shore	3
soon	6
the	1,2,3,4,5

Table 8.1 Lexicon & document lists of the 6 documents

Lexicon	Document List
hopes	6
sally	1,2
seashells	2,3,4,5,6
seashore	1,2,4,5
sells	2,3,4,5,6
shells	4
shore	3
soon	6

Table 8.2 Lexicon & document lists of the 6 documents after removing stop words

Lexicon	Document number					
	1	2	3	4	5	6
hopes	0	0	0	0	0	1
sally	1	1	0	0	0	0
seashells	0	1	1	1	1	1
seashore	1	1	0	1	1	0
sells	0	1	1	1	1	1
shells	0	0	0	1	0	0
shore	0	0	1	0	0	0
soon	0	0	0	0	0	1

Table 8.3 Index built based on the lexicon for each of the 6 documents

For a query document that contains "The shells are on the shore", it is indexed as [0,0,0,0,0,1,1,0]. To retrieve similar documents from the database for the query document, the words, corresponding to the query histogram elements that have '1' in the inverted file, are looked up. The indices of the documents listed along each of these words are extracted. Thus, documents 3 and 4 are extracted from the database. The similarity between the query document and this subset of database documents can be measured using a similarity metric. In this example, we have demonstrated that the inverted file can be used to filter database documents, which are very different from the query document, out of the search space. The database documents that are filtered out are definitely very different from the query document since they do not contain any common words. The process of filtering out unlikely database documents using the inverted file only consists of a series of simple comparison operators. The computation time required to carry out these comparison operators is negligible compared with the computation time required to compute the distances between the query document and the database documents that are filtered out. Thus, better efficiency is achieved using the inverted file structure.

To ensure that significant efficiency is achieved in retrieval using the inverted file structure in any databases, one important criterion must be met. This criterion is the number of distinct terms in each database item must match only a small

portion of the terms that are found in the lexicon. If this criterion is met, each database item is only listed along a relatively smaller number of terms in the lexicon. Therefore during retrieval, it is more likely that only a relatively smaller portion of database items, which have terms that are common to those found in the query item, is retrieved. In text-based retrieval, this criterion is often met. This is because the variety of words (terms), which appears in database documents, is usually very diverse. By compiling all the distinct words found in every database document to form the lexicon, this number of words in the lexicon is usually a lot greater compared with the set of distinct words found in each database documents. In such database, the inverted file can be used to filter out documents which do not have any words similar to the query document. Thus searching is only carried out in a set of documents which is smaller to the set of documents in the database. Therefore significant efficiency retrieval can be obtained.

8.3 FEATURES IN VQ-BASED IMAGE INDICES SUITABLE FOR THE IMPLEMENTATION OF THE INVERTED FILE STRUCTURE

Inverted file structure is chosen as the structure used to improve the efficiency for our retrieval process because our database of image indices has many features suitable for the implementation of the structure.

The inverted file for our database can be easily generated. As described above, an inverted file consists of two parts: the lexicon and the document lists. For our database, the lexicon is readily available in the form of the codeword in the codebook. This is because after the images in the database are encoded, each image vector in the images is represented by a codeword that is most similar to it. So, the set of distinct image vectors that can be found in the decoded database images will only consist of the set of codewords in the codebook.

With the lexicon available, the document list for each codeword can be constructed easily using the VQ histogram generated for each image. This is

because each histogram bin corresponds to a codeword in the lexicon and the histogram bin value of the each histogram bin also reflects the percentage of image vectors in the image encoded by that codeword. For our application, since each document list contains image names (rather than document names) encoded by its corresponding codeword, we will refer to the document list as the image list. Besides having features which can be used easily to build the inverted file, the image indices built using the proposed scheme are also suitable to allow efficient retrieval using the inverted file structure. Since the codebook built is for images from many different categories, only a relatively small subset of the entire list codewords is likely to be used for each image. Moreover, it is also very likely that similar images will use a similar subset of codewords while images that are very different will have high percentage of the codewords that are different. Thus by using the inverted file, images that are different from the query image are likely to be filtered out. With a smaller set of database images for searching, higher efficiency can be achieved.

8.4 STRUCTURE OF THE INVERTED FILE IN OUR APPLICATION

Having discussed the features in our database which are suitable for the implementation of the inverted file, we will now describe the structure of the inverted file used in our application. As stated above, the lexicon of our inverted file consists of the codewords in the codebook that is used to encode the database images. Recorded along with each codeword in the lexicon is an image list. The image list consists of a record of images that have its vectors encoded by that codeword. The percentage of the vectors in the image that are encoded by that codeword is also listed beside the image. This information is recorded because it is essential for the methods which we will describe in Section 8.5. To allow the inverted file to be read into the memory efficiently, the file size should be kept small. To achieve that, only the codeword index is used in the lexicon. For the

images recorded in the image list, instead of using the actual image file name, a number that is pre-assigned to each image is also used. The information of the cross-referencing between the image numbers and the image names is kept in a separate file. Finally, the percentage recorded along each image number in the image list is rounded off to nearest integer values of 0 to 100. This allows the percentage to be recorded using data type of 'char' instead of 'float'.

To reduce the number of images in the search space, we should attempt to reduce the number of images listed along each codeword in the inverted file. If the average number of images listed along each codeword in the inverted file is reduced, the average number of images identified as part of the search space for queries is likely to be reduced. Thus retrieval can be performed more efficiently. To achieve this without affecting the accuracy of the retrieval results, we have to ensure that only images, which have a significant number of their image vectors encoded by a codeword, are listed along that codeword in the inverted file. If only a small percentage of the image vectors are encoded by a codeword, we can usually consider such information as noise in the image and they are unlikely to contribute much in the overall appearance of the image. Such information usually can be safely left out of the inverted file without affecting the accuracy of the retrieval results. Thus when building the inverted file for our application, we will first check if any histogram bin values of an image is less than 1%. For histogram bins which have values less than 1%, we will not list that image along the codewords (in the inverted file) corresponding to those histogram bins. Bin value of less than 1% means that only less than 1% of the total image vectors are encoded by the codeword corresponding to that bin. Such percentage of image vectors usually does not contribute much to the overall appearance of the image. For example, if the image dimension is 256x256 pixels and the codeword block-size is 4x4 pixels, there are a total of 4096 image vectors in the image. Then 1% of the total image vectors are only approximately 41 image vectors. It is unlikely that this small number of image vectors will contain much important information

about the appearance of the image. Another reason for choosing 1% as the selection criterion is because in the research field of text-based retrieval using inverted file method, it is found that if a term appears in less than 1% of the document content, it usually does not contribute much to the overall information in a document [55, 82-84, 99]. Thus, we have also adopted this percentage as the selection criterion of our application.

8.5 METHODS TO REDUCE SIZE OF THE SEARCH SPACE

The basic method of using the inverted file to speed up the retrieval process is as follows:

1. For each histogram bin of the query image, check the value it has.
2. If the value is 0, go on to the next bin. If the value > 0 , go to the image list of the codeword corresponding to the histogram bin. Add the set of images in image list to the set of images in the search space.
3. When the set of images in the search space is compiled, calculate the Manhattan distance between the query image histogram and the histogram of each image in the search space.
4. Rank the image histograms, using the distances calculated, from the smallest to the largest.
5. Display to the user the images corresponding first N image histograms.

From the above description, we can see that this method of retrieving images should be more efficient than the full-search method. This is because histograms representing database images, which are not encoded by any codeword common to the ones in the query image, are filtered out of the search space. Thus the searching and ranking of histograms in the retrieval process for a query are performed on a subset of database histograms. However, using this basic method, the subset of database histograms compiled may still consist of many histograms that are very different to the query histogram. For example, even if the value of a

query histogram bin shows that less than 1% of the query image vectors are encoded using the codeword corresponding to that bin, the set of images listed along that codeword in the inverted file will also be included in the set of database images to be searched. This method of compiling the set of images to be searched may include many images that are perceptually very different from the query image. As a result, the basic method of using the inverted file to retrieve image can be improved further.

In this section, two methods to further reduce the size of search space are presented:

1. In the first method, the less significant bins in the query histogram are eliminated so as to exclude their corresponding image lists from the search space. By excluding more image lists, the number of images compiled in the search space is likely to be smaller. Thus, higher retrieval efficiency can be achieved. The significance of a bin can be judged based on the value it contains. This is because the value represents the percentage of vectors in the image that are encoded by the codeword corresponding to that bin. If a bin value has a low value, it means that only a low percentage of vectors in the image are encoded by the codeword corresponding to that bin. In this case, the significance of that codeword on the appearance of that image is low. Therefore, that bin can be considered as less significant and be eliminated. If the bin value is high, then it should be considered as a more significant bin. In this method that we denote as **IF_M1**, all bins, which have values below the threshold set by the user, are considered less significant bins. Thus, the images in the image lists corresponding to these bins will not be included in the search space. Only bins that have values greater than the threshold are considered as significant bins. The images in their corresponding image lists are included in the search space. The users can select the threshold online. For example, if the user set a percentage of 2%, then the threshold is between 0% and 2%. Thus, only bins which have values greater than 2% are

considered as significant bins. The higher the percentage set by the user, the smaller the number of images lists included in the search space is likely to be. Thus the retrieval time is also likely to be shorter. To illustrate how IF_M1 works, we will now look at the following example. Figure 8.1 shows three images histograms of query image Q, and database images I1 and I2. In this example, the codebook used to encode these images only consists of 10 codewords. Thus the total number of bins in each histogram is 10. The value shown in the each histogram bin is the percentage of image vectors in that image encoded by the codeword corresponding to that histogram bin.

Histogram bin num:	0	1	2	3	4	5	6	7	8	9
Query Q	1	2	70	9	18	0	0	0	0	0
Image I1	10	0	0	0	0	30	60	0	0	0
Image I2	30	50	0	0	0	0	0	20	0	0

Figure 8.1 Histograms of 3 images

With the basic method of using the inverted file, both images I1 and I2 will be included in the search space. Using IF_M1, if the user set the threshold at 1%, the images in the image list recorded along the codeword corresponding to bin 0 of the query image will not be included in the search space. This is because though the bin value is not zero, the percentage in query histogram bin 0 is also not greater than 1%. Images in the image list recorded along the codeword corresponding to bin 5 to 9 of the query image histogram are also not included in the search space since their values are zero. As the histogram bin 0 of image I1 is the only one which has a non-zero value common to those in the query histogram, image I1 will be filtered out of the search space. Image I2 is still included in the search space since besides being listed along codeword, corresponding to bin 0, in the inverted file, it is also listed along

codeword, corresponding to bin 1, in the inverted file. If the user states that the histogram bin value must be greater than 2%, then both images I1 and I2 will be filtered out of the search space.

2. In IF_M1, we improve the efficiency of the basic method by examining the value in each bin in the query histogram. In the second method which we denote as IF_M2, we eliminate database images from the search space by examining the number of matching significant bins between the query histogram and the database histogram. In this method, a significant bin is one with a value greater than zero in the query histogram and greater than 1% in the database histogram. For a database image to be included in the search space, the number of matching significant bins between the query histogram and the database histograms must be greater than the threshold set by the user. The threshold represents a percentage of the total number of significant bins in the query histogram set by the user. The reason for eliminating database histograms from the search space based on this threshold is because an image that is similar to the query image is more likely to be encoded by a similar set of distinct codewords as the ones used to encode the query image, and vice versa. When the percentage is set higher, the search space is likely to be smaller because fewer database histograms are likely to have that number of matching bins. Thus the retrieval time is likely to be shorter. To illustrate this method, let us look at Figure 8.1 again. If the user sets the threshold at 20%, then image I1 will be filtered out of the search space. This is because the significant bins in the histogram of image I1 only match 1 out of 5 (20%) significant bins in the query histogram. If the user increases the threshold to 40%, then image I2 will also be filtered out of the search space, since the significant bins in its histogram only match 40% of the significant bins in the query image histogram.

For both methods presented above, if the user wants the retrieval process to be performed very efficiently, a higher percentage should be selected. However, when higher percentages are selected, the possibility of some database images, which are similar to the query image being filtered out of the search space, will increase too. Thus the retrieval result may not be optimal. On the other hand, if the user wants the retrieval result to be optimal, the percentages selected should be lower.

8.6 EXPERIMENTAL RESULTS

To evaluate the retrieval performance of inverted file methods in our application, an inverted file structured retrieval module is implemented in our VQ-based CBIR system. The machine used for the experiment is an IBM compatible personal computer with 1.60 GHz Pentium IV microprocessor. The system runs on a Linux platform.

Using the module developed, two experiments are carried out. The first experiment compares the retrieval efficiency and effectiveness of method IF_M1 with that of the full-search method. The second experiment compares the retrieval efficiency and effectiveness of method IF_M2 with that of the full-search method. Both experiments are carried out on the MPEG-7 test database, CCD. This database consists of 5466 colour images. The 50 query images and their ground truths, which are provided by the developers of the database CCD, are also used to obtain the experimental results. For both experiments, the same codebook generated for the experiments in Section 6.4.2.1, with block-size of 4x4 pixels and codebook size of 1024 in colour space HSV, is used. This codebook configuration is used because it is the most effective for retrieval in our proposed VQ-based technique. The same set of VQ histograms built for the images in database CCD using this codebook is also used. Finally, the inverted file is built. The lexicon is made up of the codewords and the image list for each codeword is generated using the information in the VQ histograms. As described above in Section 8.4, in the

image list for each codeword, only images, which have 1% or more of its vectors encoded by that codeword, are listed.

To evaluate the retrieval performance of the each method, two sets of experimental results are obtained. They are:

1. Average retrieval time (in seconds)
2. Average percentage of relevant images that are in the search space after the filtering process.

The two sets of experimental results for each method are also compared with that of the full-search method.

8.6.1 RETRIEVAL PERFORMANCE OF IF_M1

Table 8.4 shows the experimental results of the full-search method and the IF_M1.

	Full-search	IF_M1					
		For image list to be in the search space, its corresponding query histogram bin value must be greater than the threshold of:					
		0%	0.5%	1%	1.5%	2%	2.5%
ART	0.17	0.1556	0.1332	0.0914	0.0912	0.065	0.0578
%RI	100	100	100	99.78	99.53	98.58	97.36

ART : Average retrieval time(in seconds) for the 50 query images

%RI : Average percentage of relevant images for the 50 query images in the search space.

Table 8.4 Retrieval results of IF_M1: Average retrieval time and average percentage of relevant images in search space for the 50 query images

The results in the table show that as the threshold set for the histogram bin value increases, the average retrieval time of the 50 query images decreases. This is because as the threshold increases, the number of query histogram bins which have values greater than the threshold decreases. Since each bin corresponds to a

codeword in the inverted file lexicon, fewer image lists are visited. When fewer image lists are visited, the number of images compiled is likely to decrease. With smaller number of images remaining in the search space, the computation time of the retrieval process decreases. The results show that when the user requires only those image lists corresponding to bin values greater than the threshold of 0.5% to be included in the search space, the average retrieval time required is only 78.4% of the full-search method. When only those image lists corresponding to bin values greater the threshold of 0% or 0.5% are included in the search space, all the relevant images for the 50 query images would appear in the search space. Thus, while the retrieval effectiveness is not affected, the retrieval efficiency using IF_M1 as compared with the full-search method has improved. When the threshold increases from 1% to 2.5%, the average percentage of relevant images for the 50 query images begins to decrease from 99.78% to 97.36%. Although there are some relevant images of the query images that are filtered out of the search space at these thresholds, the information lost (percentage of relevant images left out of the search space) is insignificant. For example, when the threshold is 2.5%, the average percentage of relevant images left out of the search space is only 2.64%, but the average retrieval time is only 34% of the full-search method.

To further analyse the experimental results, we will next look at the percentage of relevant images retrieved for each query image when only those image lists corresponding to bin values greater than threshold of 2.5% are included in the search space. Out of the 50 query images, 46 query images have all their relevant images remained in the filtered search space. Since all their relevant images are still in the filtered search space, the ranking of the relevant images for each of these 46 query images when IF_M1 is used as the retrieval method is the same as that of the full-search method. If the recall and precision graphs for each of these query images are plotted based on the retrieval results using IF_M1 (threshold of 2.5%) and the full-search method, the corresponding graphs for the two methods

will be identical. Thus using IF_M1 as the retrieval method for these 46 query images is more efficient than the full-search method, without losing effectiveness. Next, the results of the 4 query images, which do not have all their relevant images in the filtered search space, are analysed.

Query Image Number	20	31	32	45
Percentage of relevant images in the filtered search space	75	55.56	50	87.5

Table 8.5 Percentage of relevant images in the filtered search space for the 4 query images when using IF_M1 (threshold of 2.5%) as the retrieval method

Table 8.5 shows the percentage of relevant images in the filtered search space for these 4 images. From this table, we observe that majority of the their relevant images are still in the filtered search space. To observe where the relevant images, which are filtered out of the search space, are ranked in terms of their similarity to their query images, the Manhattan distances of the relevant image histograms and their query image histogram are calculated. Using the distances, the relevant images for each of the 4 query images are ranked from the most similar to the least similar. From our observation, the relevant images that are filtered out are those ranked the lowest among their respective set of relevant images. Consequently, we can also deduce that for each query image, the corresponding recall and precision graphs plotted with their retrieval results using IF_M1 and the full-search method are identical between recall percentage 0% and the maximum recall percentage which the query image is able to achieve using the IF_M1 method. Thus, the retrieval performance of IF_M1 is no worse than the full-search method for each of the 4 query images between these recall percentages.

From the above analysis, we can deduce that the overall retrieval performance of IF_M1 (threshold of 2.5%) should not be much worse than the full-search method.

This is because:

- Only 4 out of 50 query images do not have all their relevant images in the filtered search space.
- The average percentage of relevant images left out the search space is only 2.64%.
- The relevant images that are filtered out of the search space are those that are ranked the lowest among their respective set of relevant images.

Overall, we can conclude that by using IF_M1 for retrieval, we can gain much efficiency without sacrificing much retrieval accuracy.

8.6.2 RETRIEVAL PERFORMANCE OF IF_M2

Table 8.6 shows the experimental results of the full-search method and the IF_M2 method.

	Full search	IF_M2: Percentage of significant query histogram bins that database histogram bins is required to match must be greater than threshold of:			
		0%	5%	10%	15%
ART	0.17	0.1552	0.0886	0.0552	0.0352
%RI	100	100	98.36	97.96	96.68

ART : Average retrieval time(in seconds) for the 50 query images

%RI : Average percentage of relevant images for the 50 query images in the search space.

Table 8.6 Retrieval results of IF_M2: Average retrieval time and average percentage of relevant images in the search space for the 50 query images

The results of IF_M2 show that as the percentage of significant query histogram bins that database histogram bins is required to match increases, the average retrieval time of the 50 query images decreases. This is because as the percentage set by the user increases, the number of database histograms that have number of matching bins greater than the threshold decreases. Thus, with smaller number of images in the search space, the average retrieval time required by the query images decreases. When the number of matching significant bins between the database histogram and the query histogram must be greater than 0% of the significant bins in the query histogram, its average retrieval time of the 50 query images is 91.3% of the full-search method. When the number of matching significant bins must be greater than the threshold of 15%, the average retrieval time of the 50 query images is further reduced to 20.7% of the full-search method. In the table, the average percentage of relevant images that are in the filtered search space for each threshold of IF_M2 is also presented. The results show that when the number of matching significant bins must be greater than the threshold 0%, all the relevant images for the 50 query images are still in the filtered search space. As the threshold increases further, the average percentage of relevant images in the filtered search space decreases further. Although some relevant images of the query images are filtered out of the search space as the percentage set increases, the information lost (percentage of relevant images left of the search space) is insignificant. For example, when the number of matching significant bins must be greater than the threshold of 15%, the average percentage of relevant images left out of the search space is only 3.32%, but the average retrieval time is only 20.7% of the full-search method.

To further analyse the experimental results, we will next look at the percentage of relevant images retrieved of each query image when the number of matching significant bins must be greater than the threshold of 15%. Out of the 50 query images, 44 query images have all their relevant images remained in the filtered search space. Since all their relevant images are still in the filtered search space,

the ranking of the relevant images for each of these 44 query images when IF_M2 is used as the retrieval method is the same as the full-search method. If the recall and precision graphs for each of these query images are plotted based on the retrieval results using IF_M2 (threshold of 15%) and the full-search method, their corresponding graphs will be identical. Thus using IF_M2 as the retrieval method for these 44 query images is more efficient than the full-search method, without losing effectiveness. Next, the 6 query images, which do not have all their relevant images in the filtered search space, are analysed.

Query Image Number	1	11	20	27	31	45
Percentage of relevant images in the filtered search space	83.33	80	75	66.67	66.67	62.5

Table 8.7 Percentage of relevant images in the search space for the 6 query images when using IF_M2 (threshold of 15%) as the retrieval method

Table 8.7 shows the percentage of relevant images in the search space for these 6 images. From this table, we observe that at least 62.5% of their relevant images are still in the search space. To observe where the relevant images, which are filtered out of the search space, are ranked in terms of their similarity to their query images, the Manhattan distances between the relevant image histograms and their query image histogram are calculated. Using the distances, the relevant images for each of the 6 query images are ranked from the most similar to the least similar. From our observation, the relevant images that are filtered out are those ranked the lowest among their respective set of relevant images. Consequently, we can also deduce that for each query image, their corresponding recall and precision graphs plotted with retrieval results using IF_M2 and the full-search method will be identical between recall percentage 0% and the maximum recall percentage which the query image is able to achieve using the IF_M2 method.

Thus, the retrieval performance of IF_M2 is no worse than the full-search method for each of the 6 query images between these recall percentages.

From the above analysis, we can deduce that the overall retrieval performance of IF_M2 (threshold percentage of 15%) should not be much worse than the full-search method. This is because:

- Only 6 out of 50 query images do not have all their relevant images in the filtered search space.
- The average percentage of relevant images left out the search space is only 3.32%.
- The relevant images that are filtered out of the search space are those that are ranked the lowest among their respective set of relevant images.

Overall, we can conclude that by using IF_M2 for retrieval, we can gain much efficiency without sacrificing much retrieval accuracy.

8.7 CONCLUSIONS

In this chapter, we have presented two efficient image retrieval methods. These methods are based on the inverted file structure. We have also discussed the characteristics of our database image indices that made them suitable for the implementation of the inverted file structure. We have presented the results obtained from experiments conducted to evaluate the performance of the inverted file based methods. From the analysis of the results, we conclude that both methods can be a lot more efficient compared with the full-search retrieval method without sacrificing much retrieval accuracy.

CHAPTER 9

CONCLUSIONS AND FUTURE DEVELOPMENTS

9.1 CONCLUSIONS

The main aim of this thesis is to develop a good colour-based CBIR technique for image retrieval. A good colour-based CBIR technique must be effective in indexing and retrieving images. To achieve this goal, we reviewed several existing colour-based CBIR techniques which are popularly used in current CBIR systems. Then, the limitations of these techniques are assessed.

To overcome the limitations of the existing CBIR techniques, we propose an indexing and retrieval technique based on a compression technique called vector quantization. Vector quantization is a compression technique where the encoding and decoding of an image are based on blocks of pixels called vectors, instead of individual pixel. This compression technique encodes an image by representing the image vectors with the indices of the unique codewords from a codebook. Compression is achieved since the number of bits required for a codeword index is smaller than the number of bits required by an image vector. In the proposed technique, an image is indexed by a histogram generated using the indices in the

compressed image file. Retrieval of images is performed by comparing the histogram of the query image and the histogram of each database image. Using the histograms of the query and the database image, the distance, which represent the summation of the difference in frequency of each codeword index used to encode the two images, is calculated. A set of database images, which have the smallest distances, is retrieved.

In this thesis, the concepts of the proposed indexing and retrieval technique and the implementation details are described in chapter 3 and 4. To evaluate the retrieval effectiveness of the proposed technique, we have compared the retrieval performance of the proposed technique with three colour-based techniques we have reviewed. The experiments are carried out using three image databases of different sizes. The experimental results from the experiments are presented in Chapter 6. The following are the conclusions made from the experimental results:

1. The proposed VQ-based technique is more effective than the three techniques we have compared with. The proposed technique is more effective because:
 - Spatial relationships among colour pixels in images are taken into consideration when indexing the images.
 - The feature built in the proposed technique is based codewords that are made up of pixels of identical colours and also codewords that are made up of non-identical colours. Thus, information on spatial correlation between identical and non-identical colours in the image is captured in the feature.
 - The proposed technique can be applied to databases which have images of different dimensional sizes.
2. The proposed technique is robust. This is indicated by the consistency of the experimental results obtained from the experiments on three different image databases of different sizes. The experimental results on all the databases consistently show that the proposed technique outperforms the existing techniques.

3. The best compromise codebook configuration recommended to be used for our proposed CBIR technique to be effective and efficient in retrieval is codebook size 1024, codeword block-size 4x4 pixels and colour space HSV. The experimental results in Chapter 6 also show that when other reasonable configurations are used, the proposed technique can still outperform the colour-based techniques we have compared with. This shows that the retrieval performance of the proposed technique is not heavily dependent on the choice of the configurations.

A CBIR system is usually an interactive system. So a good CBIR technique must not only be effective, but also be efficient in retrieving images from the database. In order to improve the efficiency of our proposed technique, efficient multidimensional data search methods are used in the image encoding process and the image retrieval process. As encoding process mainly deals with data with medium dimensional sizes and the retrieval process deals with data with high dimensional sizes, different search methods are adopted for the two processes.

For the encoding process, a tree-structured method is adopted. The tree is constructed based on the partitioning of the search space with respect to the codewords. The codewords used for space partitioning consist of the codewords in the final codebook and also the intermediate codebooks generated from the codebook generation process. Compared with the full-search method, the computation time of the image encoding process using the tree-structure method is much shorter since only a small number of codewords in the codebook is searched. To further improve the efficiency of the tree-structure method, some information required for calculating the similarity between image vectors and codewords is precalculated and stored. This information is used during online image encoding to reduce the amount of computation required for calculating the similarity between image vectors and codewords. The search method we have adopted for the encoding process only requires between 0.98% and 4.89% of average computation time required by full-search method. Although the search

method adopted for the image encoding process is much more efficient compared with the full-search method, it does not reduce the effectiveness of the proposed CBIR technique.

For the image retrieval process, we have adopted an inverted file structure to improve its efficiency. The inverted file consists of information of the images that are encoded by each codeword and also the frequency of that codeword being used to encode that image. Two methods based on the inverted file structure are proposed. For each method, the user can determine the accuracy and the efficiency of the retrieval process by setting a threshold for the method. The higher the threshold set, the smaller is the number of histograms to be searched in the retrieval process. Both methods are capable of improving the efficiency of the retrieval process while achieving retrieval results of high accuracy.

9.2 FUTURE DEVELOPMENTS

In this research, we have developed a colour-based CBIR technique which is both effective and efficient. We have published our work in [31, 46, 91, 92, 93, 94].

In future, as the follow-up to this research, the following areas will be looked into:

1. Is the codebook configuration recommended for our proposed technique to retrieve images in databases of general domain equally effective when used to retrieve images in databases of specific domain? The research in this area is important because it allows us to make our proposed technique adaptive to databases of specific domain.
2. How can we integrate the proposed technique with other techniques which are based on other low-level image information, like text, texture and shape? The research in this area allows users to use our proposed technique together with other techniques to carry out their image retrieval effectively.
3. How can we include other low-level information like texture and shape in the codebook? By increasing the amount of image information captured in the

codebook, there is a possibility that the retrieval effectiveness of our proposed technique can be improved further.

4. Is there a way to refine the retrieval results by making use of the feedback from the user? Since the relevance of the images retrieved are very subjective, we can incorporate user feedback into our technique to achieve better retrieval results.
5. Can different search methods be applied to encoding images with different colour distribution and characteristics? This is to further improve the efficiency of the image encoding process.

APPENDIX A

Survey forms for identifying ground truth images
for each query image of Database SCD

Monash University
Gippsland School of Computing and Information Technology

Explanatory Statement

Project Title: Image Compression & Retrieval Based on Vector Quantization

Background Information

Many images are captured and stored in digital databases nowadays. Therefore, an efficient and effective image indexing and retrieval technique is needed. This project investigates an image indexing and retrieval technique based on vector quantization (VQ). An application will be developed in this project to allow the users to select a query image and search for database images which are similar to it. Effectiveness of such a system is normally determined by human judgement (i.e. whether the results returned by the system conform to human judgements).

Activities Required from Participants

- be interviewed by the researcher if required
- compare the similarities between a set of query images and their respectively set of images displayed on the website <http://www.gscit.monash.edu.au/~tengs/main.html> (images are rated for general viewing)
- complete questionnaires about the images similarities and the participants
- make himself/herself available for a further interview should that be required

(Please note: all participations in the activities listed above are voluntary in nature. Should participants encounter any discomforts during the activities, they are entitled to withdraw from them immediately. Any complaints or comments can also be made to :

Shyhwei Teng
Gippsland School of Computing and I.T.
Switchback Road, Churchill 3842,
Victoria, Australia

Phone: +61 3 99026133
Email: shyh.wei.teng@infotech.monash.edu.au

Privacy of Participants

To protect the participants' privacy, no personal information about the participants will be disclosed to the public.

Consent Form

Project Title: Image Compression & Retrieval Based on Vector Quantization

I agree to take part in the above Monash University research project. I have had the project explained to me, and I have read the Explanatory Statement, which I keep for my records. I understand that agreeing to take part means that I am willing to:

- be interviewed by the researcher if required
- compare the similarities between a set of query images and their respective set of images displayed on the website <http://www.gscit.monash.edu.au/~tengs/main.html> (images are rated for general viewing)
- complete questionnaires about the images similarities and the participants
- make himself/herself available for a further interview should that be required

There are specific information about myself the researcher must take notice before I can take part in the research activities, they are (please list) :

I would give the researchers the permission to re-use the information obtained from the activities listed above in future research work.

Do you have any experience using an image retrieval system? Yes / No (please circle)

Do you have any knowledge on the methods used in image retrieval system? Yes / No (please circle)

I understand that any information I provide is confidential, and that no information that could lead to the identification of any individual will be disclosed in any reports on the project, or to any other party

Name:(please print)

Address:

Email:

Signature: Date:

Query : Image 1



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one **only**.

1. main object

2. overall content

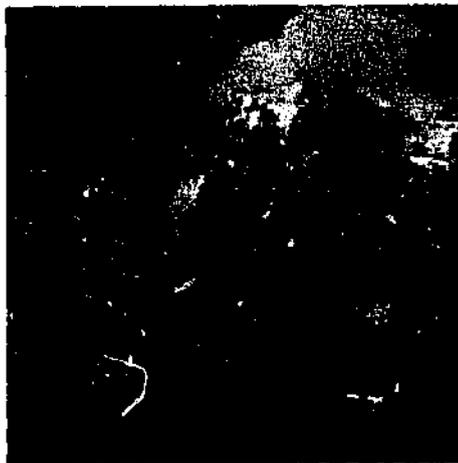
3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 2



1 []	2 []	3 []	4 []	5 []	6 []	7 []	8 []
9 []	10 []	11 []	12 []	13 []	14 []	15 []	16 []
17 []	18 []	19 []	20 []	21 []	22 []	23 []	24 []
25 []	26 []	27 []	28 []	29 []	30 []	31 []	32 []
33 []	34 []	35 []	36 []	37 []	38 []	39 []	40 []
41 []	42 []	43 []	44 []	45 []	46 []	47 []	48 []
49 []	50 []	51 []	52 []	53 []	54 []	55 []	56 []
57 []	58 []	59 []	60 []	61 []	62 []	63 []	64 []
65 []	66 []	67 []	68 []	69 []	70 []	71 []	72 []
73 []	74 []	75 []	76 []	77 []	78 []	79 []	80 []
81 []	82 []	83 []	84 []	85 []	86 []	87 []	88 []
89 []	90 []	91 []	92 []	93 []	94 []	95 []	96 []

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one **only**.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 3



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 4



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

1. main object

2. overall content

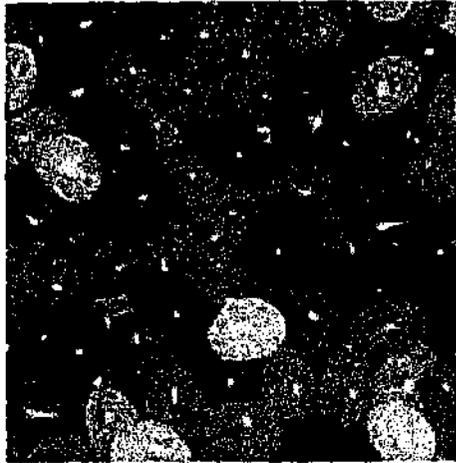
3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 5



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

1. main object

2. overall content

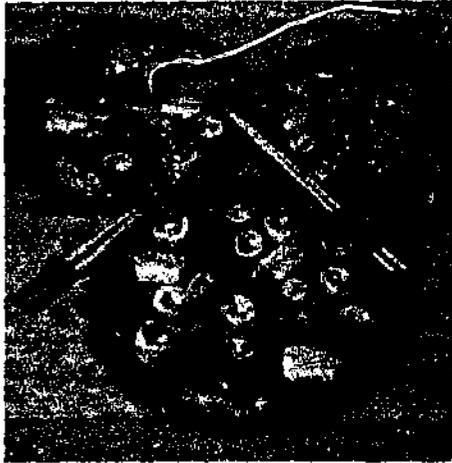
3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 6

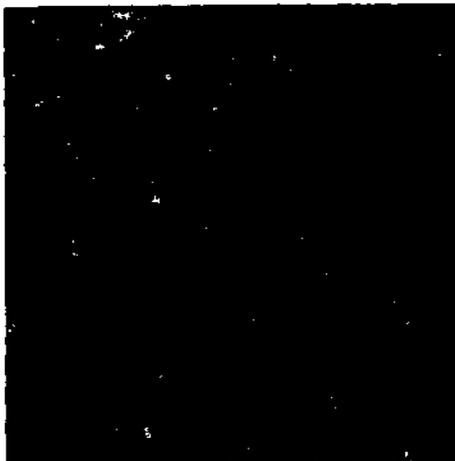


1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 7

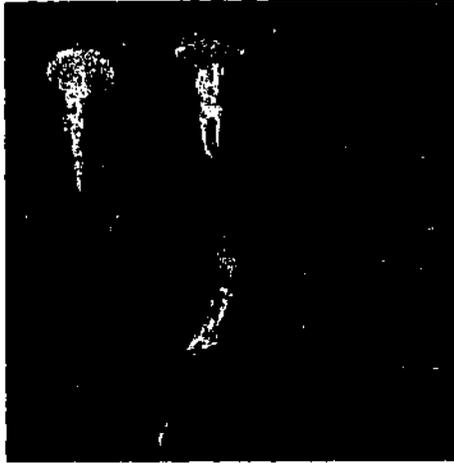


1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 8



1 []	2 []	3 []	4 []	5 []	6 []	7 []	8 []
9 []	10 []	11 []	12 []	13 []	14 []	15 []	16 []
17 []	18 []	19 []	20 []	21 []	22 []	23 []	24 []
25 []	26 []	27 []	28 []	29 []	30 []	31 []	32 []
33 []	34 []	35 []	36 []	37 []	38 []	39 []	40 []
41 []	42 []	43 []	44 []	45 []	46 []	47 []	48 []
49 []	50 []	51 []	52 []	53 []	54 []	55 []	56 []
57 []	58 []	59 []	60 []	61 []	62 []	63 []	64 []
65 []	66 []	67 []	68 []	69 []	70 []	71 []	72 []
73 []	74 []	75 []	76 []	77 []	78 []	79 []	80 []
81 []	82 []	83 []	84 []	85 []	86 []	87 []	88 []
89 []	90 []	91 []	92 []	93 []	94 []	95 []	96 []

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one **only**.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 9



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 10



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 11



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 12



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 13

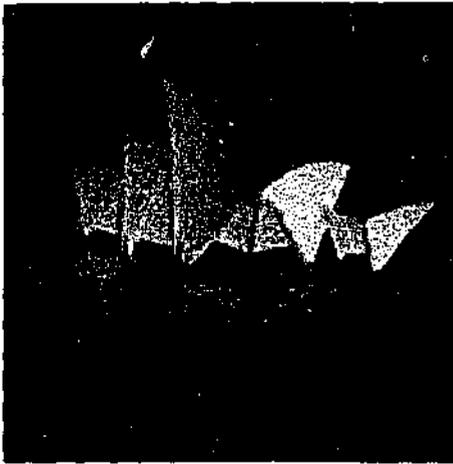


1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 14



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 15



1	<input type="checkbox"/>	2	<input type="checkbox"/>	3	<input type="checkbox"/>	4	<input type="checkbox"/>	5	<input type="checkbox"/>	6	<input type="checkbox"/>	7	<input type="checkbox"/>	8	<input type="checkbox"/>
9	<input type="checkbox"/>	10	<input type="checkbox"/>	11	<input type="checkbox"/>	12	<input type="checkbox"/>	13	<input type="checkbox"/>	14	<input type="checkbox"/>	15	<input type="checkbox"/>	16	<input type="checkbox"/>
17	<input type="checkbox"/>	18	<input type="checkbox"/>	19	<input type="checkbox"/>	20	<input type="checkbox"/>	21	<input type="checkbox"/>	22	<input type="checkbox"/>	23	<input type="checkbox"/>	24	<input type="checkbox"/>
25	<input type="checkbox"/>	26	<input type="checkbox"/>	27	<input type="checkbox"/>	28	<input type="checkbox"/>	29	<input type="checkbox"/>	30	<input type="checkbox"/>	31	<input type="checkbox"/>	32	<input type="checkbox"/>
33	<input type="checkbox"/>	34	<input type="checkbox"/>	35	<input type="checkbox"/>	36	<input type="checkbox"/>	37	<input type="checkbox"/>	38	<input type="checkbox"/>	39	<input type="checkbox"/>	40	<input type="checkbox"/>
41	<input type="checkbox"/>	42	<input type="checkbox"/>	43	<input type="checkbox"/>	44	<input type="checkbox"/>	45	<input type="checkbox"/>	46	<input type="checkbox"/>	47	<input type="checkbox"/>	48	<input type="checkbox"/>
49	<input type="checkbox"/>	50	<input type="checkbox"/>	51	<input type="checkbox"/>	52	<input type="checkbox"/>	53	<input type="checkbox"/>	54	<input type="checkbox"/>	55	<input type="checkbox"/>	56	<input type="checkbox"/>
57	<input type="checkbox"/>	58	<input type="checkbox"/>	59	<input type="checkbox"/>	60	<input type="checkbox"/>	61	<input type="checkbox"/>	62	<input type="checkbox"/>	63	<input type="checkbox"/>	64	<input type="checkbox"/>
65	<input type="checkbox"/>	66	<input type="checkbox"/>	67	<input type="checkbox"/>	68	<input type="checkbox"/>	69	<input type="checkbox"/>	70	<input type="checkbox"/>	71	<input type="checkbox"/>	72	<input type="checkbox"/>
73	<input type="checkbox"/>	74	<input type="checkbox"/>	75	<input type="checkbox"/>	76	<input type="checkbox"/>	77	<input type="checkbox"/>	78	<input type="checkbox"/>	79	<input type="checkbox"/>	80	<input type="checkbox"/>
81	<input type="checkbox"/>	82	<input type="checkbox"/>	83	<input type="checkbox"/>	84	<input type="checkbox"/>	85	<input type="checkbox"/>	86	<input type="checkbox"/>	87	<input type="checkbox"/>	88	<input type="checkbox"/>
89	<input type="checkbox"/>	90	<input type="checkbox"/>	91	<input type="checkbox"/>	92	<input type="checkbox"/>	93	<input type="checkbox"/>	94	<input type="checkbox"/>	95	<input type="checkbox"/>	96	<input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 16



1	<input type="checkbox"/>	2	<input type="checkbox"/>	3	<input type="checkbox"/>	4	<input type="checkbox"/>	5	<input type="checkbox"/>	6	<input type="checkbox"/>	7	<input type="checkbox"/>	8	<input type="checkbox"/>
9	<input type="checkbox"/>	10	<input type="checkbox"/>	11	<input type="checkbox"/>	12	<input type="checkbox"/>	13	<input type="checkbox"/>	14	<input type="checkbox"/>	15	<input type="checkbox"/>	16	<input type="checkbox"/>
17	<input type="checkbox"/>	18	<input type="checkbox"/>	19	<input type="checkbox"/>	20	<input type="checkbox"/>	21	<input type="checkbox"/>	22	<input type="checkbox"/>	23	<input type="checkbox"/>	24	<input type="checkbox"/>
25	<input type="checkbox"/>	26	<input type="checkbox"/>	27	<input type="checkbox"/>	28	<input type="checkbox"/>	29	<input type="checkbox"/>	30	<input type="checkbox"/>	31	<input type="checkbox"/>	32	<input type="checkbox"/>
33	<input type="checkbox"/>	34	<input type="checkbox"/>	35	<input type="checkbox"/>	36	<input type="checkbox"/>	37	<input type="checkbox"/>	38	<input type="checkbox"/>	39	<input type="checkbox"/>	40	<input type="checkbox"/>
41	<input type="checkbox"/>	42	<input type="checkbox"/>	43	<input type="checkbox"/>	44	<input type="checkbox"/>	45	<input type="checkbox"/>	46	<input type="checkbox"/>	47	<input type="checkbox"/>	48	<input type="checkbox"/>
49	<input type="checkbox"/>	50	<input type="checkbox"/>	51	<input type="checkbox"/>	52	<input type="checkbox"/>	53	<input type="checkbox"/>	54	<input type="checkbox"/>	55	<input type="checkbox"/>	56	<input type="checkbox"/>
57	<input type="checkbox"/>	58	<input type="checkbox"/>	59	<input type="checkbox"/>	60	<input type="checkbox"/>	61	<input type="checkbox"/>	62	<input type="checkbox"/>	63	<input type="checkbox"/>	64	<input type="checkbox"/>
65	<input type="checkbox"/>	66	<input type="checkbox"/>	67	<input type="checkbox"/>	68	<input type="checkbox"/>	69	<input type="checkbox"/>	70	<input type="checkbox"/>	71	<input type="checkbox"/>	72	<input type="checkbox"/>
73	<input type="checkbox"/>	74	<input type="checkbox"/>	75	<input type="checkbox"/>	76	<input type="checkbox"/>	77	<input type="checkbox"/>	78	<input type="checkbox"/>	79	<input type="checkbox"/>	80	<input type="checkbox"/>
81	<input type="checkbox"/>	82	<input type="checkbox"/>	83	<input type="checkbox"/>	84	<input type="checkbox"/>	85	<input type="checkbox"/>	86	<input type="checkbox"/>	87	<input type="checkbox"/>	88	<input type="checkbox"/>
89	<input type="checkbox"/>	90	<input type="checkbox"/>	91	<input type="checkbox"/>	92	<input type="checkbox"/>	93	<input type="checkbox"/>	94	<input type="checkbox"/>	95	<input type="checkbox"/>	96	<input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 17



1	[]	2	[]	3	[]	4	[]	5	[]	6	[]	7	[]	8	[]
9	[]	10	[]	11	[]	12	[]	13	[]	14	[]	15	[]	16	[]
17	[]	18	[]	19	[]	20	[]	21	[]	22	[]	23	[]	24	[]
25	[]	26	[]	27	[]	28	[]	29	[]	30	[]	31	[]	32	[]
33	[]	34	[]	35	[]	36	[]	37	[]	38	[]	39	[]	40	[]
41	[]	42	[]	43	[]	44	[]	45	[]	46	[]	47	[]	48	[]
49	[]	50	[]	51	[]	52	[]	53	[]	54	[]	55	[]	56	[]
57	[]	58	[]	59	[]	60	[]	61	[]	62	[]	63	[]	64	[]
65	[]	66	[]	67	[]	68	[]	69	[]	70	[]	71	[]	72	[]
73	[]	74	[]	75	[]	76	[]	77	[]	78	[]	79	[]	80	[]
81	[]	82	[]	83	[]	84	[]	85	[]	86	[]	87	[]	88	[]
89	[]	90	[]	91	[]	92	[]	93	[]	94	[]	95	[]	96	[]

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one **only**.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 18



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

1. main object

2. overall content

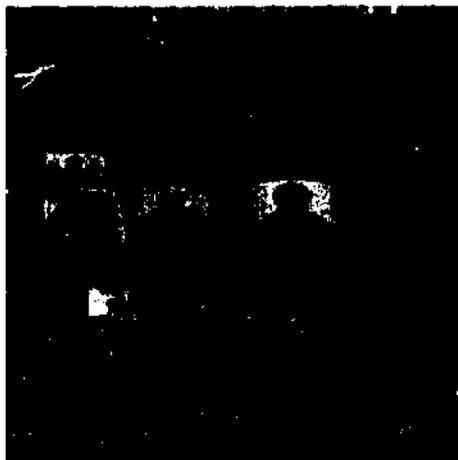
3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 19



1 []	2 []	3 []	4 []	5 []	6 []	7 []	8 []
9 []	10 []	11 []	12 []	13 []	14 []	15 []	16 []
17 []	18 []	19 []	20 []	21 []	22 []	23 []	24 []
25 []	26 []	27 []	28 []	29 []	30 []	31 []	32 []
33 []	34 []	35 []	36 []	37 []	38 []	39 []	40 []
41 []	42 []	43 []	44 []	45 []	46 []	47 []	48 []
49 []	50 []	51 []	52 []	53 []	54 []	55 []	56 []
57 []	58 []	59 []	60 []	61 []	62 []	63 []	64 []
65 []	66 []	67 []	68 []	69 []	70 []	71 []	72 []
73 []	74 []	75 []	76 []	77 []	78 []	79 []	80 []
81 []	82 []	83 []	84 []	85 []	86 []	87 []	88 []
89 []	90 []	91 []	92 []	93 []	94 []	95 []	96 []

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 20



1	<input type="checkbox"/>	2	<input type="checkbox"/>	3	<input type="checkbox"/>	4	<input type="checkbox"/>	5	<input type="checkbox"/>	6	<input type="checkbox"/>	7	<input type="checkbox"/>	8	<input type="checkbox"/>
9	<input type="checkbox"/>	10	<input type="checkbox"/>	11	<input type="checkbox"/>	12	<input type="checkbox"/>	13	<input type="checkbox"/>	14	<input type="checkbox"/>	15	<input type="checkbox"/>	16	<input type="checkbox"/>
17	<input type="checkbox"/>	18	<input type="checkbox"/>	19	<input type="checkbox"/>	20	<input type="checkbox"/>	21	<input type="checkbox"/>	22	<input type="checkbox"/>	23	<input type="checkbox"/>	24	<input type="checkbox"/>
25	<input type="checkbox"/>	26	<input type="checkbox"/>	27	<input type="checkbox"/>	28	<input type="checkbox"/>	29	<input type="checkbox"/>	30	<input type="checkbox"/>	31	<input type="checkbox"/>	32	<input type="checkbox"/>
33	<input type="checkbox"/>	34	<input type="checkbox"/>	35	<input type="checkbox"/>	36	<input type="checkbox"/>	37	<input type="checkbox"/>	38	<input type="checkbox"/>	39	<input type="checkbox"/>	40	<input type="checkbox"/>
41	<input type="checkbox"/>	42	<input type="checkbox"/>	43	<input type="checkbox"/>	44	<input type="checkbox"/>	45	<input type="checkbox"/>	46	<input type="checkbox"/>	47	<input type="checkbox"/>	48	<input type="checkbox"/>
49	<input type="checkbox"/>	50	<input type="checkbox"/>	51	<input type="checkbox"/>	52	<input type="checkbox"/>	53	<input type="checkbox"/>	54	<input type="checkbox"/>	55	<input type="checkbox"/>	56	<input type="checkbox"/>
57	<input type="checkbox"/>	58	<input type="checkbox"/>	59	<input type="checkbox"/>	60	<input type="checkbox"/>	61	<input type="checkbox"/>	62	<input type="checkbox"/>	63	<input type="checkbox"/>	64	<input type="checkbox"/>
65	<input type="checkbox"/>	66	<input type="checkbox"/>	67	<input type="checkbox"/>	68	<input type="checkbox"/>	69	<input type="checkbox"/>	70	<input type="checkbox"/>	71	<input type="checkbox"/>	72	<input type="checkbox"/>
73	<input type="checkbox"/>	74	<input type="checkbox"/>	75	<input type="checkbox"/>	76	<input type="checkbox"/>	77	<input type="checkbox"/>	78	<input type="checkbox"/>	79	<input type="checkbox"/>	80	<input type="checkbox"/>
81	<input type="checkbox"/>	82	<input type="checkbox"/>	83	<input type="checkbox"/>	84	<input type="checkbox"/>	85	<input type="checkbox"/>	86	<input type="checkbox"/>	87	<input type="checkbox"/>	88	<input type="checkbox"/>
89	<input type="checkbox"/>	90	<input type="checkbox"/>	91	<input type="checkbox"/>	92	<input type="checkbox"/>	93	<input type="checkbox"/>	94	<input type="checkbox"/>	95	<input type="checkbox"/>	96	<input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one **only**.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 21

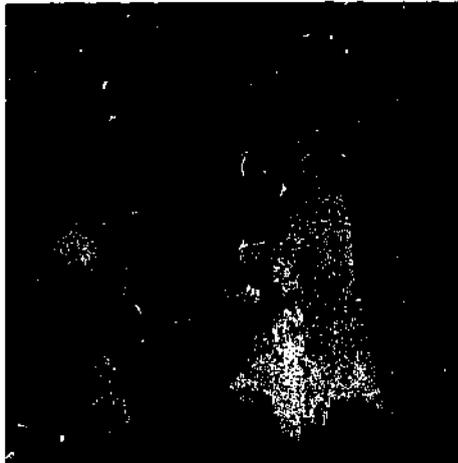


1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 22



1	<input type="checkbox"/>	2	<input type="checkbox"/>	3	<input type="checkbox"/>	4	<input type="checkbox"/>	5	<input type="checkbox"/>	6	<input type="checkbox"/>	7	<input type="checkbox"/>	8	<input type="checkbox"/>
9	<input type="checkbox"/>	10	<input type="checkbox"/>	11	<input type="checkbox"/>	12	<input type="checkbox"/>	13	<input type="checkbox"/>	14	<input type="checkbox"/>	15	<input type="checkbox"/>	16	<input type="checkbox"/>
17	<input type="checkbox"/>	18	<input type="checkbox"/>	19	<input type="checkbox"/>	20	<input type="checkbox"/>	21	<input type="checkbox"/>	22	<input type="checkbox"/>	23	<input type="checkbox"/>	24	<input type="checkbox"/>
25	<input type="checkbox"/>	26	<input type="checkbox"/>	27	<input type="checkbox"/>	28	<input type="checkbox"/>	29	<input type="checkbox"/>	30	<input type="checkbox"/>	31	<input type="checkbox"/>	32	<input type="checkbox"/>
33	<input type="checkbox"/>	34	<input type="checkbox"/>	35	<input type="checkbox"/>	36	<input type="checkbox"/>	37	<input type="checkbox"/>	38	<input type="checkbox"/>	39	<input type="checkbox"/>	40	<input type="checkbox"/>
41	<input type="checkbox"/>	42	<input type="checkbox"/>	43	<input type="checkbox"/>	44	<input type="checkbox"/>	45	<input type="checkbox"/>	46	<input type="checkbox"/>	47	<input type="checkbox"/>	48	<input type="checkbox"/>
49	<input type="checkbox"/>	50	<input type="checkbox"/>	51	<input type="checkbox"/>	52	<input type="checkbox"/>	53	<input type="checkbox"/>	54	<input type="checkbox"/>	55	<input type="checkbox"/>	56	<input type="checkbox"/>
57	<input type="checkbox"/>	58	<input type="checkbox"/>	59	<input type="checkbox"/>	60	<input type="checkbox"/>	61	<input type="checkbox"/>	62	<input type="checkbox"/>	63	<input type="checkbox"/>	64	<input type="checkbox"/>
65	<input type="checkbox"/>	66	<input type="checkbox"/>	67	<input type="checkbox"/>	68	<input type="checkbox"/>	69	<input type="checkbox"/>	70	<input type="checkbox"/>	71	<input type="checkbox"/>	72	<input type="checkbox"/>
73	<input type="checkbox"/>	74	<input type="checkbox"/>	75	<input type="checkbox"/>	76	<input type="checkbox"/>	77	<input type="checkbox"/>	78	<input type="checkbox"/>	79	<input type="checkbox"/>	80	<input type="checkbox"/>
81	<input type="checkbox"/>	82	<input type="checkbox"/>	83	<input type="checkbox"/>	84	<input type="checkbox"/>	85	<input type="checkbox"/>	86	<input type="checkbox"/>	87	<input type="checkbox"/>	88	<input type="checkbox"/>
89	<input type="checkbox"/>	90	<input type="checkbox"/>	91	<input type="checkbox"/>	92	<input type="checkbox"/>	93	<input type="checkbox"/>	94	<input type="checkbox"/>	95	<input type="checkbox"/>	96	<input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 23



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 24



1 []	2 []	3 []	4 []	5 []	6 []	7 []	8 []
9 []	10 []	11 []	12 []	13 []	14 []	15 []	16 []
17 []	18 []	19 []	20 []	21 []	22 []	23 []	24 []
25 []	26 []	27 []	28 []	29 []	30 []	31 []	32 []
33 []	34 []	35 []	36 []	37 []	38 []	39 []	40 []
41 []	42 []	43 []	44 []	45 []	46 []	47 []	48 []
49 []	50 []	51 []	52 []	53 []	54 []	55 []	56 []
57 []	58 []	59 []	60 []	61 []	62 []	63 []	64 []
65 []	66 []	67 []	68 []	69 []	70 []	71 []	72 []
73 []	74 []	75 []	76 []	77 []	78 []	79 []	80 []
81 []	82 []	83 []	84 []	85 []	86 []	87 []	88 []
89 []	90 []	91 []	92 []	93 []	94 []	95 []	96 []

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

1. main object

2. overall content

3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 25



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 26



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query ?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 27

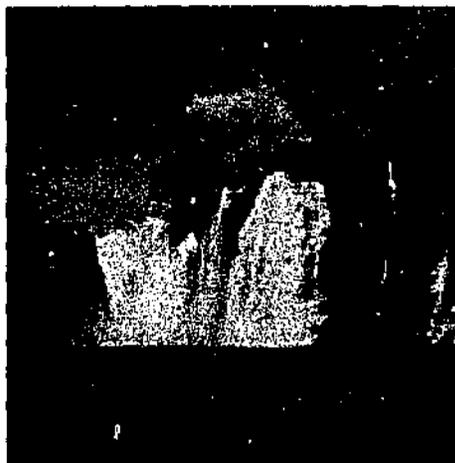


1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 28



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 29



1	<input type="checkbox"/>	2	<input type="checkbox"/>	3	<input type="checkbox"/>	4	<input type="checkbox"/>	5	<input type="checkbox"/>	6	<input type="checkbox"/>	7	<input type="checkbox"/>	8	<input type="checkbox"/>
9	<input type="checkbox"/>	10	<input type="checkbox"/>	11	<input type="checkbox"/>	12	<input type="checkbox"/>	13	<input type="checkbox"/>	14	<input type="checkbox"/>	15	<input type="checkbox"/>	16	<input type="checkbox"/>
17	<input type="checkbox"/>	18	<input type="checkbox"/>	19	<input type="checkbox"/>	20	<input type="checkbox"/>	21	<input type="checkbox"/>	22	<input type="checkbox"/>	23	<input type="checkbox"/>	24	<input type="checkbox"/>
25	<input type="checkbox"/>	26	<input type="checkbox"/>	27	<input type="checkbox"/>	28	<input type="checkbox"/>	29	<input type="checkbox"/>	30	<input type="checkbox"/>	31	<input type="checkbox"/>	32	<input type="checkbox"/>
33	<input type="checkbox"/>	34	<input type="checkbox"/>	35	<input type="checkbox"/>	36	<input type="checkbox"/>	37	<input type="checkbox"/>	38	<input type="checkbox"/>	39	<input type="checkbox"/>	40	<input type="checkbox"/>
41	<input type="checkbox"/>	42	<input type="checkbox"/>	43	<input type="checkbox"/>	44	<input type="checkbox"/>	45	<input type="checkbox"/>	46	<input type="checkbox"/>	47	<input type="checkbox"/>	48	<input type="checkbox"/>
49	<input type="checkbox"/>	50	<input type="checkbox"/>	51	<input type="checkbox"/>	52	<input type="checkbox"/>	53	<input type="checkbox"/>	54	<input type="checkbox"/>	55	<input type="checkbox"/>	56	<input type="checkbox"/>
57	<input type="checkbox"/>	58	<input type="checkbox"/>	59	<input type="checkbox"/>	60	<input type="checkbox"/>	61	<input type="checkbox"/>	62	<input type="checkbox"/>	63	<input type="checkbox"/>	64	<input type="checkbox"/>
65	<input type="checkbox"/>	66	<input type="checkbox"/>	67	<input type="checkbox"/>	68	<input type="checkbox"/>	69	<input type="checkbox"/>	70	<input type="checkbox"/>	71	<input type="checkbox"/>	72	<input type="checkbox"/>
73	<input type="checkbox"/>	74	<input type="checkbox"/>	75	<input type="checkbox"/>	76	<input type="checkbox"/>	77	<input type="checkbox"/>	78	<input type="checkbox"/>	79	<input type="checkbox"/>	80	<input type="checkbox"/>
81	<input type="checkbox"/>	82	<input type="checkbox"/>	83	<input type="checkbox"/>	84	<input type="checkbox"/>	85	<input type="checkbox"/>	86	<input type="checkbox"/>	87	<input type="checkbox"/>	88	<input type="checkbox"/>
89	<input type="checkbox"/>	90	<input type="checkbox"/>	91	<input type="checkbox"/>	92	<input type="checkbox"/>	93	<input type="checkbox"/>	94	<input type="checkbox"/>	95	<input type="checkbox"/>	96	<input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 30



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

1. main object

2. overall content

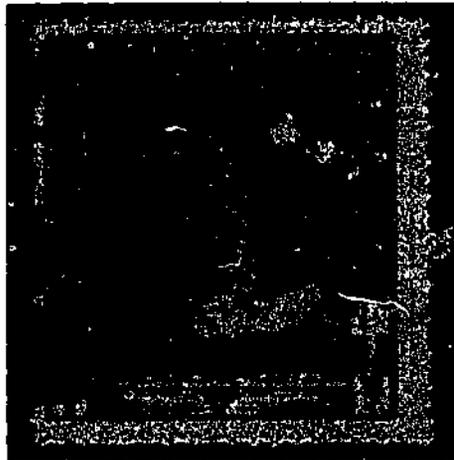
3. colour of the main object

4. background colour

5. shape of the main object

others, please specify _____

Query : Image 31

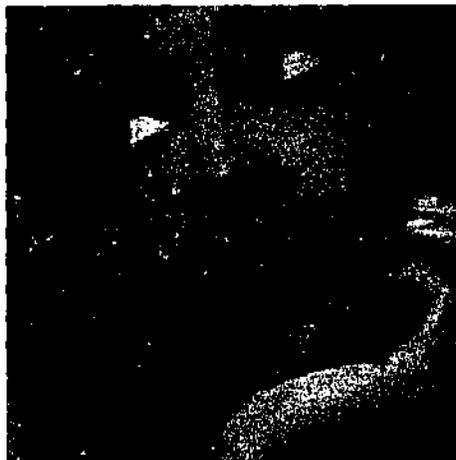


1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

Query : Image 32



1 <input type="checkbox"/>	2 <input type="checkbox"/>	3 <input type="checkbox"/>	4 <input type="checkbox"/>	5 <input type="checkbox"/>	6 <input type="checkbox"/>	7 <input type="checkbox"/>	8 <input type="checkbox"/>
9 <input type="checkbox"/>	10 <input type="checkbox"/>	11 <input type="checkbox"/>	12 <input type="checkbox"/>	13 <input type="checkbox"/>	14 <input type="checkbox"/>	15 <input type="checkbox"/>	16 <input type="checkbox"/>
17 <input type="checkbox"/>	18 <input type="checkbox"/>	19 <input type="checkbox"/>	20 <input type="checkbox"/>	21 <input type="checkbox"/>	22 <input type="checkbox"/>	23 <input type="checkbox"/>	24 <input type="checkbox"/>
25 <input type="checkbox"/>	26 <input type="checkbox"/>	27 <input type="checkbox"/>	28 <input type="checkbox"/>	29 <input type="checkbox"/>	30 <input type="checkbox"/>	31 <input type="checkbox"/>	32 <input type="checkbox"/>
33 <input type="checkbox"/>	34 <input type="checkbox"/>	35 <input type="checkbox"/>	36 <input type="checkbox"/>	37 <input type="checkbox"/>	38 <input type="checkbox"/>	39 <input type="checkbox"/>	40 <input type="checkbox"/>
41 <input type="checkbox"/>	42 <input type="checkbox"/>	43 <input type="checkbox"/>	44 <input type="checkbox"/>	45 <input type="checkbox"/>	46 <input type="checkbox"/>	47 <input type="checkbox"/>	48 <input type="checkbox"/>
49 <input type="checkbox"/>	50 <input type="checkbox"/>	51 <input type="checkbox"/>	52 <input type="checkbox"/>	53 <input type="checkbox"/>	54 <input type="checkbox"/>	55 <input type="checkbox"/>	56 <input type="checkbox"/>
57 <input type="checkbox"/>	58 <input type="checkbox"/>	59 <input type="checkbox"/>	60 <input type="checkbox"/>	61 <input type="checkbox"/>	62 <input type="checkbox"/>	63 <input type="checkbox"/>	64 <input type="checkbox"/>
65 <input type="checkbox"/>	66 <input type="checkbox"/>	67 <input type="checkbox"/>	68 <input type="checkbox"/>	69 <input type="checkbox"/>	70 <input type="checkbox"/>	71 <input type="checkbox"/>	72 <input type="checkbox"/>
73 <input type="checkbox"/>	74 <input type="checkbox"/>	75 <input type="checkbox"/>	76 <input type="checkbox"/>	77 <input type="checkbox"/>	78 <input type="checkbox"/>	79 <input type="checkbox"/>	80 <input type="checkbox"/>
81 <input type="checkbox"/>	82 <input type="checkbox"/>	83 <input type="checkbox"/>	84 <input type="checkbox"/>	85 <input type="checkbox"/>	86 <input type="checkbox"/>	87 <input type="checkbox"/>	88 <input type="checkbox"/>
89 <input type="checkbox"/>	90 <input type="checkbox"/>	91 <input type="checkbox"/>	92 <input type="checkbox"/>	93 <input type="checkbox"/>	94 <input type="checkbox"/>	95 <input type="checkbox"/>	96 <input type="checkbox"/>

Which is your main judging criteria in selecting the similar/relevant images for this query?
Tick one only.

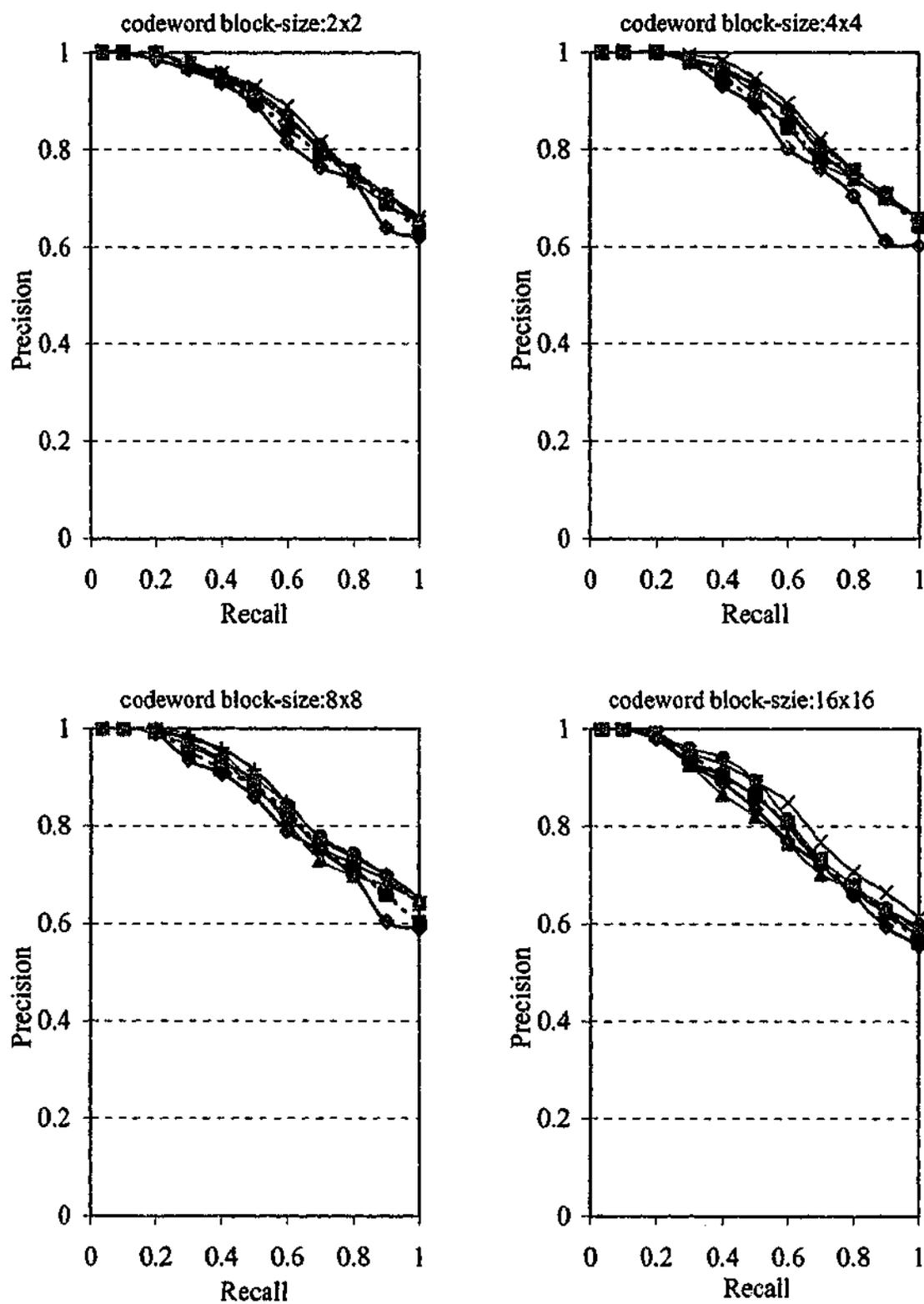
- 1. main object
- 2. overall content
- 3. colour of the main object
- 4. background colour
- 5. shape of the main object
- others, please specify _____

APPENDIX B

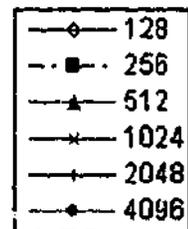
Average recall and precision graphs to show the effects of different parameters of the codebook on retrieval effectiveness. The experimental results in this appendix are obtained by using SCD as the test image database.

Appendix B1 : Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD	B1-1
Appendix B2 : Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD	B2-1
Appendix B3 : Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD	B3-1

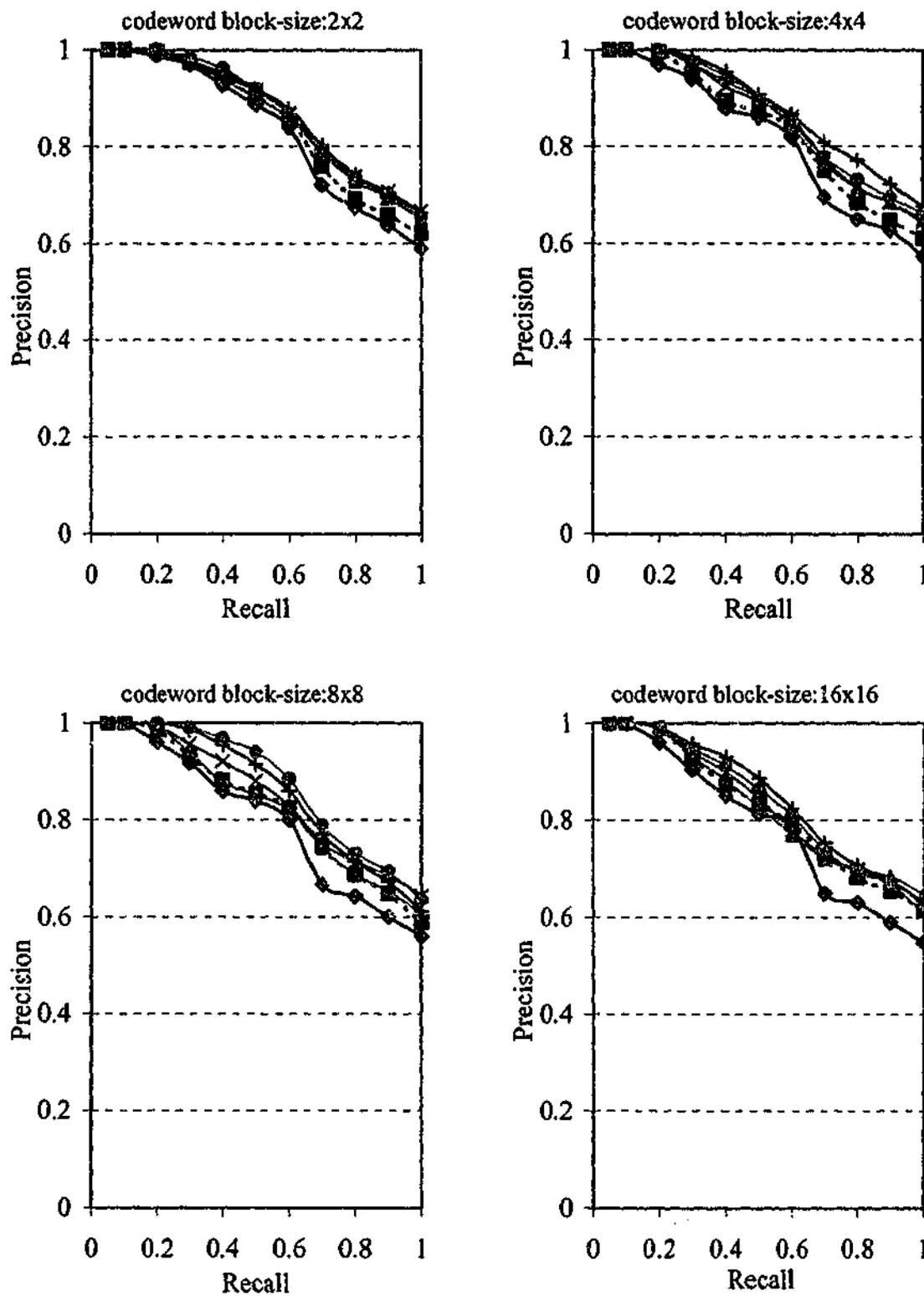
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



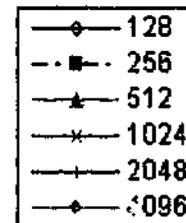
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is HSV. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



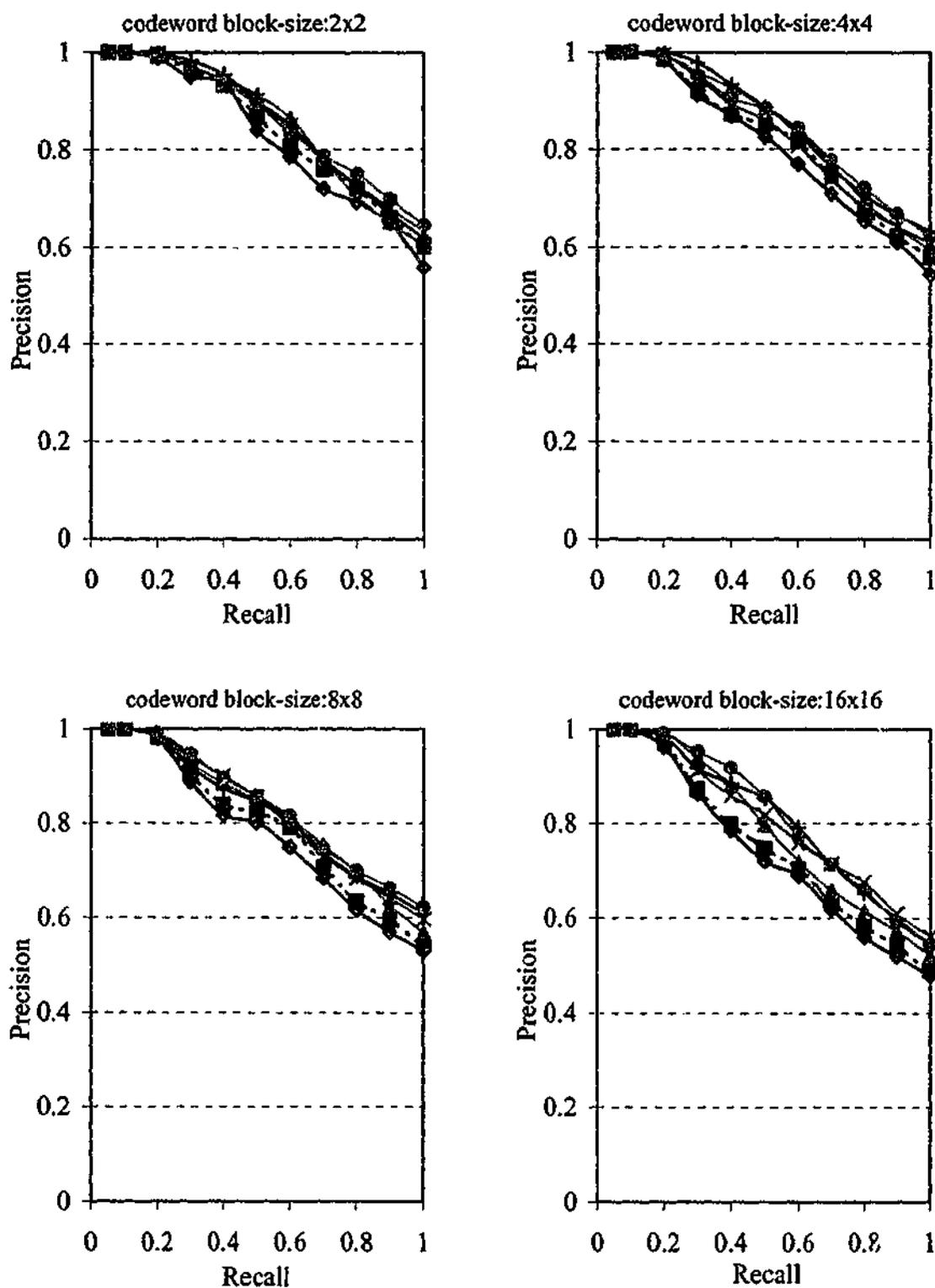
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



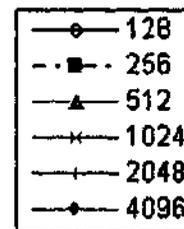
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is LUV. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



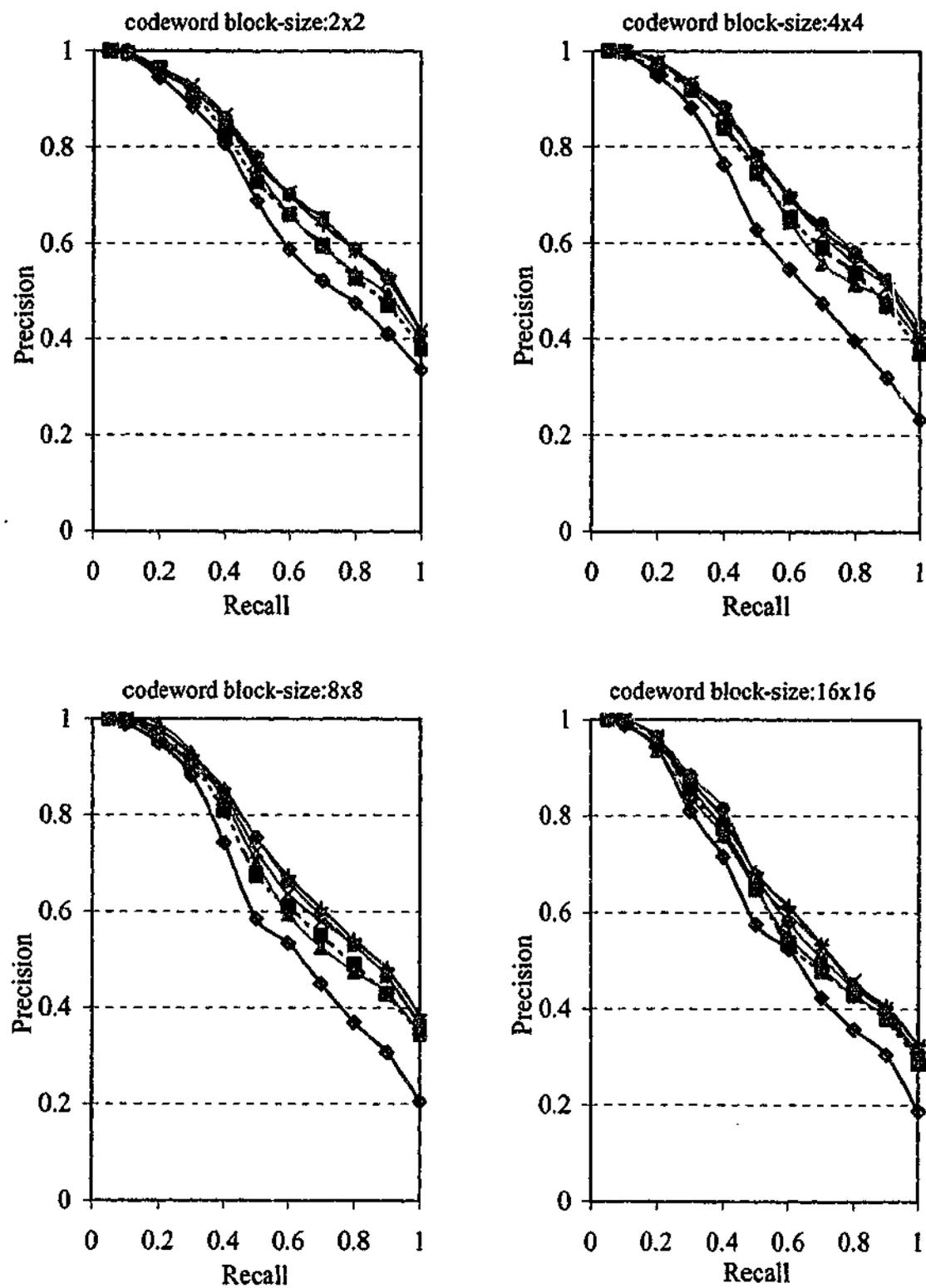
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



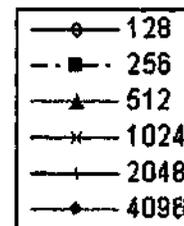
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is RGB. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



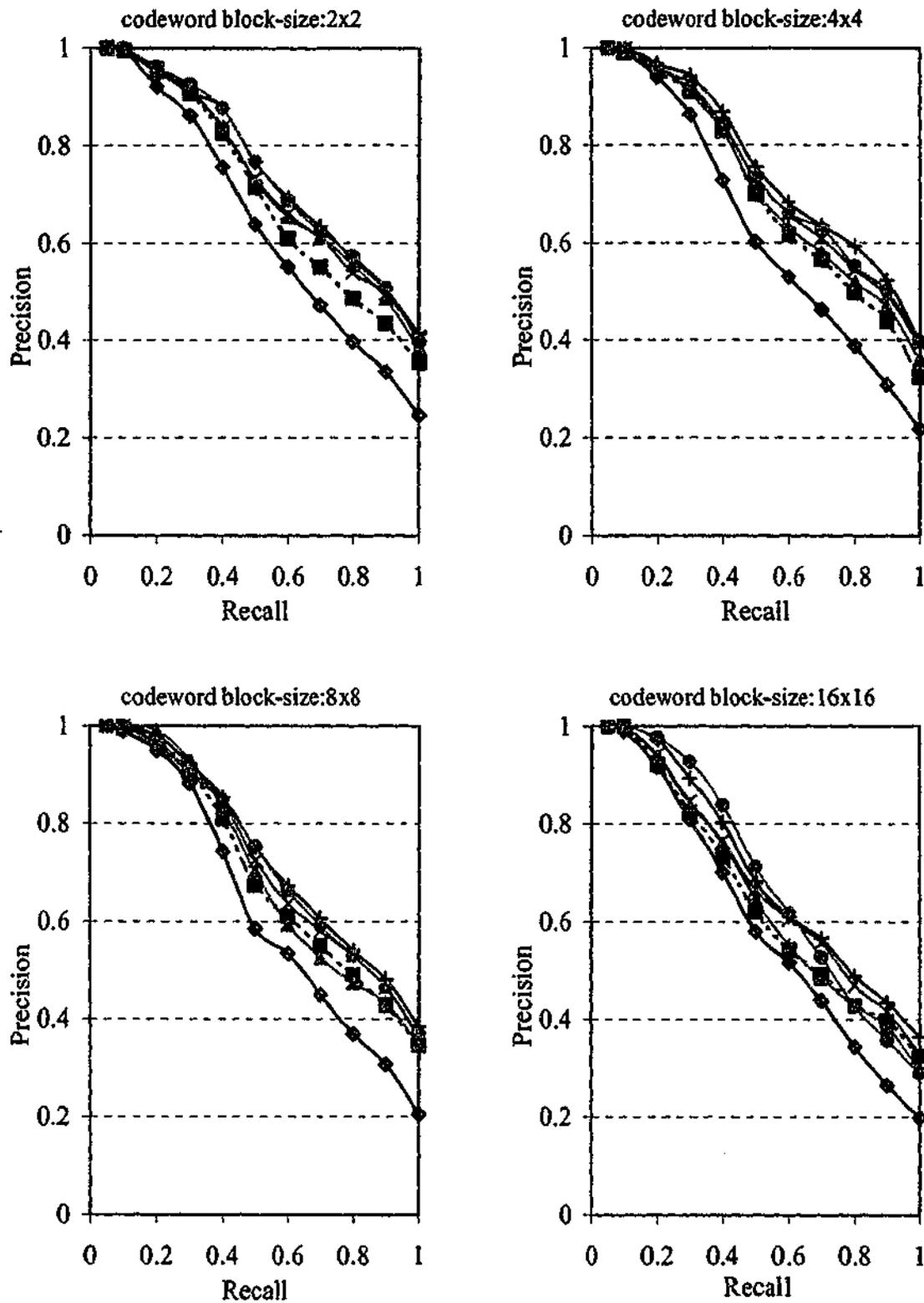
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



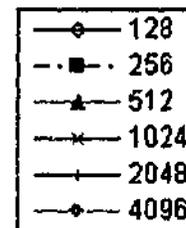
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is HSV. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



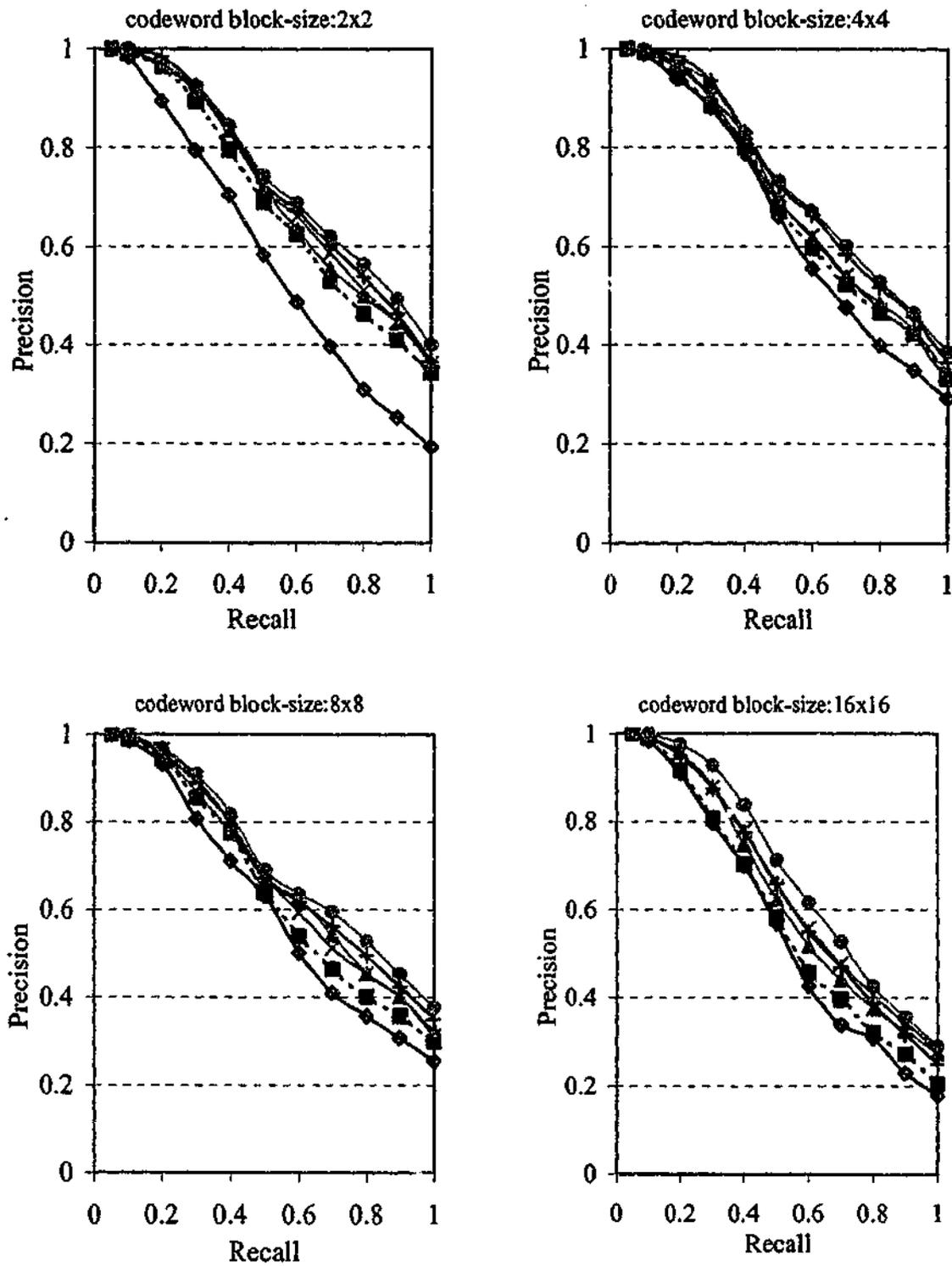
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



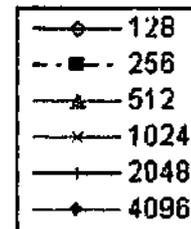
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is LUV. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



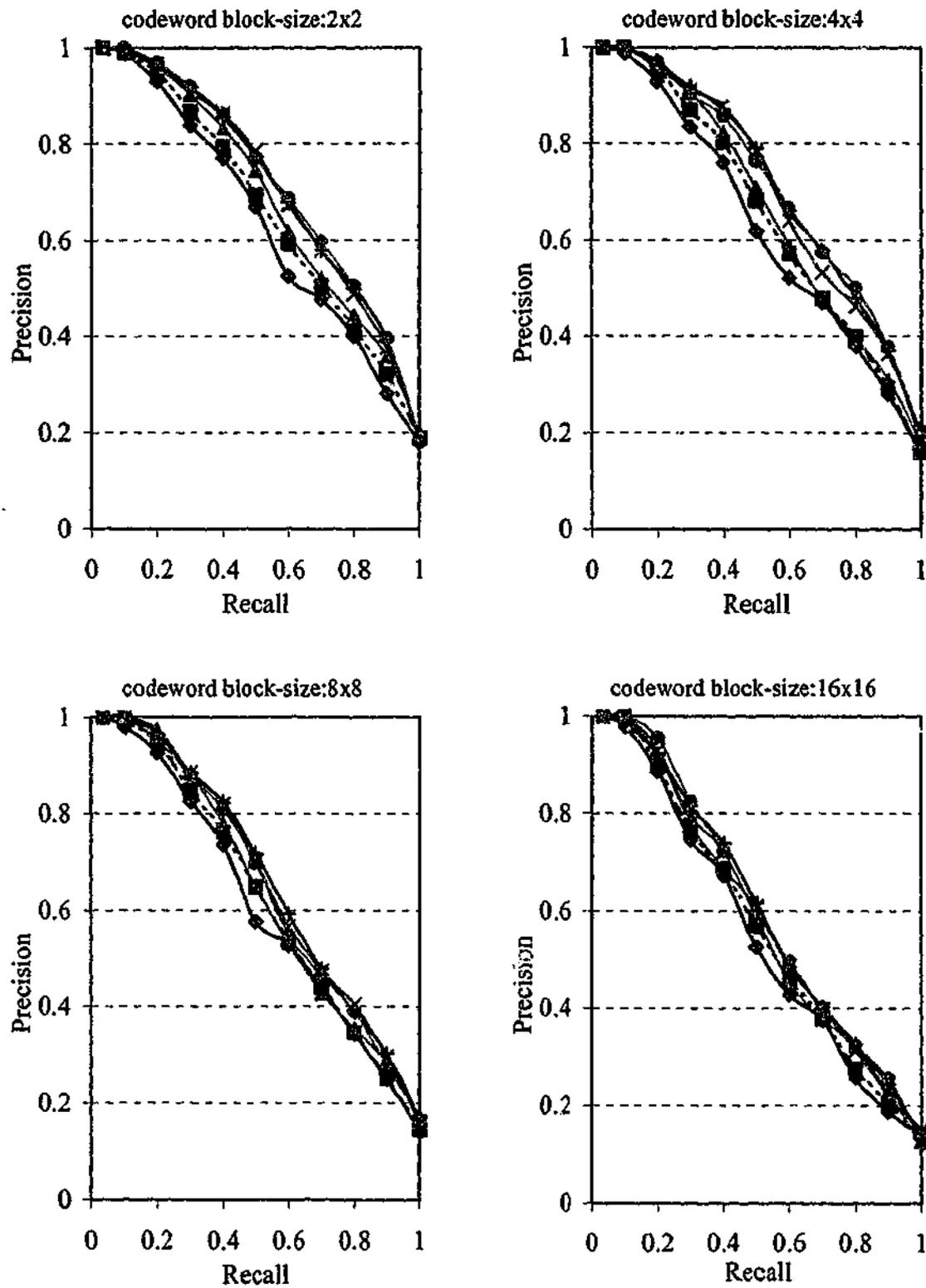
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



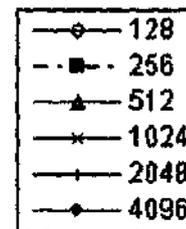
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is RGB. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



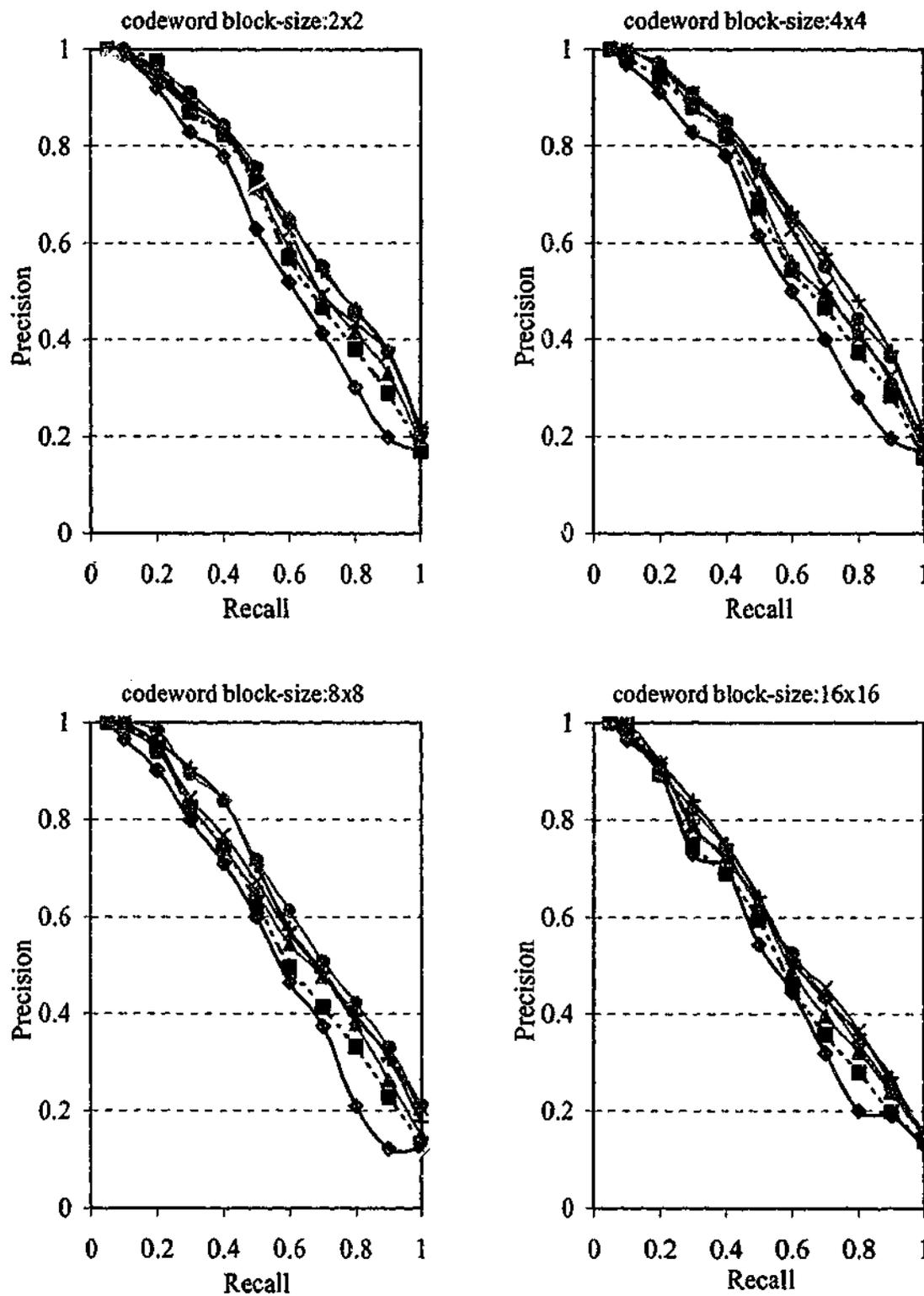
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



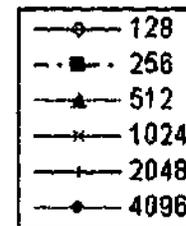
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is HSV. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



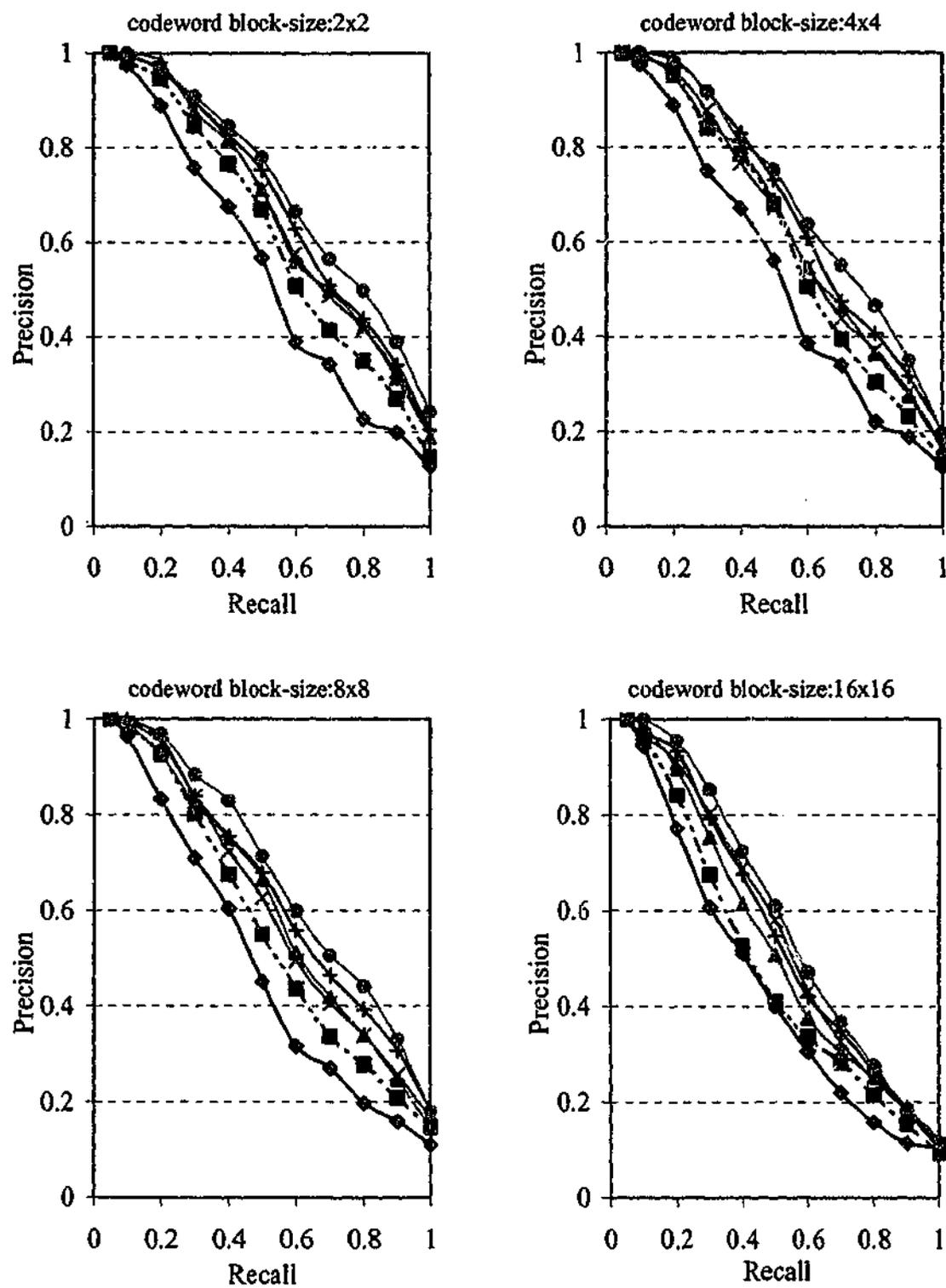
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



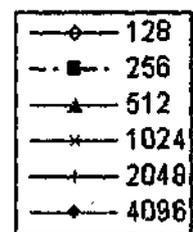
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is LUV. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



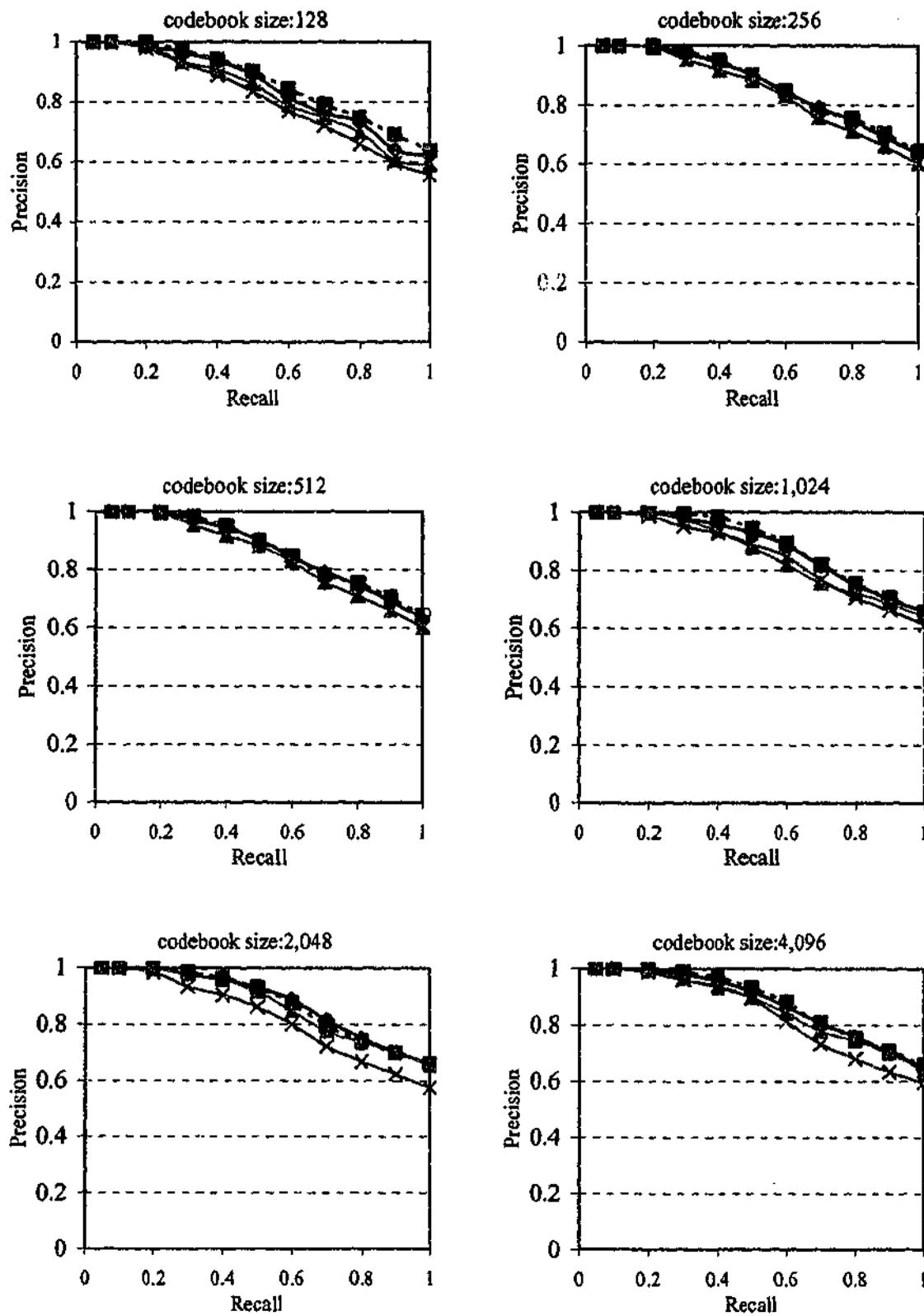
APPENDIX B1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database SCD



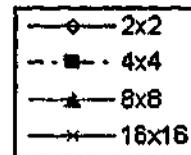
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is RGB. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



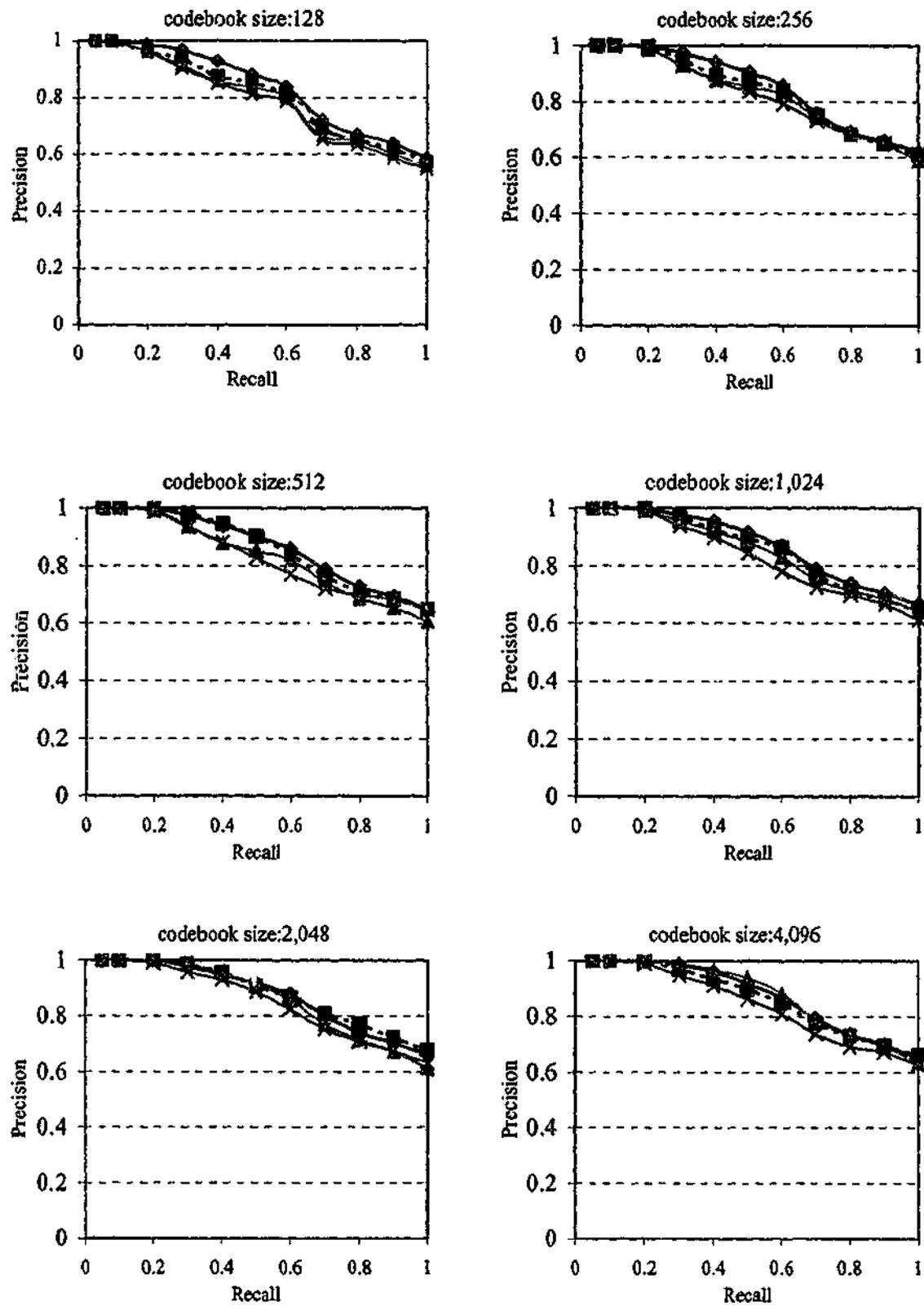
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



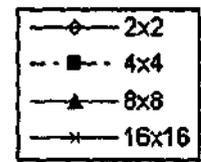
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is HSV. Experiment conducted on database SCD. Performance evaluated using GT⁷⁰. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



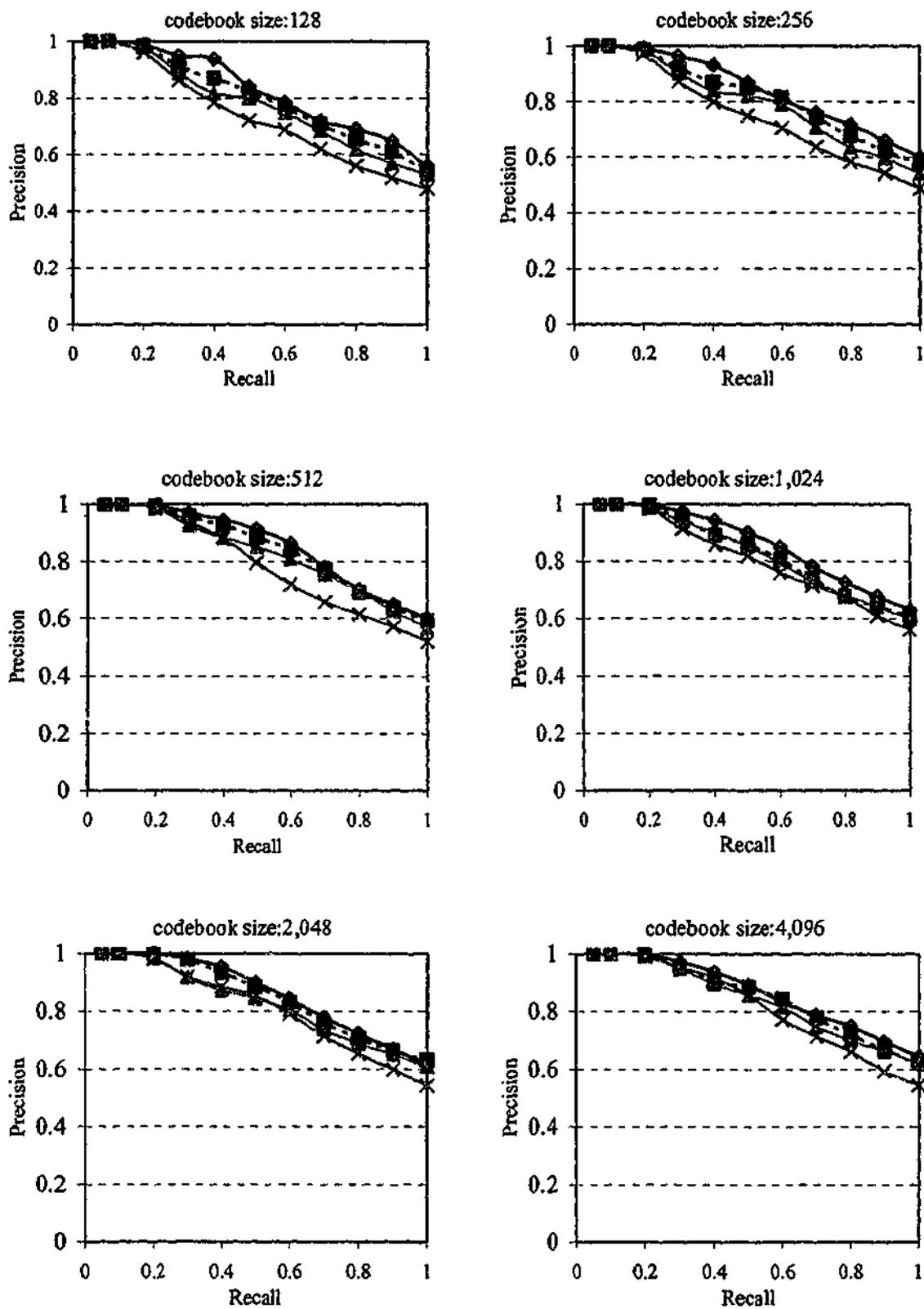
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



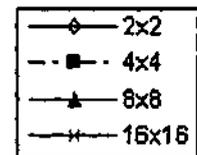
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is LUV. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



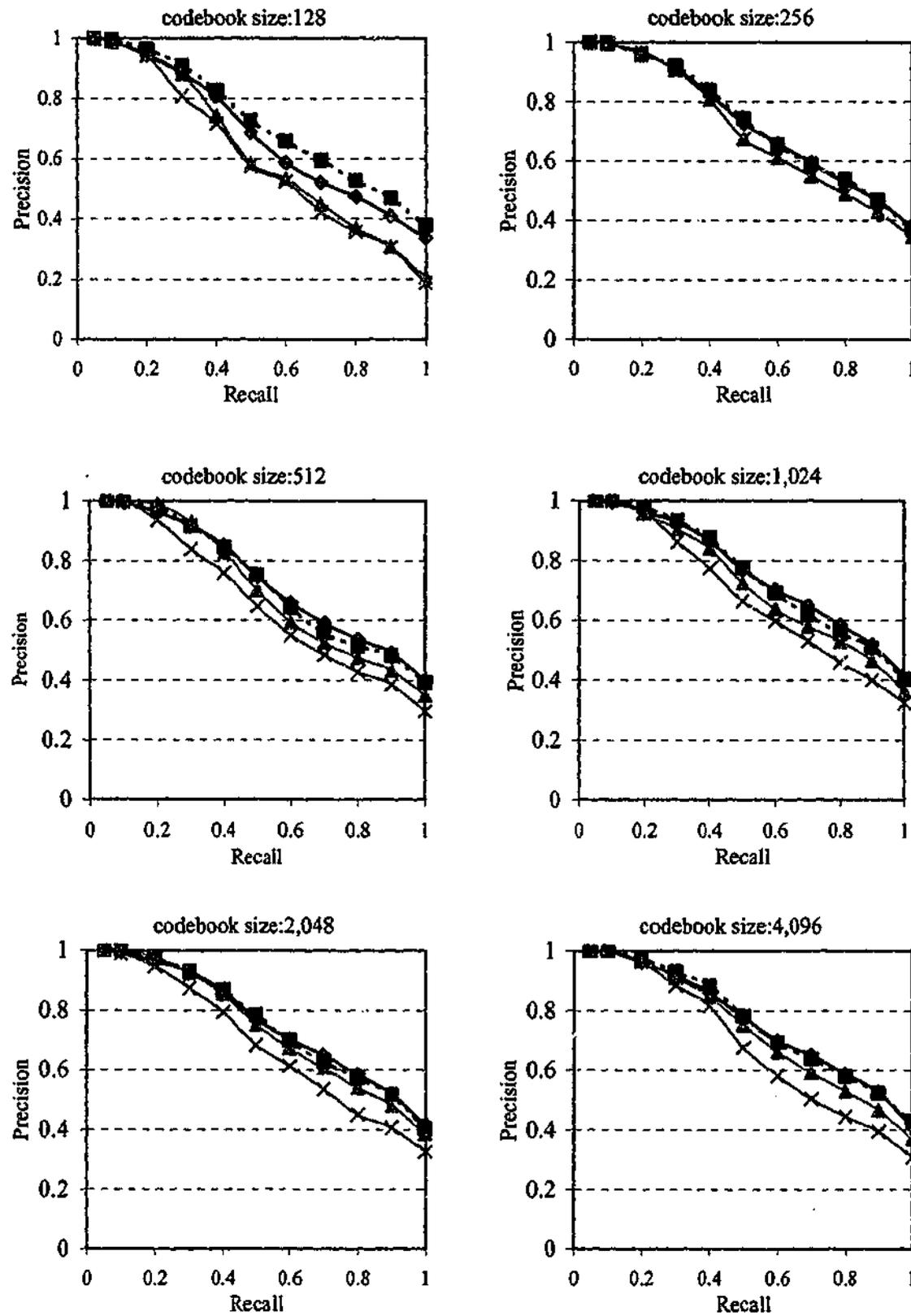
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



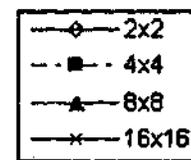
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is RGB. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



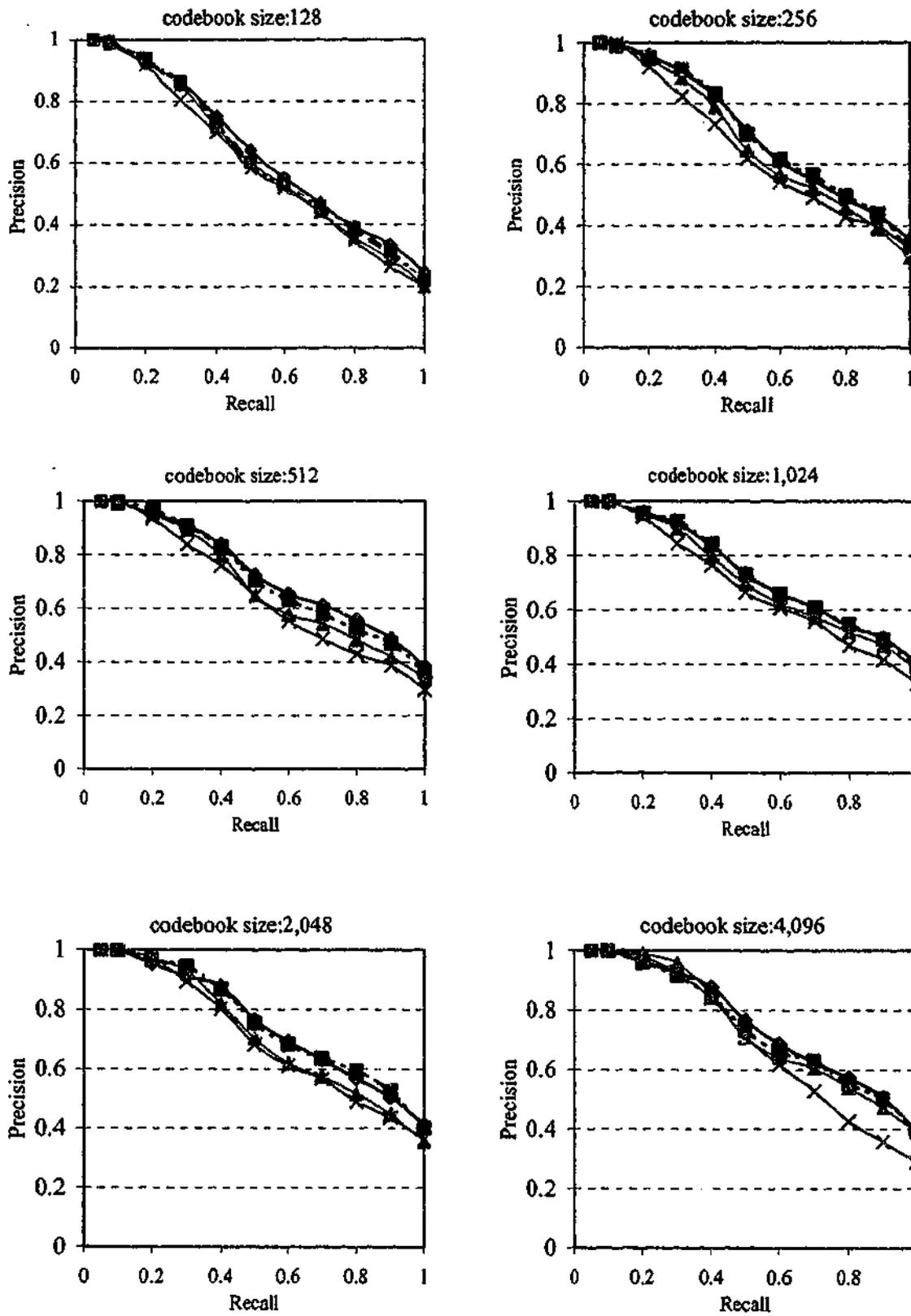
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



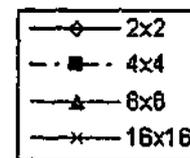
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is HSV. Experiment conducted on Database SCD. Performance evaluated using GT50. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



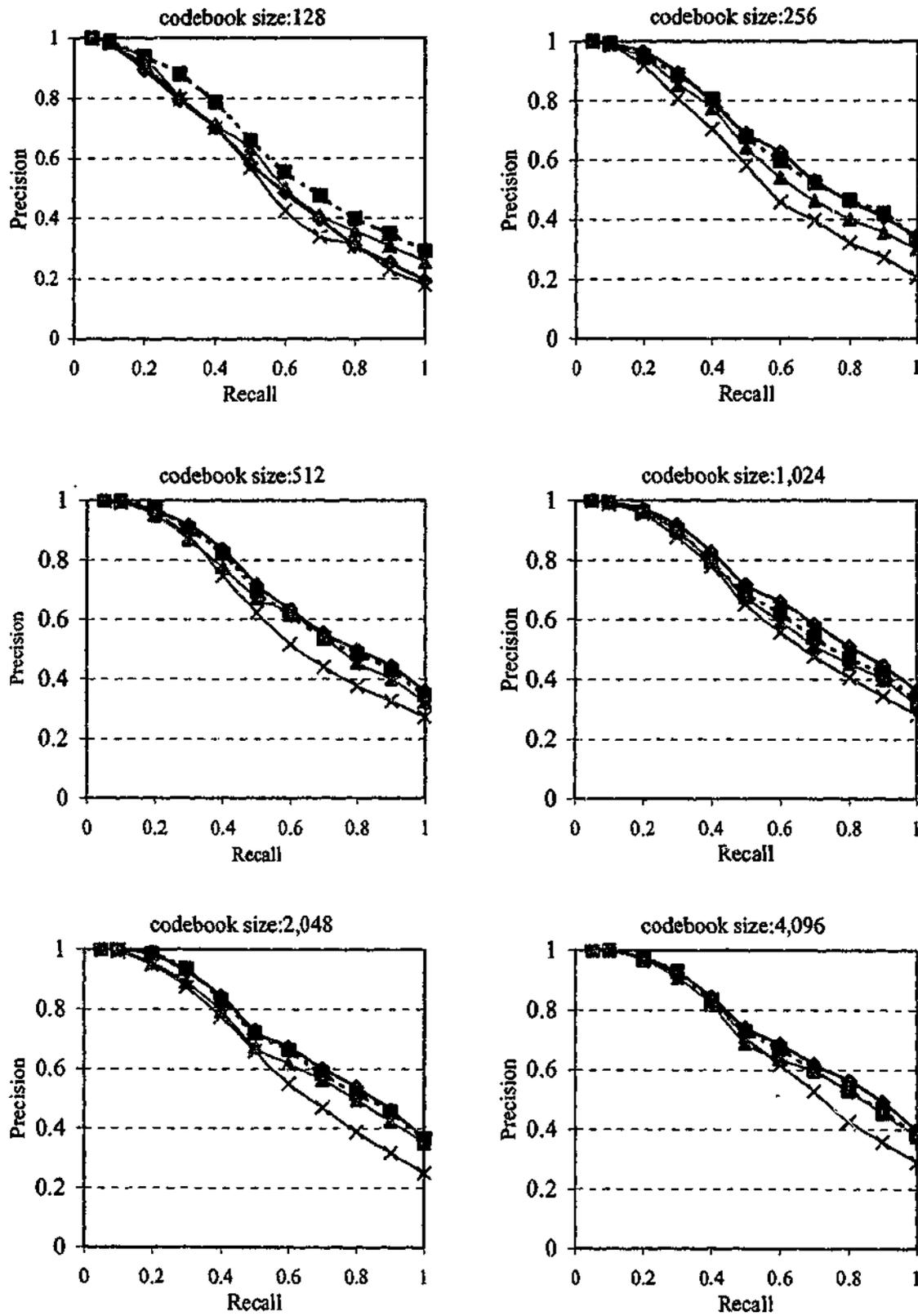
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



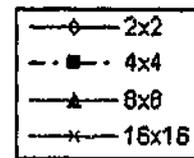
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is LUV. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



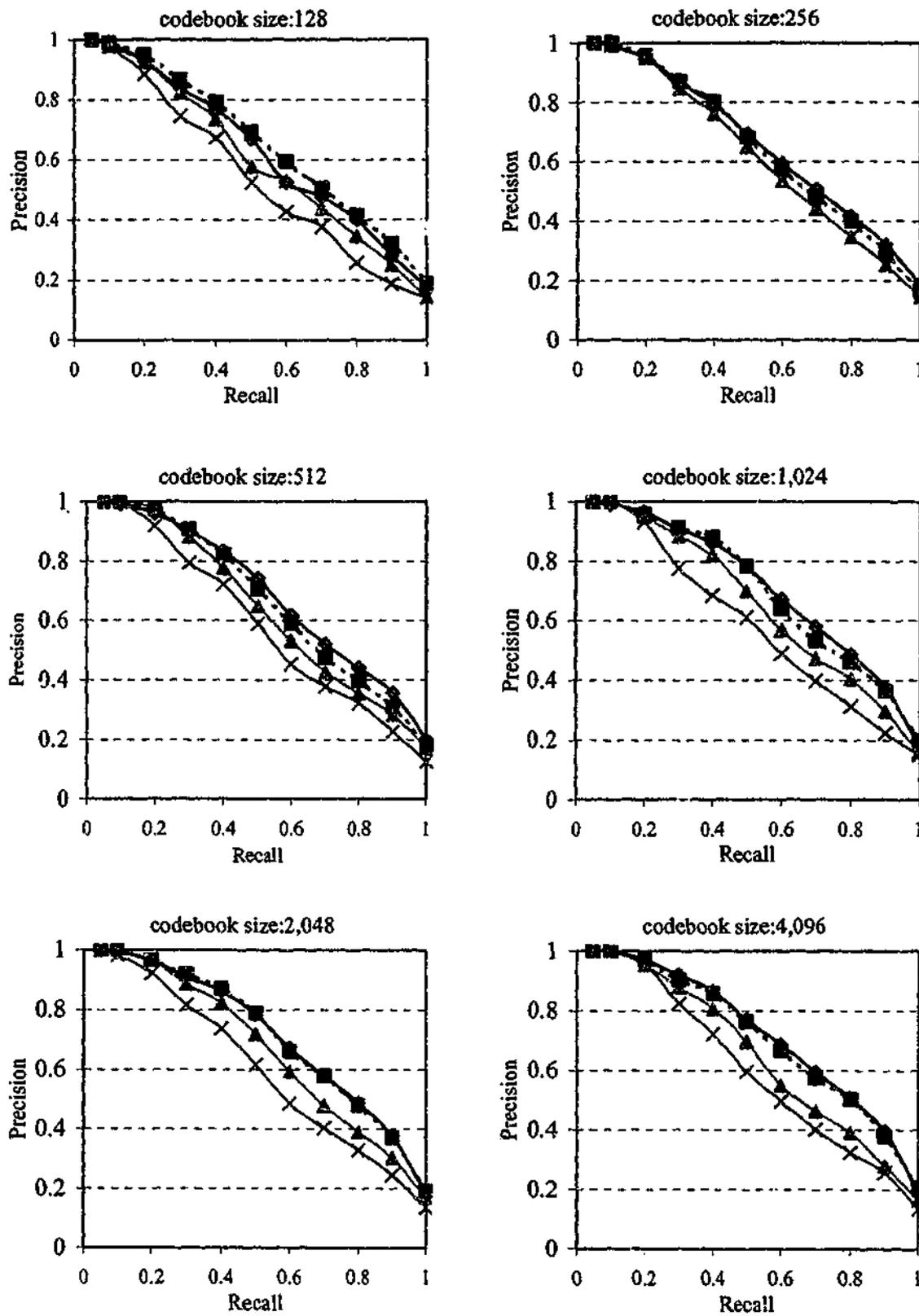
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



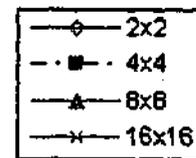
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is RGB. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



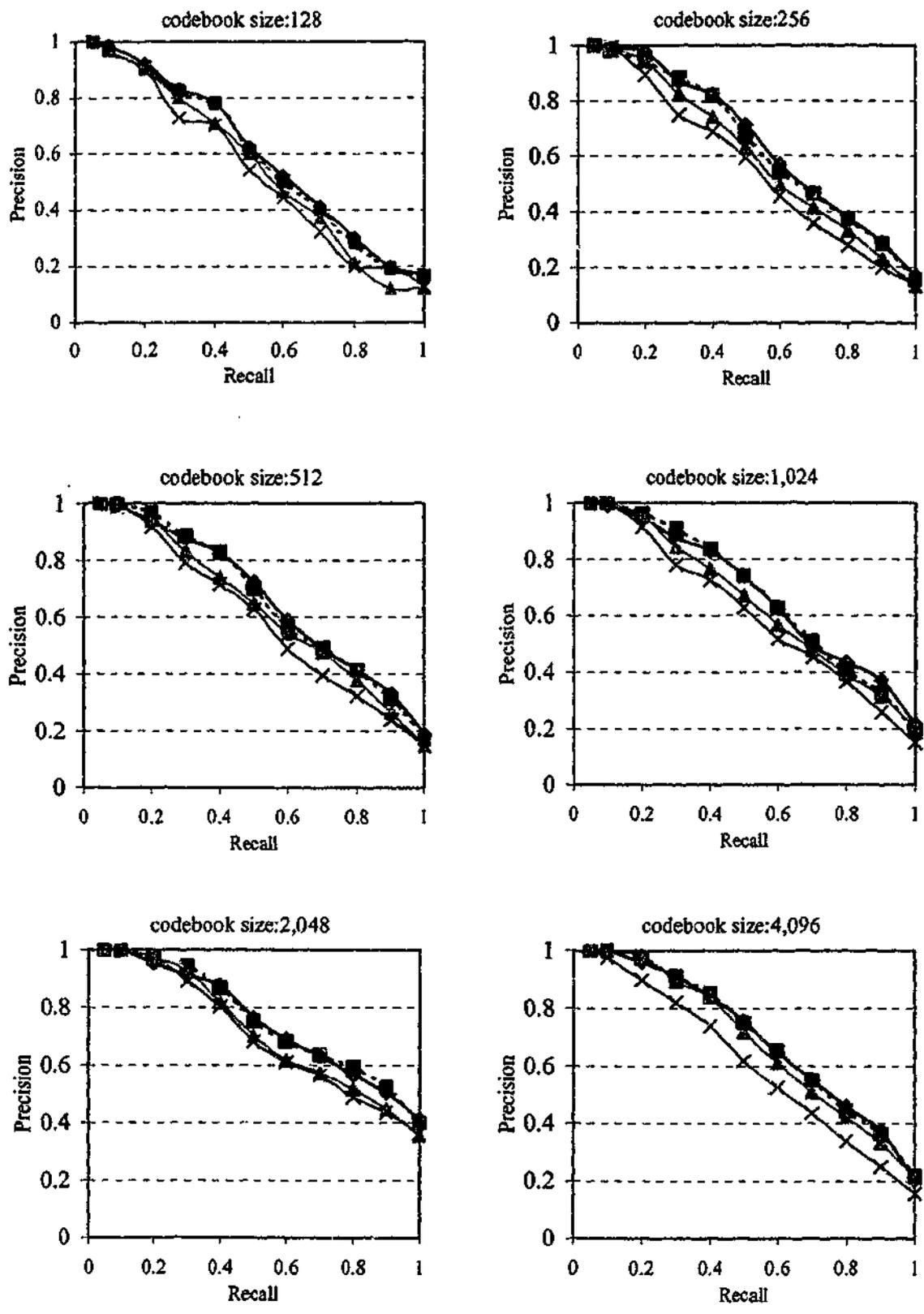
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



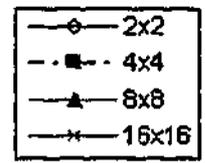
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is HSV. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



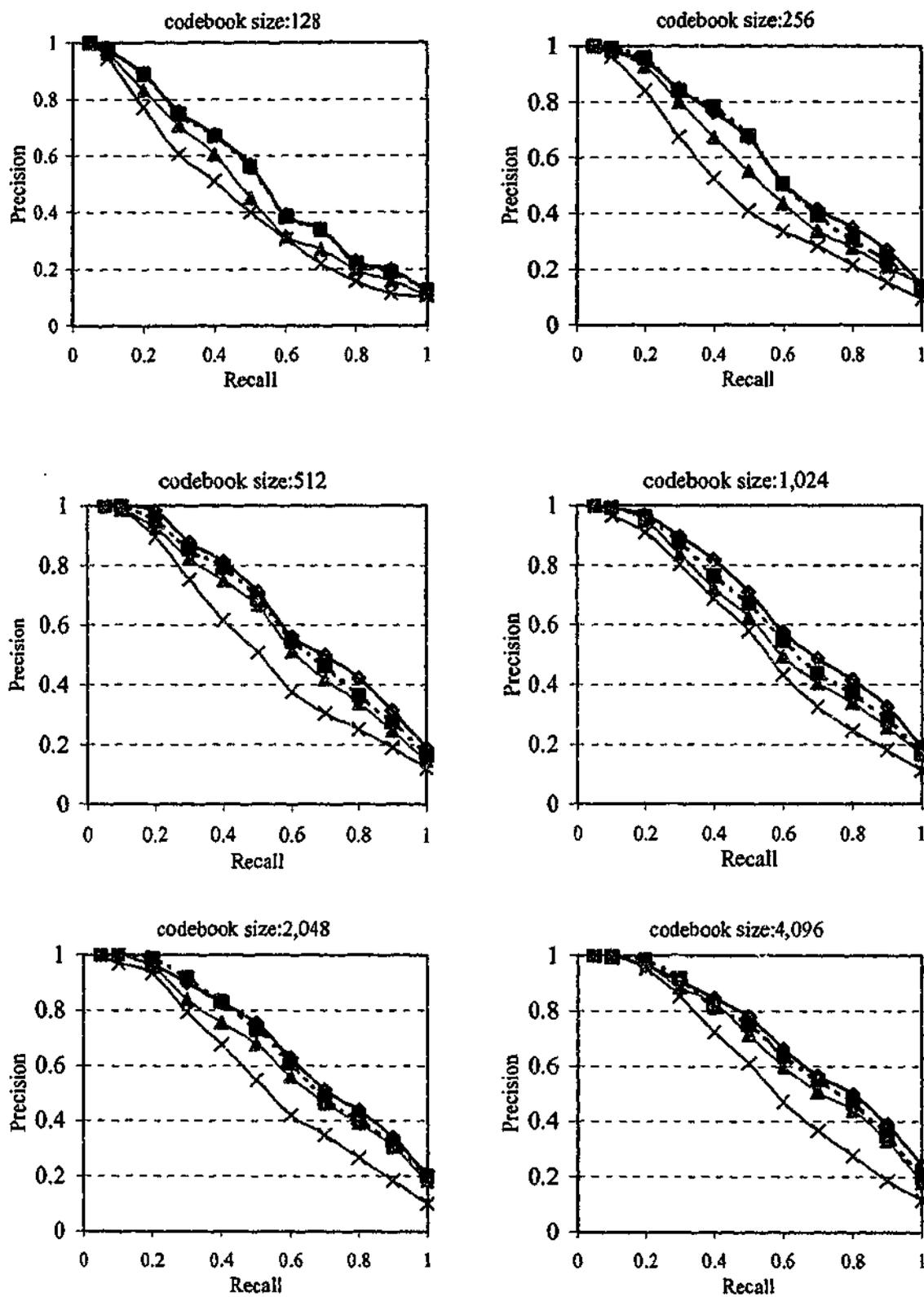
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



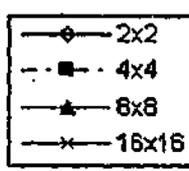
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is LUV. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



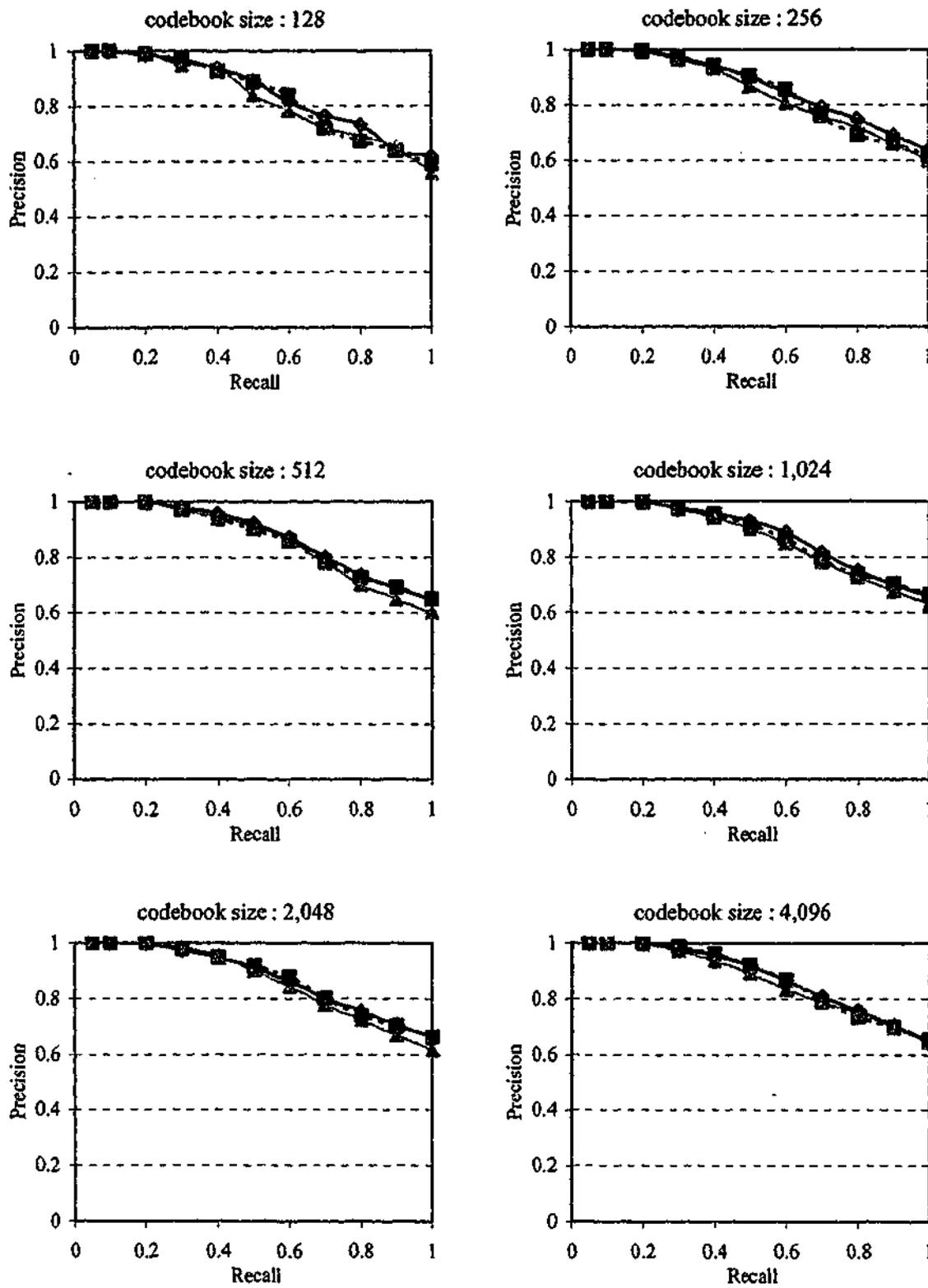
APPENDIX B2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database SCD



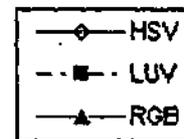
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is RGB. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



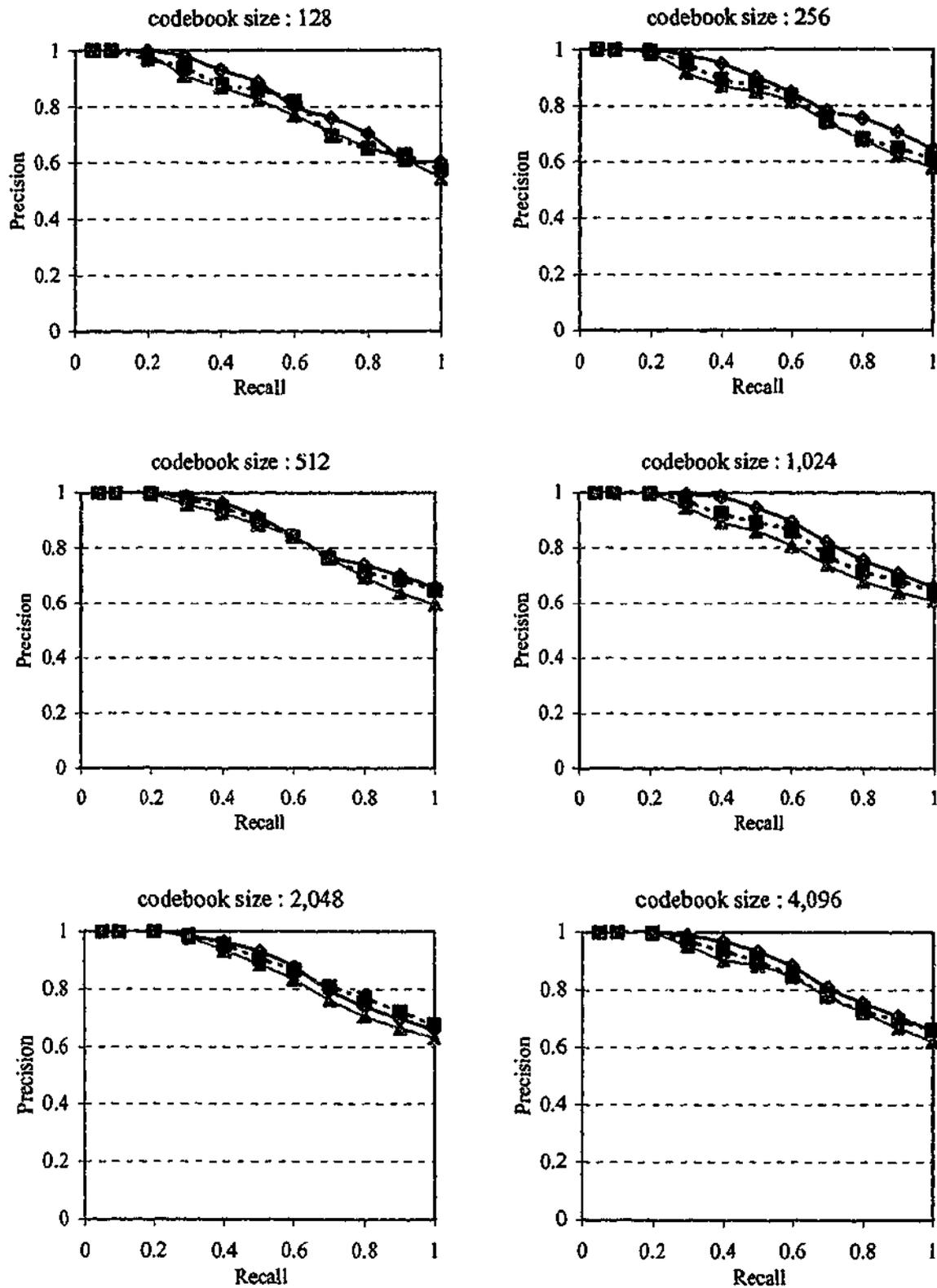
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



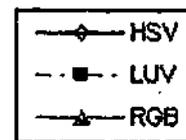
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 2x2 pixels. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



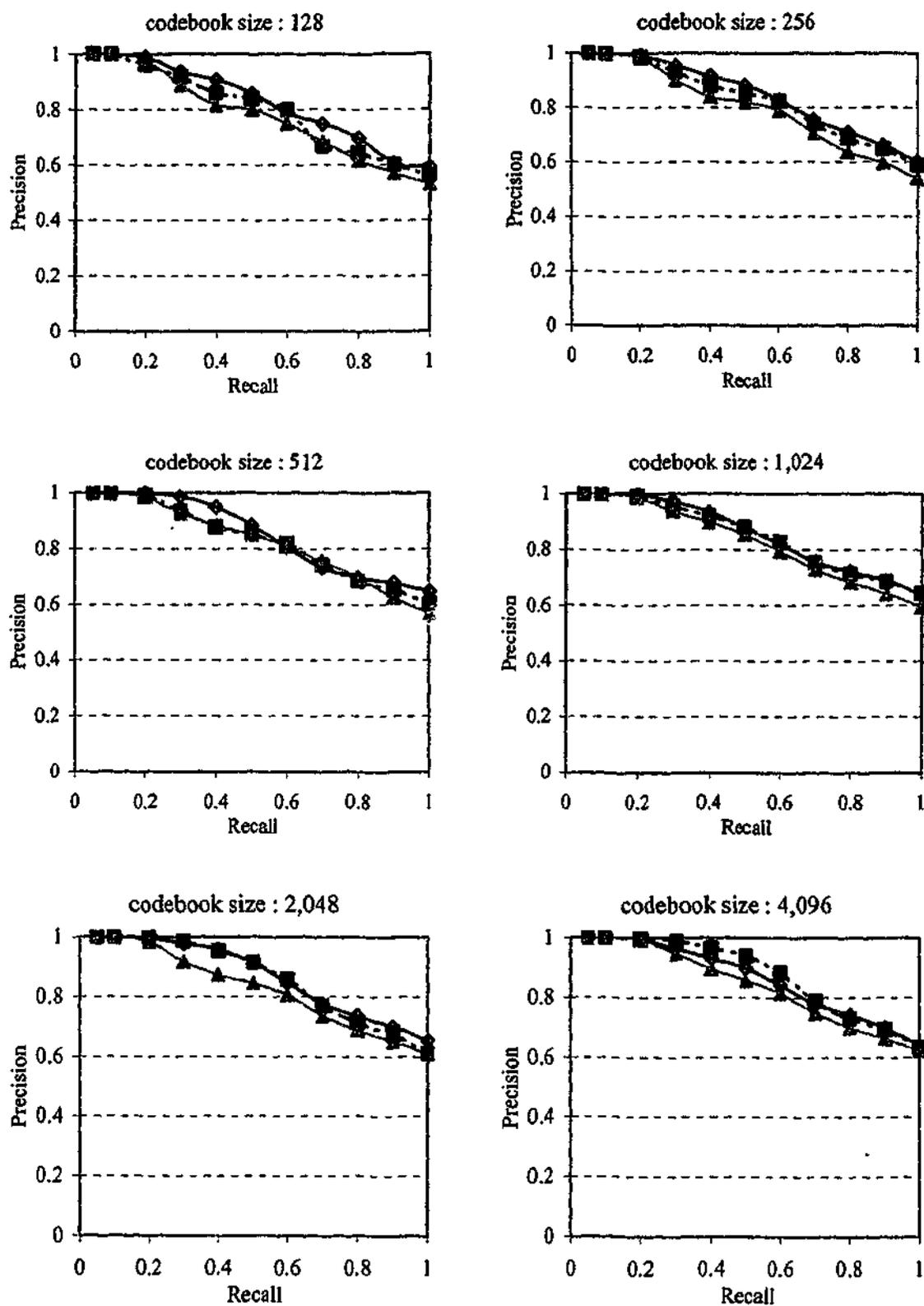
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



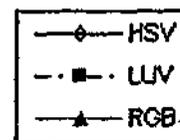
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 4x4 pixels. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



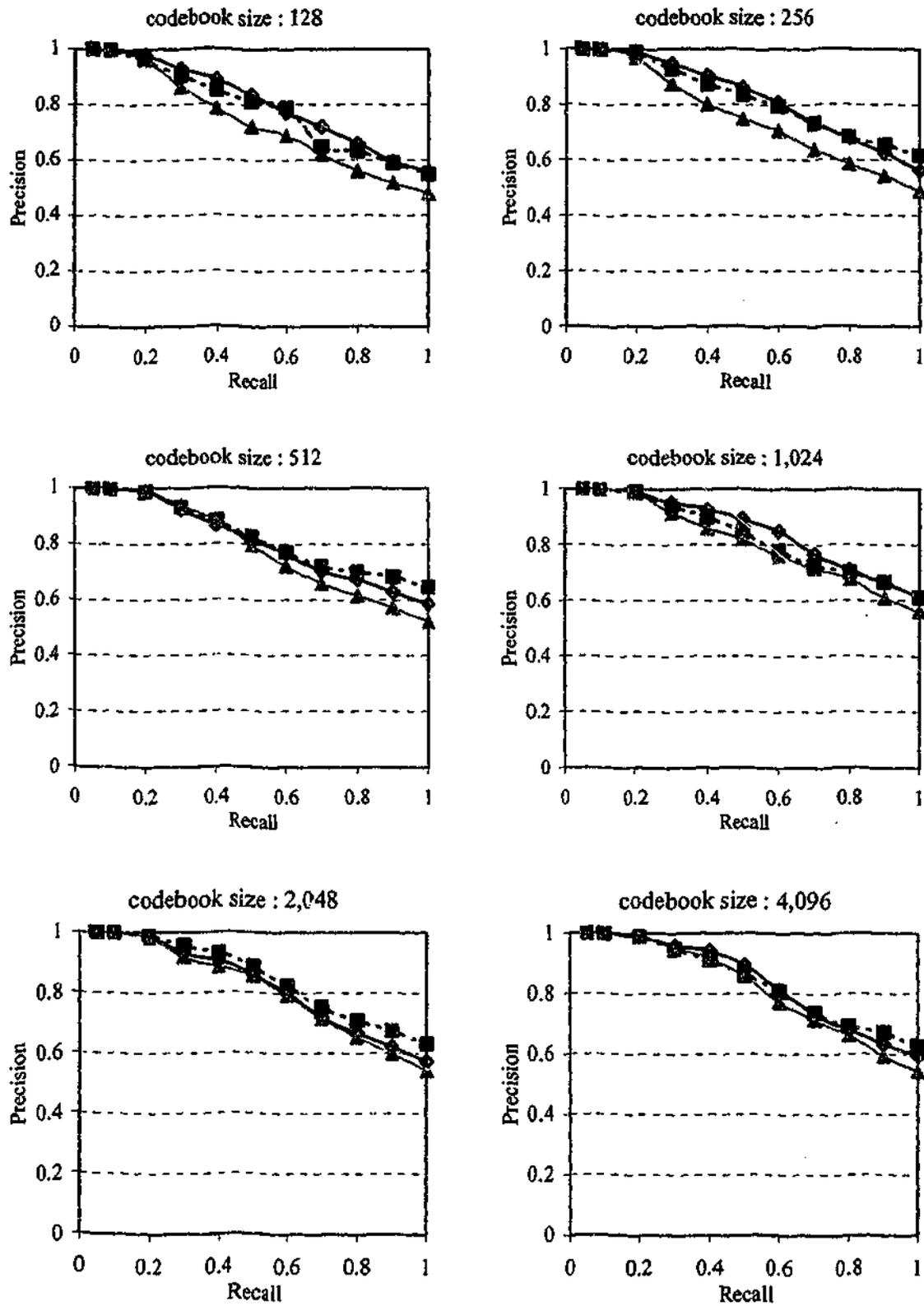
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



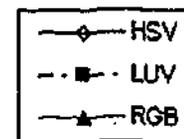
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 8x8 pixels. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



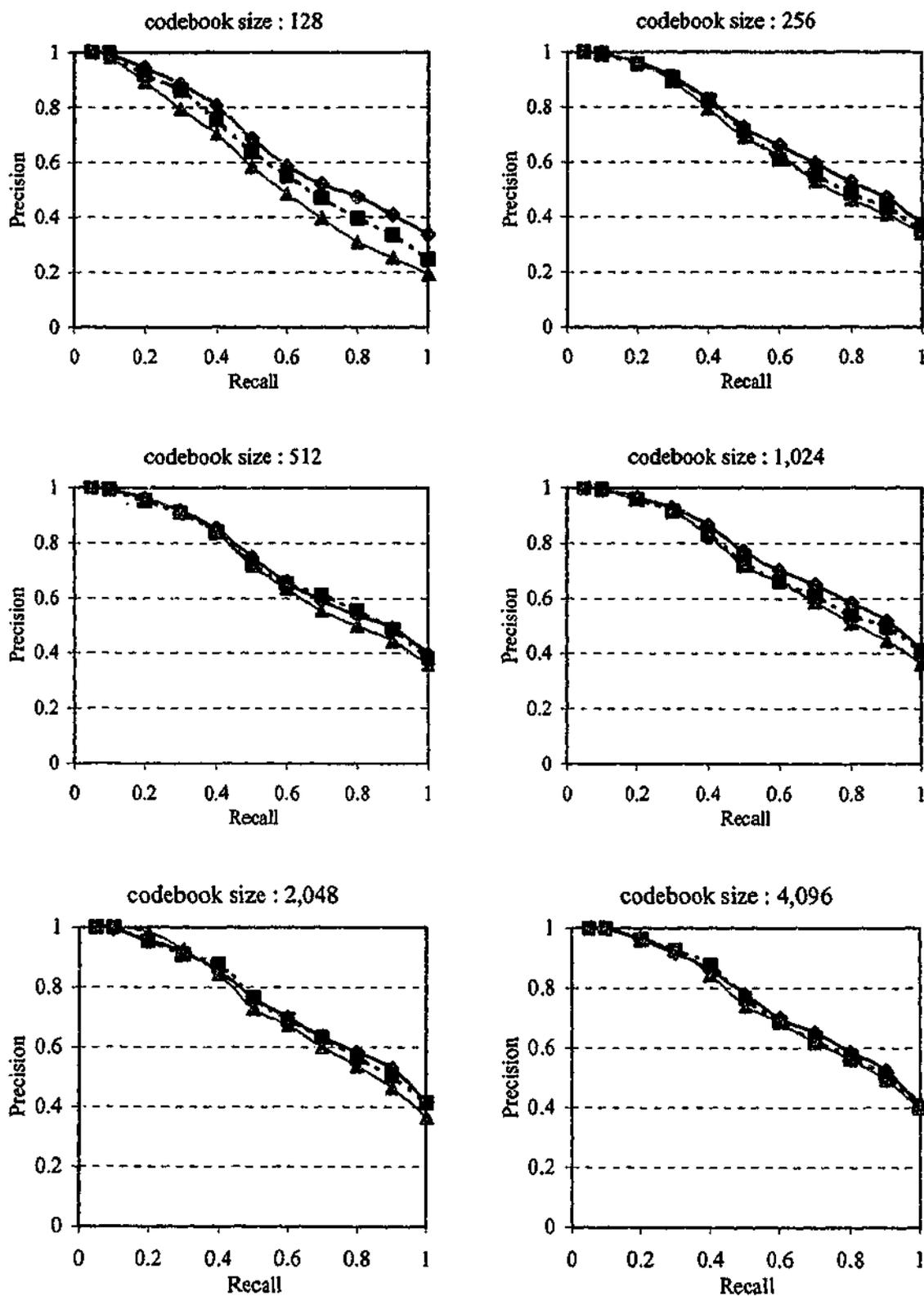
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



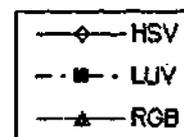
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 16x16 pixels. Experiment conducted on database SCD. Performance evaluated using GT70. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



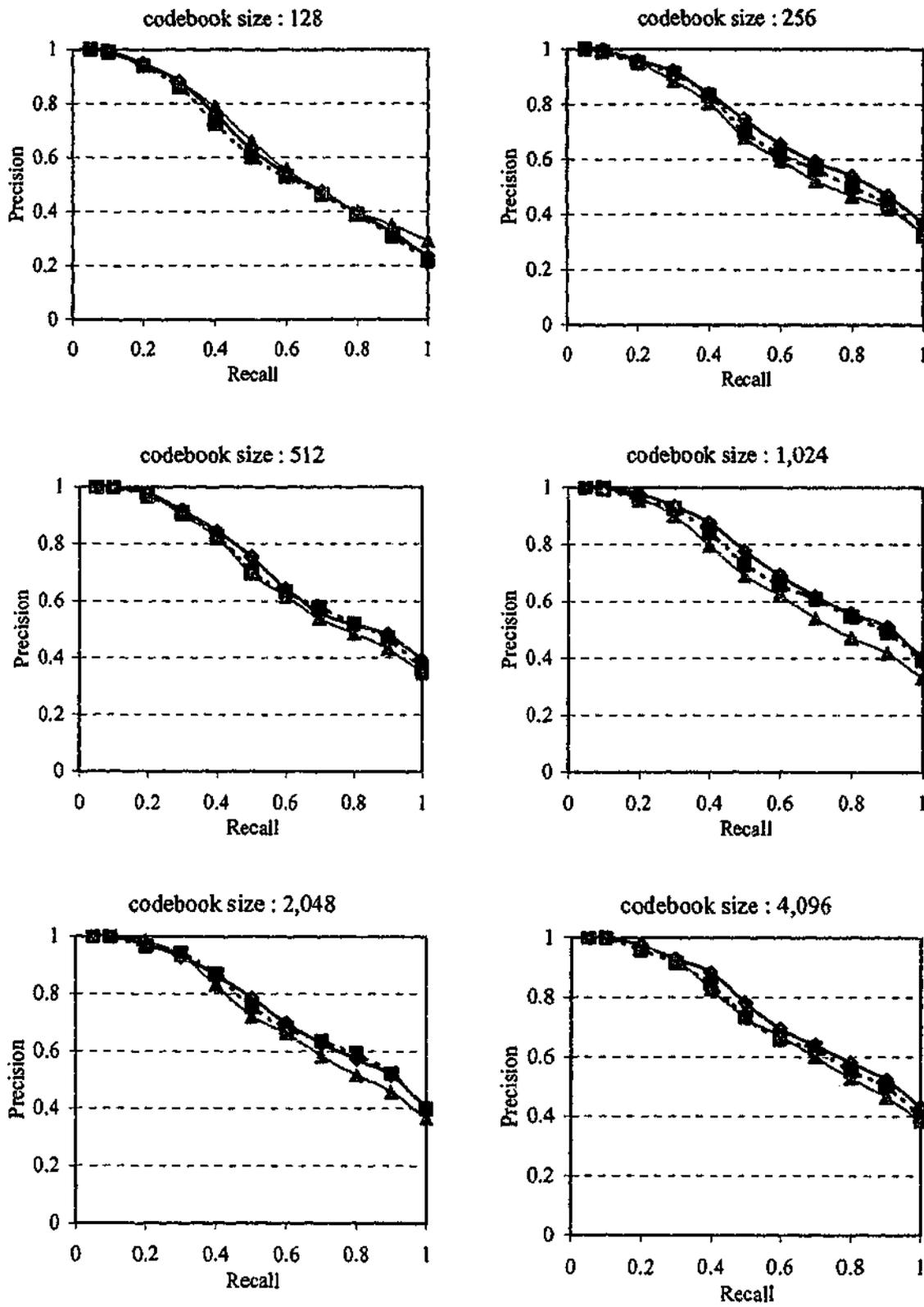
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



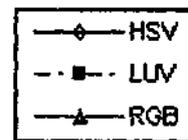
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 2x2 pixels. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



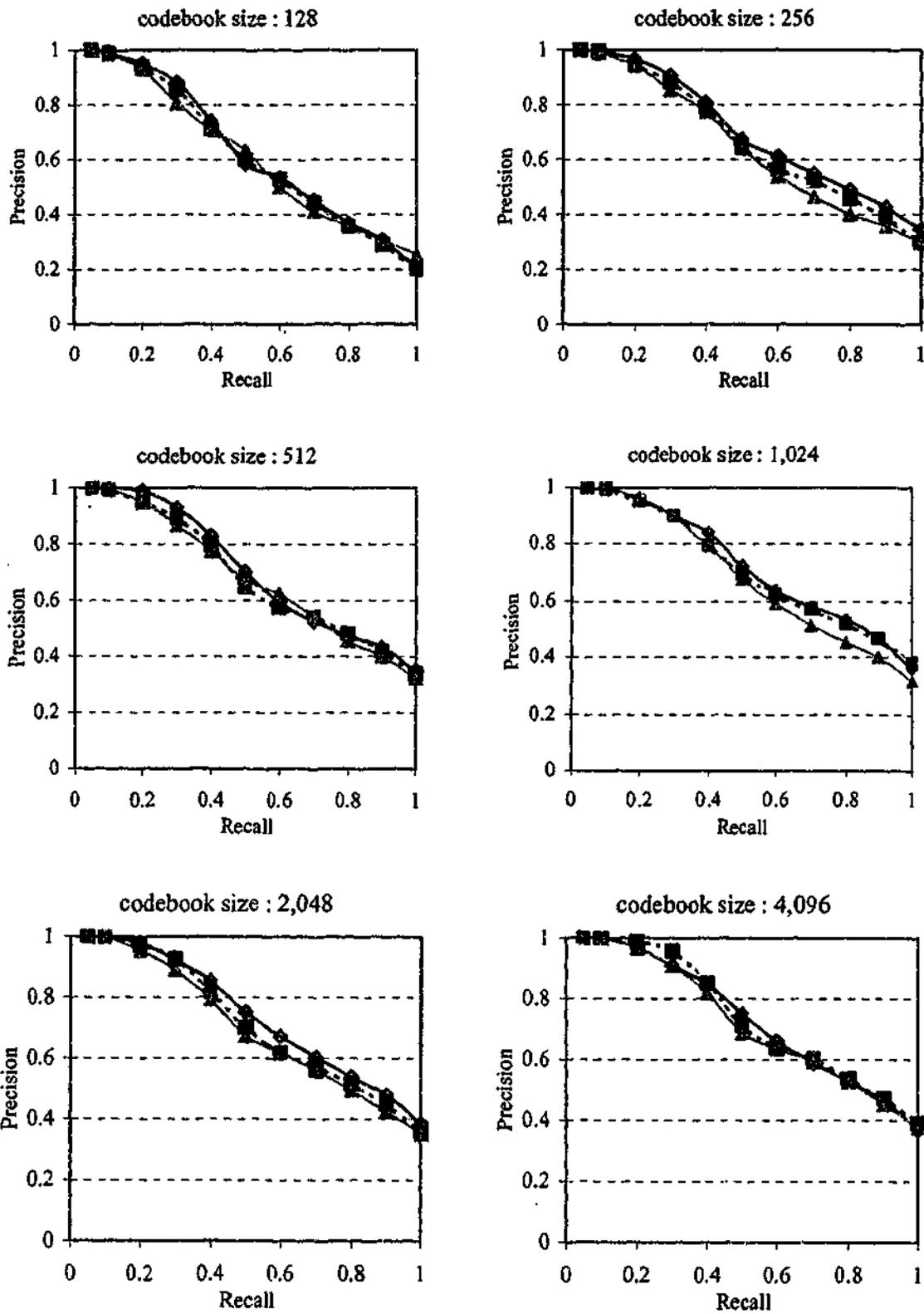
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



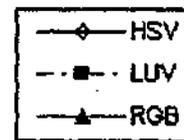
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 4x4 pixels. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



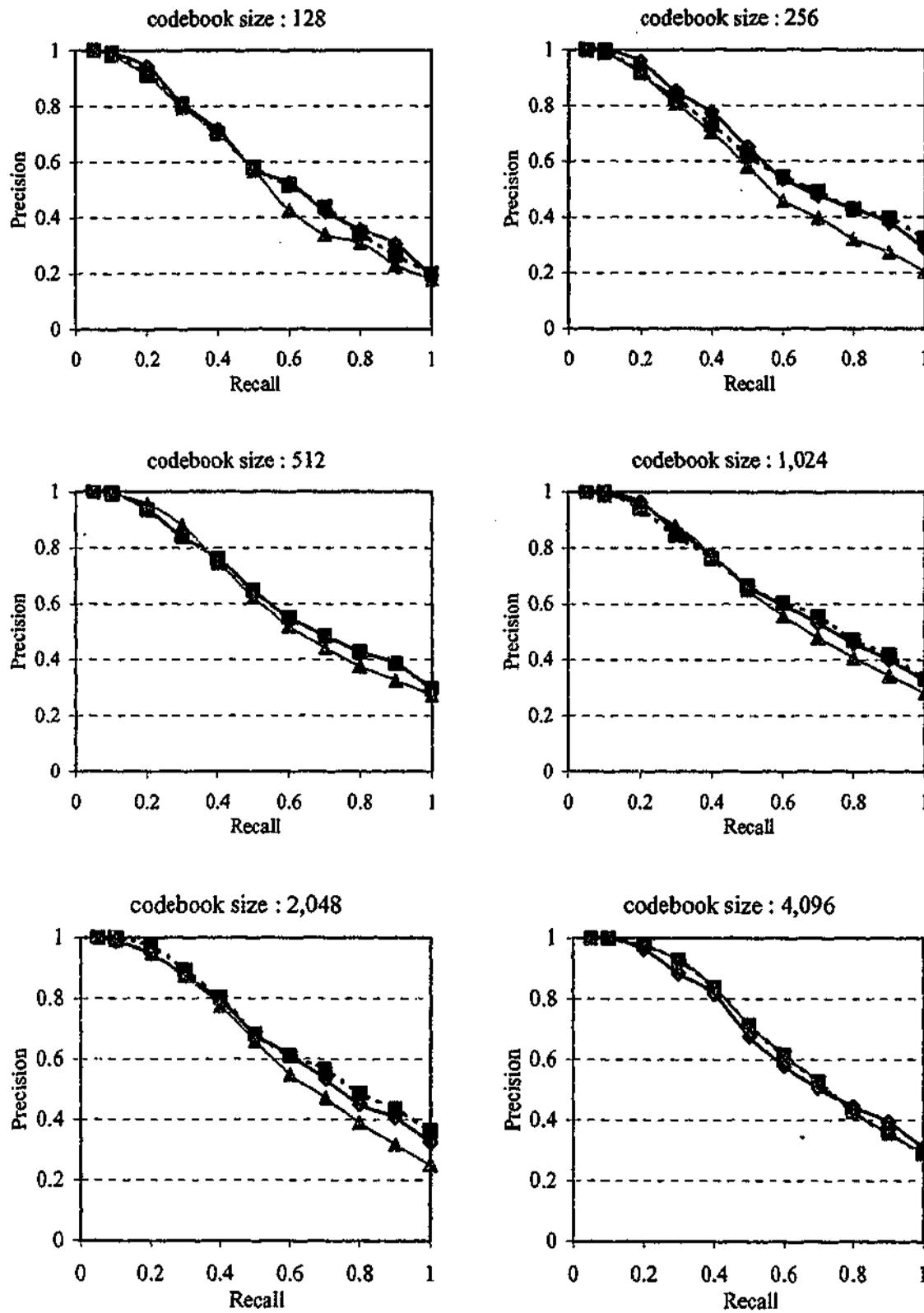
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



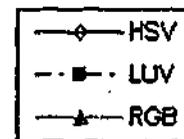
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 8x8 pixels. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



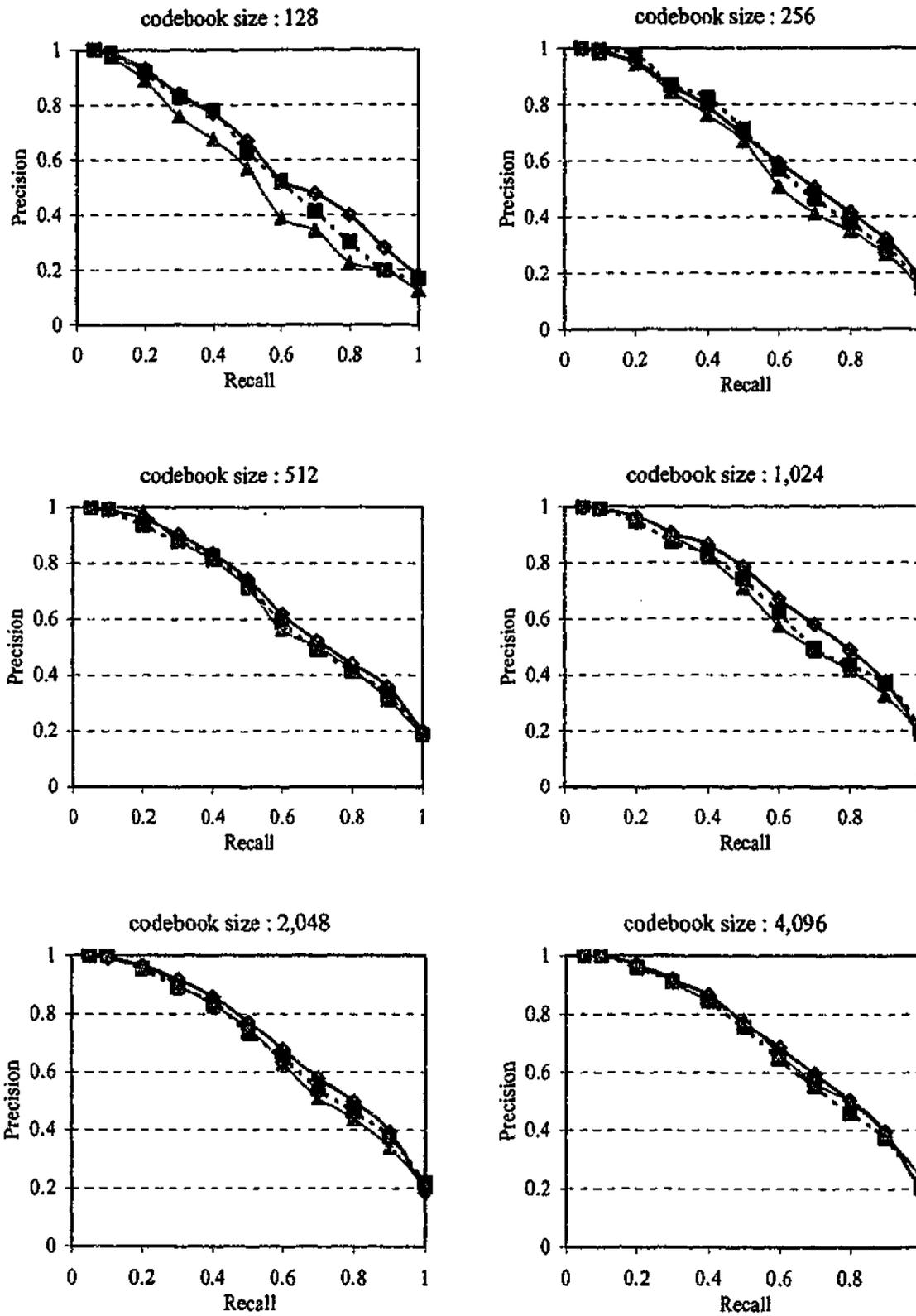
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



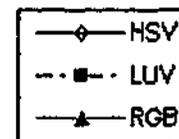
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 16x16 pixels. Experiment conducted on database SCD. Performance evaluated using GT50. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



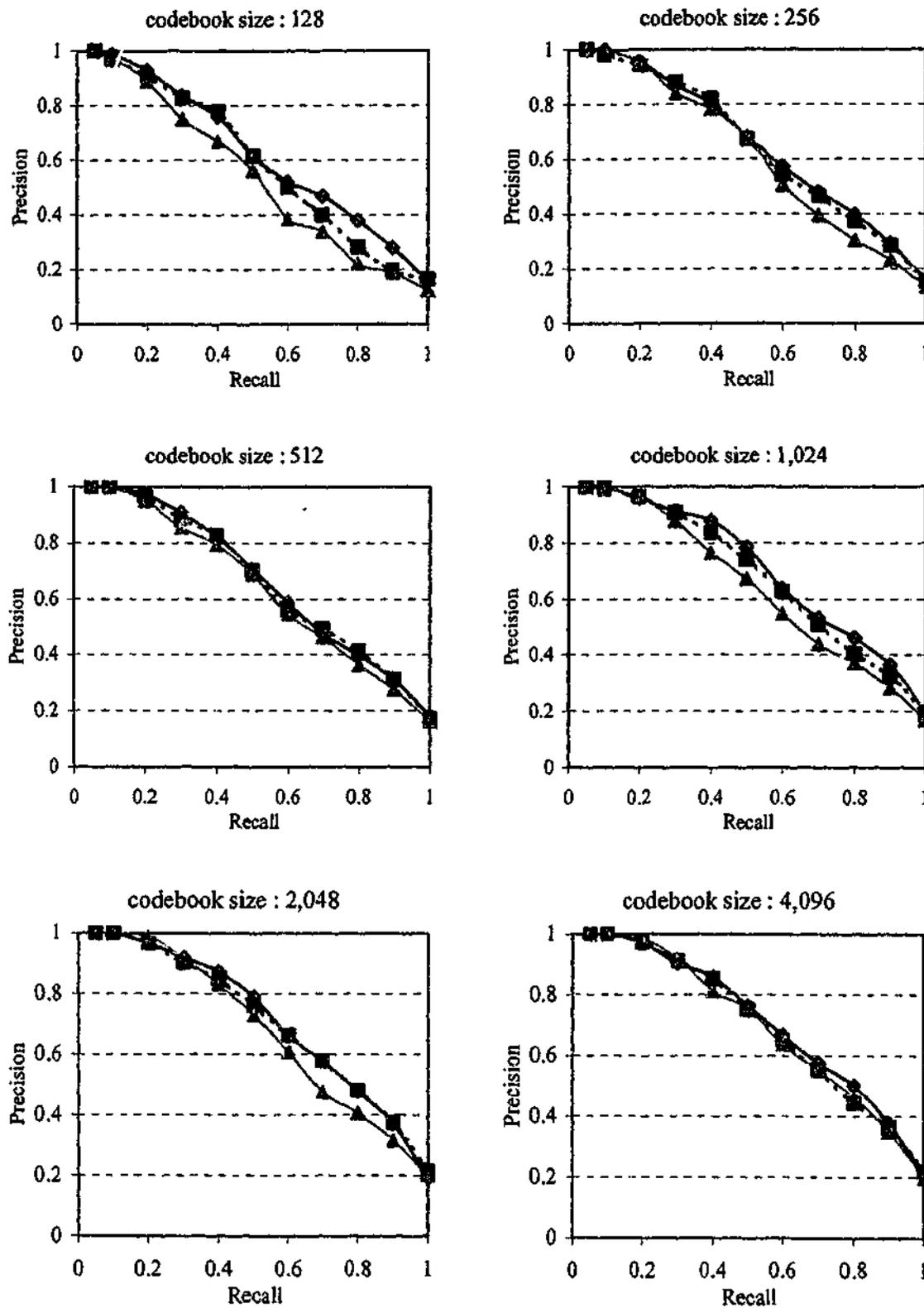
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



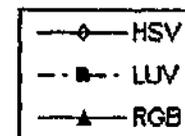
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 2x2 pixels. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



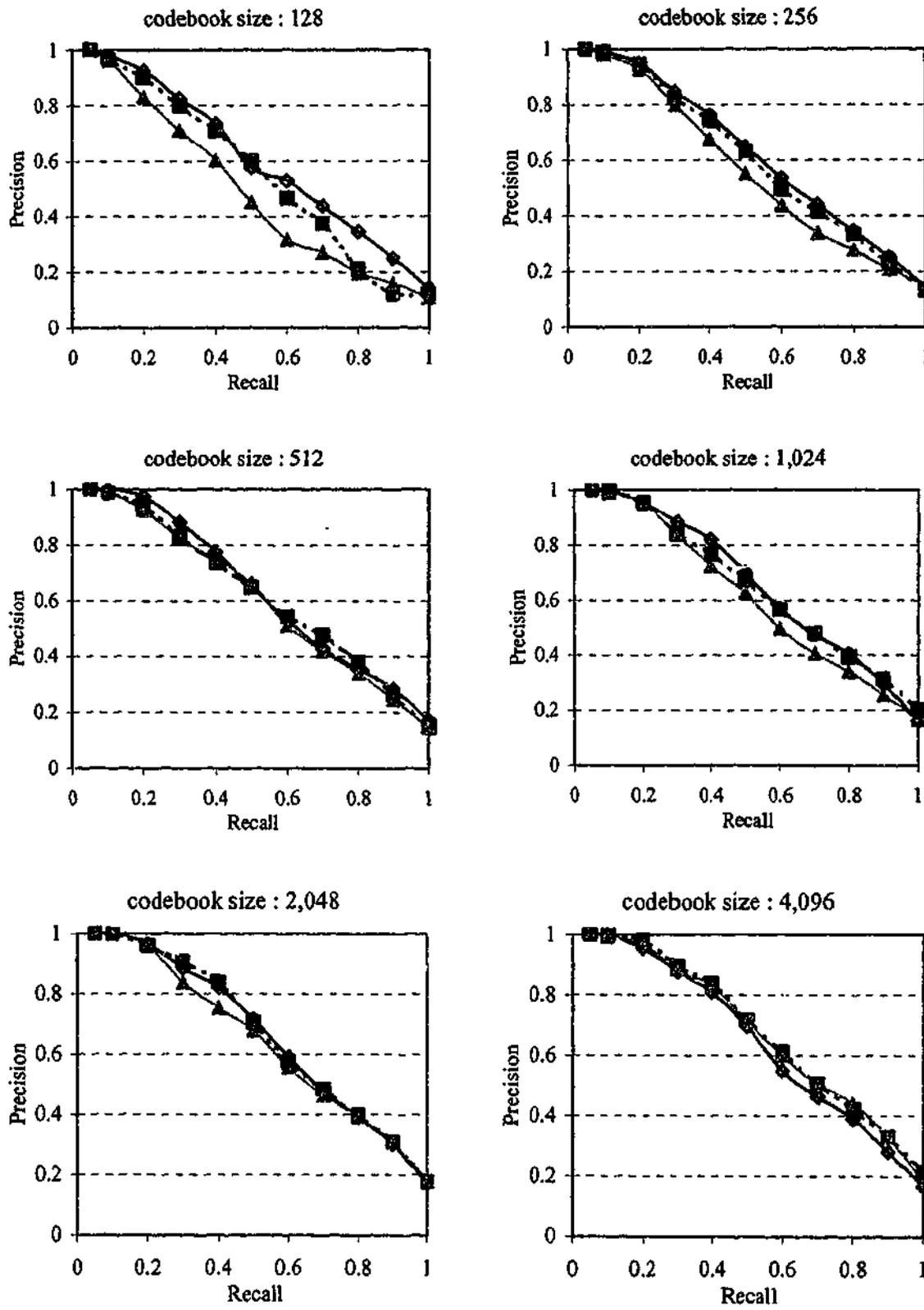
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



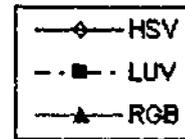
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 4x4 pixels. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



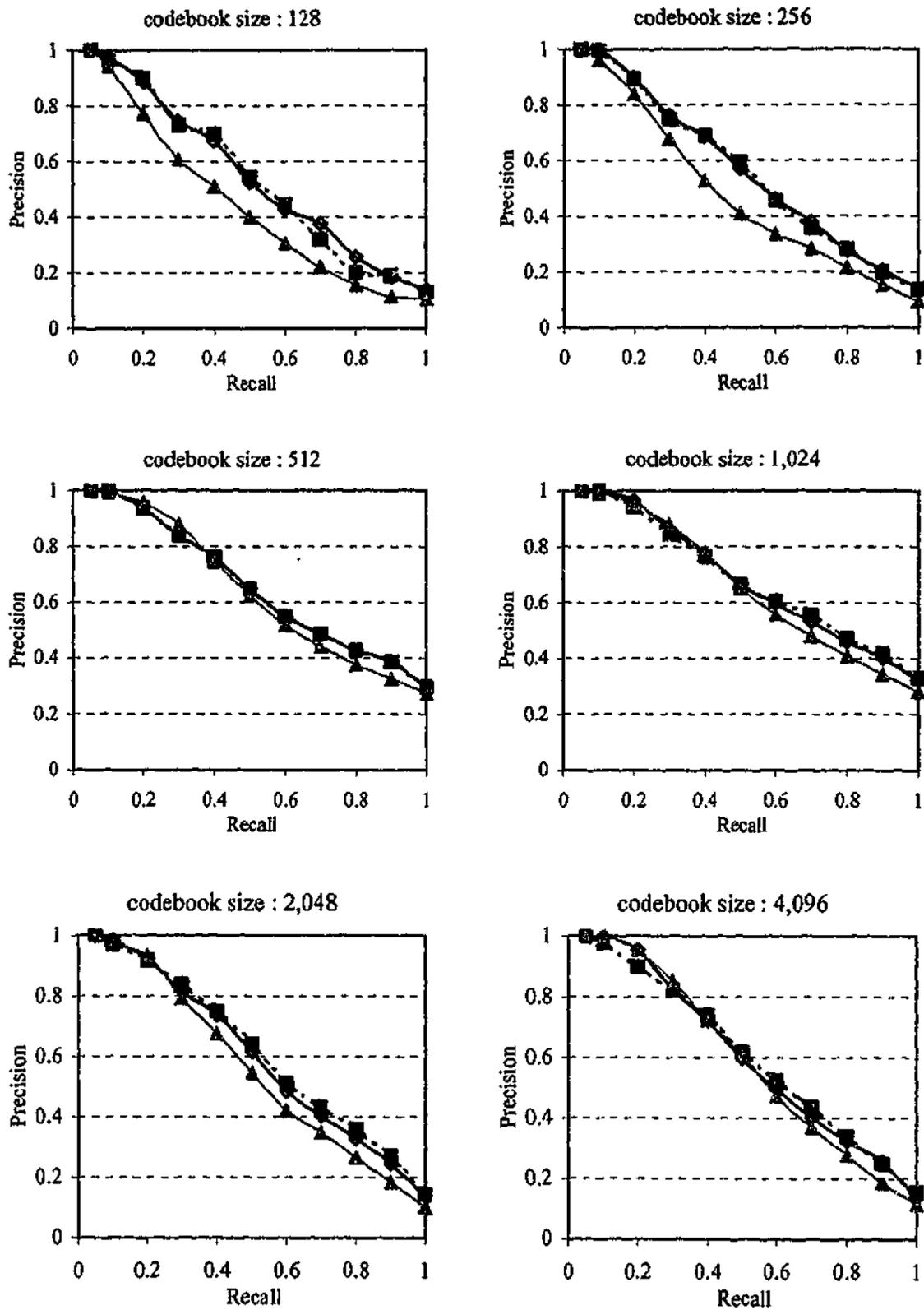
APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



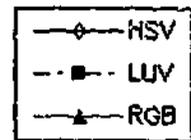
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 8x8 pixels. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



APPENDIX B3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database SCD



Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 16x16 pixels. Experiment conducted on database SCD. Performance evaluated using GT30. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.

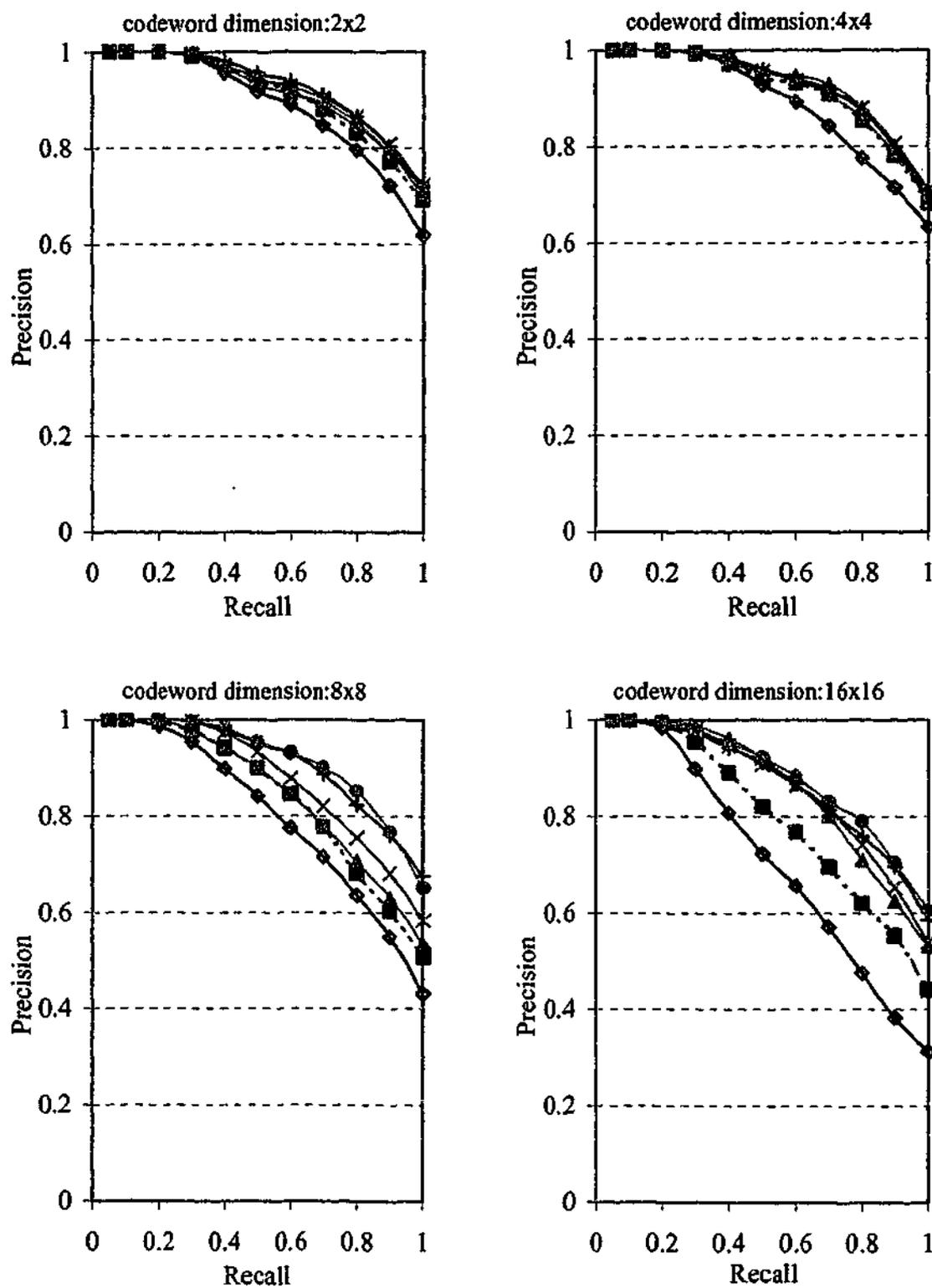


APPENDIX C

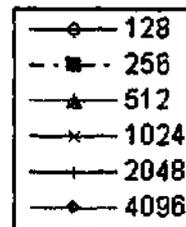
Average recall and precision graphs to show the effects of different parameters of the codebook on retrieval effectiveness. The experimental results in this appendix are obtained by using CCD as the test image database.

Appendix C1 : Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database CCD	C1-1
Appendix C2 : Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database CCD	C2-1
Appendix C3 : Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database CCD	C3-1

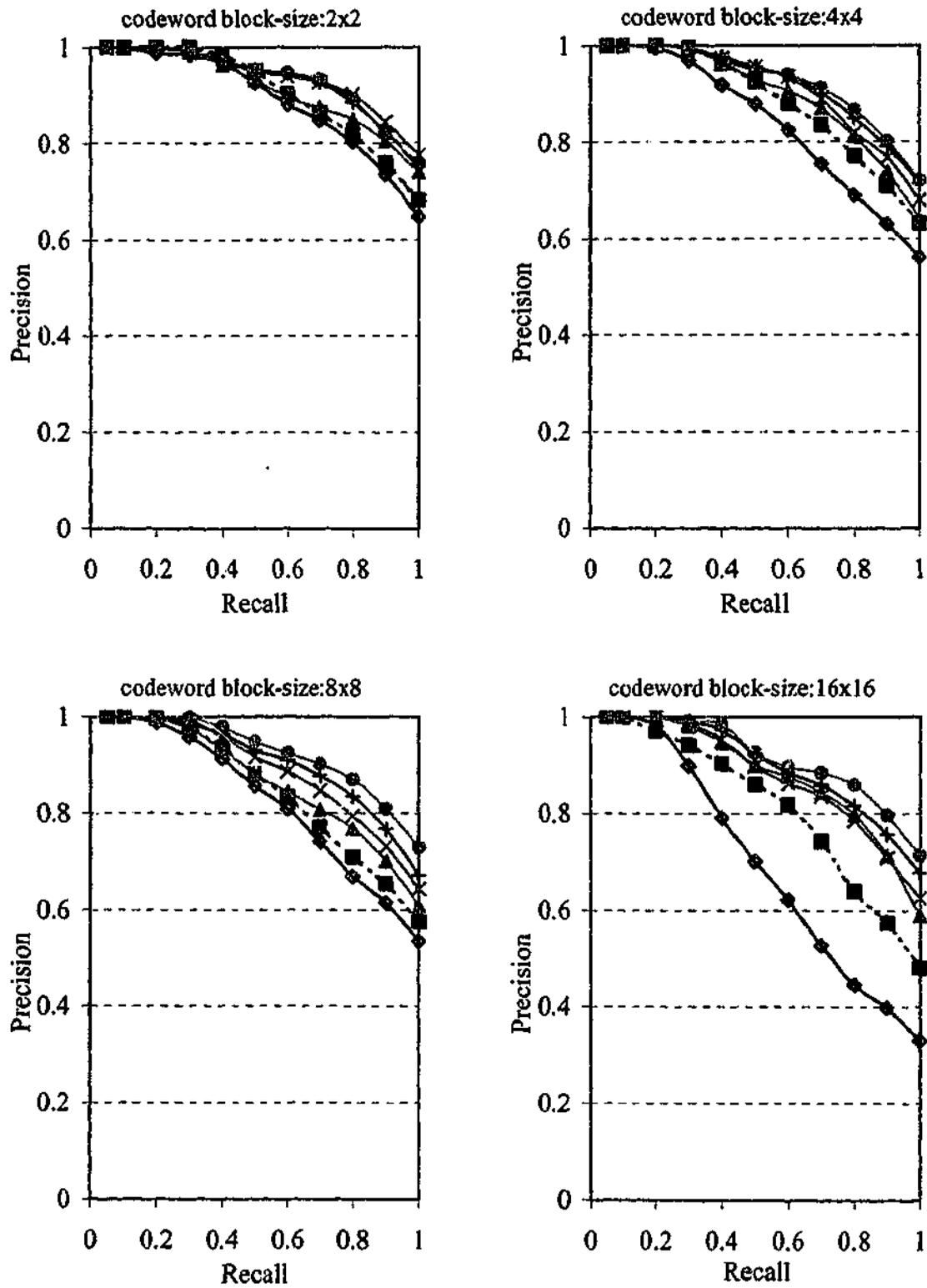
APPENDIX C1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database CCD



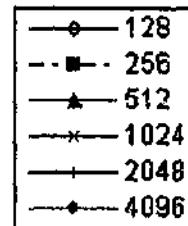
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is HSV. Experiment conducted on database CCD. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



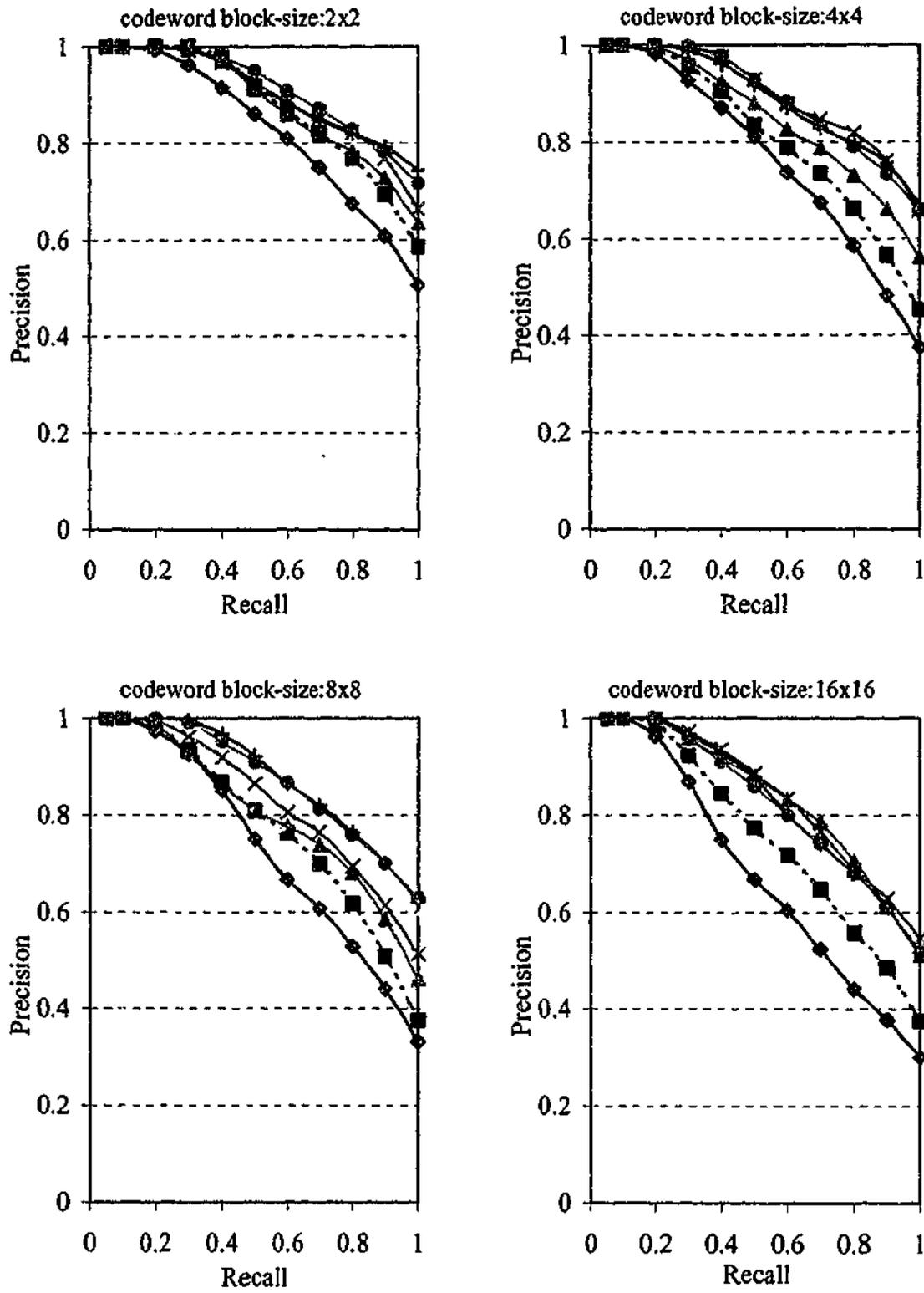
APPENDIX C1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database CCD



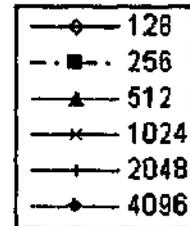
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is LUV. Experiment conducted on database CCD. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



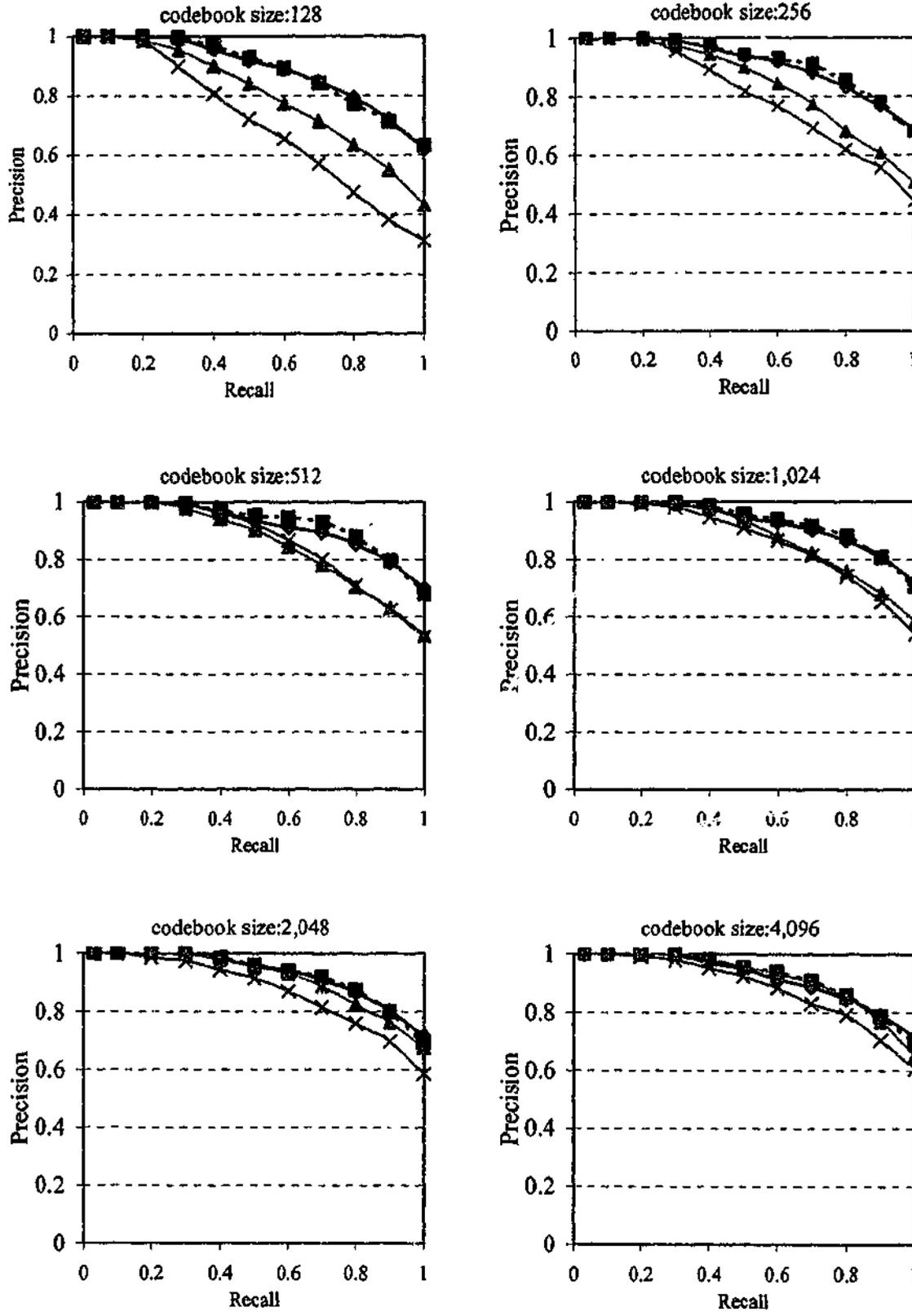
APPENDIX C1: Effects of different codebook sizes on retrieval effectiveness of VQ scheme on Database CCD



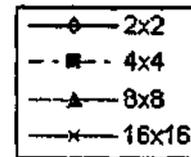
Effects of different codebook sizes on retrieval performance of VQ scheme when colour space is RGB. Experiment conducted on database CCD. For each of the 4 sets of average recall-precision graphs, the codeword block-size is constant (as shown above each set of graphs), while the codebook size varies. Legend of the codebook sizes is shown on the right.



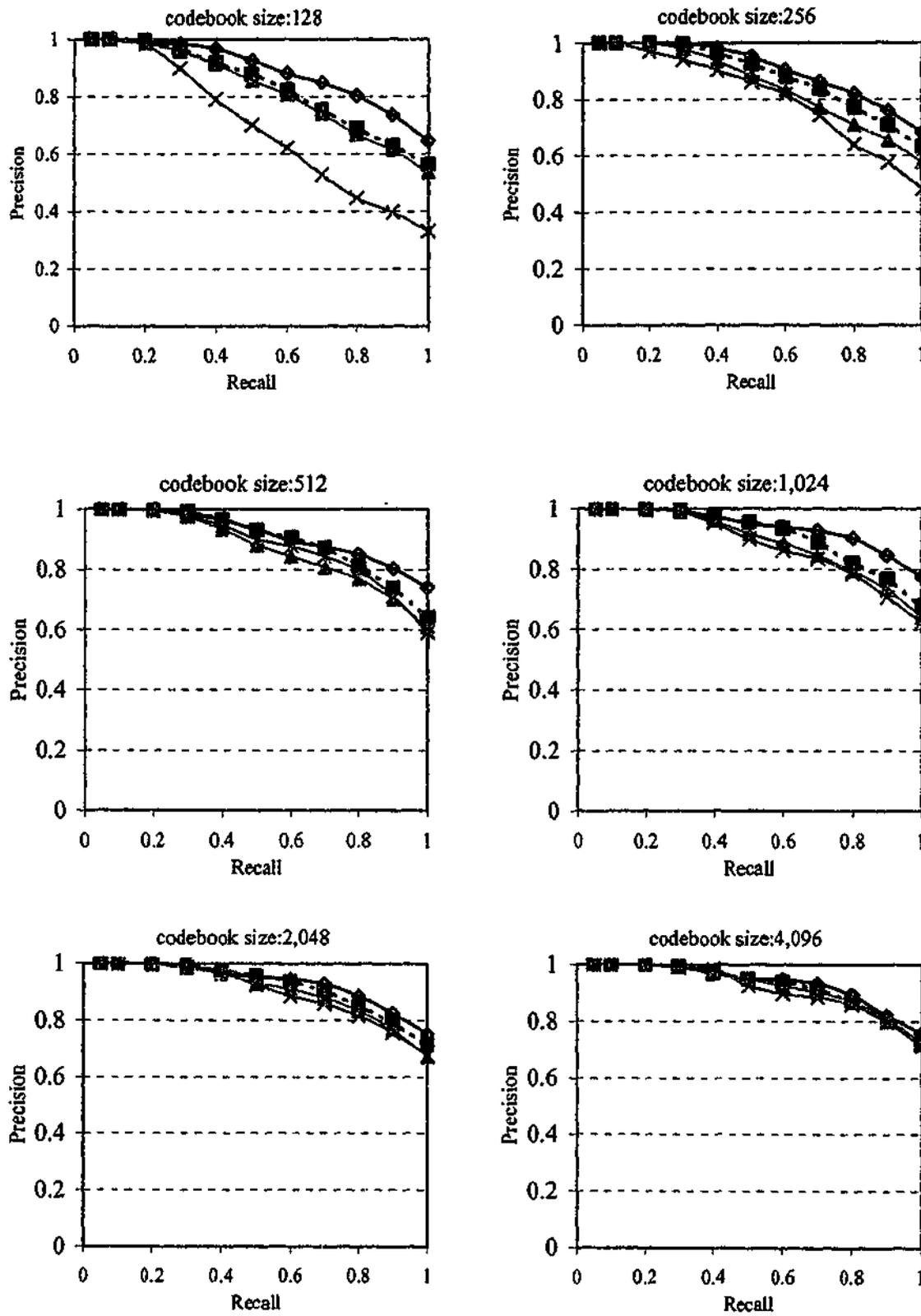
APPENDIX C2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database CCD



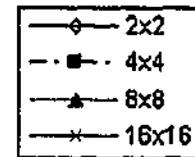
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is HSV. Experiment conducted on database CCD. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



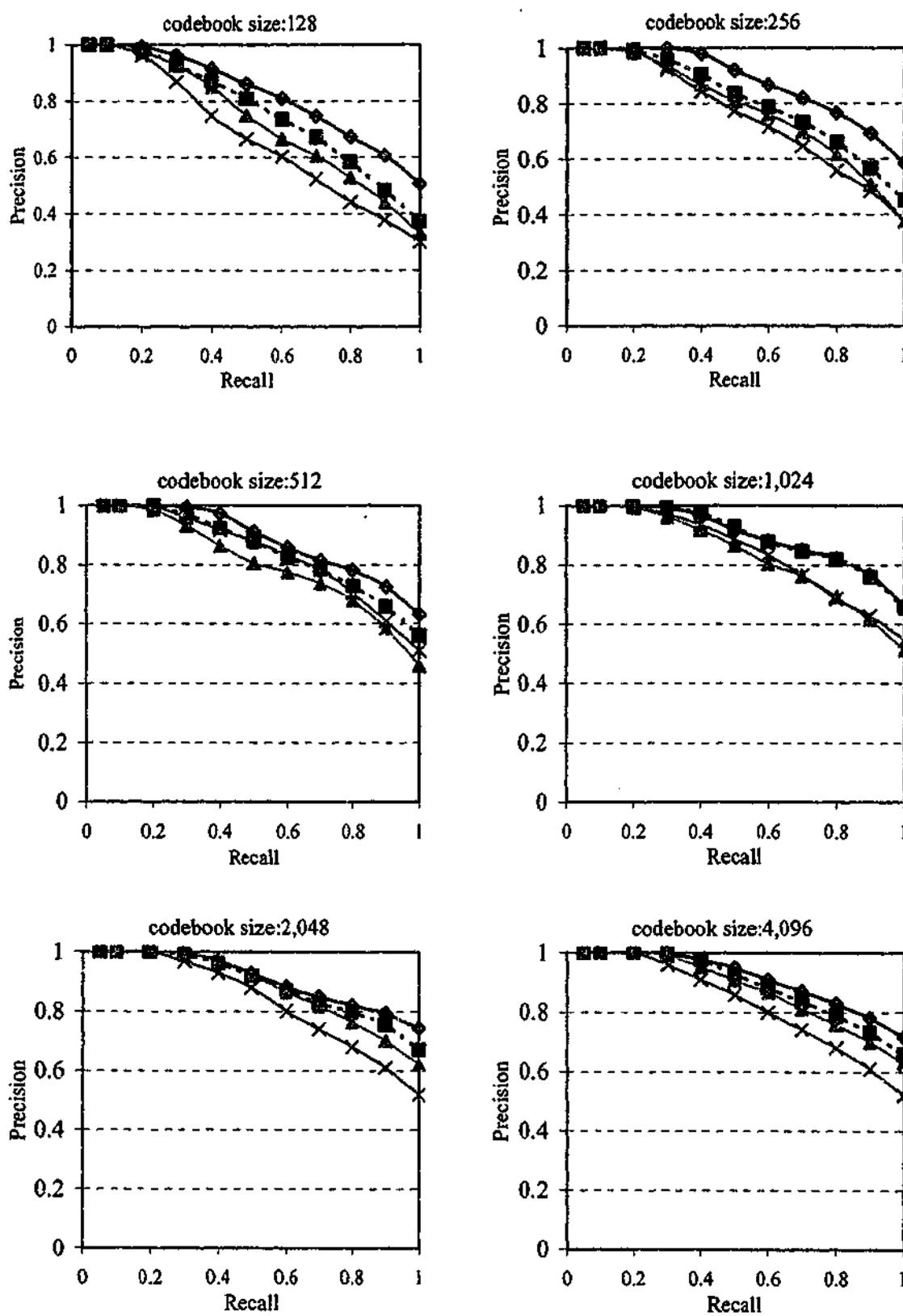
APPENDIX C2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database CCD



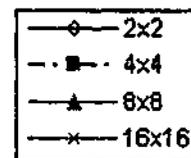
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is LUV. Experiment conducted on database CCD. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



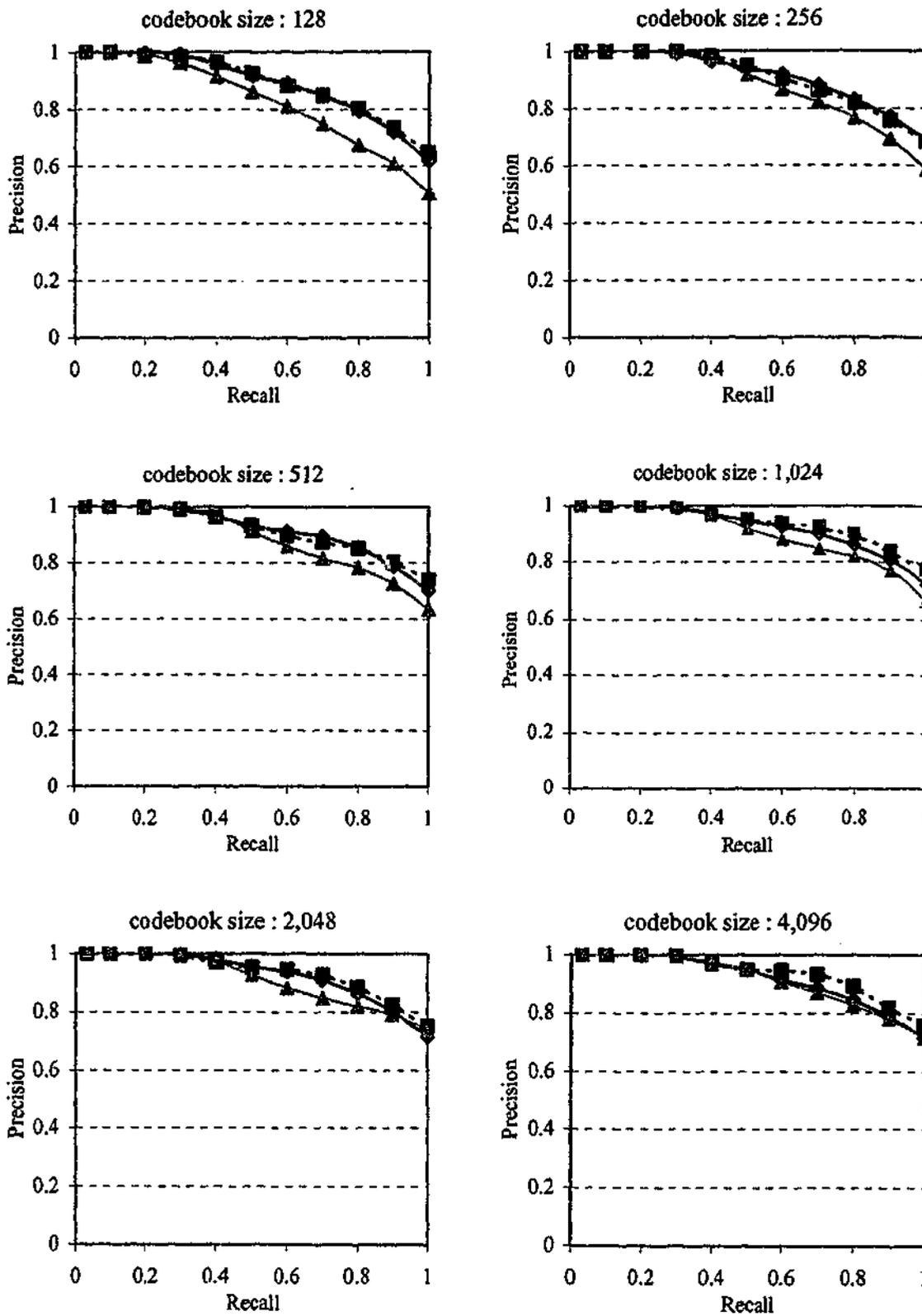
APPENDIX C2: Effects of different codeword block-sizes on retrieval effectiveness of VQ scheme on Database CCD



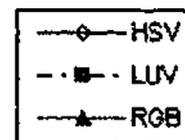
Effects of different codeword block-sizes on retrieval performance of VQ scheme when colour space is RGB. Experiment conducted on database CCD. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the codeword block-size varies. Legend of the codeword block-sizes is shown on the right.



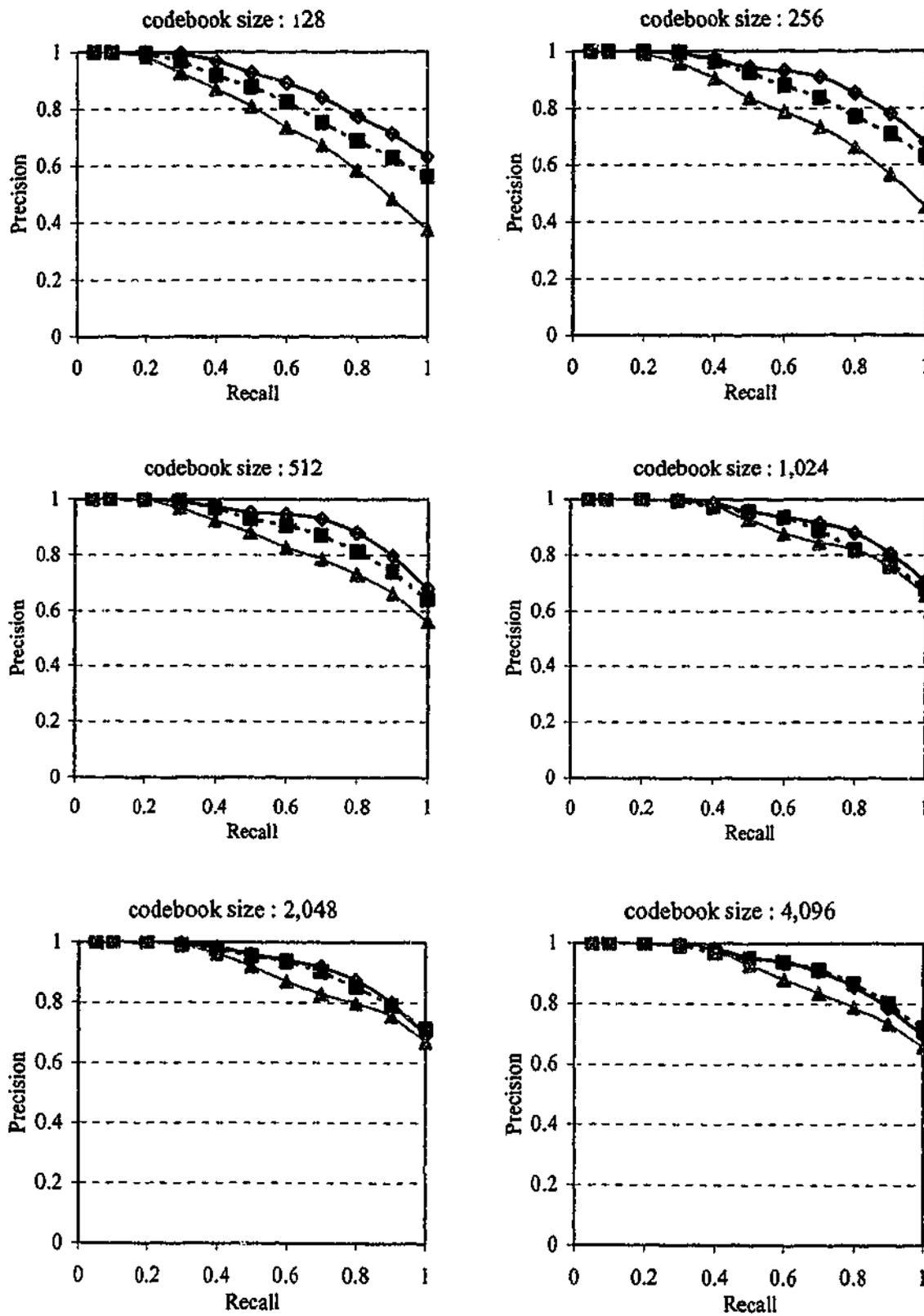
APPENDIX C3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database CCD



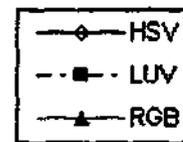
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 2x2 pixels. Experiment conducted on database CCD. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



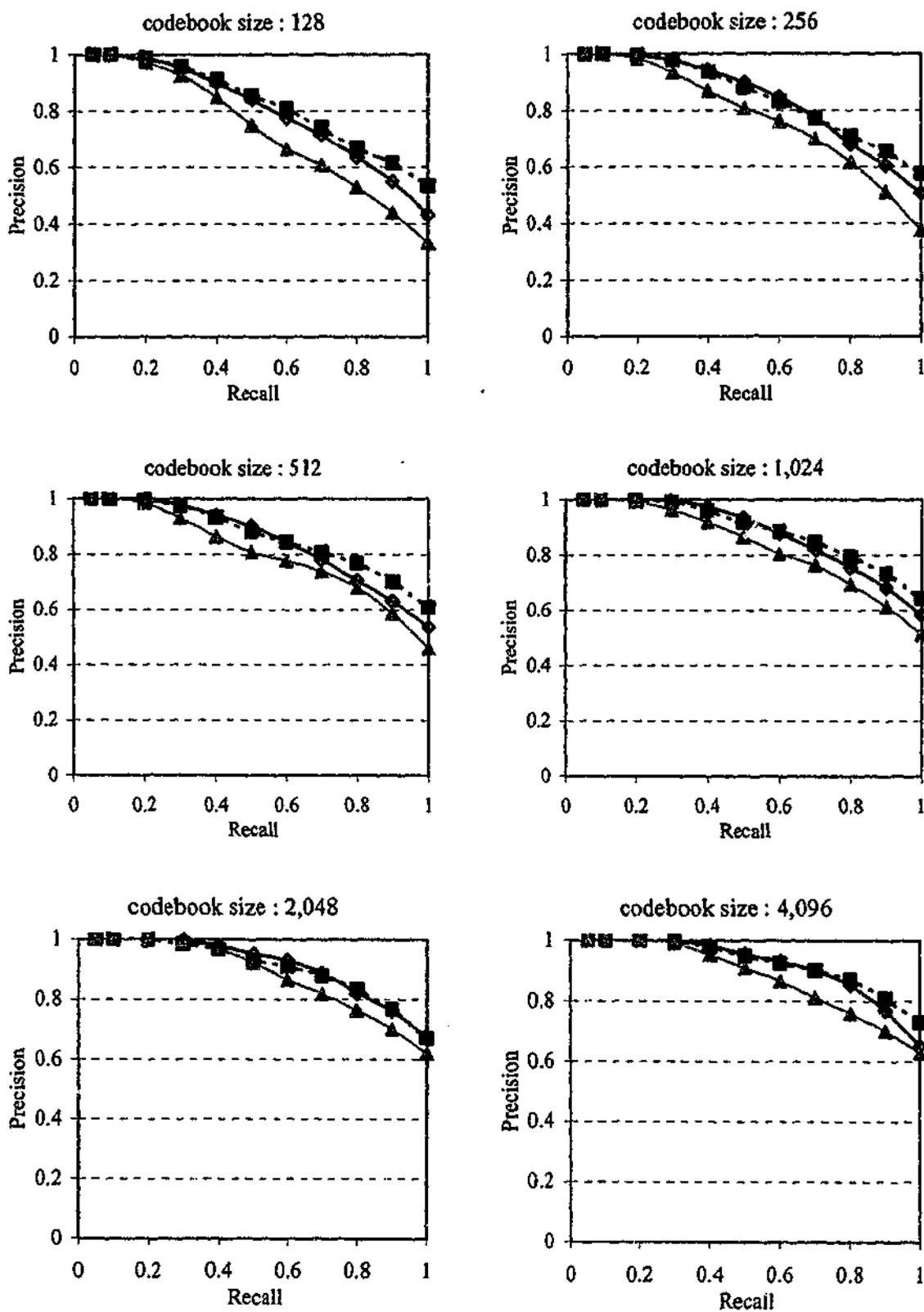
APPENDIX C3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database CCD



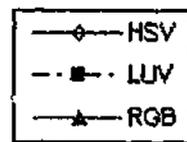
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 4×4 pixels. Experiment conducted on database CCD. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



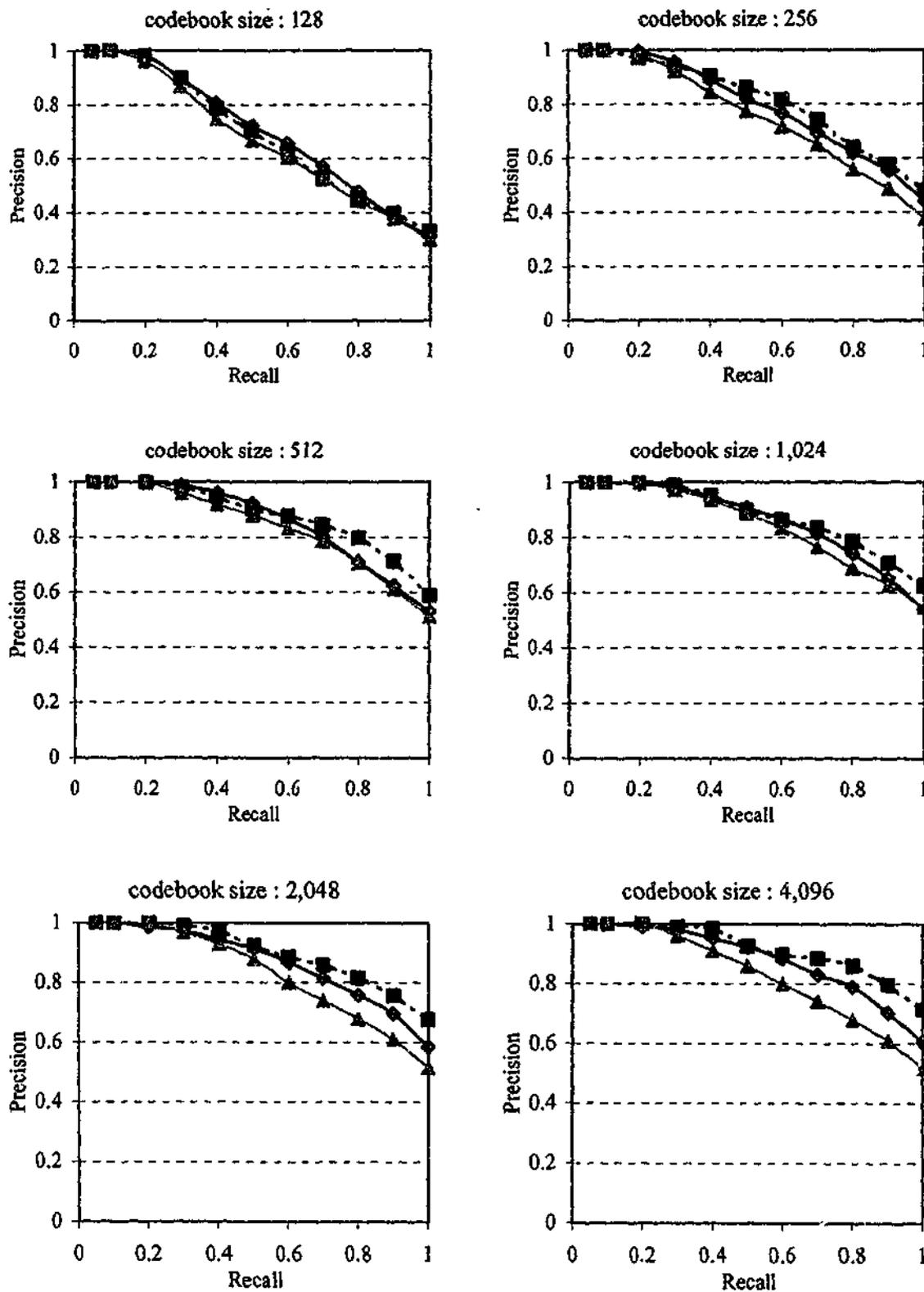
APPENDIX C3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database CCD



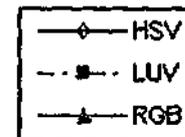
Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 8x8 pixels. Experiment conducted on database CCD. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



APPENDIX C3: Effects of different colour spaces on retrieval effectiveness of VQ scheme on Database CCD



Effects of different colour spaces on retrieval performance of VQ scheme when codeword block-size is 16x16 pixels. Experiment conducted on database CCD. For each of the 6 sets of average recall-precision graphs, the codebook size is constant (as shown above each set of graphs), while the colour space varies. Legend of the colour spaces is shown on the right.



BIBLIOGRAPHY

1. *QBIC Home Page*, <http://www.qbic.almaden.ibm.com/>
2. *VisualSEEK CBIR System*, <http://www.ctr.columbia.edu/webseek>
3. Bach, J.R. *The Virage Image Search Engine: An Open Framework for Image Management*. in *Proceedings of Conference on Storage and Retrieval for Image and Video Databases IV*. 1996. San Jose, California: SPIE.
4. Baek, S., et al., *A fast encode algorithm for vector quantization*. IEEE Signal Processing Letters, 1997. 4(12): p. 325-327.
5. Berger, T., *Rate Distortion Theory: A Mathematical Basis for Data Compression*. 1971, Englewood Cliffs, N.J.: Prentice Hall.
6. Berman, A. and L.G.R.w.M.D.M. Shapiro, *Proceedings of the Conference on Storage and Retrieval for Image and Video Databases V*, San Jose, CA, SPIE Proceedings Series, vol.3022, pp.12-21, Feb 13-14, 1997. *Efficient Retrieval with Multiple Distance Measures*. in *Proceedings of the Conference on Storage and Retrieval for Image and Video Databases V*. 1997. San Jose, CA: SPIE Proceedings Series.
7. Castelli, V. and D. Bergman, *Image Databases, search and retrieval of digital imagery*. 2002, New York, USA: John Willey& Sons, Inc.
8. Chakrabarti, K., et al. *Similarity Shape Retrieval in MARS*. in *Proc. of IEEE Int. Conf. on Multimedia and Expo (CD-ROM)*. 2000. New York, USA.
9. Chang, C. and F. Shiue, *Tree Structured Vector Quantization with Dynamic Path Search*. IEEE International Workshops on Parallel Processing, 1999: p. 536 -541.
10. Chaudhuri, B.B. and N. Sarkar, *Texture Segmentation using Fractal Dimension*. IEEE Trans. On Pattern Analysis and Machine Intelligence, 1995. 17(1): p. 72-77.
11. Cheng, D.Y., et al. *Fast search algorithms for vector quantization and pattern matching*. in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*. 1984.
12. Choi, S. and S. Chae, *Extended Mean-Distance-Ordered Search using Multiple l1 and l2 Inequalities for Fast Vector Quantization*. IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, 2000. 47(4): p. 349-352.
13. Chua, J.J., *Fast Full-Search Equivalent Nearest-Neighbour Search Algorithms*, in *School of Computer Science and Software Engineering*. 1999, Monash University: Melbourne.

-
14. Chua, S., et al. *A Concept-based Image Retrieval System*. in *International Conference on System Science*. 1994. Maui, Hawaii.
 15. Chung, K., W. Yan, and J. Wu, *A Simple Improved Full Search for Vector Quantization Based on Winograd's Identity*. *IEEE Signals Processing Letters*, 2000. 2(12): p. 342-344.
 16. Dadd, M. and A. Jon. *A Multi-Invariant Difference Measure for Grey-Scale Texture*. in *Proceedings of IEEE TENCON*. 1997.
 17. Davis, E.R., *Machines Vision Theory, Algorithms, Practicalities*. 1997: Academic Press.
 18. Deng, Y., *A Region Representation for Image and Video Retrieval*. 1999, University of California: Santa Barbara.
 19. Duda, R. and P. Hart, *Pattern Classification and Scene Analysis*. 1973, New York, N.Y: Wiley.
 20. Fairchild, M.D., *Color Appearance Models*. 1998, Menlo Park, California: Addison Wesley Longman, Inc.
 21. Faloutsos, C., *Efficient and effective Querying by image content*. *Journal of Intelligent Information Systems*, 1994. 3: p. 231-262.
 22. Faloutsos, C., *Searching Multimedia Databases by Content*. 1996: Kluwer Academic Publishers.
 23. Frakes, W.B. and R.e. Baeza-Tates, *Information Retrieval: Data Structures and Algorithms*. 1992: Prentice Hall.
 24. Friedman, J. and J.a.f.f.b.m.i.L.e.t. Bentley, *ACM Trans. Math. Software*, vol. 3, no. 3, pp. 209-226, Sept. 1977, *An algorithm for finding best matches in Logarithmic expected time*. *ACM Trans. Math. Software*, 1977. 3(3): p. 209-226.
 25. Furht, B.e., *Handbook of Multimedia Computing*. 1999, New York, USA: CRD Press.
 26. Gersho, A. and R.M. Gray, *Vector Quantization and Signal Compression*. 1992, London: Kluwer Academic Publishers.
 27. Gong, e.a. *An Image Database System with Fast Image Indexing Capability based on Colour Histograms*. in *IEEE Region 10's Ninth Annual International Conference*. 1994. Singapore.
 28. Gray, L., *Information retrieval software; recent developments in Cardbox family*. *Aslib Information*, 1992. 20(4): p. 164.
 29. Gray, R.M., *Vector Quantization*, in *IEEE ASSP Magazine*. 1994. p. 4-29.
 30. He, Y., *An Investigation into Efficient Indexing Structure for Multimedia Information Retrieval*, in *Gippsland School of Computing and I.T. (GSCIT)*. 2002, Monash University.
 31. He, Y., G. Lu, and S. Teng. *An Investigate of using K-D Tree to Improve Image Retrieval Efficiency*. in *Digital Image Computing-Techniques and Applications*. 2002. Melbourne, Australia.
 32. Hsieh, C. and Y. Liu, *Fast Search Algorithms for Vector Quantization of Image using Multiple Triangle Inequalities and Wavelet Transform*. *IEEE Transactions Image Processing*, 2000. 9(3): p. 321-328.

-
33. Hu, M.K., *Visual Pattern Recognition by Moment Invariants*. IRE Trans. Info. Theory, 1962. IT-8: p. 179-187.
 34. Huang, J., et al. *Image indexing using Color Correlograms*. in *IEEE Int. Conf. On Computer Vision and Pattern Recognition*. 1997. Puerto Rico.
 35. Idris, F. and S. Panchanathan. *Algorithm for Indexing of Compressed Images*. in *Proceedings of International Conference on Visual Information Systems, Melbourne*. Feb. 1996. p.303-308.
 36. Ioka, M., *A method of defining the similarity of images on the basis of color information*. 1989, IBM Tokyo Research Lab.
 37. Kauppinen, H., T. Seppanen, and M. Pietikainen, *An Experimental Comparison of Autogression and Fourier-based Descriptors in 2D Shape Classification*. IEEE Transaction on PAMI, 1995. 17(2): p. 201-207.
 38. Ramasubramanian, V. and K.K.-D.T.A.f.N.N.S.w.A.t.V.Q.E. Paliwal, IEEE Transactions Image Processing, vol. 40, No. 3, pp. 518-531, March 1992, *Fast K-Dimensional Tree Algorithms for Nearest Neighbour Search with Application to Vector Quantization Encoding*. IEEE Transactions Image Processing, 1992. 40(3): p. 518-531.
 39. Lecce, V.D. and A. Guerriero, *An evaluation of the effectiveness of image features for image retrieval*. Journal of Visual Communication and Image Representation, 1999(10): p. 351-362.
 40. Li, W. and E. Salari, *A Fast Vector Quantization Encoding Method for Image Compression*. IEEE Transactions on Circuits and Systems for Video and Technology, 1995. 5(2): p. 119-123.
 41. Linde, Y., A. Buzo, and R.M. Gray, *An Algorithm for Vector Design*. IEEE Transactions on Communications, 1980. COM-28: p. 84-95.
 42. Lu, G., *Communication and Computing For Distributed Multimedia Systems*. 1996, Boston, London: Artech House Publishers.
 43. Lu, G. *Image Retrieval based on Colour*. in *Conference on Storage and Retrieval for Image and Video Databases IV*. 1996. San Jose, California: SPIE Proceedings Series.
 44. Lu, G., *Multimedia Database Management System*. 1999, Boston, London: Artech House Publishers.
 45. Lu, G. and A. Sajjanhar, *Region-based Shape Representation and Similarity Measure Suitable for Content-based Image Retrieval*. Multimedia System, 1999. 7(2): p. 165-174.
 46. Lu, G. and S. Teng. *A Novel Image Retrieval Technique based on Vector Quantization*. in *Computational S. Intelligence for Modelling Control and Automation*. 1999. Viana, Austria.
 47. Lu, G. and S. Teng. *Image Compression, Indexing and Retrieval Based on Vector Quantization Are Promising*. in *The First IEEE Pacific-Rim Conference on Multimedia*. 2000. University of Sydney, Australia.
 48. Ma, W.Y. and H.J. Zhang, *Content-based Image Indexing and Retrieval*. CRC Press LLC, 1999.

-
49. Manjunath, B.S., P. Salembier, and T.I.t.M.-m.c.d.i. Sikora, Wiley, Chichester ; Milton (Qld.), 2002., *Introduction to MPEG-7 : multimedia content description interface*. 2002, Chichester ; Milton (Qld.): Wiley.
 50. McNames, J., *Rotated partial distance search for faster vector quantization encoding*. IEEE Signal Processing Letters, 2000. 7(9): p. 244-246.
 51. Miyahara, M. and Y. Yoshida, *Mathematical transform of (R, G, B) color data to Munsell (H, V, C) color Data*. Visual Communication and Image Processing, 1988. 1001: p. 650-657.
 52. Mohamad, D.G.S., S.S. Ipson, and P.o.S.A.C.o.C.V.S. Trademark Matching using Invariant Moments", C, 1995, pp. 1-439-444. *Trademark Matching using Invariant Moments*. in *Proceedings of Second Asian Conference on Computer Vision Singapore*. 1995. Singapore.
 53. Mokhtarian, F., *The Curvature Scale Space Representation: Theory, Applications and MPEG-7 Standardization*. 2002: Kluwer Academic Publishers.
 54. Mokhtarian, F. and A.K. Mackworth, *A Theory of Multiscale, Curvature-based Shape Representation for Planar Curves*. IEEE Trans. On Pattern Analysis and Machine Intelligence, 1992. 14(8): p. 789-805.
 55. Muller, H., et al., *Efficient access methods for content-based image retrieval with inverted files*. 1999, Computer Vision Group, Computing Centre, University of Geneva: Geneva, Switzerland.
 56. Netravali, A.N. and B.G. Haskell, *Digital Pictures: Representation, Compression, and Standards*. 2 ed. 1995, New York, NY: Plenum Press.
 57. Ng, R.T. and D. Tam. *Analysis of Multilevel color histograms*. in *Proc. Of Conf on Storage and Retrieval for Image and Video Databases V*. 1997. San Jose, CA: SPIE Proceedings Series.
 58. Niblack, W. and et al. *QBIC Project: Querying Images by Content, Using Colour, Texture, and Shape*. in *Proceedings of the Conference Storage and Retrieval for image and Video Databases*. 1993. San Jose, California, USA: SPIE Proceedings Series.
 59. Nielsen, J., *Usability Engineering*. 1993, Boston, MA: Academic Press.
 60. Ohm, J.R., et al., *A Set of Visual Feature Descriptors and Their Combination in a Low-level Description Scheme*. Signal Processing: Image Communication, 2000. 16: p. 157-179.
 61. Ortega, M., et al. *Supporting Similarity Queries in MARS*. in *ACM Multimedia'97*. 1997. Seattle, WA.
 62. Otterloo, P.J.V., *A Contour-Oriented Approach to Shape Analysis*. Signal Processing: Image Communication, 2000. 16: p. 157-179.
 63. Pass, G. and R. Zabih. *Histogram refinement for content-based image retrieval*. in *IEEE Workshop on Application of Computer Vision*. 1996.
 64. Pentland, A., R. Picard, and S. Sclaroff. *Photobook: Tools for Content-based Manipulation of Image Databases*. in *Proceedings of the SPIE Conference on Storage and Retrieval of Image and Databases II*. 1994.

-
65. Persoon, E. and K.S. Fu, *Shape Discrimination using Fourier Descriptors*. IEEE Transactions on Systems, Man and Cybernetics, 1977. 7(3): p. 170-179.
 66. Plataniotis, K.N. and A.N. Venetsanopoulos, *Colour Image Processing and Applications*. 2000, Berlin: Springer.
 67. Ra, S.W. and J.K. Kim, *A fast mean distance-order partial codebook algorithm for image vector quantization*. IEEE Trans. Circuits Sys. II, 1993. 40: p. 576-579.
 68. Rabbani, M. and P.W. Jones, *Digital Image Compression Techniques*. Vol. TT7. 1991, Bellvue, Washington: SPIE Optical Engineering Press.
 69. Rahman, S.M.e., *Design and Management of Multimedia Information Systems: Opportunities and Challenges*. 2001, London, UK: Idea Group Publishing.
 70. Safar, M., C. Shahabi, and X. Sun. *Image Retrieval by Shape: A Comparative Study*. in *Proc. of IEEE Int. Conf. on Multimedia and Expo (CD-ROM)*. 2000. New York, USA.
 71. Sajjanhar, A. and L. G., *A Grid based Shape Indexing and Retrieval Method*. Special Issue of Australian Computer Journal on Multimedia Storage and Archiving Systems, 1997. 29(4): p. 131-140.
 72. Sajjanhar, A. and G. Lu. *Indexing 2D Non-Occluded Shape for Similarity Retrieval*. in *SPIE Conference on Applications of Digital Image Processing XX*. 1997. San Diego, USA.
 73. Salton, G. and M.J. McGill, *Introduction to Modern Information Retrieval*. 1983: McGraw-Hill Book Company.
 74. Sangwine, S.J. and R.E.N.T.c.i.p.h. Horne, Chapman & Hall, London, 1998, *The colour image processing handbook*. 1998, London: Chapman & Hall.
 75. Sayood, K., *Introduction to Data Compression*. 2000, San Francisco, California: Morgan Kaufmann Publishers, Inc.
 76. Schauble, P., *Content-based Information Retrieval from Large Text and Audio Databases*. 1997, Boston: Kluwer Academic Publishers.
 77. Scheaffer, R.L., *Introduction to Probability, and its applications*. 1990, Belmont California: Duxbury Press.
 78. Shen, G. and M.E.C.P.-P.T.a.W.-B.F.-S.A.f.I.V.Q. Liou, IEEE Transactions on Circuits and Systems for Video Technology, vol. 10, No. 6, pp. 990-997, Sep 2000, *An Efficient Codebook Post-Processing Technique and Window-Based Fast-Search Algorithm for Image Vector Quantization*. IEEE Transactions on Circuits and Systems for Video Technology, 2000. 10(6): p. 990-997.
 79. Smith, J.R., *Integrated and Feature Image System: Retrieval, Analysis and Compression*. 1997, Columbia University.
 80. Smith, J.R. and S. Chang. *Tools and Techniques for Color Image Retrieval*. in *Proceedings of Conference on Storage and Retrieval for Image and Video Databases IV*. 1996. San Jose, California: SPIE.

-
81. Sonka, M., V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. 2nd Edition ed. 1999, New York, USA: PWS Publishing.
 82. Squire, D.M., H. Muller, and W. Muller, *Improving Response Time by Search Pruning in a Content-based Image Retrieval System using Inverted File Techniques*. 1999, Computer Vision Group, Computing Centre, University of Geneva: Geneva, Switzerland.
 83. Squire, D.M., H. Muller, and W. Muller. *Improving Response Time by Search Pruning in a Content-based Image Retrieval System, using Inverted File Techniques*. in *IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'99)*. 1999. Fort Collins, Colorado, USA.
 84. Squire, D.M., et al., *Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback*. 1998, Computer Vision Group, Computer Centre, University of Geneva: Geneva, Switzerland.
 85. Srihari, R., *Automatic indexing and content-based retrieval of captioned images*. *IEEE Computer*, 1995. 28(9): p. 49-56.
 86. Stricker, M. and A. Dimai. *Color indexing with weak spatial constraints*. in *Proc. Of the Conf on Storage and Retrieval for Image and Video Databases IV*. 1996. San Jose, CA: SPIE Proceedings Series.
 87. Swain, M.J. and D.H. Ballard, *Color indexing*. *International Journal of Computer Vision*, 1991. 7(1): p. 11-32.
 88. Swain, M.J., et al. *View-based Techniques for Searching for Objects and Texture*. in *Proceedings of Second Asian Conference on Computer Vision*. 1995. Singapore.
 89. Tamura, H., S. Mori, and T. Yamawaki, *Texture Features Corresponding to Visual Perception*. *IEEE Transactions on Systems, Man and Cybernetics*, 1995. 8(1): p. 72-77.
 90. Taycher, L., et al. *Image Digestion and Relevance Feedback in the ImageRover WWW Search Engines*. in *Proceedings of the Second International Conference on Visual Information System*. 1997. San Diego.
 91. Teng, S. and G. Lu. *Codebook generation in vector quantization used for image retrieval*. in *International Symposium on Intelligent Multimedia and Distance Education*. 1999. Baden-Baden, Germany.
 92. Teng, S. and G. Lu. *Performance study of image retrieval based on vector quantization*. in *ICCIMADE'01: International Conference on Intelligent Multimedia and Distance Education Conference*. 2001. Fargo, ND, USA.
 93. Teng, S. and G. Lu. *An evaluation of the robustness of image retrieval based on vector quantization*. in *IEEE Pacific-Rim Conference on Multimedia*. 2001. Beijing, China.
 94. Teng, S. and G. Lu. *Effects of Codebook Sizes, Codeword Dimensions, and Colour Spaces on Retrieval Performance of Image Retrieval using Vector Quantization*. in *The 3rd IEEE Pacific-Rim Conference on Multimedia*. 2002. Hsinchu Taiwan: Springer.

-
95. Tieng, Q.M. and W.W. Boles, *Recognition of 2D Object Contours using Wavelet Transform Zero-Crossing Representation*. IEEE Trans. PAMI, 1997. 19(8): p. 910-916.
 96. Torres, L. and J. Huguet, *An Improvement on Codebook Search for Vector Quantization*. IEEE Transaction on Communications, 1994. 42(2/3/4): p. 208-210.
 97. Umbaugh, S.E., *Computer Vision and Image Processing*. 1998, NJ, USA: Prentice Hall.
 98. Winograd, S., *A New Algorithm for inner product*. IEEE Trans. Comput., 1968. 17: p. 693-694.
 99. Witten, I.H., A. Moffat, and T.C. Bell, *Managing Gigabytes, compressing and indexing documents and images*. 2nd ed. 1999, San Diego, USA: Morgan Kaufmann Publishers.
 100. Yang, H.S., S.U. Lee, and K.M. Lee, *Recognition of 2D Object Contours using Starting-Point-Independent Wavelet Coefficient Matching*. Journal of Visual Communication and Image Representation, 1998. 9(2): p. 171-181.
 101. Yang, J. and C.T.o.S.V.D.a.V.Q.f.I.C. Lu, *IEEE Transactions on Image Processing*, vol. 40, No. 8, pp. 1141-1146, August 1995, *Combined Techniques of Singular Value Decomposition and Vector Quantization for Image Coding*. IEEE Transactions on Image Processing, 1995. 40(8): p. 1141-1146.
 102. Yong, I., et al., *An Analysis Technique for Biological Shape*. Computer Vision, Graphics Image Processing, 1974. 25: p. 357-370.
 103. Zhang, D., *Image Retrieval based on Shape*, in *GSCIT*. 2002, Monash University.
 104. Zhu, L., *Keyblock: An Approach for Content-based Image Retrieval*, in *Department of Computer Science and Engineering*. 2001, University of New York: Buffalo, NY.
 105. Zhu, L., A. Rao, and Z. A., *Advanced Feature Extraction for Keyblock-based Image Retrieval*. Information Systems, 2001. 19(4): p. 1-20.
 106. Zhu, L., et al. *Using Thesaurus to Model Keyblock-based Image Retrieval*. in *IEEE International Conference on Multimedia and Expo*. 2001. Tokyo, Japan.
 107. Zhu, L., C. Tang, and A. Zhang. *Using Keyblock Statistics to Model Image Retrieval*. in *The 2nd IEEE Pacific-Rim Conference on Multimedia*. 2001. Beijing, China.
 108. Zhu, L. and A. Zhang. *Supporting Multi-Example Image Queries in Image Databases*. in *IEEE International Conference on Multimedia and Expo*. 2000. New York, NY.
 109. Zier, D. and J.R. Ohm, *Common Datasets and Queries in MPEG-7 Color Core Experiments*. 1999.