Detection of Near-duplicates in Large Image Collections

A thesis submitted in fulfilment of the requirements for
the degree of Doctor of Philosophy

Jun Jie Foo
B.App.Sci (Hons),
School of Computer Science and Information Technology,
Science, Engineering, and Technology Portfolio,
Royal Melbourne Institute of Technology,
Melbourne, Victoria, Australia.

January 23, 2008
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Jun Jie Foo
School of Computer Science and Information Technology
Royal Melbourne Institute of Technology
January 23, 2008
Acknowledgments

I thank my family in Malaysia (and in Hong Kong) for their unrelenting support and encouragement, and for always believing in me.

I am very thankful to my supervisor, Dr. Seyed M. M. (Saied) Tahaghoghi for introducing me to the world of academic research. I offer my insubstantial gratitude to my second supervisor, Professor Justin Zobel, who has been an inspiration, personally and professionally. His support and advice have invaluably contributed to the quality of this thesis. I would also like to thank Dr. Ranjan Sinha for his invaluable efforts in our collaboration in various research projects. I also thank my fellow students in the Search Engine Group at RMIT for providing an enjoyable work environment.

Finally, I thank all anonymous reviewers of previous publications who have contributed to this thesis through their valued comments.
Credits

Portions of the material in this thesis have previously appeared in the following publications.

Refereed Conferences and Workshops


This work was supported by the Australian Research Council.
The thesis was written in the Vim editor on Mandrake GNU/Linux, and typeset using the \LaTeX\ document preparation system.
All trademarks are the property of their respective owners.
Note

Unless otherwise stated, all fractional results have been rounded to the displayed number of decimal figures.
Contents

Abstract 1

1 Introduction 3
  1.1 Near-duplicate image detection ................................. 7
  1.2 Thesis overview ............................................. 8

2 Background and Related Work 11
  2.1 Near-duplicate detection for text .............................. 11
  2.2 Digital image watermarking .................................. 15
  2.3 Content-based image analysis ................................ 16
    2.3.1 Feature representation .................................. 17
    2.3.2 Distance metrics .......................................... 31
  2.4 Invariant interest points and local descriptors ............... 33
    2.4.1 Existing interest point detectors and local descriptors . . 40
    2.4.2 Discussion .................................................. 47
  2.5 Image indexing ................................................ 49
    2.5.1 Locality Sensitive Hashing (LSH) .......................... 51
    2.5.2 On indexing local descriptors ............................. 55
    2.5.3 Redundant Bit Vectors (RBV) .............................. 57
  2.6 Clustering ..................................................... 60
  2.7 On efficient object-recognition approaches ..................... 62
  2.8 Techniques for near-duplicate detection ......................... 64
  2.9 Evaluation .................................................... 71
    2.9.1 Effectiveness of pairwise detection ....................... 71
    2.9.2 Clustering effectiveness .................................. 73
CONTENTS

2.9.3 Image collections ........................................ 74
2.10 Summary .................................................. 75

3 The Duplication Problem ....................................... 77
3.1 What is a near-duplicate image? ............................ 78
3.2 Near-duplication in web image search ...................... 82
3.3 Analysis of near-duplicate images on the Web .......... 82
3.4 Summary ................................................... 91

4 Efficient Near-Duplicate Image Retrieval .................. 92
4.1 Improving retrieval efficiency by pruning SIFT .......... 92
  4.1.1 Experimental setup ..................................... 95
  4.1.2 Results ................................................ 96
    Repeatability of pruned keypoints ........................ 97
    Retrieval effectiveness and efficiency .................... 99
    Further studies .......................................... 101
    Retrieval effectiveness on large collections ............ 106
    Comparative evaluation against the DPF method .......... 109
  4.1.3 Discussion ............................................. 116
4.2 Indexing of local descriptors using a modified RBV .... 116
  4.2.1 Limitations of Redundant Bit Vectors (RBV) .......... 117
  4.2.2 Indexing using a modified RBV ....................... 118
  4.2.3 Querying the modified RBV index ..................... 120
  4.2.4 Experimental setup ................................... 122
  4.2.5 Results ................................................. 123
    Retrieval accuracy ....................................... 123
    Retrieval efficiency ..................................... 125
    Related studies .......................................... 128
    Discussion ................................................ 130
4.3 Summary .................................................. 131

5 Automated Discovery of Near-duplicate Images ............. 133
5.1 Related work ............................................... 134
5.2 The discovery problem .................................... 134
5.3 Deriving the relationship graph using local descriptors 135
# List of Figures

1.1 Examples of near-duplicate images within image search results. .................. 4

2.1 Examples of RGB colour histograms from two images. .............................. 21
2.2 Diagram of band-pass filters. ......................................................... 26
2.3 The Mallat decomposition of the Lena image. ......................................... 29
2.4 Examples of blurred images using the Gaussian operator. ......................... 36
2.5 Three-dimensional scale-space axis. .................................................. 39
2.6 Process of difference-of-Gaussian at various scales. ................................ 42
2.7 Diagram illustrating the search space of a high-dimensional point. ............ 52

3.1 Examples of near-duplicate images. .................................................. 80
3.2 Image results from Google image search. ............................................. 83
3.3 Image results from Yahoo! image search. ............................................ 84

4.1 Examples of images with SIFT-detected keypoints before and after our pro-
posed reduction scheme. ................................................................. 94
4.2 Repeatability of the keypoint-reduced PCA-SIFT and other descriptors. .... 98
4.3 Effects of the number of PCA-SIFT local descriptors on retrieval accuracy,
query run-time, and index size. ..................................................... 100
4.4 Effectiveness of the original approach and the T100 method in identifying the
original image using altered ones. ................................................... 103
4.5 Effectiveness of original approach and T100 method using altered images as
query. ................................................................. 104
4.6 Effectiveness of the DPF method using various \( m \) parameters. .......... 110
4.7 Average relative rank of altered images using the DPF method. ............... 113
4.8 Average relative rank of altered images using the PCA-SIFT methods (both
the original and the T100). ........................................... 115
4.9 Effectiveness of the RBV index using various HCS parameters and dimensions. 124
4.10 Average run-time for querying the RBV index. ............................ 125
4.11 Effectiveness of search space reduction of the RBV index. ..................... 126
4.12 Effectiveness of RBV using memory-resident local descriptors. ............. 128
4.13 Growth factors pertaining to the RBV index when increasing collection sizes. 130

5.1 Process of converting local descriptors to representative units. ................ 136
5.2 Coverage, average precision, run-time, and identified edges on collections 20K
and 40K using the HPC algorithm. ..................................... 141
5.3 Coverage, average precision, run-time, and identified edges on collections 20K
and 40K using the keypoint-reduced HPC algorithm. .......................... 143
5.4 Coverage and average precision of the keypoint-reduced HPC algorithm on
collection 150K. ................................................................ 147
5.5 Run time and number of edges for collection 150K using the keypoint-reduced
HPC algorithm. .................................................................. 148
5.6 Recall and precision of the DPF method on web images. ....................... 156

6.1 Example of image answers from Google image search using query “Edvard
Munch Madonna”. .......................................................... 161
6.2 Number of non-overlapping clusters identified by the ND-CENTER algorithm. 167
6.3 Distribution of the number of images in clusters identified by the ND-CENTER
algorithm. ................................................................... 169
6.4 Average number of algorithm-identified clusters for artificial near-duplicate set. 171
6.5 Average purity and entropy using various CT values. .............................. 172
6.6 Average recall and precision over various CT values. ............................. 173
6.7 Average purity and entropy for each near-duplicate group using CT = 100. . 174
6.8 Average recall and precision for each near-duplicate group using CT = 100. . 175
6.9 Efficiency of clustering of near-duplicate images in large image collections. . 177
6.10 Ratio between seeded near-duplicate images and all images identified within
clusters. .............................................................. 179
6.11 Some examples of the algorithm-formed near-duplicate image clusters. .... 184

B.1 Average repeatability of all tested local descriptors on 50 image alteration. . 199
LIST OF FIGURES

B.2 Average relative rank of altered images using the DPF method. . . . . . . . . 200
B.3 Average relative rank of altered images using the PCA-SIFT (both the T100 and the original) methods. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 201
D.1 More examples of the algorithm-formed near-duplicate image clusters. . . . . 210
# List of Tables

2.1 Different number of taps from the Daubechies wavelet. .................. 28

3.1 Near-duplicate analysis of images from Google image search. ............ 87

3.2 Analysis of near-duplicate alterations in manually evaluated web images. . 90

4.1 Average repeatability of all local descriptors. .............................. 99

4.2 Effectiveness of keypoint-reduction approach using various threshold parameters. 105

4.3 Effectiveness of T100 method on large image collections. .................... 107

4.4 Effectiveness of T100 method on various collections and image alterations. . 108

4.5 Effectiveness of the DPF and the PCA-SIFT approaches (both original and T100 method). .............................................................. 111

4.6 Effect of HCS parameters on the RBV index size. ............................ 129

5.1 Effectiveness and efficiency of the original and keypoint-reduced HPC algorithm on collection 40K. ...................................................... 145

5.2 Effectiveness and efficiency of the keypoint-reduced HPC algorithm on large collections. .............................................................. 149

5.3 Coverage and average precision of the keypoint-reduced HPC algorithm for each image alterations on all collections. ......................... 151

5.4 Effectiveness and efficiency of the HPC algorithm and query-based approaches on web images. ...................................................... 153

6.1 Clustering results (including recall, precision, purity, and entropy) for the web collection. .............................................................. 182

B.1 Average repeatability of PCA-SIFT local-descriptors using various threshold values. .............................................................. 197
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.2</td>
<td>Average repeatability of the SIFT, SURF, and GLOH local descriptors.</td>
<td>198</td>
</tr>
<tr>
<td>C.1</td>
<td>Coverage and average precision of the HPC algorithm on collection 20K with 50 image alterations.</td>
<td>203</td>
</tr>
<tr>
<td>C.2</td>
<td>Coverage and average precision of the HPC algorithm on collection 40K with 50 image alterations.</td>
<td>205</td>
</tr>
<tr>
<td>C.3</td>
<td>Effectiveness and efficiency of the HPC algorithm using various parameters of $l$, $k$ and $T$ on collection 150K.</td>
<td>207</td>
</tr>
</tbody>
</table>
Abstract

The vast numbers of images on the Web include many duplicates, and an even larger number of near-duplicate variants derived from the same original. These include thumbnails stored by search engines, copies shared by various news portals, and images that appear on multiple web sites, legitimately or otherwise. As an example, on October 8 2007 the top twenty thumbnails returned by a web image search for the query “miles davis kind of blue” contained 11 versions of the same album cover, with numerous minor alterations reflecting reissues and similar minor variations.

Such near-duplicates appear in the results of many web image searches, and constitute redundancy, and may also represent infringements of copyright. Digital images can be easily altered through simple digital manipulation such as conversion to grey-scale, colour balance change, rescaling, rotation, and cropping. Any of these operations defeat simple duplicate detection methods such as bit-level hashing. The ability to detect such variants with a reasonable degree of reliability and accuracy would support reduction of redundancy in collections and in presentation of search results, and also allow detection of possible copyright violations.

Some existing methods for identifying near-duplicates are derived from computer vision techniques; these have shown high effectiveness for this domain, but are computationally expensive, and therefore impractical for large image collections. Other methods address the problem using conventional CBIR approaches that are more efficient but are typically not as robust. None of the previous methods have addressed the problem in its entirety, and none have addressed the large scale near-duplicate problem on the Web; there has been no analysis of the kinds of alterations that are common on the Web, nor any evaluation of whether real cases of near-duplication can in fact be identified.

In this thesis, we analyse the different types of alterations and near-duplicates existent in a range of popular web image searches, and establish a collection and evaluation ground
truth using real-world near-duplicate examples.

We present a simple ranking approach to reduce the number of local-descriptors, and therefore improve the efficiency of the descriptor-based retrieval method for near-duplicate detection. The descriptor-based method has been shown to produce near-perfect detection of near-duplicates, but was previously computationally very expensive. We show that while maintaining comparable effectiveness, our method scales well for large collections of hundreds of thousands of images. We also explore a more compact indexing structure to support near duplicate image detection.

We develop a method to automatically detect the pairwise near-duplicate relationship of images without the use of a query. We adapt the hash-based probabilistic counting method — originally used for near-duplicate text document detection — with the local descriptors; our adaptation offers the first effective and efficient non-query-based approach to this domain.

We further incorporate our pairwise detection approach for clustering of near-duplicates. We present a clustering method specifically for near-duplicate images, where our method is arguably the first clustering method to achieve a high level of effectiveness in this domain. We also show that near-duplicates within a large collection of a million images can be effectively clustered using our approach in less than an hour using relatively modest computational resources.

Overall, our proposed methods provide practical approaches to the detection and management of near-duplicate images in large collections.
Chapter 1

Introduction

Digital images are widely applied in all fields of human endeavour, from medicine, design, and law enforcement to architecture and entertainment; virtually all forms of images, such as X-rays, blueprints, mug shots, and photographs, can be stored digitally. They are also present in vast numbers on the Web, where virtually every web page contains a digital image. For instance, news articles are often accompanied by representative digital images. Although obtaining an exact count of the digital images on the Web is infeasible, an estimate from Lyra research indicates that there were more than a billion digital images on the Web in 2006.\(^1\) Given the increasing affordability of digital imaging technology — such as digital cameras and camera phones — the cheap affordability of non-volatile storage, and high-bandwidth data transmission, it seems inevitable that the volume of digital images that is captured for personal use or uploaded to the Web will continue to grow.

A major obstacle for effective management of digital images in large repositories and in environments such as the Web is that many of the images can be copies or variants of each other. Many online images can be easily appropriated without acknowledgment of the source and, accidentally or otherwise, disguised through simple processing. For instance, multiple copies of the same image can exist on the Web, where they can be altered using techniques including conversion to grey-scale, change in colour balance, rescaling, rotating, cropping, and various digital editing or filtering operations. Other image variants include the scaled-down thumbnails kept by web search engines and cached by modern operating systems, and differing versions of a single image made available by different news portals. Consider the images presented in the top 50 images that are retrieved for the query “bill clinton” from a

\(^1\)http://www.shutterfly.com/about/prs_sub_062700a.jsp
Figure 1.1: Image answers returned from Google image search for the text query of “bill clinton”. The circles indicate the near-duplicates that are found within the answers; each colour depicts a different set.
web image search. As shown in Figure 1.1, some images are clearly variants of each other but appear to come from different web sources. These image copies and variants are known as near-duplicates [Ke et al., 2004] or image replica [Qamra et al., 2005]. Near-duplicate images also present problems for private repositories since personal images are often modified for visual aesthetics. For instance, the cropping operation is a common technique used to improve image composition in digital photographs [Kelby, 2006]. Two copies of an image on the Web, or in private repositories are rarely identical at the bit-level. For example, the GIF format is considered to be lossless [Giorgianni and Madden, 1998], but this format uses an indexed 256-colour colour-space, and therefore some information is lost when an image with more colours is saved in this format. Consequently, a simple format difference between two visually identical images can render bit-level copy detection mechanisms such as the MD5 or SHA-1 checksum algorithms ineffective.

Detection of near-duplicates is of value in digital image retrieval, as it allows various edited versions or different formats of the same image to be identified. Such a task allows content owners to identify possible violations of digital image copyright. It also allows image distributors or publishers to ensure that digital images are used in accordance with an agreement, thus detecting potential breach of copyright. The vast volume of images on the Web make monitoring and tracking digital images for infringement of copyright a time-consuming and laborious process.

Another application of near-duplicate detection is the identification of information redundancy within large image repositories. Images that are derived from a shared source — possibly disguised using some form of image processing — are considered redundant as they do not contain additional core information that is not already contained within the original image. From the perspective of an administrator of a large image repository, it is useful to be able to detect such near-duplicates for purposes of maintenance and optimisation. For instance, it can be advantageous to eliminate near-duplicates, or organise them for expedited search and retrieval. It may also be useful to determine whether search results contain images that are near-duplicates of each other, so that redundant information can be organised more effectively to reduce the effort required for browsing the search results.

Hence, for copyright detection and redundancy management, it is attractive to be able to reliably identify near-duplicate images. Specifically, an effective detection scheme would allow redundant images to be accurately identified and organised effectively; such a scheme also deters distribution of unauthorised copies, as illicit distributors of image copies could be identified.
CHAPTER 1. INTRODUCTION

A field that focuses on prevention of piracy of digital images is digital image watermarking [Hartung and Kutter, 1999; Kang et al., 2003], where an encrypted mark is embedded into a digital image; the mark is expected to remain unchanged even after an image has been edited and copied [Fatès and Petitcolas, 2000]. However, copies of images can be detected only when the original released image contains the embedded watermark. Moreover, digital watermarks are of limited use for detecting image copies in large collections such as the Web, due to the numerous proprietary watermarking schemes currently in use; it is impractical for detection to cater for these different schemes.

Another approach is to search for image copies using the content itself. This approach, broadly known as content-based image retrieval or CBIR [Smeulders et al., 2000; Datta et al., 2005], aims to retrieve images that are related to a specific query using image content, usually by visual or topical similarity. In contrast to digital watermarking, content-based detection of image copies does not require any additional information to be embedded in the original image, and the detection is based on the same information that is displayed. In general-purpose CBIR methods, a score is computed for every image within a collection according to how similar it is to the query, and the user is presented with a list of images ranked by decreasing similarity score; when a user searches for images on a particular topic, only images that are semantically relevant should be returned. Most CBIR systems assume image self-similarity in their design [Smeulders et al., 2000]: an identical copy of an image will return the highest possible similarity score to indicate a perfect match. However, the problem with current CBIR systems is that none explicitly supports searching for different versions of a given image.

While our focus is related to the problem of general image retrieval, the definition of relevance for a retrieval task in our problem domain can be more easily defined. In general-purpose image retrieval, the definition of relevance is somewhat ambiguous; the topical or semantic similarity between two images is subject to human interpretation. It is difficult to find a universally accepted standard of similarity. In this thesis, we define a pair of images as near-duplicates only if they (are suspected to) share the same original image source.

In this work, we investigate a content-based approach that specifically targets near-duplicate images. In addition to copyright detection, this approach can also be used to detect redundant copies of images within large collections.
1.1 Near-duplicate image detection

We focus on the problem of identifying near-duplicate images where one is derived from another, or both are derived — directly or indirectly — from a shared image source. The focus of our research is to investigate techniques to efficiently and accurately identify near-duplicate images, and therefore address the problems discussed earlier. For a search of near-duplicate images in a large image collection to quickly yield results, the detection algorithm needs to be fast and scalable. The algorithm also needs to perform at a high level of effectiveness; it should be robust against image degradation and photometric, lighting, and various geometric variations — such as cropping, scaling, and rotation — that are common in near-duplicate images. A high level of effectiveness also implies that the algorithm-identified results closely resemble those of a human observer, allowing operation with minimal human intervention.

Previously proposed methods for detecting near-duplicate images have limited ability to address the range of problems discussed above; most are sensitive to image changes such as cropping, contrast levels, brightness, or any degradation in image quality [Chang et al., 1999; Berrani et al., 2003; Yang et al., 2006; Maret et al., 2006]. Thus far, the approach proposed by Ke et al. [2004] is perhaps the only robust method for detecting near-duplicate images. The high-level of effectiveness of this method is due to the application of computationally expensive PCA-SIFT local descriptors; these descriptors represent local image regions that are insensitive to degradations in image quality, digital editing operations, and various photometric changes. However, a major drawback of this method is that, depending on the image dimensions and quality, each image can produce hundreds to thousands of local descriptors, all of which are required for the comparison between two images. Due to the impractical search time, this method has limited application for large image collections.

Another method is proposed by Qamra et al. [2005], which uses non-metric similarity measures on typical CBIR image features for this task. This method is also not specifically designed for the detection of near-duplicates, but for robust detection of images with similar visual content [Li et al., 2003]. While this method is efficient, it is sensitive to image degradation and photometric variations. Qamra et al. also note that their approach is not as effective as that of Ke et al. [2004]. Some recent methods that target near-duplicate video keyframes — that have application in video scene and domain tracking [Ngo et al., 2006] — assume that near-duplicate images and keyframes are equivalent [Zhang and Chang, 2004; Chum et al., 2007; Zhao et al., 2006]; but we believe they are within different problem domains. Near-duplicate video keyframe detection considers keyframes that are taken under
varying angles and timeframes. None of these keyframe detection methods show high levels of effectiveness for detecting images derived from the same source. Other methods — as we elaborate in Chapter 2 (see page 64) — have also been proposed in this domain, but the methods presented in this thesis are the first practical solution to efficient and accurate identification of near-duplicate images within large collections.

1.2 Thesis overview

In this thesis, we develop methods to efficiently and accurately identify near-duplicate images in large collections. We investigate several areas of interest relating to near-duplicate image detection. Specifically, we pose the following research questions:

1. What is a near-duplicate image?
   - What are the different kinds of image copies and near-duplicate images on the Web?
   - Is there a difference between near-duplicate images in controlled collections and near-duplicates on the Web?

2. Can we improve the efficiency of the computationally expensive PCA-SIFT local-descriptor-based near-duplicate detection method?
   - How can the efficiency of the PCA-SIFT local descriptors be improved for detecting near-duplicate images, while maintaining effectiveness?
   - Are the improvements scalable to large image collections?
   - Are there more efficient alternatives of indexing local descriptors for near-duplicate detection?

3. Can we effectively detect near-duplicate images using local descriptors without a query example?
   - How do we develop a non-query-based detection method?
   - How does the effectiveness and efficiency of this method compare to a query-based method?
   - Can we use this method for large image collections?

4. Can we use a clustering approach to organise near-duplicate images?
1.2. THESIS OVERVIEW

- How do we develop a clustering technique for near-duplicate images?
- Is this method effective for large image collections and real near-duplicate examples on the Web?

In Chapter 2, we explore background material relating to near-duplicate detection, and general content-based image retrieval; we also discuss previous approaches and related concepts for detection of near-duplicate images.

In Chapter 3, we discuss the scope of our problem, and the kinds of near-duplicates that are common on the Web; we also describe the methodology used to gather real-world near-duplicate examples from the Web. We gather approximately 19,000 images from the Web and manually analyse the different kinds of near-duplicates present in this collection. We use these images in subsequent chapters for testing of near-duplicate detection methods to observe the effectiveness of these methods on real-world data.

In Chapter 4, we describe a method of pruning the large number of local descriptors using a simple ranking and thresholding scheme to improve the efficiency of the local descriptor-based detection. We compare the effectiveness of our newly proposed method with the baseline methods of Ke et al. [2004] and Qamra et al. [2005]. We show that using our approach, the comparison between two images can be limited to using only approximately 10% of the original number of local descriptors, reducing the search time by a factor of 70 while maintaining comparable levels of effectiveness. We also investigate an alternative indexing structure known as Redundant-Bit Vectors (RBV) [Goldstein et al., 2004]; this structure was originally proposed as a high-dimensionality indexing structure with low memory requirements. We describe modifications to adapt the RBV structure for near-duplicate image detection, and at the same time address the limitations of the original RBV structure. Experiments on our modified RBV structure indicate that, while it does not yield the same level of effectiveness as the indexing approach of Ke et al. [2004], it yields a satisfactory level of effectiveness and is substantially more compact in terms of memory requirements.

Most of the previously proposed search methods are limited to using a query example by which to compare an image collection, and few have explored the pairwise detection of near-duplicates without the use of a query example. In Chapter 5, we propose an approach to automatically sift a collection of images to using a discovery method — without using a query — that automatically identifies the pairwise near-duplicate relationship between any two images. We use the same image local descriptors, and adapt a method that was originally used for near-duplicate text document detection for images. This method allows a
collection of images to be precomputed so that search and retrieval of near-duplicate images that correspond to a specified query within a collection can be returned rapidly. We present detailed discussion and experiments of the discovery method against the query-based baseline method. We show that our proposed approach is capable of achieving a level of effectiveness that, while not superior to those achieved by the baseline query-based approaches, is comparable; our experiments also show that our discovery method is also more efficient than the approach of Qamra et al. [2005], while achieving a higher level of effectiveness.

In Chapter 6, we describe a method for clustering near-duplicate images using a thresholding scheme that adapts a standard clustering technique for text-document detection; we also use our newly proposed discovery method as part of this process. Our clustering method allows near-duplicates and redundant image copies to be quickly identified, and therefore improve the organisation and management of such instances in large repositories. Experiments show that our proposed method produces highly accurate clustering results, being capable of efficiently and accurately identifying near-duplicate image clusters within large image collections while using modest computational resources. This is arguably the first near-duplicate clustering method that demonstrates a high level of effectiveness and efficiency on such collection sizes.

We summarise this research and draw conclusions in Chapter 7.

All our experiments initially use a small image collection (≤ 20,000) to test the robustness of our proposed methods against 50 common digital alterations, and photometric changes. They are followed by large-scale experiments where the same tests are performed on large image collections (of up to a million). Through experimentation with large-scale image collections and real near-duplicate web examples, we demonstrate our methods to be more effective and efficient than previously proposed methods in detecting near-duplicate images, and that all our proposed methods can scale well for a million images. The proposed methods in this thesis is perhaps the first practical approach for detecting near-duplicate images in large collections.
Chapter 2

Background and Related Work

Near-duplicate image detection and retrieval has only recently begun to receive attention from researchers. While near-duplicate image detection — by definition the detection of images that are close to being duplicates — may appear to be within the domain of conventional content-based image retrieval, the problem statement differs. Consider a query example for which a collection of images is searched: a traditional content-based retrieval system, which is designed to retrieve visually similar images, that returns a mixed-bag of similar images and near-duplicates may be deemed highly effective; in contrast, a near-duplicate detection system that yields the same set of result is undesirable given that only a subset of the images in the mixed-bag is of interest. Thus, the main distinction between a near-duplicate image detection system and a general image retrieval system is that the former focuses on images that are redundant or are (suspected to be) derived from the same source.

In this chapter, we discuss near-duplicate text document detection and digital image watermarking in the context of near-duplicate detection. We review the foundations of content-based image retrieval — that is, image feature representation, indexing, and search — and discuss features of robust computer vision, and examine the state-of-the-art approaches to near-duplicate detection, where we identify the limitations that motivate our research.

2.1 Near-duplicate detection for text

Near-duplicate text document detection, within the domain of information retrieval, is related to near-duplicate image detection. The detection of near-duplicate text documents is relevant to our work, in that the problem has similarities to that of near-duplicate image detection. While the field of near-duplicate document detection is well-researched, direct application
of text-document detection techniques for images is hindered by the fact that digital images typically have only limited associated textual information. For example, on the Web, images are rarely accompanied by corresponding text documents that explicitly describe them; they are merely accompanied by anchor text or simple keywords.

Nevertheless, we believe that some techniques developed for documents can be adapted for the image domain. In this section, we discuss some near-duplicate text document detection techniques that are adapted and used for images in our work.

There are two main approaches to detection of near-duplicate documents: full-text-based [Yang and Callan, 2006] and fingerprint-based [Manber, 1994]. The full-text-based approach uses document statistics such as relative frequency of words between documents — much like traditional text information retrieval approaches — to determine the likelihood that one document is a near-duplicate of another. Thus a list of documents can be ranked using document statistics, where the first document has the highest estimated likelihood of being a near-duplicate of a given example. Techniques such as relative frequency matching [Shivakumar and Garcia-Molina, 1998] and the identity measure [Hoad and Zobel, 2003] are examples of this approach. The full-text-based approach is suitable for applications where the task requires a list of documents to be evaluated for near-duplicates based on one specified document example (a query), just as with like a traditional search engine, a particular text query is used to find a list of documents. This is also commonly known as the one-to-many paradigm [Hoad and Zobel, 2003].

The fingerprint-based approach is more suitable for the evaluation of all near-duplicate pairs within a given text document corpus, as it is more efficient in the processing required for pair-wise evaluation of all documents [Bernstein and Zobel, 2004]. The identification of all instances of near-duplicates within a given collection is also known as the many-to-many or discovery problem [Bernstein and Zobel, 2004].

For the adaptation of such techniques to the image domain in this work, we focus on the fingerprint-based approach, and do not investigate the full-text-based approach. For near-duplicate images, the one-to-many paradigm has been addressed by current techniques [Ke et al., 2004; Meng et al., 2003; Qamra et al., 2005; Yang et al., 2006]; it is also a common approach in content-based image search, as described in Section 2.3. Our adaptation of text-based techniques focuses on the discovery — many-to-many — paradigm, which has not been well-explored in the near-duplicate detection domain (see Section 2.8).
2.1. NEAR-DUPLICATE DETECTION FOR TEXT

Document fingerprinting

In most of the fingerprint-based approaches, documents are converted into compact representations, or fingerprints, that are comprised of a set of hash values (integers) derived from substrings of a document [Yang and Callan, 2006]. The effectiveness of a fingerprint-based approach is directly influenced by a few parameters [Hoad and Zobel, 2003], namely generation, granularity, size, and substring selection strategy:

Generation. Fingerprint generation describes the process that is used to generate a representation for a given string (or substring). This process is performed using a hash function, where a string is converted into a hash value; a hash function guarantees reproducibility of hash values for the same string. An ideal function is one that is efficient to compute and minimises the probability of collision — sharing of identical hash values — of two different substrings [Ramakrishna and Zobel, 1997].

Granularity. Fingerprint granularity refers to the size of a substring; the selection strategy ranges from the number of characters in a string [Manber, 1994] to the number of words in a sentence [Heintze, 1996]. A high granularity increases the likelihood of false matches due to the increased probability of matching more substrings of any two documents; for example, a substring of a single character provides the highest granularity and is likely to match large numbers of documents. In contrast, a low granularity has an opposite effect as fewer substring matches can result from the increased selectivity of the matching criterion. For example, a substring of “the” will probably result in more document matches than the substring of “thespian”.

Size. The size of the fingerprint affects the amount of processing required to find a document; it also has a direct impact on the effectiveness of the evaluation, as longer fingerprints tend to yield more matches [Heintze, 1996]. While longer fingerprints generally yield greater accuracy they also incur larger storage overhead.

Substring selection. There are many strategies for choosing substrings from documents by which to produce the hash values. These include position-based selection using offset values, and frequency-based strategy using substrings (or phrases) based on occurrence statistics. Hoad and Zobel [2003] provide a detailed survey of various substring selection strategies.

Using a fingerprint-based scheme with the appropriate choice of these parameters, a near-
duplicate document can be represented such that a pair is likely to share common patterns in their representations; common patterns can be identified by analysing the document fingerprints. Although various fingerprint-based techniques have been proposed [Manber, 1994; Brin et al., 1995; Broder et al., 1997; Shivakumar and Garcia-Molina, 1998; Bernstein and Zobel, 2004], the basic process of these techniques are quite similar: Text documents are first parsed into representative substrings (or units) of either contiguous words or characters — also known as shingles — that can be indexed in an inverted file [Zobel and Moffat, 2006; Witten et al., 1999]. Each index entry contains the postings list of documents (IDs) in which a particular unit occurs, along with any auxiliary information, such as unit offsets from the beginning of a document, or the frequencies of unit occurrences within a document. Most fingerprint-based techniques exploit the postings list to evaluate the pair-wise near-duplicate relationship of a given document corpus; the principal differences lie in substring selection heuristics [Bernstein and Zobel, 2004].

Manber [1994] proposed counting of the number of identical postings lists using the inverted index of the substrings, but observed limited effectiveness in detecting near-duplicate documents. Broder et al. [1997] proposed counting the number of all possible document pairings in each postings list to identify the number of co-occurring units in any two documents; they also propose a document clustering method using this approach. Although this method is effective, it is costly, as the number of unique document pairings that can be generated is quadratic in the length of the postings list. Shivakumar and Garcia-Molina [1998] address the scalability issues of exact counting of document pairings by introducing a filter-and-refine scheme, based on hash-based probabilistic counting. The key idea is to set an upper-bound on the number of unique document pairs by coarse counting in the first-pass — using a hash table — to discard pairs that do not have sufficient co-occurring units. This method is efficient: given a hash table of sufficient size, the number of identified edges can be dramatically reduced. Bernstein and Zobel [2004] proposed using only shingles (contiguous words of a specified length) that occur in more than one document; they discard unique shingles that do not contribute towards the identification of near-duplicates. To discards unique shingles efficiently, they adapt the hash-based probabilistic counting [Shivakumar and Garcia-Molina, 1998] to incrementally discard unique shingles using multiple passes, where the length of the shingles is increased at each pass. Bernstein et al. showed that this method can be used to identify near-duplicate documents with good scalability. Instead of using the postings list from the inverted index, Haveliwala et al. [2000] showed that document fingerprints can be generated using a bag-of-words that are selected from anchor text, and from text surrounding
the URLs. They showed that locality sensitive hashing [Indyk and Motwani, 1998] indexing technique (see Section 2.5.1) can be used to generate distinctive document fingerprints. They also proposed a document clustering method similar to that of Broder et al. [1997].

In this thesis, we investigate the adaptation of the fingerprint-based approach, specifically the approaches of Shivakumar and Garcia-Molina [1998], for near-duplicate image detection, as we discuss in Chapter 5. We also address the relatively unexplored many-to-many problem in near-duplicate image detection. We adapt the clustering techniques proposed by Broder et al. [1997] and Haveliwala et al. [2000] for this domain, as we describe in Chapter 6.

### 2.2 Digital image watermarking

One of the aims of near-duplicate detection is to allow effective detection of unauthorised copies of images. Digital image watermarking has been proposed as a solution for this problem.

In digital image watermarking, a watermark signal is computed and embedded by the distributor in the original image (also known as the host image) such that the signal can be recovered for verification from the host image or from copies of it. A watermark signal should ideally be secure, imperceptible, robust, and remain in the host image through all forms of digital image processing [Hartung and Kutter, 1999]. The watermark signal is recoverable only if the correct cryptographically secure key is used, preventing distributors of unauthorised copies from tampering with the watermark. Digital watermarking techniques can be broadly categorised into visible and invisible. Visible watermarks, where a message or logo is displayed to the consumer, have limited application [Hartung and Kutter, 1999], and we do not consider these in our discussion.

Research in this field focuses on the design, embedding, and recovery of watermark signals from digital images. Digital image watermarking techniques can be broadly categorised into two groups: spatial and frequency. Spatial watermarks [Wolfgang and Delp, 1996; Darmstaedter et al., 1998] are those that are designed in the spatial domain — the original information space that corresponds directly to the image data — and embedded into the pixel data of a host image; the watermark and host image are usually of the same dimensions. Frequency watermarks, also known as spectral watermarks, are designed in the frequency domain — a transform of the spatial domain using mathematical analysis functions. Examples of frequency watermarks include those based on the discrete cosine transform [Cox et al., 1997; Podilchuk and Zeng, 1998], the Fourier transform [Lin et al., 2001], and the discrete
wavelet transform [Xie and Shen, 2004; Kang et al., 2003]. A major challenge of digital image watermarking is the derivation of a scheme that is resilient to geometric variations such as scale, rotation, and translation. Early work in this field focused on watermark signals that are robust against compression and filtering schemes, whereas schemes robust against geometric variations have recently received more attention [Lin et al., 2001; Kang et al., 2002; 2003; Xie and Shen, 2004].

The recovery of a watermark is achieved using non-blind or blind techniques. Non-blind watermarking techniques [Johnson et al., 1999; Kang et al., 2002] require the host image for recovery of watermark signals within a suspect image. While this approach is suited for situations in which the host image is assumed to be present, it has limited use when the host image is unavailable. Another approach, blind watermarking [Kang et al., 2003; Lin et al., 2001] can recover a signal from an image without the presence of the original host. This is typically achieved using some form of modulation simulation whereby the content of the host image is considered to be noise or distortion; these blind techniques are designed so that, if the distortion is reproducible during the process of recovery, it can be suppressed to recover the watermark signal even without the host image [Hartung and Kutter, 1999].

Digital image watermarks are inherently invasive, that is, they require that additional information be embedded within images. In an event that the ownership of a digital image is disputed, the watermark signal can be extracted from the image for verification purposes. However, for retrieval and comparison of digital images in an environment such as the Web, digital watermarking techniques have limited application. Detection can be costly as each encountered image may need to be processed for each of the watermarking schemes in use. Robustness, particularly given the diversity of image manipulations seen in practice on the Web, remains an open question. In contrast, near-duplicate image detection, which is similar to content-based image retrieval but focuses on near-duplicate images, is advantageous in that every image in a collection is profiled only once [Chang et al., 1999]. For these reasons, we focus on near-duplicate image detection, which emphasises the effective and efficient retrieval of redundant or unauthorised copies within image collections, and do not consider digital watermarking techniques further.

2.3 Content-based image analysis

Near-duplicate image detection is related to Content-based Image Retrieval (CBIR), in that it focuses on detection and retrieval of only the images that are (or are suspected to be)
copies of the same source, whereas CBIR aims at finding visually similar or topically relevant images with regards to a specific information need. While both fields are similar in their goal — that is, to detect images that meet certain requirements — their criteria are different.

Research in CBIR and near-duplicate detection focuses on techniques for image representation and similarity computation within large image repositories that accurately reflect assessments of human observers for their respective tasks; this allows the laborious process of manual image evaluation to be automated. A major obstacle in practical CBIR is the sensory or semantic gap [Smeulders et al., 2000], that is, there exists a disconnect in the notion of relevance and similarity between humans and automated computations. In near-duplicate detection, the notion of similarity is narrower (and less subjective) than that of CBIR; we are only concerned with images that are redundant, or are copies of the same source. Nevertheless, the fundamentals of CBIR such as feature representation and similarity computation are relevant to understanding of near-duplicate image detection. We continue with descriptions of the techniques and features commonly applied in CBIR.

### 2.3.1 Feature representation

In image databases, it is useful to retrieve image data by matching the meaning, also known as similarity search [Santini and Jain, 1995; Smeulders et al., 2000]. For example, a user may want to search for images that depict a particular subject, object, or event. To enable similarity search in an image database, a summary of visual features is automatically extracted for every image as a representation of its content. These visual features can also be used to distinguish images from each other, and serve as a guide to image content [Rui et al., 1999].

Visual features such as colour, texture, and shape are commonly used in CBIR; these may be specified over the entire image or over specific regions of the image. Other features based on local invariants such as interest points, which are typically applied in computer vision, can also be applied in this domain [Loupia et al., 2000; Sebe et al., 2002], but they are mainly used for object-recognition and stereo matching (see Section 2.4). Once these features are extracted, they can be indexed and stored in a feature database. This process is commonly known as feature extraction [Smeulders et al., 2000; Rui et al., 1999] or profiling [Chang et al., 1998].

The extracted features are typically stored as a vector of numbers (integers), where the size of the vector is fixed for the given collection. Using such a vector representation, the similarity between two images can be estimated by computing the similarity between their
respective vectors. On the assumption that the closer the feature data, the higher the level of similarity [Rui et al., 1999], we can calculate the similarity of images — using the vector representation — within a given collection with respect to a query (see Section 2.3.2).

We follow with a discussion of the features that are commonly applied in the CBIR literature, but we focus on invariant interest points as these are relevant to our work in Section 2.4.

Modelling colour

Colour is probably the most commonly used image feature in CBIR [Smith and Chang, 1996b; Stricker and Dimai, 1996]. It is relatively tolerant and robust against photometric changes such as luminance and intensity variation, slight occlusions, and minor changes in scale and orientation [Smith, 2002b].

Before we discuss colour feature presentation, we describe some common colour models [FairChild, 2005] that are used to quantify colours mathematically for digital representation. The range of reproducible colours in a colour model is known as a colour gamut or simply a gamut [Hunt, 2004]. Using a colour model, a colour can be generally represented as three values or dimensions, also commonly known as a channel; the combination of some proportion of each channel produces the desired colour. Popular colour models that are applied in CBIR include the RGB, HSV, YC_bC_r, and perceptually significant colours [Ford and Roberts, 1998; Gonzalez and Woods, 1993].

The most common and widely understood RGB colour model represents colour using Red, Green, and Blue channels, corresponding loosely to the ability of the human visual system in discerning the red, green, and blue components of white light [Hunt, 2004]. This colour model is advantageous in its simplicity, in that it models the hardware process of displaying digital images; most colour CRT monitors use red, green, and blue phosphors that are excited by electron beams to light up a colour pixel. Similarly each pixel in an LCD display can be further divided into three sub-pixels of red, green, and blue.

The HSV model, which represents Hue, Saturation, and Value, is a colour model that is derived from the RGB colour model [Hunt, 2004]; it is also referred to as the HSB or HSI (for Hue, Saturation, and Brightness or Intensity) model. This colour model separates the luminance component — represented by Value — providing a higher resemblance of the manner by which people describe colour; separation of luminance, which is basically the measure of light reflected from a surface, is a desired property of an ideal colour model [Ford
2.3. CONTENT-BASED IMAGE ANALYSIS

and Roberts, 1998]. The HSV model also improves the perceptual uniformity [Poynton, 2003] that the RGB model lacks; that is, the distance between two values in the RGB model is disproportional to the visual differences.

The YC_rC_b colour model similarly represents colour using three values, Y for luminance, C_r for red chrominance, and C_b for blue chrominance; red and blue chrominance are essentially the difference between the red value and luminance, and the blue value and luminance, respectively. This colour model is used in the JPEG image compression standard, and also the MPEG compressed video encoding standard.

While not strictly a colour model, the perceptually significant colours (or culture colours) model human perception. The design of perceptually significant colours is motivated by culture, that is, most people typically describe basic colours derived from a simple set instead of in detail. Perceptually significant colours consists of a set of colours routinely used by humans to describe colour; these include black, grey, white, red, green, yellow, blue, purple, orange, pink, brown, blue-green, and light blue [Carson et al., 1997; Berlin and Kay, 1999]. These colours can be modelled based on certain numerical ranges that are specified using one of the colour models such as RGB or HSV.

In the following sections, we discuss some common colour feature representations that are used in CBIR.

Colour features

Once the human perceived colours of an image are mapped to a numerical representation by a colour model, the difference between two images can be quantified mathematically using their extracted colours (or features).

The most commonly used colour image feature is arguably the colour histogram. A colour histogram captures the colour distribution of an image, and is simple to implement. It is created by quantising the colour value of each pixel into a particular colour range, also known as a bin. Each bin represents the proportion of colour pixels that fall within that colour range; a large number of bins indicates a fine quantisation granularity, whereas a small number of bins produces the opposite effect. The choice of bin size often depends on the application [Bovik, 2000].

Colour histograms can be computed using various colour models; the selection depends on the application. For example, using the RGB colour model, each channel can be individually analysed to obtain three histograms, each of which depicts the distribution of pixels within
its corresponding colour channel. The granularity of quantisation can also vary between each channel; for instance, using the HSV or YCbCr colour models, the brightness or luminance channel can be quantised using a different bin size from the saturation or chrominance channels. Different bin sizes produce variable-width histogram bins, whereas an equi-width histogram is generated using a uniform bin size. The use of a single channel histogram such as the luminance is sometimes known as the intensity histogram, and is commonly used to represent the distribution of the pixels of grey-scale images; the intensity histogram also reflects the distribution of lighting within the image.

A colour histogram can be denoted as a vector $I_{\text{hist}}(0,\ldots,n,\ldots,N-1)$ with $N$ as the bin number which records the amount of pixels that fall within that range. Given an image $I[X,Y]$ of $X \times Y$ pixels, and a gamut of $K$ for a selected colour model, a histogram can be extracted as [Smith, 2002a]:

\[
I_{\text{hist}}[n] = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \begin{cases} 
1 & \text{if } Q(I_{xy}) = n \\
0 & \text{otherwise}
\end{cases} 
\]  

(2.1)

where $Q(\cdot)$ is essentially the quantisation function defined as:

\[
Q(I_{xy}) = \left\lfloor \frac{I_{xy}N}{K} \right\rfloor 
\]  

(2.2)

Figure 2.1 shows the histogram representation of two similar images using the RGB colour model; each channel is quantised using 256 bins. The complete histogram of an image is usually represented as a vector of integers, or floating-point values, each recording the number of pixels within their corresponding colour range. It is also necessary to normalise the complete histogram so that the size of the image does not affect the distribution of colours.

A drawback of the colour histogram is that it stores only the colour distribution of an image, and all spatial information — that is, the locations of particular colours — is lost. Two images may share a similar colour distribution regardless of the spatial layout of the colours within them. For example, the colour histogram representation of an $8 \times 8$ chessboard and a speckled image with an even distribution of black and white will be identical; thus the colour histogram is not useful for distinguishing between images with similar colour distribution. The colour histogram — when applied alone — has limited application for near-duplicate image detection due to its susceptibility to variations of colour, luminance, and geometric variations such as cropping [Meng et al., 2003]. Consider an image that is cropped to retain
2.3. **CONTENT-BASED IMAGE ANALYSIS**

![RGB histograms of two images using 256 bins for each colour channel.](image)

**Figure 2.1:** The RGB histograms of two images using 256 bins for each colour channel. As shown, images with similar colour distribution will produce similar patterns of histogram representation. The figure shows images of the Sphinx, in Cairo; courtesy of FreeDigitalPhotos.net.

only the centre region, presumably the region of interest; its colour histogram could be substantially different from that of the original image.

A colour representation that closely resembles the colour histogram is the Colour Coherence Vector (CCV) [Pass et al., 1996]. It captures additional information of image pixel distribution by distinguishing between **coherent** and **incoherent** pixels; the former refers to pixels that belong to contiguous and sizeable regions of an image, whereas the latter refers to isolated pixels. The CCV is computed in the same manner as the colour histogram, except each histogram bin value — which is typically the normalised number of pixels in a quantised colour range — is analysed as two values instead: number of coherent and incoherent pixels. The CCV has been shown to outperform colour histograms at the expense of higher computational complexity [Li et al., 2002b; Ma and Zhang, 1998b]. Although the CCV was designed to be more robust than the original colour histogram, it was shown to yield only slightly better performance in retrieving visually similar images when compared to the colour histogram [Ma and Zhang, 1998a]; Pass et al. [1996] also observe only slight improvement in robustness against various imaging conditions and photometric changes, as compared to the colour histogram.
The **colour correlogram** is a colour representation method that was proposed by Huang et al. [1997] to improve robustness to minor changes such as scale, occlusion, and viewpoint variation. The idea of correlograms is to measure colour similarity between two images by taking into account the spatial correlation between the image objects of similar colours. The colour correlogram first quantises every image pixel into \( m \) bins. For a pixel \( p \) of colour \( c_i \), a correlogram calculates the probability that there exists a pixel of colour \( c_j \) from another image with a distance of \( k \) from pixel \( p \) for \( k \leq d \), where \( d \) denotes the maximum of a set of fixed distance values [Huang et al., 1997]. Using this method, an \( m \times m \) image will produce a correlogram that maps the probabilities of all values of \( c_i \) and \( c_j \), for \( i,j \in [m] \), and \( k \leq d \). While Ma and Zhang [1998a] and Huang et al. [1997] have shown that the correlograms outperform the colour histogram and the CCV — in that the correlograms are more robust against slight changes in scale and viewpoint variations — they are not explicitly designed to differentiate between near-duplicate images, but images with visually similar colours. This means that, even though it improves upon the colour histogram and the CCV, the correlogram does not make a distinction between images with similar colours and those that are near-duplicates.

A highly compact colour representation method can be derived using colour moments, proposed by Stricker and Orengo [1995]. Colour moments are very efficient in terms of memory requirements in that only nine values — as compared to the number of bin values for a colour histogram — are stored for each digital image. The colour moments of namely mean \((\mu)\), standard deviation \((\sigma)\), and skew \((\theta)\) of each colour channel are used to describe the distribution of image pixel values. For an image \( I[X,Y] \), the moments can be calculated as [Stricker and Orengo, 1995]:

\[
\mu^c = \sum_{c=0}^{C} \frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} I_{xy}^c \quad (2.3)
\]

\[
\sigma^c = \sum_{c=0}^{C} \left( \frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} (I_{xy}^c - \mu^c)^2 \right)^{\frac{1}{2}} \quad (2.4)
\]

\[
\theta^c = \sum_{c=0}^{C} \left( \frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} (I_{xy}^c - \mu^c)^3 \right)^{\frac{1}{3}} \quad (2.5)
\]

where \( \mu^c \) indicates the mean of the pixel values for a component \( c \). Thus, the number of colour moments depends on the colour model used; typical colour models such as the HSV and \( YC_rC_b \) yield nine numerical values. Ma and Zhang [1998a] show that colour moments yield
comparable effectiveness — in retrieval of visually similar images — to the colour histogram and the CCV. The colour moments are also advantageous in their compactness.

In an attempt to model the human perception of colour images, Chang et al. [2000] show that the colours within an image can be represented at various levels using colour filters; each filter is essentially used to represent colour at a different level of detail. They use a combination of a colour histogram (based on the HSV model), colour moments, and shape characteristics [Leu, 1991] of each colour set, such as mean, variance, spread and elongation. Li et al. [2003] show that this method of colour representation, when used in combination with texture features, yields high effectiveness in retrieving visually similar images. Meng et al. [2003] have also shown that this method of representation can also be applied for near-duplicate detection when used in conjunction with appropriate means of comparison. Thus we use this approach of colour representation as one of the baseline approaches for comparative evaluation. We elaborate on this approach further in Section 2.8. We follow with a description of some common texture features that are used in CBIR.

**Texture Features**

Texture describes the spatial distribution of primitive patterns or the intrinsic characteristics of the surface of an object, and is particularly useful for differentiating between objects of the same colour [Manjunath and Ma, 2002]. For example, colour will most likely fail to distinguish images with objects of similar colours, or those of similar colour distribution. To distinguish between these image instances within a large collection, their texture features can be used instead. We discuss some commonly used texture features, where we focus on the wavelet texture analysis that is most relevant to our work.

Many CBIR systems have been developed to allow image search by specification of features using combined texture and colour [Faloutsos et al.; Pentland et al., 1994; Smith and Chang, 1996a]. Texture analysis techniques have been developed in both the spatial and frequency (transform) domains, and can be broadly categorised into statistical and structural approaches [Nixon and Aguado, 2002]. The statistical approaches are developed directly using spatial image statistics, whereas structural approaches exploit the repetitive nature of primitive patterns in most textures. The co-occurrence matrix [Aksoy and Haralick, 1998] is a commonly applied statistical approach that analyses the grey level statistics such as distance, direction, and intensity of pixel pairs of a given image. Structural approaches such as the Fourier and wavelet transforms are widely used for texture analysis [Zhang et al., 2000;
Manjunath and Ma, 1996]. Texture analysis using random field models such as the Markov or Wold random field models have also been successfully applied in image retrieval [Ohanian and Dubes, 1992; Pentland and Liu, 1999]. Another common analysis technique is the Tamura feature [Tamura et al., 1978] that classifies texture into six categories devised to correspond to human visual perception [Howarth and Rüger, 2004]: coarseness, contrast, directionality, line-likeness, regularity, and roughness.

Texture has been extensively studied using multi-resolution analysis in the transform domain. Manjunath and Ma [1996] evaluated various wavelet representations of texture, and reported that wavelet transforms — especially the Gabor wavelet transform — are highly suitable for general image retrieval tasks on the Brodatz texture album [Brodatz, 1966]. The wavelet transform is an increasingly popular tool for texture analysis due to its effectiveness and efficiency [Jacobs et al., 1995; Wang et al., 1997; 2001; Natsev et al., 2004; Albuz et al., 2001]. Here, we limit our discussion to the wavelet transform as it is used in the work of Meng et al. [2003] and Qamra et al. [2005] for detecting near-duplicate images; we use them in one of the baseline methods used in this thesis. A thorough tutorial of the wavelet transform is beyond the scope of this thesis, and is detailed in the work of Antonini et al. [1992], Strang and Nguyen [1996], Valens [1999], and Williams and Amaratunga [1994].

The Wavelet Transform

The wavelet transform is a signal processing tool that can be used to analyse and isolate frequency patterns present in a signal [Strang and Nguyen, 1996]. When applied to digital images, the wavelet transform can identify the location and frequency of image data, which is useful for similarity evaluation between two images.

Digital images are essentially visual data that can be represented by a two-dimensional signal $f(x, y)$. In signal processing, continuous signals are often one-dimensional, described as a function of time. Processing a two-dimensional signal using a one-dimensional signal processing method is straightforward, as the former can be processed one dimension at a time. We digress briefly to describe the concept of the wavelet transform on continuous signals due to its relevance to the discrete wavelet transform.

Given a sample one-dimensional signal $f(t)$, its wavelet transform is defined as:

$$W_{\tau,s} = \frac{1}{\sqrt{s}} \int f(t)\psi^* \left( \frac{t - \tau}{s} \right) dt$$

(2.6)

where $\psi(t)$ is the wavelet or the basis function, and $s$ and $\tau$ are the scale and translation, respectively. The $*$ denotes a convolution of the basis function and the signal. The computation
of $W_{\tau,s}$, which is repeated for all possible values of $s$ and $\tau$, shows the integrated convolution of the basis function and the sampled signal. It also shows how a signal of $f(t)$ is decomposed into a set of basis functions $\psi_{s,\tau}$, known as wavelets. A basis function, or a wavelet, is a *prototype* or basic building block by which the sampled signal can be reconstructed, using only the *coefficients*. The coefficients of any signal for a given wavelet constitute the wavelet transform of that signal; these coefficients, along with the basis function, can be used to reconstruct the original input signal as long as the wavelet fulfills *admissibility* and *regularity* conditions [Antonini et al., 1992].

The wavelet serves as a window function with finite length to analyse the signal at the given interval, where it is repeatedly computed — by dilating and shifting the wavelet by a fixed interval — over the entire signal. Using the above equation, any signal corresponding to a given scale ($s$) of the basis function will yield a large product. The wavelet transform analyses the level of *energy* within the signal, which is calculated using the area under the wavelet curve [Valens, 1999].

Using Equation 2.6, the basis function with a given $s$ and $\tau$ is first placed in position $t = 0$, where the product of the signal and wavelet is obtained; it is then integrated over all $t \geq 0$. The $\tau$ parameter relates to the time information, as it relates to the position of the wavelet function as it shifts along the signal; the scale $s$ is used to either dilate or compress a signal. At high scales, the low frequency signals can be resolved with good frequency resolution but poor time resolution; this means that the dilated basis function can be used to resolve frequency information for a large portion of a signal [Valens, 1999]. At low scales, the temporal information of high-frequency signals can be resolved, yielding good time resolution with poor frequency resolution. This indicates a contracted basis function can be used to clearly analyse a small portion of the signal to accurately resolve time information.

There are an infinite number of wavelets or basis functions by which a signal can be analysed; the common ones can be grouped into families such as Daubechies, Coiflets, Symmlets, or Gabor. The choice of the wavelet families is dependent on the application as there is a trade-off between frequency and time localisation [Williams and Amaratunga, 1994].

### The Discrete Wavelet Transform

The continuous wavelet transform is computed by changing the scale of the basis function, shifting this function in time while multiplying the input signal, and finally integrating the product over all time. This generates a large number of wavelet coefficients from the input...
CHAPTER 2. BACKGROUND AND RELATED WORK

Figure 2.2: This figure shows band-pass filter. The original input signal $f(x)$ is decomposed using high and low-pass filters; $1L$, $1H$, $\ldots$ $nL$, $nH$ denote the subsampled high- and low-pass components of the signal.

signal, most of which are redundant. This is also computationally expensive, even when the wavelet series [Chui, 1992] that are derived from discretising the wavelet transform is used.

Since digital images are essentially discrete representations of continuous signals — where the one-dimensional function over time can be visualised as the fluctuation of data pixels along a row or a column — the discrete wavelet transform [Antonini et al., 1992] can be used. The discrete wavelet transform can also be used for two-dimensional image pixel data $f(x,y)$ by first processing the row pixel data, and then each column of the one-dimensional transformed pixel data. Each colour component of an image can also be separately processed as an independent image, which is computationally expensive as each image is processed the same number of times as the number of components. An alternative is to first convert an image to contain only grey-level pixels, hence each image only has to be processed once [Tahaghoghi, 2002]. In this discussion, frequency (Hz) refers to spectral data of an image — which is the pixel values of an image — where the time component (t) is simply the spatial location of the pixel along a row or column.

The discrete transform retains the properties of good time resolution and poor frequency resolution at high frequencies, and good frequency resolution and poor time resolution at low frequencies. For digital images, this is used to exploit the human visual system as it is more sensitive to low frequency data (overall characteristics) than high frequency data (small details). For example, given an image depicting a beach, human observers notice more of the overall characteristics of the beach rather than detailed information such as the graininess of
the sand.

While the concept of discrete wavelet transform is essentially the same as that of the continuous transform, it is a multi-resolution analysis method derived from sub-band coding [Croisier et al., 1976; Woods, 1990]. The discrete wavelet transform decomposes an input signal into coarse and detailed coefficients using a scaling function [Mallat, 1989] and wavelet function, respectively. These functions are also known as band-pass filters, where the input signal is analysed at different scales of various cut-off frequencies or bands. In this manner, a series of high-pass and low-pass filters with complementary bandwidth is used to separate the input signal into its corresponding high and low frequencies. The high-pass filter complements the low-pass filter, in that they retain only the high and low frequency information from the input signal, respectively. These filter pairs are also commonly known as the quadrature mirror filters [Strang and Nguyen, 1996].

The low-pass filter reduces the input signal to half of its original frequency range; thus any frequencies above this range in the input signal will be removed by the low-pass filter. For this reason, the low- and high-pass filters are also commonly known as the half-band filters. Once the input signal passes the half-band filter, half of input signal can be removed, or subsampled, according to the Nyquist sampling theorem [Nyquist, 1928]; by subsampling the signal, the number of samples are reduced by a factor of two.

This process of sub-band coding — also known as the Mallat decomposition [Mallat, 1989] — can be repeated for further signal decomposition. For a given signal of \( n = x^j \) samples, \( x \) is the factor by which the signal can be subsampled — which is typically 2 — and \( j \) denotes the possible level of decomposition. Figure 2.2 shows a diagram of the band-pass filters where the process of Mallat decomposition of an input signal is depicted. The Mallat decomposition can be computed using [Williams and Amaratunga, 1994]:

\[
c_{m-1,k} = \frac{1}{\sqrt{2}} \sum_{j=0}^{n-1} c_{m,j} a_{j-2k}
\]

\[
d_{m-1,k} = \frac{1}{\sqrt{2}} \sum_{j=0}^{n-1} c_{m,j} (-1)^j a_{N-1-j+2k}
\]

where \( c_m \) denotes a signal in scale \( m \), and \( c_{m-1} \) and \( d_{m-1} \) correspond to, respectively, the coarse and detailed coefficients of this signal in scale \( m - 1 \); \( a_{j-2k} \) and \( (-1)^j a_{N-1-j+2k} \) represent the low and high pass filter coefficients, respectively, for a given wavelet.

As with the continuous wavelet transform, there are various wavelets that can be used with the discrete wavelet transform; the Haar [Jacobs et al., 1995], Daubechies [Wang et al.,
Daubechies-4 | Daubechies-6 | Daubechies-8 | Daubechies-10
--- | --- | --- | ---
0.683013 | 0.470467 | 0.325803 | 0.226419
1.183013 | 1.141117 | 1.010946 | 0.853944
0.316987 | 0.650365 | 0.892201 | 1.024327
-0.183013 | -0.190934 | -0.039575 | 0.195767
-0.120832 | -0.264507 | -0.342657 | -0.045601
0.049818 | 0.043616 | 0.046504 | 0.109703
-0.014987 | -0.014987 | -0.008827 | -0.017792
| | | 0.004717 |

Table 2.1: Different sets of wavelet coefficients from the Daubechies wavelets ranging from 4-taps to 10-taps; the taps reflect the number of sample points considered for the wavelet and the input data.

In Figure 2.3, the process of the discrete wavelet analysis of an image is shown, where the band-pass filters are used to produce the low- and high-pass coefficients. The extracted coefficients reflect the energy content of an image at varying scales, and the concatenation of these coefficients at all decomposition levels comprises the discrete wavelet transform of the original signal; the number of coefficients is the same as that of the original signal. These coefficients can be further concatenated to form a high-dimensional feature vectors, such as those of colour features.

In this thesis, we use the discrete wavelet transform, specifically the Mallat decomposition and the Daubechies wavelets as in the work of Meng et al. [2003] and Qamra et al. [2005].
Figure 2.3: The wavelet decomposition of the Lena image. This representation is also commonly known as the Mallat diagram. The image has been post-processed for representation using a 1-level decomposition. The top left quadrant represents the lowest frequency band; the surrounding quadrants denote the high-level details.
Shape Features

Shape is another visual cue on which human perception relies; it can be used to identify objects of interest within an image. Researchers believe that human perception (or high level semantics) can potentially be mapped to different classes of interesting objects, so that they can improve image understanding in machines [Chen and Wang, 2002; Jing et al., 2003]. A CBIR system that retrieves images based on shape similarity between images is also known as a region-based system.

To extract shape information from an image, segmentation techniques [Shi and Malik, 2000; Gdalyahu et al., 2001] can be used to isolate regions of interest for further processing [Carson et al., 1999; Nixon and Aguado, 2002; Jing et al., 2003]. Image segments can be identified in one of the two ways [Ma and Manjunath, 2000]: using local discontinuities in image pixel intensity (or colour) values, which is essentially an edge detection process [Nixon and Aguado, 2002], or using homogeneous regions within the image, which can be identified by texture analysis (see Section 2.3.1). Popular edge detection techniques include the Sobel [Duda and Hart, 1973], Canny [Canny, 1986], Harris-function [Harris, 1993], Pre-witt [Prewitt, 1970], and the Hough transform [Duda and Hart, 1972]. These techniques, in general, use gradient and orientation information arising from lines and curves from local image structures — which we do not elaborate here — to identify shape boundaries; they can be useful for identification and categorisation of image shapes. Texture analysis, on the other hand, allows individual segments to be identified based on different texture patterns [Ma and Manjunath, 2000]. Once segmentation is performed on an image using these techniques, shape attributes — including circularity, orientation, and moment invariants [Leu, 1991] — can be extracted from these individual regions. Thus, the similarity between two images can be computed based on their shape attributes.

The work of Carson et al. [1997], known as Blobworld, shows that individual regions that are segmented from an image can be further processed to extract colour and texture information, so that they can be used to describe the segmented objects. Ma and Manjunath [1999] also show a similar approach that specifies features based on individually segmented regions in their NeTra CBIR system. The effectiveness of these approaches is limited by the inaccuracies of image segmentation [Jing et al., 2003]; where single objects can be occasionally over-segmented into multiple regions, and multiple objects can sometimes be falsely segmented as one. Wang et al. [2001], and Li et al. [2000] show that an integrated region-matching scheme can be used to reduce the negative impact of inaccurate segmentation
by allowing dynamic matching of one region to multiple regions using a form of similarity measure; their approach is known as the integrated region matching.

Intuitively, given a highly effective region-based retrieval system, it is conceivable that shape information can be used for near-duplicate detection, since near-duplicate images share identical objects or regions that can be used for matching. Nevertheless, existing region-based systems that are designed for CBIR are unsuitable for near-duplicate detection as the matching criteria caters for similarity and not equality; that means that the distinction between images that are visually similar and those that are near-duplicates is not clear. This reason has led us to focus on highly robust features such as interest points and local descriptors, which we discuss in Section 2.4; we do not further discuss shape features or region-based systems. In the following section, we describe some common distance metrics that are applied in CBIR.

### 2.3.2 Distance metrics

Once the image features are computed for a given collection, the pairwise similarity between two images can be calculated and quantified using their feature representation. Image features are typically stored as high-dimensional vectors, and therefore the comparison between two images can be determined using geometric distance measures. While similarity and distance are often used interchangeably, the latter is in fact used to measure the dissimilarity between two images. However, since the similarity can be derived from a inverse relationship of the distance — that is $(1 - \text{distance})$ — between two images [Santini and Jain, 1995], it can also be used to measure the similarity of two images. There are many distance measures that have been developed to assess image similarity; here we discuss some that are commonly applied in CBIR.

Given two feature vectors $\mathbf{I}_a$ and $\mathbf{I}_b$ with $N$ dimensions, the Minkowski [Naber, 1992] distance measure can be defined as:

$$D_{\text{mink}}(\mathbf{I}_a, \mathbf{I}_b) = \left( \sum_{n=0}^{N-1} |I_a[n] - I_b[n]|^r \right)^{\frac{1}{r}} \quad (2.9)$$

where $r$ is the order of norm, and values of $r = 1, 2, \infty$ are commonly used [Stricker and Orengo, 1995; Androutsos et al., 1998]. Setting $r = 1$ produces the $L_1$ distance (or the $L_1$ norm), commonly known as the Manhattan or city-block distance. The $L_2$ norm or Euclidean distance, which measures the shortest distance between two points, is computed when $r = 2$. 
The \( L_\infty \) norm can also be defined as \[\text{Stricker and Orengo, 1995}\]:

\[ D_\infty(\vec{I}_a, \vec{I}_b) = \max_{1 \leq n \leq N} |I_a[n] - I_b[n]| \]  \hspace{1cm} (2.10)

where the distance between two points is computed as the greatest distance between them in any dimension. Ma and Zhang [1998b] show that the \( L_1 \) and \( L_2 \) distance metrics yield good retrieval results for colour and texture feature vectors. These norms are arguably the most commonly used distance metrics for comparing two feature data in CBIR [Androutsos et al., 1998; Ma and Zhang, 1998a; Aslandogan and Yü, 1999; Smeulders et al., 2000], and also for indexing data in high-dimensional spaces [Weber et al., 1998; Böhm et al., 2001; Fonseca and Jorge, 2003].

Androutsos et al. [1998] show that the angular distance between two vectors can also be used to measure similarity between two images. The angular distance, which is commonly used in text retrieval for ranking documents, is also known as the Cosine measure [Baeza-Yates and Ribeiro-Neto, 1999]. The angular distance between two images can be computed as [Androutsos et al., 1998]:

\[ D_{\cos}(\vec{I}_a, \vec{I}_b) = 1 - \frac{2}{\pi} \cos^{-1} \left( \frac{\vec{I}_a \cdot \vec{I}_b}{|\vec{I}_a||\vec{I}_b|} \right) \]  \hspace{1cm} (2.11)

Androutsos et al. [1998] show that the angular distance yields a greater retrieval effectiveness than the \( L_1 \) and \( L_2 \) norms — both of which produce identical results — for comparing image colour histograms that are represented using the RGB colour space.

However, the recent work of Li et al. [2003] shows that, using images that are represented using colour and texture features that are stored as high-dimensional vectors, the \( L_1 \), \( L_2 \), and angular distance measures are inadequate for measuring image similarity as they often overlook similar images during retrieval. To address this shortcoming, they propose the use of dynamic partial functions (DPF) for measuring image similarity. The DPF is defined as:

\[ DPF(\vec{I}_a, \vec{I}_b) = \left( \sum_{\Delta d_i \in \Delta m} \Delta d_i^p \right)^{\frac{1}{p}} \]  \hspace{1cm} (2.12)

where

\[ \Delta d_i = |I_a[i] - I_b[i]| \]

\[ \Delta m = \text{smallest } m \Delta d_i \text{'s of } (\Delta d_1, \ldots, \Delta d_N) \]

and \( m \) is the number of features (vector elements) observed to have the smallest distances between two images. Using all features of an image for distance computation, by setting \( m = \ldots \)
2.4 Invariant interest points and local descriptors

The goal of near-duplicate image detection is to effectively and efficiently detect image pairs that are near-duplicates of each other. Due to digital editing operations, accidental or otherwise, these image instances are not always identical and may vary in dimension, scale, photometric, and imaging conditions. The image features that we have discussed so far are not designed to be robust against these conditions, but are instead developed for general detection of images that approximate the visual cues — such as colour, texture, and shape — of the human visual system.

In the next section, we discuss robust image features that can be used for near-duplicate detection.

2.4 Invariant interest points and local descriptors

The goal of near-duplicate image detection is to effectively and efficiently detect image pairs that are near-duplicates of each other. Due to digital editing operations, accidental or otherwise, these image instances are not always identical and may vary in dimension, scale, photometric, and imaging conditions. The image features that we have discussed so far are not designed to be robust against these conditions, but are instead developed for general detection of images that approximate the visual cues — such as colour, texture, and shape — of the human visual system.

Li et al. show that the DPF outperforms the $L_1$, $L_2$, and angular distance metrics in retrieving visually similar images that are represented by a combination of colour and texture features. Meng et al. [2003] and Qamra et al. [2005] further show that the DPF measure can be used to detect near-duplicate images. Hence, we use the DPF measure in this thesis for comparing colour and texture features, which we discuss further in Section 2.8.

In the next section, we discuss robust image features that can be used for near-duplicate detection.
image space. The notion of interest points can be traced back to the development of corner and region (or blob) detectors. Corner detectors [Moravec, 1981] were originally used to find robust image points for object tracking in robotics. Region detectors are those that detect stable image regions — often centred around well-defined points — that are more interesting (brighter or darker) than the surrounding area; this is typically indicated by a high gradient level in multiple orientations, which can be determined using local derivatives [Harris and Stephens, 1988].

Interest points are not widely applied in CBIR due to the computational complexity of identifying such features; depending on the complexity of the image, each can typically produce up to a few thousand interest points. Moreover they are unsuitable for detection of visually similar images as the matching of two interest points typically require a high degree of similarity in (more than one) local region that is based on a detected corner [Loupias and Sebe, 2000; Tian et al., 2001]; this is a relatively stringent requirement given that visually similar images do not necessarily share such properties.

Interest point detectors have been extensively studied over the years. They can be computed using methods that are based on contour [Pikaz and Dinstein, 1994; Mokhtarian and Suomela, 1998], intensity [Moravec, 1981; Harris and Stephens, 1988; Smith and Brady, 1997], or parametric model [Rohr, 1992; Baker et al., 1998]. The contour-based interest points require a contour structure (or shape) to be extracted from an image so that intersection or inflexion points can be determined. Intensity-based interest points are more straightforward in that they are computed using the grey-scale pixel values of an image. Parametric models fit template models (used for analysis) to an image structure, and are limited to certain interest point structures such as the L-corners [Rohr, 1992]. We focus our discussion mainly on the intensity-based interest points as they are the most commonly applied interest point detector in computer vision [Schmid et al., 2000], and are the most relevant to our work.

Interest point detectors typically operate in two main steps: first, they detect characteristic and robust image points or regions in an image, for which distinctive feature vectors — also known as local descriptors — are then computed to be used for matching. A match between two local descriptors can be estimated using geometric distance measures (such as those discussed in Section 2.3.2), where a match is deemed by a distance within a certain threshold [Ke and Sukthankar, 2004; Mikolajczyk et al., 2005; Bay et al., 2006]. Therefore, the correspondences between two images can be determined using the number of matching local descriptors. In computer vision literature, corner points and regions are also sometimes referred to simply as interest points or features. For consistency, we use the terms interest
points and regions interchangeably in this thesis, as most interest points possess a supporting region. In the following sections, we discuss the basic concepts of scale-space point detection, and discuss popular detectors and descriptors that are used for image matching.

**Scale-space point detection**

The idea behind robust interest point detectors is the detection of points or regions that are resistant to changes in image conditions, and that can be further processed for specific applications. Here, we discuss some fundamentals of scale-space theory that are relevant to the interest points detectors that are used in this thesis; theoretical foundations of scale-space theory are detailed in the works of Witkin [1983], Sporring et al. [1997], and Lindeberg [1998].

The majority of the scale-invariant interest point detectors and local descriptors are derived from scale-space theory [Witkin, 1983; Lindeberg, 1994; 1998] to address issues of scale representation in image data. Images are inherently multi-scale in nature, as they may contain multiple local structures; thus, it is difficult to determine a suitable scale at which to describe image data. For example, a picture of scenery may consist of multiple objects in the foreground and background, where each can be more appropriately described using its respective scale.

The concept of scale-space representation is to use a set of images to represent the different resolution levels [Witkin, 1983]. This enables image data to be described at each resolution (or scale), so that image data can be parameterised in scale; allowing fine-scale local structures of an image to be successively suppressed when the scale parameter is increased. This is useful for image analysis as it allows image data to be recorded at all scales; this representation also accounts for any unknown scale variations that may occur.

For an image \( I(x, y) \), a scale-space representation can be computed by performing a convolution operation using a Gaussian function with uniformly increasing scales of \( \sigma \):

\[
\begin{align*}
    f(x, y; \sigma) &= g(x, y; \sigma) * I(x, y) \\
    \text{where} \\
    g(x, y; \sigma) &= \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} 
\end{align*}
\]  

(2.13)  

(2.14)

and the initial condition of \( f(x, y; 0) = I(x, y) \) corresponds to the original image. The \( \sigma \) following a semicolon in each function indicates that the convolution operation is only performed over \( x, y \), where \( \sigma \) is the defined scale level. The discretised convolution operation
over a digital image using a Gaussian function is calculated as:

\[ f(x, y; \sigma) = \sum_{m} \sum_{n} g(m, n; \sigma) I(x - m, y - n) \]  

(2.15)

The convolution process is analogous to image Gaussian smoothing at a given scale level. It produces a blurring effect to an image, with the amount of blur controlled by adjusting the scale of the Gaussian function; an increased scale parameter gives the effect of a coarse representation of the image as only the obvious structures are apparent. Figure 2.4 shows the convolution operation with the Gaussian function using a series of increasing scale parameters. For simplicity, we henceforth use the notation \( x = (x, y) \), with "\( \ast \)" indicating a discrete convolution.

The Gaussian function has been shown to be a good choice for scale-space analysis as it reflects the neurophysiological responses of a human observer [Lindeberg, 1997; Witkin, 1983]. The Gaussian function is also widely accepted as a function that satisfies the key requirement of a multi-scale representation [Sporring et al., 1997; Romeny, 2003; Witkin, 1983]; that is, the structures at larger scales (coarse levels) can only be simplified versions of the structures at smaller scales (or finer levels), and that the method or function used to
suppress fine-scale structures should not create any accidental artifacts [Lindeberg, 1997].

Using the scale-space representation, differential functions can be computed at every scale as:

$$\partial_{x^i} f(x; \sigma) = (\partial_{x^i}(g(x; \sigma)) * I(x))$$  \hspace{1cm} (2.16)

where \( i \) denotes the \( i \)th order of derivative. The convolution is performed on the derivative of the Gaussian smoothing function; this process also constitutes the basic Gaussian derivative operator [Lindeberg, 1994]. When a combination of this operator and other differential functions — such as the Laplacian, difference-of-Gaussian, or Harris — are used, they can serve as a detector for a variety of features such as interest points, edges, blobs, and corners at multiple scales [Lindeberg, 1998]. Without scale-space detection, these differential functions are susceptible to scale variations as the size of the region (scale) is fixed, causing the operator response to be dependent on the relationship between the region and scale of the Gaussian function. However, these functions can be adapted to be scale-invariant when combined with the Gaussian derivative operator.

To achieve scale invariance in the derivatives, scale-normalisation is required due to a well-defined property of such an approach; the values of the derivatives diminish with scale, which means that the numerical value of the derivative responses in coarse scales (smoothed data) decreases. This is an undesirable quality for scale invariance [Lindeberg, 1998]. Scale-normalised derivatives using the Gaussian operator has been extensively studied [Lindeberg, 1994; 1997; 1998], and can be defined as:

$$\partial_{x^i}^{\text{NORM}} f(x; \sigma) = \sigma^i(\partial_{x^i}(g(x; \sigma)) * I(x))$$ \hspace{1cm} (2.17)

where the \( i \)th order (partial) derivatives are normalised with respect to the observed scale. Given two images \( I(x) \) and \( I'(x') \), where \( x' = sx \) of different scales can have derivatives relatable by:

$$\partial_{x^i} I(x) = \partial_{x'^i} I(x')$$ \hspace{1cm} (2.18)

where

$$\partial_{x'^i} I(x') = s^i \partial_{x^i} I(sx)$$ \hspace{1cm} (2.19)

And let us suppose that the scale of the Gaussian operator \( \sigma \) is normalised by the same scale factor, the relationship becomes:

$$\sigma^i \partial_{x^i}(g(x; \sigma) * I(x)) = s^i \sigma^i \partial_{x'^i}(g(x'; s\sigma) * I(x'))$$ \hspace{1cm} (2.20)

\hspace{1cm} (2.21)
Thus, the response of the normalised derivatives are:

\[
\partial_x^\text{NORM} f(x; \sigma) = \partial_{x'}^\text{NORM} f'(x; s\sigma)
\]  

(2.22)

We can see that the scale-normalised derivatives will yield the same numerical values at corresponding scales, provided that one fundamental requirement of scale-normalisation is satisfied; that is, if an image is rescaled by a scaling factor \(s\), then the scale at which the maximum derivative response is assumed must be relatable by the same scale factor (when measured in units of \(\sqrt{\sigma}\)) [Lindeberg, 1998]. As such, the amplitude of the normalised derivatives in the scale-space representation should correspond at points \((x; \sigma)\) and \((x'; \sigma')\) when \((x' = sx, \sigma' = s^2\sigma)\). With this property, it can be shown that given a scale-normalised differential function \(f(x; \sigma_n)\) at a particular scale, \(\sigma_n = s^n\sigma_0\), the derivative of the initial scale \(\sigma_0\) can be separated from its successive scales \(\sigma_n\) by a constant scale factor [Lindeberg, 1998; Mikolajczyk and Schmid, 2001].

Thus far, we have discussed an approach for representing images at multiple scales such that they can be related in a scale-invariant manner. While representing images in multiple scales is advantageous, it is conceivable that a suitable local scale may be required at some point for further analysis. Lindeberg [1998] showed that the scale-space representation can be designed such that the image representations are made invariant to scale (or automatic scale selection). This is achieved by selecting a scale based on local maxima or minima — local extrema — over scale of multiple scale-normalised derivatives; that is, the scale level is selected when the differential functions attain a local maximum with respect to the scale.

Once the differential function responses are calculated for a set of scales, local extrema can be detected simultaneously over scale and spatial location by sampling all pixel points (or intensity values) in an image, resulting in three-dimensional scale-space extrema [Lindeberg, 1998; Schmid et al., 2000] The local extrema of the function responses (maxima or minima) over scale are those that correspond to the local image structures that are centred around a point [Lindeberg, 1998]. The scales at which the extrema of these responses exist are also known as characteristic scales, as they obey scale invariance under rescaling of image patterns [Mikolajczyk and Schmid, 2002].

For digital images, the detection of scale extrema can be defined in discrete scale-space as:

\[
F_{\text{max}}(x_0, y_0, \sigma_0) \geq F_{\text{max}}(x_{0+i}, y_{0+j}, \sigma_{0+j}), \text{ for } (i, j) \in \{-1, 0, 1\}
\]

(2.23)

where \(\sigma\) can be sampled within a range of scale levels whereby the ratio between \(\sigma_j\) and \(\sigma_{j+1}\) remains approximately constant. \(F_{\text{max}}(.)\) denotes the scale-space maxima of the scale-
2.4. INVARIANT INTEREST POINTS AND LOCAL DESCRIPTORS

Figure 2.5: The three-dimensional scale-space axis, where the black dot represents the scale-space extremum when its intensity value is greater than all its 26 neighbouring points; 8 in local scale, and 18 neighbouring points in scales above and below.

normalised derivative response of the image, as shown in Figure 2.5. Note that this process can also be extended for minima detection. As shown in this figure, the scale-space extrema can be obtained by comparing a sample point to its eight neighbours — in a $3 \times 3$ region — and nine neighbours in the scales above and below it; a local extremum or interest point is detected if the sample point has a value larger or lower than all of its neighbouring points [Lowe, 2004].

This process is commonly known as scale-space extrema detection, and is the basis of popular scale-invariant interest point detectors. Such an approach was first introduced by Crowley [1981] where he described a pyramidal difference-of-Gaussian filtering scheme [Crowley and Parker, 1983; 1984] to detect the presence of a scale-space extremum when its value exceeded a given threshold. Existing scale-invariant interest point detectors are similar to this approach, differing mainly in the differential function used for building the scale-space representation of an image.

So far, we have discussed only the properties of scale invariance but not those of true affine invariance; that is, invariance to orthographic projection — or variations in a two-dimensional image as a result of a projection from a three-dimensional object. Affine variations of image correspondences in computer vision typically refer to changes in viewpoint of the specific captured scene [Tuytelaars and van Gool, 2000; Schaffalitzky and Zisserman, 2002]. Thus scale variation is a restricted form of affine variation, in that scale changes can be regarded as a consequence of distance variations between the viewer and the captured scene.
Lowe [2004] notes that scale-invariant detectors — specifically the SIFT detector (see Section 2.4.1) — can tolerate minor affine distortions where viewpoint changes can vary as much as 50°, but typically fail when there are extreme affine variations [Lowe, 2004; Mikolajczyk and Schmid, 2004]. He also notes that while truly affine invariant detectors have not been proposed, there has been some work that extends from scale-invariant detectors by resampling images from local affine projections [Tuytelaars and van Gool, 2000; Schaffalitzky and Zisserman, 2002], which are typically more expensive to compute, and are also more sensitive to noise [Lowe, 2004]. Thus we focus on scale-invariant detectors and not affine invariant detectors in this thesis. We show in later chapters that the property of scale-invariance alone is sufficient for the application as near-duplicate images typically have limited affine variations.

2.4.1 Existing interest point detectors and local descriptors

As mentioned previously, interest point detectors are typically comprised of two functions: interest point detection, and construction of interest point representation using local descriptors. Many existing interest point detectors use differential functions based on the Gaussian derivative operator, where they differ mainly in the differential functions applied. When computed using a three-dimensional scale-space representation, a scale-invariant interest point is selected if an extremum (maximum or minimum value) can be detected simultaneously in the scale and spatial axes.

Given a digital image, a naïve approach is to examine each pixel (using the intensity value) within the image and assign a scale extremum if it exists in the sampled scales. Mikolajczyk and Schmid [2001] show that this approach yields unstable results; that is, a large number of corresponding points between two images of different scales do not have interest points with a scale proportional to scale factor between the images. Thus differential functions are useful as they permit a more stable localisation of the interest points in both scale and space [Lindeberg, 1998; Lowe, 1999; Mikolajczyk and Schmid, 2001]. For instance, Lindeberg [1998] demonstrates that the Laplacian-of-Gaussian can be used to localise the scale-space extrema, while Lowe [1999] detects interest points using the difference-of-Gaussian. Mikolajczyk and Schmid [2001] further show that a combination of Harris function and Laplacian-of-Gaussian — known as Harris-Laplace — can similarly be applied for this task. Mikolajczyk and Schmid [2004], and Lowe [2004] also demonstrate that the trace and determinant of Hessian can be applied to localise interest points with more stability. Each function has its advan-
tages and disadvantages, and the choice of which is dependent on the task at hand. Here, we do not discuss each of these methods in detail as they are beyond the scope of this thesis; instead we discuss their application in popular interest point detectors and local descriptors.

Local descriptors can be seen as separate from interest point detectors, in that descriptors can be generated from a localised region (centred around an interest point) that is detected by any interest point detector. Local descriptors are represented by feature vectors that are computed from these detected regions, such that the similarity of two interest regions can be calculated and quantified numerically by their vector distances. This can be computed using common distance measures such as the $L_1$ and $L_2$ measures, as described in Section 2.3.2. Here, we describe four local descriptors, and their corresponding interest point detectors, that are commonly used in object-recognition and scene matching, such as the Scale Invariant Feature Transform (SIFT), PCA-SIFT, Speeded-Up Robust Features (SURF), and the Gradient Location and Orientation Histogram (GLOH).

**Scale Invariant Feature Transform (SIFT)**

The Scale Invariant Feature Transform or SIFT that was developed by Lowe [2004] is arguably the most well-known local descriptor in the computer vision and object recognition domain. The SIFT detector uses the difference-of-Gaussian function to build the scale-space representation, and subsequently for interest point detection; each detected interest points is also referred to as a *keypoint* [Lowe, 1999; 2004]. Here we describe interest point detection using the difference-of-Gaussian as proposed by Lowe [1999] in some detail as it is relevant to our work.

Lowe [1999] shows that the difference-of-Gaussian can be efficiently computed by taking the difference between the two images, each of which is computed by a Gaussian operator at a different scale. The difference-of-Gaussian can be computed as:

$$
\text{DoG}(x) = (g(x, k\sigma) - g(x, \sigma)) \ast I(x)
$$

$$
= f'(x', k\sigma) - f(x, \sigma)
$$

(2.24)

where $x = (x, y)$, $k$ is the constant scale factor, and $f'(.)$ and $f(.)$ denote the image representations built using the Gaussian operator that are separated by this scale factor. Once the scale-space representation is computed, the image is further downsampled by a factor of 2 and the process is repeated; the set of sampled images within a given image size is known as an *octave*. This process is depicted in Figure 2.6. Lowe further shows, empirically, that a
Figure 2.6: The process of deriving the difference-of-Gaussian (DoG) images by getting the difference between the Gaussian smoothed images separated by a constant scale factor. The process is repeated for each octave, on an image that is half the size of the previous octave.

Scale factor of $k = 2^{\frac{1}{3}}$ yields good results; this also means that each octave is processed by the Gaussian operator using three different scales that are separated by a constant factor. As shown earlier in Figure 2.5, using this representation, points are selected only if their intensity value is greater or smaller than all the neighbouring pixels with a larger and smaller scale. Brown and Lowe [2002] show that, using the location of these selected points, the quadratic Taylor expansion [Marsden and Tromba, 2003] can be used to accurately localise the extrema at the scale-space. Briefly, the quadratic Taylor’s expansion is a mathematical concept that can be used to linearly approximate a particular function in a neighbourhood of a given point. Brown and Lowe [2002] show that the scale-space extrema can be computed using:

$$\text{DoG}(x) = \text{DoG}(x_0) + \partial_x \text{DoG}(x_0)x + \frac{1}{2}x^2 \partial_{xx} \text{DoG}(x_0)$$  \hspace{1cm} (2.25)

where $x_0$ is the sampled point, and $x = (x, y, \sigma)$ is the offset from the sampled point. Thus the interpolated extremum is attained by setting the derivatives of the quadratic Taylor’s expansion to zero, which is calculated to be [Brown and Lowe, 2002]:

$$\hat{x} = (\partial_{xx} \text{DoG}(x))^{-1} \partial_x \text{DoG}(x)$$ \hspace{1cm} (2.26)

We note that the notation used here is different from those of Brown and Lowe [2002] and Lowe [2004] as they have expressed the expansion using the matrix form. The Taylor’s
expansion using the matrix form is further detailed in Equation A.1 (page 195). Lowe [2004] further improved the reliability of the difference-of-Gaussian function by rejecting interest points that have intensity values \((\text{DoG}(\hat{x}))\) less than 0.03 (intensity values are in the range of \([0,1]\)). This is attained by evaluating the function value using Equations 2.25 and 2.26.

In addition to rejecting interest points with low intensity values, Lowe [2004] shows that the trace and determinant of the Hessian matrix — which are computed on the difference-of-Gaussian — can be used to reject interest points that are selected due to edge responses, and are therefore unstable to small amounts of noise. The reason is that a poorly defined interest point of the difference-of-Gaussian function yields a large principal curvature value — which can be computed using the Hessian matrix — across the observed edge, and a small value across the perpendicular direction. The principal of curvature of the difference-of-Gaussian is calculated using the location (at the observed scale) of a given interest point as [Lowe, 2004]:

\[
H = \begin{bmatrix}
\partial_{xx} \text{DoG}(x,y) & \partial_{xy} \text{DoG}(x,y) \\
\partial_{xy} \text{DoG}(x,y) & \partial_{yy} \text{DoG}(x,y)
\end{bmatrix}
\] (2.27)

The eigenvalues of the Hessian matrix reflect the principal of curvature of the difference-of-Gaussian function [Lowe, 2004], where the trace and determinant — or sum and product of the eigenvalues [Marsden and Tromba, 2003] — can be defined as:

\[
\text{Tr}(H) = \partial_{xx} \text{DoG}(x,y) + \partial_{yy} \text{DoG}(x,y) \quad (2.28)
\]

\[
\text{Det}(H) = \partial_{xx} \text{DoG}(x,y)\partial_{yy} \text{DoG}(x,y) - (\partial_{xy} \text{DoG}(x,y))^2 \quad (2.29)
\]

such that the ratio of principal curvature — which is detailed in the work of [Lowe, 2004] — can be efficiently calculated using:

\[
\frac{\text{Tr}(H)^2}{\text{Det}(H)} = \frac{(r + 1)^2}{r} \quad (2.30)
\]

where \(r = 10\) has been empirically observed to effectively reject poorly localised interest points [Lowe, 2004].

While the interest points that are derived from this function are rotation variant, Lowe [1999] shows that a detected interest region can be normalised to attain rotation invariance. This is achieved by rotating a region — which is centred around the point being considered — to the dominant gradient orientation sampled within the neighbourhood of the same region.
The gradient magnitude and orientation can be calculated as [Lowe, 1999]:

\[
m(x, y) = \sqrt{(\text{DoG}(x + 1, y) - \text{DoG}(x - 1, y))^2 + (\text{DoG}(x, y + 1) - \text{DoG}(x, y - 1))^2}
\]
\[
\theta(x, y) = \tan^{-1}\left(\frac{\text{DoG}(x, y + 1) - \text{DoG}(x, y - 1)}{\text{DoG}(x + 1, y) - \text{DoG}(x - 1, y)}\right)
\]

where \(m(x, y)\) and \(\theta(x, y)\) are calculated using the pixel differences of the smoothed Gaussian image with the scale closest to that of the detected interest point [Lowe, 2004]. The dominant orientation can be observed by computing a histogram with 36 bins that represent the 360° range of orientation. The histogram is formed by sampling the computed orientations from the region centred around the interest point, and each sample is further weighted by the gradient magnitude. The dominant orientation is indicated by the peak in the histogram, and multiple dominant orientation may be created if there exist any orientations (as observed in the histogram) that are not separated from the most dominant one by 20% [Lowe, 2004]; thus there can be two identical interest points with different orientation. To achieve rotation invariance, once the interest points (or keypoints) and their dominant orientations are detected, the region surrounding each interest point is sampled using the Gaussian smoothed image that is selected by the scale of the interest point. The coordinates of the sampled region, and the previously computed gradient magnitudes and orientations, can be rotated relative to the dominant orientation of the interest point being considered.

The SIFT descriptor is generated by a three-dimensional histogram that records gradient and orientation information for the region centred around the considered interest point. The spatial location of the sampled region is then quantised into a \(4 \times 4\) grid, and the gradient angles for that region are quantised into 8 orientations; this results in a \(4 \times 4 \times 8 = 128\)-element feature vector. For invariance against illumination variations, this vector is normalised using the square root of the sum of the squared components [Lowe, 2004].

While the SIFT descriptor can be used to characterise regions derived from other interest point detectors, the original approach uses the difference-of-Gaussian function. For simplicity, we refer to the original approach as the SIFT feature; it is used as a baseline approach in this work. We implement the SIFT keypoints as shown in the work of Lowe [2004], where the SIFT local descriptors are generated using the software provided by Lowe.\(^1\)

Gradient Location and Orientation Histogram (GLOH)

The Gradient Location and Orientation Histogram (GLOH) is a local descriptor that improves the robustness and distinctiveness of the SIFT descriptors [Mikolajczyk and Schmid, \(^1\)http://www.cs.ubc.ca/~lowe/keypoints/]
2.4. INVARIANT INTEREST POINTS AND LOCAL DESCRIPTORS

While the GLOH can be used to in combination with a variety of interest point detectors, Mikolajczyk and Schmid [2003] show that a combination of the Hessian-affine [Mikolajczyk et al., 2005] regions that are described using the GLOH yields the highest effectiveness — outperforming the SIFT descriptors — for the task of identifying corresponding regions between images with variations of viewpoint and angle.

To compute Hessian-affine regions, the Hessian matrix can be used to find the strong responses from the second derivatives of the Gaussian derivative operator [Mikolajczyk et al., 2005]:

\[
H = \begin{bmatrix}
\frac{\partial^2 f}{\partial x^2}(x, y; \sigma) & \frac{\partial^2 f}{\partial x \partial y}(x, y; \sigma) \\
\frac{\partial^2 f}{\partial x \partial y}(x, y; \sigma) & \frac{\partial^2 f}{\partial y^2}(x, y; \sigma)
\end{bmatrix}
\]  

(2.33)

where the second order derivatives are computed at several scales using the Gaussian operator, as defined in Equation 2.16. Unlike the scale-space extrema detection in the SIFT detector — where the extrema in scale and space are localised using the Taylor quadratic expansion — the extrema in the spatial location are selected using the Hessian matrix, and the characteristic scale is selected based on the maximum response of the scale-normalised Laplacian-of-Gaussian [Marsden and Tromba, 2003; Mikolajczyk and Schmid, 2004], which is essentially the trace of the above-defined Hessian matrix; this is computed as: [Mikolajczyk and Schmid, 2004]:

\[
\text{LoG}(x, y; \sigma) = \sigma^2 \left| \frac{\partial^2 f}{\partial x^2}(x, y; \sigma) + \frac{\partial^2 f}{\partial y^2}(x, y; \sigma) \right|
\]  

(2.34)

such that the scale can be selected when the maximum response from the Laplacian-of-Gaussian is obtained [Mikolajczyk and Schmid, 2004]. As discussed earlier, affine invariance can be achieved by resampling scale-invariant points in local affine projections (see Section 2.4); this is similarly applied in the Hessian-affine regions for affine and rotation invariance using a process of affine normalisation. The work of Mikolajczyk et al. [2005] contains a detailed description of the affine normalisation process.

For each detected region, GLOH uses a log-polar grid [Schwartz, 1977] with three different radii and eight angular directions (where only two radii are computed with angular directions) to map the region. Each of the \(2 \times 8 + 1 = 17\) bins is further quantised using 16 levels resulting in a histogram of 272 bins; Principal Component Analysis (PCA) [Jolliffe, 1986] — a dimensionality reduction technique — is then used to reduce the histogram to a 128-element feature vector.

As discussed earlier, while we are not particularly concerned with affine invariance in this work, we also use the Hessian-affine regions as a baseline, since they are used as supporting
regions for the GLOH descriptors. To generate the Hessian-affine regions and the GLOH descriptors, we use the software provided by Mikolajczyk.\(^2\)

**Speeded-Up Robust Features (SURF)**

In recent work, Bay et al. [2006] describe an approach to detect interest points and local descriptors known as *Speeded-Up Robust Features* (SURF). They show that the determinant of Hessian matrix can be used for selection of interest points in both scale and location; their approach is similar to the approach by Lowe [1999] and that of the SIFT detector [Lowe, 2004]. They also show that this method is more efficient to compute than those based on combined or multiple functions such as the Hessian-affine function [Bay et al., 2006].

Bay et al. [2006] show that the determinant of Hessian, as defined in Equation 2.33, can be used to approximate the second order derivatives of the Gaussian operator, such that they can be used to derive Gaussian filters at multiple scales. These filters can be then be iteratively convolved with an image to build the scale-space representation. The maxima of the determinant-of-Hessian for an interest point is then selected by using the same approach as the SIFT detector, by comparing the neighbouring 26 pixels within the scale of the considered point and those with greater and smaller scales [Lowe, 2004]; The maxima of the determinant of Hessian are further localised using the quadratic Taylor’s expansion, as used in the SIFT detector by Brown and Lowe [2002] and Lowe [2004]. For orientation invariance, the dominant orientation is determined from the region surrounding the detected interest points, similar to the approach of the SIFT detector. This approach of interest point detection is also known as the *Fast-Hessian* method.

To compute the SURF local descriptors a 20 × 20 patch surrounding a detected interest point is first divided into 16 subregions. The Haar wavelet responses of the horizontal and vertical directions (\(d_x\) and \(d_y\)) for each subregion are then computed. Next, the sum of the wavelet responses (\(\sum d_x\) and \(\sum d_y\)) over each subregion is calculated, along with the magnitude (\(\sum |d_x|\) and \(\sum |d_y|\)) of both horizontal and vertical responses, each computed individually. Using this approach, a local descriptor of 4 × 4 × 8 = 128 dimensions is generated for each interest region. Although smaller descriptors (36 and 64 dimensions) can be computed by varying the size of the subregions, Bay et al. [2006] show that descriptors of 128 dimensions have the best performance; they show that this descriptor can be computed efficiently, with accuracy comparable to both SIFT and GLOH in identifying correspondences.

\(^2\)http://www.robots.ox.ac.uk/~vgg/research/affine
amongst images that are captured with variation of viewpoints angles, scale, and lighting conditions.

We use the SURF descriptors that are generated by the software provided by Bay et al.\(^3\) as a baseline.

### PCA-SIFT

The PCA-SIFT is a local descriptor developed by Ke and Sukthankar [2004]. The PCA-SIFT descriptor extends the SIFT method in that it uses the same information as the SIFT-detected interest points (or keypoints); that is, the scale, location, and (dominant) orientation.

As with the SIFT descriptors, the PCA-SIFT descriptors are generated for multiple interest points when there are multiple dominant orientations. The PCA-SIFT descriptor uses the vertical and horizontal gradients of the regions that are centred around the detected interest points. The \(x\) and \(y\) gradients are sampled using a \(41 \times 41\) patch, resulting in a \(2 \times 39 \times 39 = 3042\)-element vector. This vector is then projected onto 36 dimensions using the Principal Component Analysis (PCA) dimensionality reduction technique [Jolliffe, 1986]. When compared to the SIFT descriptors, the PCA-SIFT descriptors are considerably more compact. It has also been shown to be highly efficient while achieving comparable effectiveness in identifying image correspondences [Ke and Sukthankar, 2004]. The PCA-SIFT descriptors was also shown to be the most distinctive when compared with the other local descriptors such as GLOH, SIFT, and SURF [Mikolajczyk et al., 2005].

In this thesis, we use the PCA-SIFT descriptors that are generated using the SIFT-detected interest points; we use the other local descriptors as baselines. We implement the interest point detection using the difference-of-Gaussian function and the PCA-SIFT local descriptors according to the framework as shown in the work of Ke and Sukthankar [2004] and Ke et al. [2004].

#### 2.4.2 Discussion

Thus far, we have discussed several interest point detectors and local descriptors that have been applied in computer vision. These have been studied and compared for object recognition and scene matching tasks [Schmid et al., 2000; Mikolajczyk and Schmid, 2003; 2004; Mikolajczyk et al., 2005; Bay et al., 2006]. The evaluation of these interest point detectors for computer vision tasks generally involves the identification of correspondences between

\(^3\)http://www.vision.ee.ethz.ch/~surf/download.html
an original image and a set of corresponding images that are rotated, JPEG-compressed, blurred, and captured under various focal lengths (zooming), viewpoint, or lighting conditions. The effectiveness of the interest point detectors and local descriptors are evaluated based on the reproducibility of the interest points (characterised by the descriptors) between the original image and its corresponding images.

However, the choice of an interest point detector and local descriptors for image representation in near-duplicate detection is not immediately apparent, and our choice of the difference-of-Gaussian function (with the application of the PCA-SIFT local descriptors) has not been justified. Mikolajczyk and Schmid [2004] show that the difference-of-Gaussian does not have a rate of repeatability as high as the Hessian-affine detectors for images of viewpoint changes. Nevertheless, the difference-of-Gaussian function has been shown to yield a rate of repeatability comparable to many existing interest point detectors [Mikolajczyk and Schmid, 2001; 2004; Bay et al., 2006], while outperforming them considerably in terms of computational speed [Mikolajczyk and Schmid, 2004; Bay et al., 2006]. Zhao et al. [2006] also show that the difference-of-Gaussian function to be comparable to the Hessian-affine in detecting robust interest points. The difference-of-Gaussian function is also used in the SIFT detector [Lowe, 2004] — a widely applied interest point detector in computer vision and object recognition [Lowe, 1999; Mikolajczyk and Schmid, 2003; Ke and Sukthankar, 2004; Ke et al., 2004].

In this thesis, we use the difference-of-Gaussian function as shown in the work of Lowe [2004]. For our application, we believe that this is an acceptable trade-off as near-duplicate images typically possess less variance than those considered in stereo matching and robust object-recognition; we do not consider viewpoint invariance to be vital for near-duplicate detection. While detecting images of the same subset, but with viewpoint changes may have interesting applications — such as panoramic stitching [Brown and Lowe, 2003] and scene tracking [Skrypnyk and Lowe, 2004] — they have limited use for near-duplicate matching. We believe that images of substantial difference in viewpoint are generally not regarded as a near-duplicate, since they cannot be considered as redundant information. It is also arguable whether images of such substantial viewpoint changes infringe on copyright, as these changes usually render an image to be substantially different from the original version.

Lowe [2004] also observed that the advantages provided by affine-invariant detectors do not necessarily outweigh the computational cost they incur, especially for substantial image collection sizes. Additionally, Ke et al. [2004] have shown, via thorough experimentation on a wide array of image alterations — including severe scaling, cropping, or shearing — that
the difference-of-Gaussian detector and the PCA-SIFT descriptors can be used to effectively identify correspondences between the original images and their near-duplicates, and that these descriptors can tolerate minor affine distortions. Our choice of PCA-SIFT local descriptors is hence justified by the fact that it is effective for this application, and at the same time, more efficient and compact than other descriptors. Thus the difference-of-Gaussian and the PCA-SIFT descriptors are suitable candidates for our application, as we later demonstrate in Chapter 4 (page 97).

2.5 Image indexing

Once feature data is extracted from an image, it can be used to compare two images. A naïve approach is to compare each image feature in turn and derive — using a distance or similarity measure — an estimate of the closeness of the two images. This approach, also known as a linear sequential search or a brute-force search, is impractical for large image collections as the search time is largely dictated by the size of the collection. A common approach to facilitating efficient search and comparison within large image collections is to index their feature vectors using indexing techniques, such that they can be used to organise the feature data more efficiently [Böhm et al., 2001].

Visualising an image as a single vector in a high-dimensionality vector space, a collection of images can be regarded as a set of \( N \) points \( P = \{ p_0, \ldots, p_N \} \) in a \( d \)-dimensional space \( \mathbb{R}^d \), where the estimated similarity between two images is correlated to the computed distance (in this space) between their vectors. Each image can be represented using one or more feature vectors; using features such as colour or texture, an image is typically represented using a single feature vector, whereas, for more complex local features such as local descriptors, multiple feature vectors are used. For images represented by multiple vectors, the distance between an image pair can be similarly calculated using the aggregated distance between multiple points in a high-dimensional space, instead of using only the distance between a single point. Thus, a small distance between two vectors reflects a high similarity between the overall image representations; and in the case of multiple feature vectors, a high similarity is indicated with a small aggregated distance. Here, we discuss the basic indexing methods but focus on the Locality Sensitive Hashing [Gionis et al., 1999] and Redundant Bit Vectors [Goldstein et al., 2004], as these are the most relevant to our work. For this discussion, each image is assumed to be a single feature vector; we describe indexing of images with multiple features vectors in Section 2.5.2.
Given large collection sizes, and images represented using high-dimensional feature vectors — where tens to hundreds of dimensions are typical — indexing techniques are often reduced to linear sequential search, where all points \( p \in P \) have to be evaluated; this is commonly known as the \textit{curse of dimensionality} [Castelli, 2001]. To reduce the dimensionality of the feature vector sets, \textit{dimensionality reduction} techniques such as PCA [Jolliffe, 1986] and factor analysis [Fodor, 2002] are applied. Even so, the dimensionality of feature vectors remain high. To facilitate efficient retrieval, feature vectors are indexed using high-dimensionality indexing structures [Castelli, 2001].

High-dimensionality indexing schemes are generally designed to support the different forms of queries that are commonly employed in content-based search. The main query categories are [Castelli, 2001; Faloutsos et al.]:

**Range search** A range search allows a query to be specified based on a given range; for example if a \( d \)-dimensional point \( p(x_0, \ldots, x_d) \); for \( p \in P \) represents features of an image, a range \( \{x_0 < t_0, \ldots, x_d < t_d \} \) — where \( t \) is the specified threshold — can be specified such that only points within this range are returned. This form of similarity query is most suitable for \textit{query-by-features}, where features are specified separately [Faloutsos et al.].

**\( k \)-Nearest-Neighbour (\( k \)-NN) search** This is arguably the most common similarity query employed in CBIR. This form of query returns a list of \( k \) answers most similar to a given query, and is typically applied with the query-by-example method [Faloutsos et al.]. The exact nearest-neighbour search is a form of the \( k \)-NN search where only the closest point \( p \in P \) to the query example \( q \in \mathbb{R}^d \) is returned (\( k = 1 \)). The \( k \)-NN search is an instance where \( k > 1 \) points are returned; hence all points \( p \in P \) need to be evaluated to derive a list of points ranked by their distance to the specified example. This process is exhaustive in that the distance to every point in the collection is computed with respect to a given query.

**Approximate-Nearest-Neighbour (ANN) search** The approximate nearest-neighbour search is a less stringent form of the nearest-neighbour search, in that the returned answers are only approximated to be nearest-neighbours of the specified query. Given an example query \( q \in \mathbb{R}^d \), find a point \( p \in P \) such that for all \( p' \in P \), \( d(p, q) \leq (1 + \epsilon)d(p', q) \); point \( p \) is known as the \( \epsilon \)-approximate nearest neighbour of \( q \) for a small constant \( \epsilon \), and \( d(\cdot, \cdot) \) denotes the distance between two points. This is an efficient approach for high-dimensionality indexes as it does not compute all points exhaustively;
this efficiency makes it attractive for large collections [Böhm et al., 2001]. Moreover, the exactness of the \( k \)-NN search is typically not required for most practical applications [Indyk and Motwani, 1998]. Thus, we apply the ANN indexing methods for near-duplicate images in this thesis.

High-dimensionality index structures include the \( R \)-tree [Guttman, 1984; Wu and Bretschneider, 2004], \( TV \)-tree [Lin et al., 1994], \( k\)-\( d \)-\( B \)-tree [Egas et al., 1999], \( SS \)-tree [White and Jain, 1996b], \( SR \)-tree [White and Jain, 1996a; Katayama and Satoh, 1997], and the \( X \)-tree [Bertsch et al., 1996]. Each index structure has advantages and disadvantages with regards to search performance, index size, maintenance cost, and supported query types [Weber et al., 1998; Castelli, 2001]. The \( k\)-\( d\)-\( B \)-tree is known to have limited performance for range and NN search, while the \( R \)-tree family (including the improved \( R^* \)-tree [Beckmann et al., 1990]) has better space utilisation [Katayama and Satoh, 1997; Böhm et al., 2001]. The \( SR \)-tree, \( SS \)-tree, and the \( TV \)-tree are limited to NN and ANN search as they do not support range queries. Katayama and Satoh [1997] have also shown the \( SR \)-tree to yield better performance than the \( R^* \)-tree and the \( SS \)-tree.

Although most of the index structures that support \( k \)-NN search can be modified for ANN search, Indyk and Motwani [1998] propose the locality sensitive hashing technique that supports ANN search, which exhibits superior efficiency to predominant index structures such as the \( SR \)-tree [Gionis et al., 1999]. The LSH index has been widely adopted for indexing high-dimensional data in fields ranging from text information retrieval [Haveliwala et al., 2000; Stein, 2007] to multimedia retrieval [Bawa et al., 2005; Chum et al., 2007]. In recent work, Goldstein et al. [2003] propose the redundant bit vectors, which support ANN search; the RBV scheme has been shown to yield better performance than the LSH index in terms of memory requirements and space utilisation [Goldstein et al., 2004], although its effectiveness has not been compared in depth to LSH.

In Chapter 4, we apply the LSH indexing scheme for near-duplicate image detection, and extend the RBV index for the same purpose. In the following sections we discuss these two index structures in detail.

### 2.5.1 Locality Sensitive Hashing (LSH)

As discussed previously, given a set of points \( P \), the distance between two points in approximate nearest-neighbour search can be defined as [Gionis et al., 1999]:

\[
D(q, p) \leq (1 + \epsilon)D(q, P) \tag{2.35}
\]
where $q$ is the query point, and $D(q, P)$ is the distance of $q$ to the closest point in $P$. This limits the search to only points with distances no greater than $(1 + \epsilon)$ times the distance of the nearest point to $q$. LSH uses a family of hash functions to ensure that the probability of collision of two points is closely related to the distance between them; two points will share a hash value (also known as a bucket) with a probability commensurate to the distance between them in the $\mathbb{R}^d$ space.

In general, a family of hash functions $H = \{h : P \rightarrow Q\}$ — where a set of points in $P$ is mapped to a set $Q$ using function $h$ — is known as $(r_1,r_2,p_1,p_2)$-sensitive [Indyk and Motwani, 1998] for a similarity or distance function $D(.,.)$ between the points in $P$ for any $q,p \in P$:

- if $p \in \beta(q,r_1)$ then $Pr_H[h(q) = h(p)] \geq p_1$
- if $p \notin \beta(q,r_2)$ then $Pr_H[h(q) = h(p)] \leq p_2$

where $\beta(q,r)$ denotes the set of points with $D(q,p) \geq r$. Where a point $p$ is within a radius of $r_1$ from the point $q$ (or query), the probability that these two points are hashed to the same value exceeds that of $p_1$. Conversely, if point $p$ is not within the radius of $r_2$ from point $q$, then the probability that these two points collide does not exceed the rate of $p_2$. Additionally, the property of locality-sensitivity for a distance function is only satisfied when $p_1 > p_2$ and $r_1 < r_2$ (or $r_1 > r_2$ for a similarity measure) [Indyk and Motwani, 1998].
A family of hash functions can be defined as $G = \{g : P \rightarrow Q^k\}$ using a parameter $k$ and $L$, such that:

$$g_i(p) = (h_1(p), \ldots, h_k(p))$$  \hspace{1cm} (2.36)

for $i = 1, \ldots, L$, and $h_1 \ldots h_k$ chosen randomly from the family function $H$ with replacement [Indyk and Motwani, 1998; Gionis et al., 1999; Datar et al., 2004]. The parameter $k$ is used to maximise the probability of collision between two points that are close to each other, while minimising the chances of collision between two points that are not. Thus, the probability of collision within the index is dictated by the $k$ random bits that are selected to create the hash. The $L$ parameter determines the fraction of false negatives, where the use of $L$ independent indexes increases accuracy as those that are missed by one index may be retrieved by other indexes [Indyk and Motwani, 1998; Gionis et al., 1999; Datar et al., 2004].

Gionis et al. [1999] show that this family of functions can be efficiently computed using a Hamming space $H^d$ [Gionis et al., 1999; Bawa et al., 2005] for $d$ dimensions, whereby each $d$-dimensional vector $p(x_1, \ldots, x_d)$ can be mapped to a Hamming cube $H^{d'}$ with $d' = Cd$ (where $C$ denotes the largest coordinate in $P$), transforming vector $p$ to a binary Hamming string $p'$:

$$p' = Unary_C(x_1) \ldots Unary_C(x_d)$$  \hspace{1cm} (2.37)

where $x_i$ is replaced with its unary representation ($x_i$ ones followed by $C - x_i$ zeroes). Since the Hamming distance reflects the number of bits that differ between two Hamming strings, the $L_1$ distance between two embedded points is preserved by this transformation to a binary vector [Gionis et al., 1999].

To transform a given vector (or point) $p$ to a binary vector of $p'$, Gionis et al. [1999] show that $L$ subsets of $\{I_1, \ldots, I_L\}$ of $\{1 \ldots d'\}$ can be created, such that $k$ elements are selected uniformly at random from $\{1 \ldots d'\}$ for each $I$. Thus, a function $g_i(p)$, for $i = 1, \ldots, L$, can be derived by a projection of vector $p$ onto a set $I$, where $I$ consists of $k$ coordinates that are sampled randomly with replacement from $\{1, \ldots, d'\}$. Therefore, the binary vector $p'$ can be calculated from the projection of vector $p$ onto $I$, where the coordinate positions from $I$ that correspond to each dimension of $p$ (from $1, \ldots, d$) can be selected, and so the bits in those positions can be concatenated to form $p'$. Using this approach, the LSH function can be defined as $g(p)$ for a set of points $p \in P$, where there are a total of $L$ LSH functions that are selected from the $H^{d'}$ family; this approach is shown to be equivalent to computing Equation 2.36 [Gionis et al., 1999]. As the number of buckets can be high depending on the
Algorithm 1 Summary for building the LSH index.

Require: Database of $N D$-dimensional vectors $x$, pre-defined $L$ parameter.

Embed (transform) all vectors into the Hamming Cube.

for $i = 0$ to $L - 1$ (every hash function) do
    Initialise hash table $HT_i$ by generating a random hash function $g_i(.)$
end for

for $i = 0$ to $L - 1$ (every hash function) do
    for $j = 0$ to $N - 1$ (every vector in database) do
        Hash $x_j$ in bucket $g_i(x_j)$ of hash table $HT_i$.
    end for
end for

Cardinality of set $P$, a second level of standard hashing is used to map the contents of $g_i(p)$ to a hash table. The second level of hash function is defined as [Gionis et al., 1999]:

$$h(p') = (a_1 \cdot x'_1 + \ldots + a_d \cdot x'_d) \mod M$$

(2.38)

where $p'$ is the transformed binary vector, and $a_1 \ldots a_k$ are selected using a random selection from $\{0, \ldots, M-1\}$. The size of each hash table $M$ is determined using:

$$M = \alpha \frac{n}{B}$$

(2.39)

where $n$ is the total number of points in a collection, and $B$ denotes the size of the hash bucket; $\alpha$ is the utilisation value of each hash bucket [Indyk and Motwani, 1998]. These are the tunable parameters for the LSH index, and are critical for effectiveness and efficiency of this indexing scheme. The process for building the LSH index structure is summarised in Algorithm 1.

When comparing two points using the LSH index, the hash value for a single point is used to retrieve the corresponding hash bucket; the neighbourhood search is limited to only those points that fall within that bucket. Thus, the neighbourhood of an approximate match is reduced considerably to those that share identical hash values, and two images — that are each represented as a single feature vector — can also be matched based on this hash value. All points sharing identical hash values (collisions) within a given hash table are estimated by the $L_1$ distance (due to the Hamming distance) to be closer to each other than those that do not share hash values. Therefore, the search space of an approximate nearest-neighbor
Algorithm 2  Summary for querying the LSH index.

Require: Database of \( N \) \( D \)-dimensional vectors \( x \), a query vector \( q \), \( K \) (number of approximate nearest neighbours), pre-generated \( L \) hash tables \( HT \).

Set \( ANS \) to \( \emptyset \)

for \( i = 0 \) to \( L - 1 \) (every hash table) do

Add points found in bucket \( g_i(q) \) of table \( HT_i \) to \( ANS \).

end for

if \( ANS \neq \emptyset \) (at least one nearest neighbour) then

Return \( |ANS| \) (or \( K \)) nearest neighbours.

end if

match is greatly reduced, to those points (or vectors) that share identical hash values; the process of querying the LSH index is summarised in Algorithm 2. While there are recent variants of LSH that support the \( L_2 \) distance [Datar et al., 2004; Bawa et al., 2005; Andoni and Indyk, 2006], we use the \( L_1 \) norm as it was shown to be sufficiently accurate for near-duplicate image detection [Ke et al., 2004]. Moreover, Gionis et al. [1999] show that the use of the \( L_2 \) distance yields only slight improvements over the \( L_1 \) norm, where limited impact is observed [Gionis et al., 1999]. In this thesis, we implement locality sensitive hashing as detailed in the work of Gionis et al. [1999], and experiment with the \( L \) and \( k \) parameters, as described in Chapter 5 (page 145).

2.5.2 On indexing local descriptors

Images typically contain many local descriptors, and so an exhaustive search for the nearest-neighbour — within a collection — of every local descriptor in an image is computationally expensive. The number of comparisons required is in the order of \( O(q \times n) \), where \( q \) is the number of local descriptors within an image, and \( n \) denotes the number of local descriptors in the entire collection of images.

For query evaluation, all candidate matches are returned from a particular index for every local descriptor in a query image. Hence, each image is treated as a bag-of-points, simulating a multi-point query evaluation. While index structures such as the SR-tree [Katayama and Satoh, 1997] or the \( k-d-B \)-tree [Egas et al., 1999] could be applied for indexing local descriptors, the LSH structure has been shown to yield higher efficiency for indexing high-
CHAPTER 2. BACKGROUND AND RELATED WORK

dimensional data [Bawa et al., 2005; Philbin et al., 2007; Stein, 2007]. Ke et al. [2004] have also shown that the LSH indexing scheme can be applied for highly accurate detection of near-duplicate images. Thus, in this thesis, we use the same framework as Ke et al. [2004]: we maintain two auxiliary index structures — file table (FT) and keypoint table (KT) — to map SIFT keypoints (and PCA-SIFT descriptors) to their corresponding images; an entry in KT consists of the file ID (index location of FT) and keypoint information (x and y location, scale, orientation, and the local descriptor). To search for the nearest-neighbour of a local descriptor using the LSH index, we can perform the same point query as described in Section 2.5.1. Once the approximate nearest-neighbours are identified for all local descriptors in the query image, a cumulative score can be computed for all matching descriptors.

Thus, using the LSH index, the cost of comparing two images largely depends on the cardinality of the set of local descriptors $P$ in any given image. For example, an image with 1 000 local descriptors requires 1 000 point queries to the LSH index for evaluation. As each image is treated as a bag-of-points, processing each bag exhaustively to find the best matching image can be computationally intensive.

As described in the previous section, the effectiveness and efficiency of an LSH index is largely dictated by tunable parameters $k$, $L$ (or $l$), and $B$; these are respectively the random bits selected for hashing, the number of indexes, and the number of hash buckets used. In the work of Ke et al. [2004], $l$, $k$, and $B$ are set to the empirically determined optimal values 20, 450, and 20; the utilization parameter $\alpha$ is set to 0.5. While they do not report in detail the exact methodology used for the empirical evaluation, they have shown that these values yield high effectiveness for near-duplicate detection in a moderate-sized collection of 20 000 images. Nevertheless, in Chapter 5 (page 145), we empirically investigate the $l$ and $k$ parameters as the image collections used in this thesis are considerably larger than those used by Ke et al.

Ke et al. [2004] have shown that a filtering phase can be applied to matched interest points (and their local descriptors) using robust estimators such as RANdom SAmple Consensus (RANSAC) [Fischler and Bolles, 1981] to reduce the number of false positive matches. RANSAC is used to estimate if a set of data points can be fitted to a particular model; the ones that fit are known as inlier matches, and the rest are outliers. Thus RANSAC can be similarly used to approximate a transformation between one point (from the original image) to another (from the corresponding image) by iteratively selecting a random subset of matching points that are assumed as inliers, where they are fitted to this transformation model for hypothetical testing [Fischler and Bolles, 1981]. Mikolajczyk and Schmid [2004]
and Lowe [2004] use similar schemes for robust filtering of false positives in matched interest points. They show that a small number of matched points (as few as 3 [Lowe, 2004]) can be iteratively and randomly sampled from a large number of matched points to geometrically verify and discard matches due to false positives; this ensures high precision in the returned answers. We refer to the literature [Fischler and Bolles, 1981; Matas and Chum, 2002; Forsyth and Ponce, 2003] for detailed discussion of this geometric verification technique. We apply filtering for retrieval tasks using the approach of Ke et al. [2004].

2.5.3 Redundant Bit Vectors (RBV)

Goldstein et al. [2004] propose Redundant Bit Vectors (RBV) for high-dimensionality search for multimedia data. The algorithm consists of three key elements:

1. Approximate high-dimensional spherical regions by tightened hyper-rectangles.
2. Partition the query space to promote redundancy in the index.
3. Represent each partition with an efficient bit vector.

As described in Section 2.5, the conventional nearest-neighbour matching problem is usually formulated as a point query over spheres of fixed or variable radius. The \( \epsilon \)-range search can be applied to return all objects with distances within an \( \epsilon \) threshold [Böhm et al., 2001] if more than one object is required. The distances between two objects are commonly measured in some \( L_p \) metric. However, Goldstein et al. [2003] formulate this as a rectangle search problem, where each point \( p \) in a \( D \)-dimensional space is replaced by the smallest hyper-rectangles of radius \( c \) enclosing the hyper-sphere with centre point \( p \). With this approach, the data space is searched using approximated rectangular regions instead of spheres.

To create an RBV index, all data points are mapped onto data hyper-rectangles, where each dimension of these rectangles is partitioned into \( Q \) bins. The choice of \( Q \) determines the number of disjoint intervals between the rectangles and is specified by the user [Goldstein et al., 2003]. Every dimension of the hyper-rectangle is projected onto its respective axis, where each partition is a bit-vector that reflects the overlap test between the interval boundaries. Each bit in a bit-vector serves as an identifier, and corresponds to a hyper-rectangle within the collection. Each adjacent bit vector (within the same dimension) may have corresponding positions (at the bit-level) set to 1 if the hyper-rectangle overlaps the interval boundaries, leading to bit vectors that are redundant. The process for building the RBV
Algorithm 3  Algorithm for building redundant bit vectors [Goldstein et al., 2004].

**Require:** Database of $N$ $D$-dimensional vectors (or points) $x$, hypercube half-side length $\epsilon$, $Q$ partitions for per dimension.

Sort database by increasing value of the most selective dimension.

for $i = 0$ to $D - 1$ (every dimension of a point) do
  for $k = 0$ to $N - 1$ (every point in the database) do
    $T_{2k} = x_{ki} + \epsilon$.
    $T_{2k+1} = x_{ki} - \epsilon$.  (find cube boundaries)
  end for
  Sort $T$.
  for $k = 0$ to $Q - 2$ do
    $B_{ik} = T_{[2kN/Q]}$  (select $Q - 1$ dividers from sorted boundaries)
  end for
end for

Create bit vectors $BV[D \times Q \times (\frac{N}{32} \text{ machine words})]$, where $N$ elements are packed into machine words

for $i = 0$ to $D - 1$ (every dimension) do
  Initialise and set all bits in $BV_i[Q \times \frac{N}{32}]$ in dimension $i$ to 1
  for $k = 0$ to $N - 1$ (every point) do
    Use following overlap test to create $BV$:
      for $j = 0$ to $Q - 2$ do
        if $x_{ki} + \epsilon \leq B_{ij}$ then
          $BV_{i,j+1,k} = 0$
        end if
        if $x_{ki} - \epsilon \geq B_{ij}$ then
          $BV_{ijk} = 0$
        end if
      end for
  end for
for $j = 0$ to $Q - 1$ do
  $lo[j]$ = index of first “1” bit in $B_{0j}$ (rounded up to machine word boundary)
  $hi[j]$ = index of last “1” bit in $B_{0j}$ (rounded down to machine word boundary)
end for
2.5. IMAGE INDEXING

index is detailed in Algorithm 3: the \( \text{lo} \) and \( \text{hi} \) arrays are used to denote the first and last “on” bits in the bit vector of the most selective dimension; this results in a tightly packed number of 1 bits with preceding and trailing 0 bits. The rationale is that the result will always be dictated by the number of bits set to 1 in the first selected dimension; omitting bits that are set to 0 for bitwise processing does not have any affect on the resultant bit vector [Goldstein et al., 2004]. Thus the \( \text{lo} \) and \( \text{hi} \) bits are, respectively, used to speed up the search process by keeping track of the leading and trailing number of 0 bits for the first dimension.

To search for approximated neighbours for a given point, say \( q \), a spatial bit vector for a given dimension is selected if the partition includes the query point, that is, has its corresponding bit set to 1. The resultant spatial bit vector after performing bitwise \( \text{AND} \) operations on all selected vectors from every dimension will return the data rectangles that contain the query point. Using this approach, each dimension of a data point is assumed to be independent of each other, points are assessed using only one dimension at any one time [Goldstein et al., 2003]. The search process using the Redundant Bit Vectors is detailed in Algorithm 4: the \( \text{lo} \) and \( \text{hi} \) arrays dictate the minimum number of machine words that have to be processed during querying, which improves the efficiency of query evaluation. This indexing method is also highly compact as it uses a single bit to represent a vector within a given dimension, resulting in \( 32 \times \) savings as compared to an integer array of identifiers (assuming a 32-bit machine word). Moreover, Goldstein et al. [2004] show that this approach exploits the size of the machine words, in that a single bitwise \( \text{AND} \) operation can effectively discriminate 32 vectors, for a 32-bit the word size.

Goldstein et al. [2004] also demonstrate that the RBV index excels for applications where there is a large fraction of negative queries — that is, queries for which no answers are expected. For such applications, they report significant gains in efficiency, and a reduction of memory requirement in comparison to LSH. This makes the RBV highly advantageous for indexing high-dimensional feature vectors while maintaining low spatial complexity. However, this scheme has not been shown to be effective for positive queries [Goldstein et al., 2003; 2004]. In this thesis, we experiment with RBV to extend their applicability for positive queries; that is, we show that this scheme can be adapted for positive queries using some intuitive modifications. We further discuss this approach in Chapter 4, where we show the modified RBV indexing scheme can be used for near-duplicate image detection.
Algorithm 4 Algorithm for querying the redundant bit vectors [Goldstein et al., 2004].

Require: Database of $N$ $D$-dimensional vectors (or points) $x$, query point $q$, boundary arrays $B[D \times Q - 1]$ for all $i$ dimensions, completed bit-vectors $BV[D \times Q \times \frac{N}{32}]$.

Create and initialise resultant bit-vectors $R[\frac{N}{32}]$.

$i = \text{smallest index such that } x_0 \leq B_{0i}$

for $j = lo[i]$ to $hi[i]$ (for every machine word within the range of $lo$ and $hi$) do

$R_j = BV_{0ij}$ (find the correct bit vectors in the first dimension)

end for

for $k = 0$ to $D - 1$ (for every dimension of a given point) do

$i = \text{smallest index such that } x_k \leq B_{ki}$

for $j = lo[i]$ to $hi[i]$ do

$R_j = R_j \& BV_{kij}$ (performed only once for every machine word)

end for

end for

for $n = 0$ to $N - 1$ (for every point in database) do

if $R_j$th-bit is 1 then

return $x_n$

end if

end for

2.6 Clustering

The approaches that we have discussed so far assume that the retrieval task involves a query example. Given a collection of images, it is useful to be able to identify the specific classes of images that correspond to the nature of their content; this allows images to be categorised for more effective retrieval [Chen et al., 2005]. For example, images that depict flowers can be categorised as such, so that the class can be used to contain only images of this nature. This also has valid application for near-duplicate detection as images that are near-duplicates of one another can be categorised into the same class (or group).

The general task of associating images to various classes is within the domain of image clustering [Gdalyahu et al., 2001; Li et al., 2002a], and image classification [Tong and Chang, 2001], both of which can be viewed as machine learning problems [Bishop, 2007; Alpaydin, 2004]. While both techniques can be applied for the task of finding specific classes of im-
ages, their approach differs [Arabie et al., 1996]. Image classification typically assigns image instances to a set of predefined classes, whereas image clustering aims to discover a set of (undefined) classes from the image instances. Thus, classification is generally considered to be supervised learning, whereas clustering is regarded as an unsupervised learning task [Arabie et al., 1996]. Here we limit our discussion to image clustering as it is more relevant to our work; we refer to relevant literature [Tong and Chang, 2001; Cox et al., 2000; Wang et al., 2005] for descriptions of image classification techniques.

Image clustering can be regarded as data clustering, given that image features are typically represented as a high-dimensional vector (or point). Here, we digress briefly for a discussion of data clustering techniques. Data clustering can be largely divided into two categories: hierarchical and partitional, each of which can be further sub-categorised into agglomerative or divisive approaches [Jain et al., 1999]. Hierarchical clustering methods generate a nested series of clusters, whereas partitional methods produce clusters in a single level (flat clusters). Agglomerative approaches begin the clustering process by treating each data point as a distinct cluster, where they are progressively merged to form bigger clusters if a criterion is satisfied. In contrast, divisive approaches treat a given set of data points as a single cluster where additional clusters can be formed at a later stage. We refer to the relevant literature [Jain and Dubes, 1988; Jain et al., 1999] for further descriptions of these clustering methods.

Image clustering is a well explored domain where various data clustering techniques such as *k*-means [MacQueen, 1967; Kanungo et al., 2002], and graph partitioning [Jain and Dubes, 1988; Wu and Leahy, 1993] techniques have been applied. The *k*-means is a simple and popular divisive clustering technique. The idea is to specify *k* initial centroids, given a set of points, so that each centroid can be used to group neighbouring points that are within a distance under a certain norm (see Section 2.3.2); additional clusters can be formed using a centroid computed based on the points within each cluster. One of the drawbacks of the *k*-means is that the *k* initial centroids need to be specified a priori [Jain et al., 1999], and a poor selection of this parameter can adversely affect the quality of the cluster formation. While *k*-means is not widely applied directly on image features for cluster formation, it has been used for clustering image pixels for image segmentation [Pappas, 1992], and local region clustering [Sivic and Zisserman, 2003]. Zheng et al. [2004] have also shown that given an appropriate *k* parameter, this method can be applied for effective clustering of visually similar images.

The graph partitioning technique is an agglomerative method that treats the data space
(of the image feature data) using a graph representation; this differs from the vector-based representation as used by the $k$-means [Jacobs et al., 2000]. Using this representation, images are represented as nodes and a set of weighted edges connecting them reflect the affinity between the nodes; the weights between each node can be modelled using a distance function.

Using a graph representation, clustering can be performed by identifying the disjoint groups that accurately reflect the number of distinct classes, with the aim of organising the nodes into groups so that the between-group similarity is low and the within-group similarity is high [Shi and Malik, 2000]. A common technique used for graph partitioning is known as a normalised cut [Shi and Malik, 2000]. A cut [Wu and Leahy, 1993] is a measure of the total weight of the edges that need to be removed for a bipartition; the minimum cut value indicates an optimal bipartition between two sets of nodes. Shi and Malik [2000] propose an improved normalised cut that also considers the edge weights of the nodes within each disjoint set relative to the entire graph; this partitioning method is used in the work of Chen et al. [2005], where visually similar images are shown to be effectively clustered using appropriate distance functions.

While the approaches that we have discussed can potentially be applied for near-duplicate image detection, they are mainly designed for clustering visually similar or topically relevant images. Clustering for near-duplicate images is not a well-explored domain, where only few researchers have attempted this problem with limited success; we discuss this further in Section 2.8.

In Chapter 6, we explore a different approach to clustering near-duplicate images, where we use an agglomerative graph partitioning method that is adapted from techniques used for text document clustering (see Section 2.1). We show that the invariant local descriptors (bag-of-vectors) for each image — which are highly distinctive — can be treated as a bag of words, so that near-duplicate text document clustering techniques can be applied for identifying near-duplicate clusters.

### 2.7 On efficient object-recognition approaches

In recent literature, there has been some development in scalable object-recognition on large image databases [Sivic and Zisserman, 2003; Grauman and Darrell, 2005; Nistér and Stewénius, 2006; Philbin et al., 2007]; these works aim at improving the scalability of the bag-of-points approach to indexing high-dimensional image local descriptors for object-recognition.
2.7. ON EFFICIENT OBJECT-RECOGNITION APPROACHES

Sivic and Zisserman [2003] show that $k$-means can be used to quantise large numbers of local descriptors into clusters, where the descriptors in each cluster are closest to their respective cluster centres. Each cluster is known as a visual word. Each descriptor is thus mapped to a visual word where an immediate match can be found using the $L_1$ distance. Thus, the set of feature vectors of an image is mapped to a vocabulary of $k$ words, or a single $k$-dimensional vector. Nistér and Stewénius [2006] propose an efficient search process that applies a vocabulary tree to the $k$ clusters that are generated using the approach of Sivic and Zisserman [2003]. Instead of using each cluster as a visual word, the vectors are further partitioned into $k$ groups where clustering can be recursively applied to each of the defined cluster. This forms a hierarchical tree with a maximum level $L$ of $k$-branches. Thus, a search for a matching local descriptor involves only $kL$ comparisons. Philbin et al. [2007] show that the original approach of Sivic and Zisserman [2003] can be further improved by using approximate $k$-means, as most of computational complexity lies in the computation of the distance between the vectors and their cluster centres. Their comparison against the approach of Nistér and Stewénius [2006] — known as hierarchical $k$-means — suggests that the approximate $k$-means is superior in terms of effectiveness and efficiency. Grauman and Darrell [2005] show that a set of feature vectors can be directly used to match two images by approximating the distance between them using Earth Mover’s Distance (EMD); the EMD is essentially a measure of the effort required to transform one set of feature vectors to another. Their approach was shown to yield high efficiency in finding correspondences between image scenes [Grauman and Darrell, 2005].

While these methods have been shown to improve the scalability of the bag-of-points approach, the goal is efficient object recognition. Using these approaches, each feature vector (descriptor) is typically generalised to a single unit, where effectiveness of image matches is largely dependent on the discriminative power of a vector of units [Sivic and Zisserman, 2003; Grauman and Darrell, 2005]. For object recognition, images are deemed a match if they share objects or structures; these objects can be matched under varying viewpoints, scale, and orientation. For near-duplicate image detection, we are only concerned with images that are possibly derived from the same digital source, as we discuss further in Chapter 3. As shown in the work of Ke et al. [2004], the distinctiveness of each local descriptor in a bag-of-points approach is useful for discriminating between images with common objects (or structures) and images that are suspected to be from the same source. We believe it is conceivable that these methods can be adapted for highly efficient near-duplicate image detection; however we do not explore these approaches in this thesis.
CHAPTER 2. BACKGROUND AND RELATED WORK

2.8 Techniques for near-duplicate detection

In the following sections, we review related work on near-duplicate image retrieval, focusing on two promising techniques that we use as baseline approaches: dynamic partial functions and the PCA-SIFT-based method.

Dynamic Partial Functions (DPF)

Li et al. [2003] propose that DPF (described in Section 2.3.2), can be used to measure the perceptual difference between two images, and show it to yield high effectiveness in retrieving visually similar images [Li et al., 2003]. As described in Section 2.3.2, the DPF is a distance function that uses a sampling technique to measure the similarity between two images by dynamically comparing only those feature vector elements that contribute the smallest distance [Meng et al., 2003; Qamra et al., 2005]. This is considerably different from the conventional distance metrics such as the $L_1$ or $L_2$ norms, where all elements are used. Meng et al. [2003] further show that the DPF can be applied for near-duplicate detection where it outperforms the $L_1$ and $L_2$ metrics. Their work is perhaps the first to demonstrate that colour and texture are promising features for near-duplicate image detection.

As with CBIR, features such as colour and texture extracted from images are used for similarity computation. Meng et al. [2003] record the colour distribution of an image with 11 perceptual colours that are specified using the HSV colour model, and therefore produce an 11-bin colour histogram (see Section 2.3.1). Eight additional features are extracted from each bin, including the mean and variance (in each of the channels), and also two shape characteristics: elongation and spread [Meng and Chang, 2003]. Elongation characterises the overall shape of the portions that contain colours from each of the 11-bins, whereas spread measures the local distribution of each colour (how well it is scattered) within an image. We refer to the relevant literature [Leu, 1991] for detailed descriptions for how to compute spread and elongation.

For texture representation, Meng et al. [2003] apply the discrete wavelet transform as in the work of Wang et al. [1997], using the Daubechies wavelet in four scales (or level) of decomposition (with three quadrants each) for each level (see Section 2.3.1). For extraction of texture, the image is first converted into grey-scale; then, at each level of decomposition, the mean, variance, elongation, and spread values are extracted from each quadrant [Meng and Chang, 2003] to produce $4 \times 3 \times 4 = 48$ texture values. To further analyse the texture and colour of an image, 11 sub-images are produced using the perceptual colours, where
each image contains only one perceptual colour. For each sub-image, the same wavelet decomposition is applied, and only the mean value is recorded for each of the four levels and the three quadrants — resulting in $11 \times 4 \times 3 = 132$ values. Thus, each image is represented in terms of a 279-element feature vector.

The results reported in the work of Meng et al. [2003] and Qamra et al. [2005] are observed using two separate collections; one for training, and the other for testing. The training collection consists of approximately 80 000 images that are created using 40 artificial image alterations for each of the 2000 images that are randomly selected from the Corel Photo CD collection. The test collection contains 40 000 images, where half of the collection are planted using 500 unique images with 40 artificial alterations each; the 500 images are used as queries to retrieve the near-duplicates.

Meng et al. and Qamra et al. show that using the training data, the DPF method can be used to observe the optimal $m$ parameter. They also show that using the optimal setting, the DPF method is effective in retrieving near-duplicate images. However, this method suffers from low precision at high recall level (see Section 2.9) — or low accuracy when a large number of relevant near-duplicates are retrieved. Meng et al. [2003] extend the DPF function by showing that a slightly higher level of effectiveness can be achieved using a combination of statistical analysis and sampling methods. Motivated by the lack of flexibility in a fixed-size $m$ parameter of the DPF function (discussed in Section 2.3.2), they propose enhanced versions of the DPF using techniques including thresholding, sampling, weighting, and a combination of these methods. However, these variants — in particular the Weighted-Sampling-Thresholding method (WST) that was shown to be most effective — have a negative impact on the efficiency of the algorithm. That is, the effectiveness of these enhanced versions depends largely on the sampling on the given collection, and are therefore dependent on the image collection size [Qamra et al., 2005].

Although a 10% gain in precision was observed in the enhanced model (WST method) [Meng et al., 2003; Qamra et al., 2005], the problem of low precision at high recall remains. While Qamra et al. have shown that the LSH index can be used with the DPF sampling method for large collections of a million images, the methodology is not reported in detail, where it is only preliminarily investigated. Ke et al. [2004] have also shown that the WST method is not as effective as the PCA-SIFT-based approach. Nevertheless, given that the DPF method is probably the first method to address near-duplicate detection in large image collections, we use the original DPF proposed by Meng et al. [2003] a baseline; we refer to this approach as the DPF method.
PCA-SIFT-based detection

Ke et al. [2004] demonstrate near-perfect accuracy in near-duplicate detection by representing images using the SIFT-detected interest points that are characterised by PCA-SIFT local descriptors [Ke and Sukthankar, 2004] (described in Section 2.4). The summary of the three detection phases of the SIFT interest point, and the process of PCA-SIFT local descriptor generation, is as follows:

1. For each image, identify all scale-space extrema (or interest points) using a difference-of-Gaussian (DoG) function computed using a pyramidal scheme; an image is successively reduced by half the size where the process is repeated.

2. Reject poorly localised and unstable extrema if they are below the threshold intensity level and ratio of principal curvature; the rejection thresholds directly dictate the number of interest points (or keypoints) that are used in the subsequent phases.

3. After all stable extrema are identified, assign a dominant orientation to each keypoint for rotation invariance. The orientations are computed using gradients of the regions surrounding the identified extrema; if there is more than one dominant orientation, multiple interest points are created with different orientations. A stable extremum (scale and location information) and its dominant orientation form a keypoint.

4. Compute PCA-SIFT local descriptors using the SIFT-detected keypoints.

Lowe [2004] uses the threshold intensity level that maximises the number of detected points, such that it increases the probability of matches between two images. Lowe [2004] and Ke et al. [2004] do not experiment with lower intensity levels; in Chapter 4 (page 92), we experiment with this threshold to improve the efficiency of this approach.

Local descriptors computed using PCA-SIFT have been shown to be more compact and distinctive than other local descriptors [Ke and Sukthankar, 2004; Mikolajczyk and Schmid, 2003]. By using patches of $41 \times 41$ surrounding each keypoint, with its orientation rotated and canonically aligned, PCA-SIFT uses the horizontal and vertical gradient maps to generate a $2 \times 39 \times 39 = 3042$-element descriptors; each represented by a feature vector. For compact representation, each vector is then projected to a low-dimensional feature space using PCA; this is performed using a pre-computed eigenspace.\footnote{We use the eigenspace provided to the author by Ke and Sukthankar [2004].} Ke et al. [2004] have empirically observed
that descriptors of $n = 36$ feature spaces yield high effectiveness in detecting near-duplicate images; a match is flagged if the $L_2$ distance (Euclidean) is less than 3 000.

To calculate the similarity between two images, Ke et al. [2004] employ LSH [Gionis et al., 1999] to index local descriptors using the approach described in Section 2.5.2. Due to potentially high levels of false positive matches introduced by the $L_1$ distance approximation of the LSH index, they apply additional processing to reduce the number of falsely matched local descriptors. They show that the application of the $L_2$ distance of 3 000 and a geometric verification step such as RANSAC (see Section 2.5.2) for verification of the returned candidates yields good results for near-duplicate detection.

Their results are observed on two separate image collections; one consists of approximately 12 000 fine art images, and another 20 000 images that are randomly sampled from the MM270K collection [Media Graphics International, 2007]. Approximately half of the first collection (6 000 images) are seeded using 40 artificial near-duplicate alterations — identical to those of Meng et al. [2003] and Qamra et al. [2005] — on 150 unique images; the unique images are used to query for the altered images. The second collection contains approximately 4 000 planted images created using an additional 13 severe image alterations on 314 images [Ke et al., 2004]; their approach was observed to yield a very high level of effectiveness for these collections. They also show that the level of effectiveness produced by their approach is significantly higher than that of the WST method by Meng et al. [2003] as described earlier. One major drawback of this approach is that it does not scale well. In Chapter 4 (page 99), we observe that the evaluation time for a single query to retrieve near-duplicate images in a moderate-sized collection of 20 000 can easily exceed a minute. However, to date, this is the most effective approach in this domain; hence, we use this method as a baseline, and refer to this approach as the PCA-SIFT method. In this thesis, we propose an approach to improve the efficiency of this method.

Other approaches

Using the same PCA-SIFT descriptors, Zhao et al. [2006] propose a matching strategy that they refer to as the Local Invariant Point One-to-One Symmetric Matching (LIP-OOS). Here, local descriptors (points) are matched only when an approximated nearest-neighbour relationship between two descriptors is mutual, that is, if point $A$ is the approximated nearest-neighbour of point $B$, then this predicate also holds when comparing points $B$ to $A$. This contrasts with the approach of Ke et al. [2004], where symmetry is not required.
for a match. Zhao et al. [2006] also propose the Local Invariant Points Index Structure (LIP-IS) to facilitate the OOS matching between local descriptors. While Zhao et al. [2006] claim that the combination of LIP-OOS and the LIP-IS approach to yield a slightly better effectiveness than that of Ke et al. [2004], the results are observed using 600 TRECVID video keyframes [Smeaton et al., 2006], where it consists of 150 pairs of near-duplicate keyframes; keyframes are essentially representative images extracted from video footage. The results in effectiveness and efficiency have only been preliminarily evaluated on a limited dataset of 600 video keyframes [Zhao et al., 2006], and the performance of their approach on larger collections are unknown. Their data set is derived from the work of Zhang and Chang [2004] (see Section 2.8), in which a similar definition of near-duplication is adopted; near-duplicates include those of varying viewpoints, or those captured in differing time frames. This is considerably different from the definition of Ke et al. [2004], which we use in this thesis; we further discuss the distinctions in Chapter 3. We also observe that the evaluation of the LSH structure is inconsistent as the LIP-OOS scheme is used instead. This implies that LSH indexing scheme is used for symmetric matching, which is different from the approach of Ke et al. [2004]; the implication of using the symmetric matching with LSH is not discussed. Zhao et al. also note that the LSH indexing scheme is approximately 2.5 times faster than their approach. While the LIP-OOS and LIP-IS have been shown to be promising for associating video keyframes in broadcast domain [Ngo et al., 2006], they are within a slightly different domain. For these reasons, we do not further discuss the LIP-OOS method.

Yang et al. [2006] propose decomposing an image hierarchically into multiple quadrants using its geometric centroids to increase the robustness of colour features specifically for the purposes of near-duplicate image detection. Each centroid is computed in the RGB colour model (see Section 2.3), and a centroid value is calculated for each colour channel. Each image is represented using a 126-dimensional feature vector. Their results show that the centroids are susceptible to variations of contrast changes; only minor scale changes are explored. All observations were confined to a small data set of 5000 images that contains limited image variations, and only four queries were used [Yang et al., 2006]. No comparisons were made against predominant work in this area including highly effective methods of Ke et al. [2004] and Qamra et al. [2005]. In this thesis, we do not experiment with this method.

Maret et al. [2006] show that 162-dimensional feature vectors of colour and texture — essentially a subset of the features used by Meng et al. [2003] — can be indexed using R-tree [Guttman, 1984; Wu and Bretschneider, 2004] to facilitate detection of near-duplicate images. The feature representation is computed using 10 perceptual colours that are specified
using the HSV colour model, and the Gabor wavelet coefficients [Maret et al., 2006]. However, they show that this approach is outperformed by both the DPF approach and the PCA-SIFT method for near-duplicate detection; the experiments were performed using approximately 20,000 images and image alterations that are derived from the work of Ke et al. [2004]. While Maret et al. [2006] show that their extension of this method yields improvements in terms of the false positive and false negative rates, their method remains less effective when compared with the approach of Ke et al. [2004]. We do not further discuss this approach in this thesis.

Clustering near-duplicate images

While some work has addressed the query-based detection of near-duplicate images, clustering of near-duplicates has received little attention. Chang et al. [1998] propose a simple approach to divide the data space of the given set of feature vectors into multiple cells, with each cell used for indexing and storing similar feature vectors. A cluster is formed using similar feature vectors within each cell, and also by collapsing the cells adjacent to it. Chang et al. [1998] (and also Chang et al. [1999]) show, via an experimentation using 192-dimensional feature vector of Daubechies wavelet coefficients, that this technique is effective for the task of near-duplicate detection. However, their technique makes a fundamental assumption about the locality of feature vectors within the data space: that is, feature vectors from near-duplicate images will always fall into adjacent cells [Chang et al., 1998]. For their experiments, they inserted a sequence of 10 artificial modifications from a single image — that range from simple sharpening to re-quantisation — into a collection of 30,000 images.

Other important near-duplicate alterations — such as rotation, scaling, and cropping — are not explored in their work, where the effectiveness of this clustering method is observed from a perspective of a retrieval task. The effects of a larger data set of more severe near-duplicate alterations on the clustering method is not addressed. For these reasons, we do not further discuss this approach in this thesis.

Kim [2003] shows that the discrete cosine transform coefficients (feature vector of 35 dimensions) can be used to detect near-duplicate images. The DCT is similar to the discrete wavelet transform, but uses a cosine function as a basis instead (see Section 2.2). The detection experiment in this work involves using a single query example with 13 artificially created near-duplicates — similar to the work of Chang et al. [1999] — including water colouring, noise distortion, and insertion of text [Kim, 2003]: all near-duplicates are retrieved from a
collection of 40,000 images. He also uses the \( k \)-means clustering approach (see Section 2.6) where five examples, each with 10 image alterations, are inserted into the image collection; there is no justification for the removal of the 3 image alterations. Kim [2003] note that this method is not robust against slight cropping, rotation, and scaling. The observed results in his work indicate that this method fails to retrieve corresponding images of these slightly more severe alterations (using five queries); the efficiency of this scheme is also not discussed in his work. Due to the lack of robustness, we do not further discuss this approach.

Zhang and Chang [2004] show that an Attributed Relational Graph (ARG) can be applied for near-duplicate image detection using a combination of machine learning and stochastic processes to model the transformation from one graph to another. An attributed relational graph is constituted of a set of vertices, a set of edges, and a set of attributes. A vertex represents a part of an image, and an edge reflects the relationship between two vertices. Zhang and Chang use salient regions that are computed from an image using a corner detector, where each corner can be described using local descriptors. They describe each region using eleven feature vectors that include colour distribution (that are specified using the RGB colour model), spatial locations, and the Gabor wavelet coefficients (see Section 2.3.1). Thus, each local descriptor is used to represent a vertex in the ARG. Zhang and Chang use this method to automatically identify near-duplicate pairs within a TRECVID video dataset, but their definition of near-duplication includes variations in viewpoint and scene changes that is relatively broad compared to that described in this thesis (see Chapter 3, page 78). The results as observed on the TRECVID data set [Smeaton et al., 2006] using this method suffers from low effectiveness [Zhang and Chang, 2004]; the data set is limited to 600 video keyframes, and scalability is not addressed. For these reasons, we do not experiment with this method in this thesis. Although their method is not strictly a clustering approach, Zhang and Chang are perhaps the first to address the challenges of automatically identifying all near-duplicate image instances within a given collection without the use of a query example.

Chum et al. [2007] show in recent work — that extends the work of Sivic and Zisserman [2003] — that a bag-of-vectors can be quantised into a single vector to achieve higher efficiency in near-duplicate video keyframe detection. While their approach is not strictly a clustering method, they show that it is applicable for pairwise detection of near-duplicate video keyframes. However, their experimental results were reported to be comparable to only a colour histogram, where the histogram was not specifically designed and tested for robustness against variations in imaging parameters common in near-duplicates; this indicates the relatively low effectiveness of their approach [Chum et al., 2007]. Importantly, their
2.9. EVALUATION

Definition of near-duplication is ambiguous as they define a near-duplicate pair as being two images (keyframes) with a calculated distance between their histogram less than a certain threshold. We do not further discuss this approach.

In this thesis, we show that a simple text document clustering approach can be adapted for near-duplicate detection in large image collections using invariant local descriptors; we discuss this further in Chapter 6.

2.9 Evaluation

To measure the effectiveness of retrieval experiments of near-duplicate images, we apply the standard information retrieval evaluation metrics of recall and precision that are widely used in CBIR [Ma and Zhang, 1998a; Aslandogan and Yu, 1999; Smeulders et al., 2000], defined as [Witten et al., 1999]:

\[
\text{recall} = \frac{\text{relevant images retrieved}}{\text{total relevant images in collection}} \quad (2.40)
\]

\[
\text{precision} = \frac{\text{relevant images retrieved}}{\text{total images retrieved}} \quad (2.41)
\]

For a specific query, recall measures the proportion of the relevant answers (images) that are retrieved, whereas precision measures the ratio of returned answers (images) that are relevant [Baeza-Yates and Ribeiro-Neto, 1999]. Relevant images are typically those that are pre-defined (by a human observed) within a collection, or artificially generated prior to experimentation, so that the metrics can be applied to a set of returned answers to gauge the ratio of relevant images. In this thesis, we deem two images as being relevant to each other, if and only if they are near-duplicates of one another.

When performing multiple distinct queries, the average recall and average precision are used to represent the average performance of an algorithm. The paired t-test [Hull, 1993] is used to confirm to statistical significance of many of the observed results throughout this thesis; the confidence level of 95% is used in all cases. All the experiments in this thesis are conducted on a personal computer with the specifications listed in Appendix E (page 211).

2.9.1 Effectiveness of pairwise detection

The effectiveness of our discovery (or pairwise detection) approach in Chapter 5 can be evaluated by assessing the relationship graph, where each edge connecting two nodes reflect
the computer-identified near-duplicate relationship between two images. A relationship graph can also be visualised as an affinity matrix, where every element in a row corresponds to every other element in a column; each entry in the matrix represents the relationship between the corresponding elements. An ideal human-evaluated relationship graph of the entire image collection, otherwise known as a reference graph [Bernstein and Zobel, 2004], provides a benchmark for measuring a near-duplicate detection algorithm. The evaluation metrics of coverage and density [Bernstein and Zobel, 2004], which are similar to recall and precision, respectively, can be used for the evaluation of such tasks. The coverage of a computer-generated relationship graph is the completeness relative to the reference graph, while the density is the proportion of edges that are correct.

Generation of a complete reference graph for a collection of images is impractical; it is difficult and labour-intensive, since the near-duplicate relationship of one image to another needs to be known a priori for the entire image collection. However, a reference graph can be easily generated if there is a collection where all near-duplicates for every image are known. We can then evaluate the coverage of a relationship graph using the ratio of pre-determined edges that are identified in the artificial reference graph. For instance, 10 groups of 5 near-duplicate images each will generate $10 \times 5 \times 4 \times 2 = 100$ edges. In Chapter 5, we use an artificially generated reference graph using a collection of known near-duplicate images.

With the reference graph, we can use the coverage measure, defined as [Bernstein and Zobel, 2004]:

$$\text{Coverage} = \frac{E_i}{E_r}$$  \hspace{1cm} (2.42)

where $E_i$ denotes the total number of algorithm-identified edges that also appear in the set of reference edges, and $E_r$ indicates the total number of reference edges. Our experiments in Chapter 5 uses a set of artificially generated near-duplicate images that are inserted to a larger database of images. Thus, the true density of a relationship graph cannot be evaluated without a complete reference graph of the entire collection. To estimate density, we can only select edges from the computer-generated graph to determine whether near-duplicate relationships exist between connecting nodes — a labour-intensive task. For a less resource-intensive evaluation, we use the average precision of the algorithm-identified relationship graph, defined as [Bernstein and Zobel, 2004]:

$$\text{Average Prec} = \frac{1}{J} \sum_{j=1}^{J} \frac{E_j}{E_t}$$  \hspace{1cm} (2.43)
where $E_j$ and $E_t$ denote, respectively, the near-duplicate edges and the algorithm-identified edges of an image that is used for coverage estimation; $J$ denotes the total number of images used for coverage estimation. The average precision measures the ratio of near-duplicate edges to the total identified edges in that graph for each image used in coverage estimation.

### 2.9.2 Clustering effectiveness

In Chapter 6, we present a method to cluster near-duplicate images. For clustering of near-duplicates, a well-formed cluster that is deemed to be of high quality is one that consists of only images that are near-duplicates of each other; it should also ideally contain a complete set of near-duplicates from within the given collection. Thus, the effectiveness of a clustering algorithm can be examined by measuring the quality of the clusters that are identified by the algorithm [Chen et al., 2005].

To measure the quality of the generated near-duplicate clusters, we use the two measures of purity and entropy [Chen et al., 2005]. Given a set of $N$ images that belong to $c$ distinctive near-duplicate sets — denoted by $1, \ldots, c$ — that are formed into $m$ clusters $C_j, j = 1, \ldots, m$, purity for a single cluster $C_j$ can be defined as [Chen et al., 2005]:

$$ p(C_j) = \frac{1}{|C_j|} \max_{k=1,\ldots,c} |C_{j,k}| $$

where $|C_j|$ is the size of the cluster (number of images), and $C_{j,k}$ denotes a set of images in cluster $C_j$ that belong to a near-duplicate group $k$. Here, $k$ denotes a predefined or known near-duplicate set a priori; the use of a predefined near-duplicate set is discussed further in Chapter 6 (page 166). We note that $c$ and $m$ are independent variables, as the expected number of clusters and the number of algorithm-identified clusters do not necessarily coincide.

Purity is the ratio of the size of the dominant near-duplicate set within a cluster to the cluster size. This measure is similar to the precision metric, since $C_{j,k}$ returns the number of relevant images in the dominant near-duplicate set of the cluster. Entropy for cluster $C_j$ is defined as:

$$ h(C_j) = -\frac{1}{\log c} \sum_{k=1}^{c} \frac{|C_{j,k}|}{|C_j|} \log \frac{|C_{j,k}|}{|C_j|} $$

This measure is used to quantify the distribution of the different near-duplicate sets within a cluster; values are normalised to be between 0 and 1. A low entropy indicates that the cluster consists of primarily a single near-duplicate set, whereas a high entropy reflects a mixture of different sets. Thus, in contrast to purity, an ideal clustering algorithm will
form all clusters with entropy values of 0, as each cluster contains only images that are near-duplicate instances of one other. Additionally, we also measure the average purity and entropy of each near-duplicate set, respectively defined as:

\[
p(k) = \frac{1}{m_k} \sum_{j=1}^{m_k} p(C_j), \quad h(k) = \frac{1}{m_k} \sum_{j=1}^{m_k} h(C_j)
\]

(2.46)

where \(m_k\) denotes the number of clusters containing images from near-duplicate set \(k\). Although purity and entropy are informative measures, they are insufficient indicators of the content of each cluster [Chen et al., 2005]. To this end, we also report recall and precision with respect to each near-duplicate set; these are defined as:

\[
R(k) = \frac{\sum_{j=1}^{m_k} |C_{j,k}|}{|k|}, \quad Pr(k) = \frac{\sum_{j=1}^{m_k} |C_{j,k}|}{\sum_{j=1}^{m_k} |C_j|}
\]

(2.47)

where \(|k|\) indicates the size of the known near-duplicate set \(k\). When used with purity and entropy along with \(m_k\), recall and precision are accurate indicators of cluster quality in terms of completeness and clustering precision.

### 2.9.3 Image collections

To create our test collection, we select 200 images at random from the Corel Photo CD collection (volume Seven and volume Twelve) [Corel Corporation, 1994], and perform 50 digital alterations, yielding 10 000 images; the specific alterations are discussed in Chapter 3 (page 79). We consider all alterations of an image to be near-duplicates of the original. We refer to this as the seed collection throughout this thesis. We also do not take into account the different categories — as presented in the Corel Photo CD — as they are treated as random samples of images. We believe that the distribution of the image categories should not have to be taken into account in this case, since only images derived from the 50 digital alterations and their originals are considered relevant to each other.

To test the effectiveness of our approaches using large collections, we use four separate collections of 20 000, 150 000, 300 000, and 1 000 000, each including the 10 000 images of the seed collection, and the balance selected at random using images — gathered from the Web — in the SPIRIT collection [Joho and Sanderson, 2004]. We refer to these four image collections respectively, as 20K, 150K, 300K, and 1M.

We use separate collections for our seed and noise images, as there is no way to guarantee that the images randomly selected from the SPIRIT collection do not include near-duplicates
that could affect our evaluation; this is especially important for our clustering experiments in Chapter 6. To minimise the discrepancies between the quality of images between the two collections, we select only images with similar dimensions to those of the seed collection. We also identify identical (in the bit-level) images using the MD5 [Rivest, 1992] digests and remove them. All images are normalised to uniform dimensions, with 512 pixels in the longer edge.

To gauge the effectiveness of the algorithms on real-world examples, we also use an image collection of approximately 19,000 examples that we gathered from the Web by querying commercial search engines (by text). Near-duplicates in this collection are manually identified, as we discuss further in Chapter 3 (page 82). This collection is used in addition to the controlled collections.

2.10 Summary

In this chapter, we discussed the relevant near-duplicate text document detection techniques; specifically the document fingerprinting approaches that are commonly applied for identifying near-duplicate pairwise similarity of text documents. We have also discussed the relevant field of digital image watermarking approaches designed for detection of copyright violation, and its limitations for the near-duplicate detection.

We provided an overview on the fundamental aspects of CBIR, such as image feature representation and indexing techniques, and their relevance to near-duplicate detection. In particular, we have described the colour, texture, and shape features that are commonly applied in CBIR for detecting visually similar images; we also discussed the limitations of these features for near-duplicate detection. We described image representation using interest points and local descriptors that are used in this work; they are robust and invariant against image changes — such as photometric, lighting, and geometric variations — that are common in near-duplicates.

In this chapter, we also reviewed some high-dimensionality indexing techniques that are relevant to our work. We described in some detail two techniques namely Locality Sensitive Hashing and Redundant Bit Vectors. We also discussed some basic concepts of image clustering that are relevant to our work. We followed with a review of existing approaches on near-duplicate image detection, and discussed the limitations that motivate our research. We outlined the two best previous methods for near-duplicate detection that we use as baselines, specifically the DPF method and the PCA-SIFT-based method. Finally, we provided
an overview of the evaluation methodologies and the image collections that are used for our experiments in this thesis.
Chapter 3

The Duplication Problem

In an environment such as the Web, it is common to find multiple versions of the same image; examples include thumbnails stored by web search engines that are served in response to user queries; copies shared by various news portals; and images that are — accidentally or otherwise — appropriated or re-used from one website to another. In the absence of an effective detection system, such near-duplicates present problems of digital rights management and collection management.

Near-duplicate image detection has been investigated from several perspectives, including digital watermarking, content-based image retrieval (CBIR), and stereo matching, as described in Sections 2.2, 2.3, and 2.4. However, most proposed solutions are tested on controlled image collections, often derived from digital watermarking testbeds such as Stir-Mark [Fatès and Petitcolas, 2000], or from personal or proprietary collections. Moreover, the alterations studied are usually artificially generated, and their relevance to web collections is unclear. While most studies assume that artificial alterations serve as a rough guide to the kinds of near-duplicates extant in web images, there has been no study of the image alterations that commonly occur on the Web. Indeed, results of existing near-duplicate detection approaches on such controlled collections may not reflect their effectiveness on more representative collections. An explicit evaluation of the kinds of alterations that exist in web images is useful for such experimentation.

In this chapter, we report two related investigations. First we describe the artificial image editing operations that are commonly used in the near-duplicate detection literature [Ke et al., 2004; Meng et al., 2003; Qamra et al., 2005], and define the scope of our problem. Second, we analyse images from the Web; we gather the results returned by a web search.
engine for popular image queries, and manually examine these to identify instances and types of near-duplicates. These human-evaluated near-duplicate web images are used for experimentation in later chapters.

3.1 What is a near-duplicate image?

In recent work, Jaimes et al. [2002] and Zhang and Chang [2004] have grouped near-duplicate images into three main categories by the nature of change from the original:

- **Scene** Images that differ due to addition, occlusion, or movement of foreground or background objects.
- **Camera** Images that are captured with variations in camera viewpoint or angle.
- **Image** Images that differ due to digital editing operations such as cropping and filtering, and colour, contrast, or resolution alteration.

Most research in the literature assumes near-duplicate images — otherwise known as near-identicals, or image replicas — to belong to any one of these categories, and therefore treat these image instances as a general near-duplicate problem. As described in Chapter 2 (page 64), Zhang and Chang [2004] treat video keyframes that are captured a few seconds (or sub-seconds) apart as near-duplicate images, and the image variations from these keyframes can belong to any one of the categories described above; Zhao et al. [2006], and Chum et al. [2007] use similar data sets in their work. However, such an approach targets a broad range of image alterations, which we believe is counter-productive for purposes of detecting copyright infringement and image redundancy for two reasons:

1. Image copies or near-duplicate images that are suspected of infringing copyright that are derived from the original or source image, rather than images that contain similar visual content; images of camera and scene changes are typically images within the latter category.

2. Redundant images are those that do not add content value alongside those already at hand. In a web search, redundant images are those that appear in the result set after a near-duplicate (of image changes) has previously been returned to the user.

Images of scene and camera changes are less restrictive in terms of the kinds of derivation that are possible. We believe the detection of these instances are better suited for applications
3.1. WHAT IS A NEAR-DUPLICATE IMAGE?

such as content tracking of video footage [Chum et al., 2007; Wu et al., 2007], where the aim is to detect similar scenes in different sources of video clips.

In this work, we are mainly concerned with the detection of copyright infringement and with redundancy management; thus, we focus on the detection of derivatives created through image changes, where we assume that the original and its derivatives share a digital source, and that the derivation is evident to a human observer. However, we do not limit our detection strictly to images that are — or are suspected to be — derived from the same source; some images may be near-identical without a common digital source. Examples of such instances include multiple photographs of the same scene, or multiple photographs of a painting. These are considered to be redundant images, where the derivation is obvious to the human observer. Although these images are within the scope of camera changes — those captured with variations in camera viewpoints — there are subtle differences. We are interested only in images captured under minimal or no variations in viewpoint and angle; this also implies that there would be minimal distortion of subjects within the image. Such images are not syntactically identical, but they may still be of interest for detection of copyright infringement and for redundancy management. The rationale is that any images that are reproduced or captured under severe changes in viewpoint or angle have limited resemblance to the original image; thus, the value of such reproductions is questionable.

Artificially created near-duplicate images are often used to assess the effectiveness of near-duplicate detection algorithms where the alterations are adapted from StirMark [Fatès and Petitcolas, 2000]. It is easy to simulate image changes with digital image editing software. In this thesis, we use a list of 50 digital alterations similar to those found in the work of Ke et al. [2004], Meng et al. [2003], and Qamra et al. [2005]. The number in parentheses is the number of instances for each alteration type.¹ These are:

1. **jpeg to gif**: format change from JPEG to GIF (1)
2. **colorise**: each of the red, green, and blue channels are tinted by 10% (3)
3. **contrast**: increase and decrease contrast (2)
4. **contrast (sev)**: increase and decrease contrast to $3 \times$ that of the original image (2)
5. **crop (5-30%)**: crop 95%, 90%, 80%, and 70% of the image, preserve centre region (4)
6. **crop (40-90%)**: crop 60%, 50%, and 10% of the image, preserve centre region (3)
7. **despeckle**: apply "despeckle" operation of ImageMagick (1)

¹All alterations are created using the ImageMagick software, http://www.imagemagick.com.
Figure 3.1: Near-duplicate images derived from a single original (top left); using the list of alterations described in this Section 3.1. We selectively show examples of 16 representative alterations (out of 50) and omit jpeg_to_gif and despeckle operations. The figure shows images of the British Airways London Eye, on the River Thames; courtesy of FreeDigitalPhotos.net.
3.1. WHAT IS A NEAR-DUPLICATE IMAGE?

8. border: a frame size 10% of image is added using random colors (4)
9. rotation: rotate image (by 90°, 180°, and 270°) about its centre (3)
10. scale-up: increase scale by 2×, 4×, and 8× (3)
11. scale-down: decrease scale by 2×, 4×, and 8× (3)
12. saturate: alter saturation by 70%, 80%, 90%, 110%, and 120% (3)
13. intensity: alter intensity level by 80%, 90%, 110%, 120% (4)
14. intensity (sev): alter intensity level by 50% and 150% (2)
15. saturation (sev): alter saturation by 110%, and 120% (2)
16. rotate+crop: rotate the image (by 90°, 180°, and 270°), crop 50% in centre region (3)
17. rotate+scale: rotate the image (by 90°, 180°, and 270°), decrease scale 4× (3)
18. shear: apply affine warp on both x and y axes using 5°, and 15° (4)

We note that even though we normalise image sizes after these transformations, alteration 10 and 11 can still effect substantial image changes as scaling up will extrapolate pixel information, whereas scaling down will cause information loss. The list represents a sample of the types of digital alterations in Stirmark [Fatès and Petitcolas, 2000] suitable for testing the effectiveness of near-duplicate image detection schemes. The complete set of digital operations as described in SitMark may be counter-productive, as they are designed to test the robustness of a digital watermarking scheme across even small alterations; for instance, the alteration of crop can range from 95% to 5%, in 1% decrements. While the observation of the embedded digital watermark retrieved from the two of the same image that are cropped with one percent difference may be interesting for observing the robustness of digital watermarks, it is not the case for content-based detection algorithms, as the image content may not have changed considerably.

Figure 3.1 illustrates some examples of near-duplicate images that are created from one original image; the image at the top left is the original, and the rest are artificial near-duplicates derived from this original. Some of these digital alterations severely affect the appearance and content of the image; for example, the crop operation — also known as digital zooming if the image is resized to its original dimensions — can create images that appear completely unrelated despite being derived from the same source. However, the detection of these instances is still important as they are related to the original source image.
CHAPTER 3. THE DUPLICATION PROBLEM

We use these alterations to create near-duplicate images for experimentation in this thesis, as we show in later chapters.

3.2 Near-duplication in web image search

Duplicate and near-duplicate images — variants derived from the same original image as described earlier — are common on the Web. Consider the examples in Figure 3.2 and Figure 3.3, where the query “miles davis kind of blue” was issued, in turn, to the Google and Yahoo\(^2\) text-based image search engines, on August 3, 2007. As we can see, there are only seven unique images in the top 20 images returned by Google, and eight unique images returned by Yahoo; by default 20 images are returned on the first page of results. Most of the returned images are identical copies, and others are clearly derived from one another, reflecting revisions of the same image.

This presents a problem for users of these image search services, who are subjected to viewing multiple apparent copies of the same image in the search result. It can be argued that the reason for these reissues is the absence of other images that are associated with a given query, but when a wide variety of images correspond to a given query (as is the case in the Figures 3.2 and 3.3), we question whether these reissues should be presented. We conjecture that grouping near-duplicates and identical copies can improve the usability of image search, and allow a greater diversity to be presented to the users.

3.3 Analysis of near-duplicate images on the Web

To study a sample collection of images representative of the variety and level of duplication present on the Web, we decided to use information from an existing web image search engine, such as Google Flickr, Ditto, Yahoo!, or AltaVista. These search engines use text metadata associated with images to retrieve images relevant to keyword queries. For instance, many related images for “miles davis” are annotated with his name, or with keywords such as “jazz musician” or album titles, as was shown in the previous examples in Figures 3.2 and 3.3. Since they do not use the visual content itself, such systems are essentially text search engines. However, we can use such systems to collect a pool of images associated with popular subjects for further manual analysis. We have observed that results returned for popular image queries are typically highly relevant, and irrelevant images can be eliminated by human evaluators

\(^2\)http://images.search.yahoo.com
3.3. ANALYSIS OF NEAR-DUPLICATE IMAGES ON THE WEB

Figure 3.2: The top 20 image answers returned for an image query of “miles davis kind of blue” using the Google search engine (http://images.google.com). The image query was issued on August 2, 2007.
Figure 3.3: The top 20 image answers returned for an image search of “miles davis kind of blue” using the Yahoo! search engine (http://images.search.yahoo.com). The image query was issued on August 2, 2007.
3.3. ANALYSIS OF NEAR-DUPLICATE IMAGES ON THE WEB

during manual assessment.

We use the Google Image Search Engine\(^3\) — which we refer to as GISE — and a set of popular image queries that users supply to this system as determined from the Google ZeitGeist report for 2005.\(^4\) Google ZeitGeist reports on the general user search trends, and contains statistics on millions of Google searches over a set period of time. The report contains categories of search trends ranging from general, such as queries originating from particular countries, to specific, such as Google news queries, and Google image queries; the Google image queries are further grouped into a few sub-categories such as popular music groups or popular nature queries. At the time of the study, the Google ZeitGeist was the only comprehensive report on search trends that the author was aware of.

Making an arbitrary decision, we selected five categories from the list of Google image queries, each with five popular image queries from January to October of 2005; these categories are Celebrities; Kids-related; Heroes; Cars; and Deceased. At the time of this study, image queries from November to December of 2005 were not yet available. Using these 25 image queries, we use the image URLs (as indexed by Google) to retrieve a list of images associated with the corresponding query.\(^5\) For instance, the URL http://images.google.com/images?q=angelina+jolie&start=0 is used to retrieve the first page of image results for query “Angelina Jolie”; we do not filter for any particular file types or file sizes. We also do not explicitly check for robot.txt files, or exclude personal web pages and photo sharing sites such as flickr photos, as we expect the image results from Google to have been filtered. GISE returns a maximum of 1000 answers for each query; the result sets for our queries contain an average of 750 images each. In total, we gathered 18,765 images from 25 different image queries. Images with identical URLs were removed.\(^6\) All images were converted to the JPEG format and resized to 512 pixels in the longer edge, preserving aspect ratio.

We used the ranked list of image URLs returned by GISE for each query to retrieve the original images from each host web site. We then manually examined these ranked images to identify clusters of near-duplicate images within the returned answers. We limited our evaluation to the first 20 clusters due to lack of resources for such a laborious task, as the formation of each human-identified near-duplicate cluster requires the perusal of the entire

\(^3\)http://images.google.com
\(^4\)http://www.google.com/press/zeitgeist/archive.html
\(^5\)All image queries were issued on September 15, 2006.
\(^6\)The list of image URLs returned by the Google index is available at http://www.cs.rmit.edu.au/~jufoo/ZeitGeistURLS.tgz
result set; all images are manually evaluated by the author. We selected 20 clusters as this is typically the number of answers shown on the first page of image results by commercial web search engines. The seed for the first cluster was the top-ranked image, and images considered to be near-duplicates of this image were allocated to this cluster. Similarly, the seed for the second cluster was the top-ranked image that did not belong to the first cluster. Using Figure 3.2 as an example: the first image in the list — shown at the top left — is used as the seed for the first cluster while the fifth and the seventh images are seed images for the second and third clusters, respectively. The evaluation process used for each of the 25 query subjects is as follows:

1. Select the first 20 distinct images from the answers returned by GISE; only images relevant to the subject are selected.

2. Manually inspect the entire result set to identify clusters of near-duplicates for each of these 20 images. Clusters with no identified near-duplicate images are deemed to be singletons.

3. Within each cluster, manually identify and categorise the kinds of alterations for each image.

Table 3.1 shows the 25 different queries, which are evenly distributed across five major subject groups as listed in Google ZeitGeist 2005. The subjects used are listed in the second column. The third column shows the number of images returned by GISE for each subject; the total number of images within the 20 clusters is indicated within parentheses. Column four shows the cumulative number of near-duplicate images manually identified in the 20 clusters, and the percentage relative to the total number of images returned by GISE. Note that the counts of near-duplicate images for each subject do not include the seed image in each of the 20 clusters. The final column lists the number of distinct images identified within the first 20 answers (as returned by GISE) of each subject. For the 25 queries we discovered 1 512 near-duplicate images in 500 separate clusters within the collection of 18 765 images, indicating a high level of image redundancy.

As indicated in Table 3.1, we find that for the ZeitGeist subject groups Celebrities, Kids-related, Cars, Heroes, and Deceased, there are on average approximately 7%, 4%, 2%, 9%, and 18% near-duplicate images respectively. The percentage of identified near-duplicate instances is an average of all near-duplicate instances of all subjects within each subject group. Even though we have evaluated only 20 clusters for each query subject, we find that the number of near-duplicate instances is surprisingly high; the statistics for each query
### Table 3.1: Popular image queries from Jan. to Oct. 2005 according to Google Zeitgeist.
The third column shows the number of returned images from Google image search for each subject; within parentheses are the number of images within clusters. Column four shows the number and percentage (within parentheses) of near-duplicate images that are manually identified within each result set. The last column shows the number of unique (distinct non near-duplicates) images in the first 20 answers of each subject.

<table>
<thead>
<tr>
<th>Zeitgeist Groups</th>
<th>Query Subject</th>
<th>number of images</th>
<th>number of near-duplicates (%)</th>
<th>unique images in first 20 answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celebrities</td>
<td>50 cent</td>
<td>830 (164)</td>
<td>144 (17.3)</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Angelina Jolie</td>
<td>565 (53)</td>
<td>33 (5.8)</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>David Beckham</td>
<td>661 (44)</td>
<td>24 (3.6)</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Carmen Electra</td>
<td>682 (63)</td>
<td>43 (6.3)</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Britney Spears</td>
<td>633 (34)</td>
<td>14 (2.2)</td>
<td>19</td>
</tr>
<tr>
<td>Kids-related</td>
<td>The Simpsons</td>
<td>785 (58)</td>
<td>38 (4.8)</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>South Park</td>
<td>859 (90)</td>
<td>70 (8.1)</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Garfield</td>
<td>845 (42)</td>
<td>22 (2.6)</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Disney</td>
<td>741 (21)</td>
<td>1 (0.1)</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Snoopy</td>
<td>712 (33)</td>
<td>13 (1.8)</td>
<td>20</td>
</tr>
<tr>
<td>Cars</td>
<td>Ferrari</td>
<td>703 (35)</td>
<td>15 (2.1)</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Lamborghini</td>
<td>673 (54)</td>
<td>34 (5.0)</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>BMW</td>
<td>679 (20)</td>
<td>0 (0.0)</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Porsche</td>
<td>781 (34)</td>
<td>14 (1.8)</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Mustang</td>
<td>773 (22)</td>
<td>2 (0.3)</td>
<td>19</td>
</tr>
<tr>
<td>Heroes</td>
<td>Batman</td>
<td>837 (74)</td>
<td>54 (6.5)</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Harry Potter</td>
<td>777 (90)</td>
<td>70 (9.0)</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Yoda</td>
<td>859 (111)</td>
<td>91 (10.6)</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Spiderman</td>
<td>832 (102)</td>
<td>82 (9.9)</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Superman</td>
<td>750 (82)</td>
<td>62 (8.3)</td>
<td>17</td>
</tr>
<tr>
<td>Deceased</td>
<td>Bob Marley</td>
<td>818 (147)</td>
<td>127 (15.5)</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Tupac</td>
<td>799 (132)</td>
<td>112 (14.0)</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Kurt Cobain</td>
<td>766 (165)</td>
<td>145 (18.9)</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Aaliyah</td>
<td>715 (169)</td>
<td>149 (20.8)</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Terri Schiavo</td>
<td>690 (173)</td>
<td>153 (22.2)</td>
<td>16</td>
</tr>
</tbody>
</table>
subject are detailed in Table 3.1. The numbers in the last column indicate that redundant images are present even in the first 20 answers as returned by GISE.

From the same table we observe trends relating to topic and the volume of near-duplicate images. For instance, there appear to be considerably more near-duplicates within the **Deceased** category, whereas images within the **Cars** category have only a small percentage of near-duplicate instances. The **Deceased** subject group contains images of deceased people for whom there may be only a limited number of images in circulation, thus creating a high likelihood of duplication in a large set. Observe that the **Celebrities** group contains fewer than half the number of near-duplicate instances as compared to **Deceased** group, with 7% and 18% on average, respectively. We believe this is explained by the significantly larger pool of images of these celebrities in circulation on the Web, with new images of these celebrities less likely to be derived from pre-existing images.

We observe two interesting factors relating to the varying amounts of near-duplicate images between each query: **relative rarity** and **popularity**. Consider the **Cars** subject group. We observe a decreasing trend in the number of near-duplicate instances from rare cars to relatively common ones (from 5% to 0% for “Lamborghini” to “BMW”, respectively). We speculate that rare cars such as “Ferraris” and “Lamborghinis” will have more near-duplicate instances than common cars, since there are considerably fewer personal images of rare cars. In contrast, we observe images associated to the query of “BMW” to contain many images of personal vehicles that are likely to be unique. Drawing another example from the **Celebrities** subject group, popular rap music icons such as “50 cent” are observed to have a considerable number of near-duplicate images of popular album covers and movie posters, many of which appear to share the same digital source. Conversely, we find that images for “David Beckham” do not contain as many near-duplicates, as there are fewer definitive images, such as album covers or movie posters; images of this celebrity are mainly shots of soccer matches, that we observe to be less popular targets for duplication.

**Types of alterations**

When categorising the process used to derive an image from the original, it is unrealistic to expect that near-duplicate images on the Web will have been subjected to only one alteration, and therefore, the artificial alterations from previous research [Ke et al., 2004; Meng et al., 2003; Qamra et al., 2005] are unsuitable for this evaluation process. It is challenging to identify the kinds of alterations to which an image may have been subjected; for simplicity,
3.3. ANALYSIS OF NEAR-DUPLICATE IMAGES ON THE WEB

we categorise each image on the basis of its most obvious alteration. For instance, if an image has been slightly cropped and rotated, it is categorised by only the rotation. If multiple alterations are obvious, the modification is classified under “combination”.

Based on our observation, the common types of alterations found among web images are:

1. **combination**: Images with one or more major alterations; this includes artistic manipulations.
2. **scale**: Images that are close to or identical to original images but differ in size or dimensions.
3. **crop/zoom**: Images that are cropped from or show portions of original images.
4. **P.I.P.**: Picture-In-Picture. Images that contain another image within it.
5. **contrast (+/-)**: Images with visibly different contrast levels.
6. **border/frame**: Images that are framed or bordered versions of original images.
7. **grayscale**: Images that are converted into grey-scale.
8. **recolouring**: Images with obvious colour changes.
9. **mirror**: Images that are mirrored versions of original images.
10. **rotate**: Images that are rotated.

Within a cluster, there is often an image that is more complete or has higher resolution than the others. It is often plausible that this is the original image from which the others are derived. We use the additional category original to label these images; however, any classification is tentative, as it must change if a new candidate original image is discovered. Typically, we expect at least one best candidate for original image within each cluster. However, some clusters may not contain any candidates to be considered for this category, as images within the cluster may have been subjected to obvious alterations; these clusters will not have any images labeled as original.

As shown in Table 3.2, we find that besides the category original, the most common is the combination category with 391 instances; this is understandable as it is an umbrella category containing combinations of other common alterations, as well as artistic effects such as collages. This is followed by the categories scale, crop/zoom, P.I.P., contrast (+/-), and border/frame, with respective near-duplicate instances of 368, 354, 134, 92, and 85. The remaining alteration groups, grayscale, recolouring, mirror, and rotate are relatively rare, with an average of less than 20 instances each. The category with the most images is
 CHAPTER 3. THE DUPLICATION PROBLEM

<table>
<thead>
<tr>
<th>Alteration Type</th>
<th>Celebrities</th>
<th>Kids-related</th>
<th>Cars</th>
<th>Heroes</th>
<th>Deceased</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>100</td>
<td>96</td>
<td>99</td>
<td>97</td>
<td>118</td>
<td>510</td>
</tr>
<tr>
<td>combination</td>
<td>62</td>
<td>59</td>
<td>8</td>
<td>110</td>
<td>152</td>
<td>391</td>
</tr>
<tr>
<td>scale</td>
<td>72</td>
<td>42</td>
<td>16</td>
<td>95</td>
<td>143</td>
<td>368</td>
</tr>
<tr>
<td>crop/zoom</td>
<td>87</td>
<td>24</td>
<td>21</td>
<td>69</td>
<td>153</td>
<td>354</td>
</tr>
<tr>
<td>P.I.P.</td>
<td>15</td>
<td>13</td>
<td>6</td>
<td>43</td>
<td>57</td>
<td>134</td>
</tr>
<tr>
<td>contrast (+/-)</td>
<td>13</td>
<td>4</td>
<td>3</td>
<td>18</td>
<td>54</td>
<td>92</td>
</tr>
<tr>
<td>border/frame</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>18</td>
<td>56</td>
<td>85</td>
</tr>
<tr>
<td>grayscale</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>27</td>
<td>33</td>
</tr>
<tr>
<td>recolouring</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td>29</td>
</tr>
<tr>
<td>mirror</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>rotate</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3.2: Number of near-duplicate alterations amongst manually evaluated web images. All alterations are broadly categorised by type. The last column shows the total number of instances for each kind of alteration across all subjects.

original with 510 images; this means that there are 510 images that could be — or are suspected to be — the original image. This finding is within our expectation as there are approximately 100 instances, on average, across five major subject groups; this translates to approximately one best candidate for the original image for each cluster. Note that some clusters contain more than one instance of the original category due to the difficulty of identifying a canonical original; this is reflected by the relatively larger number of original instances in the Deceased subject group.

Overall, we observe that there is a relatively skewed distribution in the kinds of image alterations that are present on the Web; there is a considerably larger pool of images from the categories of scale, crop/zoom, and P.I.P. than the other categories; this is consistently observed across all five major subject groups. These are interesting findings as they reflect the discrepancies in the characteristics of real near-duplicate examples and artificially created ones. The latter assumes an even distribution of the different kinds of image alterations, whereas our analysis shows that this is unrealistic. Hence, the level of effectiveness as observed in experimentation on a controlled collection may not necessarily translate to the same level of effectiveness in practice.
Our observations suggest that artificial collections are inadequate indicators of the performance of near-duplicate image detection algorithms, given that some image instances — such as those of the combination category — cannot be easily simulated. By performing this study, we have also established ground truth by which to evaluate near-duplicate detection schemes. We believe that real examples as shown here are invaluable for measuring the performance of near-duplicate detection schemes in a real-world environment. This is one of the data sets used for evaluation of near-duplicate detection algorithms in later chapters. Although we have taken a speculative approach in this analysis, it is the first such study regarding image near-duplication on the Web.

3.4 Summary

We have discussed the different kinds of digital editing operations used to derive image alterations that are commonly used to evaluate near-duplicate image detection algorithms. We have also discussed the scope of the problem that we investigate in this thesis. We performed the first ever study of the near-duplicate images on the Web to gain insight into their characteristics. At the same time, we also established ground truth by which to evaluate near-duplicate detection algorithms for real collections. Although somewhat preliminary, our observations in this chapter indicate that artificial collections are inadequate to gauge the performance of these algorithms for images on the Web, as there are considerable discrepancies between the controlled collections and the artificial ones. In the next chapter, we discuss a more efficient method of detecting near-duplicate images using local descriptors; we evaluate our approach using both the artificial collection using the alterations as described earlier in this chapter, and the ground truth as established for the sampled web images.
Chapter 4

Efficient Near-Duplicate Image Retrieval

To allow near-duplicate images to be detected in a large collection, an algorithm needs to be efficient. A previous approach proposed by Ke et al. [2004] uses invariant local descriptors (generated from DoG interest point detector) to detect near-duplicate images with near-perfect accuracy; but the described method has limited application for large collections due to its high computational cost.

In this chapter, we propose a novel pruning strategy for the DoG interest point detector used in the SIFT detector — as described in Chapter 2 (page 41) — for near-duplicate image matching. We show this approach to be simple and highly effective at reducing the number of interest points detected in an image. We also show, empirically, that the reduction in number of interest points leads to only negligible loss in effectiveness for matching near-duplicate images.

Using the pruned set of interest points (characterised by local descriptors), we propose an alternative index structure — a modified Redundant Bit Vector (RBV) — that boasts a considerable reduction in memory usage for purposes of query processing. This alternative index structure is also highly compact for storage of moderate-sized collections.

4.1 Improving retrieval efficiency by pruning SIFT

The SIFT interest points, also known as keypoints, that are generated from difference-of-Gaussian (DoG) regions, have been shown to be highly effective for identifying correspondences in image objects [Lowe, 2004]. Using the DoG detector as computed in SIFT, the
number of generated interest points is typically in the order of $10^3$, depending on the image content, size, and complexity. Such quantities of interest points are often crucial for object-recognition and stereo matching, which enables even small occluded objects with varying viewpoints to be reliably matched [Lowe, 2004]. For retrieval tasks on large collections, however, such a substantial number of interest points can reduce any efficient index structure to be no more efficient than a sequential search.

In the work of Lowe [2004], one of the criteria for rejecting poorly localised peaks is to use an intensity threshold, where the threshold value was selected as 0.03 somewhat arbitrarily; a peak is rejected if its value is below this threshold. Use of this default value generally yields a substantial number of interest points; for instance, the number of interest points that are detected from an image of $512 \times 342$ pixels from our collection is observed to be an average of 1 594.

We hypothesise that near-duplicate image retrieval may not require the full set of interest points, as the retrieval task is often aimed at near-duplicate images displaying high perceptual similarity [Meng et al., 2003; Qamra et al., 2005], where the variations in viewpoint (affine distortions) are limited; we discussed this earlier in Chapter 3 (page 78). Even cropped images will possess many similar traits — such as visible objects, subjects, texture, and shape — to those of the original.

To reduce the number of SIFT keypoints, we propose varying the intensity threshold that is applied to discard candidate local peaks (or extrema). The rationale is that near-duplicate images typically share a substantial number of matching local descriptors, and we conjecture that perhaps a subset may suffice; we believe that using the original number of SIFT keypoints for near-duplicate images is excessive, as a subset of descriptors between two images can likely already discriminate against those that are not near-duplicates. Ke et al. [2004] have also shown that using geometric verification techniques such as RANSAC [Fischler and Bolles, 1981], a small number of points — three to five matching pairs — that are iteratively sampled from a large pool of matched points can be used to reliably verify whether matches are indeed true positives [Lowe, 2004; Ke et al., 2004].

There are two ways that this inclusion threshold value can be used to reduce the cardinality of the keypoint set:

1. Select the $N$ most significant keypoints as ranked by the intensity value.
2. Raise the inclusion threshold value to discard keypoints.

Both methods can be used to reduce the number of keypoints. However, the second method
Figure 4.1: Images on the left show the original SIFT keypoints; images on the right show the number of SIFT keypoints with our proposed reduction scheme. These images are courtesy of FreeDigitalPhotos.net.
4.1. Improving Retrieval Efficiency by Pruning SIFT

yields an unstable number of keypoints, since an image with a lower average intensity may not have any detected keypoints. Hence, we apply the first approach, which sets an upper-bound limit on the number of keypoints selected in this phase; images have fewer detected keypoints than the threshold are not pruned. We experiment with varying the inclusion threshold to determine the optimal value, resulting in a selection of a set of most significant keypoints. We use the term upper-bound loosely, since some keypoints may share the same location and scale information with multiple orientations as described in Chapter 2 (page 41), resulting in approximately 15% more keypoints generated in the subsequent phase [Lowe, 2004]. Thus, by using a simple ranking scheme, we reduce the cardinality of local descriptors without actually modifying the properties of each individual DoG-detected interest point. Figure 4.1 shows examples of images processed with and without the ranking scheme.

Once the SIFT keypoints are generated, we characterise each detected point using a local descriptor, as described in Chapter 2 (page 47). Here, we use the PCA-SIFT local descriptors instead of the original SIFT descriptors as it has been shown to be highly effective for near-duplicate images [Ke et al., 2004]; it is also more compact as each descriptor is represented using a 36-element feature vector instead of 128 in the original descriptor.

4.1.1 Experimental setup

We now describe the series of experiments used to empirically gauge the effectiveness of our approach. For our test collections, we use image collections 20K, 150K, and 300K, and 1M, as described in Chapter 2 (page 74).

To evaluate the effectiveness of our keypoint-reduction strategy, we vary the number of detected keypoints and compare their effectiveness against that of the original number of keypoints as observed in our collection, and use a subset of 1000, 100 and 10 most significant keypoints. We refer to these as threshold values.

Next, we evaluate the percentage of local descriptor (computed from the keypoints) matches between a query image and each of its image alterations; this percentage is relative to the detected keypoints in the query image. Two local descriptors are deemed to be a match if their respective descriptors are nearest neighbours, and within 0.7 times the $L_2$ distance of the second nearest neighbour; this method is known as the repeatability test — commonly used to evaluate the repeatability of local descriptors in computer vision and object recognition [Mikolajczyk and Schmid, 2003; Lowe, 2004; Bay et al., 2006]. For an accurate evaluation, we use sequential scan for the nearest neighbour search on the collection.
of local descriptors, as efficient index structures — such as the LSH index as described in Chapter 2 (page 51) — are approximate nearest neighbour algorithms and unsuitable for this particular experiment.

To compare the effectiveness of the keypoint-reduced PCA-SIFT local descriptors to other popular local descriptors, such as SIFT, SURF, and GLOH as described in Chapter 2 (pp. 41—47), we also perform the repeatability test on these descriptors and compare them against our keypoint-reduced PCA-SIFT local descriptors. Due to the exhaustive computation required when using several keypoint thresholds, specifically for this study we use 100 random query sets of 50 altered images each (half of the seed collection), and evaluate the percentage of local descriptor matches between each image and its alterations.

All subsequent experiments are performed using the LSH index with 200 queries; the indexing of local descriptors is described in Chapter 2 (page 55). For retrieval experiments, we use an identical framework to that of Ke et al. [2004], differing only in the number of SIFT keypoints — characterised by PCA-SIFT local descriptors — used.

We evaluate the effects of our keypoint-reduction strategy at all threshold levels, (1000, 100, and 10) and compare them against the original approach in terms of retrieval effectiveness, where the original image is used to retrieve all altered versions. For this experiment, the local descriptors are indexed using the LSH structure. We apply the standard recall and precision metrics, as described in Chapter 2 (page 71). We also measure query run-time, \(^1\) and index size.

We also evaluate the effectiveness of our approach by posing each altered image as a query to retrieve only its respective original image (from which it is derived); for this purpose, we do not consider other altered images that are relevant to a given query. In addition, using a more stringent evaluation, we assess the retrieval effectiveness of our approach by posing each altered image as a query to retrieve all other altered images that are derived from the same original; the results of this retrieval experiment are also compared against that of the original approach. We also additionally test our approach against the DPF method [Meng et al., 2003] (see Chapter 2, page 64) that employs standard colour and texture features.

4.1.2 Results

Here, we present our results on the retrieval effectiveness of our keypoint-pruning strategy on numerous image alterations and discuss the effectiveness and efficiency of our approach.

\(^1\)Query run-time is measured as the total *elapsed time*. 
in terms of repeatability, recall and precision, query run-time, and index size.

**Repeatability of pruned keypoints**

As shown in Figure 4.2, the effects of keypoint-reduction are presented for all threshold values with 50 alterations, where the alterations are grouped into 18 categories for simplicity; additional detailed results for each image alteration can be found in the Figure B.1 (page 199). The percentages are averaged over 100 queries. We experiment with threshold values of 1000, 100, and 10; these values reduce the average number of local descriptors per image to approximately 1059, 130, and 14, respectively. The figure shows that our keypoint-reduction strategy using various threshold values has little effect on the percentage of local descriptors matched within the threshold criterion (of the repeatability test) for most image alterations. This implies that our pruning strategy is appropriate for near-duplicate image matching.

The variation of the percentage of matching local descriptors amongst different image alterations are expected as some alterations severely affect the local descriptors, yielding lower overall descriptor matches. These trends are relatively stable across all levels of reduction, which leads us to believe that a small subset of keypoints is indeed sufficient for this application. This is an important finding as it suggests that effectiveness can be maintained by severely reducing the cardinality of a set of local descriptors for a single image. For some alterations, the slight increase in the percentage of matching descriptors for some threshold values, compared to the default approach, is explained by the fact that the percentage of matched descriptors is relative to the number of detected keypoints in the image. The percentage of matching local descriptors naturally increases when the number of correct matches is relatively stable for a reduced set of local descriptors.

To further study our keypoint-reduced local descriptors, we test our approach against other popular descriptors. Figure 4.2 shows the repeatability of the keypoint-reduced PCA-SIFT local descriptor against other local descriptors; the percentages are averaged over 100 queries of 50 alterations each. We observe that almost all local descriptors exhibit similar trends for all 50 alterations, and the repeatability of the local descriptors is quite consistent for each alteration. For instance, the SIFT descriptor is shown to be superior for the alterations rotate+crop, shear, and crop (40–90%), as it clearly yields a better repeatability than other descriptors, but such effectiveness is not replicated across other alterations. With the SURF descriptor, it is observed to be superior for only alterations of rotate+scale, and
CHAPTER 4. EFFICIENT NEAR-DUPLICATE IMAGE RETRIEVAL

Figure 4.2: Repeatability (%) of keypoint-reduced PCA-SIFT local descriptors in various altered images; the numbers in the parentheses for PCA-SIFT denote the applied threshold value. Average repeatability of other local descriptors (SURF, SIFT, GLOH) are also shown; all local descriptors are computed using the default regions (see pp. 41—47). The results are averaged over 100 sets of images, with 50 alterations each. The alterations are grouped into 18 categories.
4.1. IMPROVING RETRIEVAL EFFICIENCY BY PRUNING SIFT

<table>
<thead>
<tr>
<th>Descriptors</th>
<th>pca-sift (orig)</th>
<th>pca-sift (1000)</th>
<th>pca-sift (100)</th>
<th>pca-sift (10)</th>
<th>sift</th>
<th>surf</th>
<th>gloh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. repeatability (%)</td>
<td>50.4</td>
<td>49.2</td>
<td>54.4</td>
<td>53.1</td>
<td>55.4</td>
<td>55.6</td>
<td>56.0</td>
</tr>
<tr>
<td>Avg. no. of descriptors</td>
<td>1,594</td>
<td>1,059</td>
<td>130</td>
<td>14</td>
<td>1,594</td>
<td>869</td>
<td>925</td>
</tr>
<tr>
<td>Descriptor length</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>128</td>
<td>64</td>
<td>128</td>
</tr>
</tbody>
</table>

Table 4.1: Average repeatability (%) of each local descriptor on all image alterations; it also shows the characteristics of each local descriptor type as observed for our image collection.

contrast (sev). This is similarly observed for all local descriptors. Details results of these local descriptors on individual image alterations is further shown in Figure B.1 (page B.1).

As shown in Table 4.1, the average repeatability for all image alterations is approximately within the range of 50% to 56%. This is an interesting finding, indicating that certain digital alterations affect the repeatability of one set of local descriptors as much as another; and that none of the local descriptors has better effectiveness for all image alterations. While it is evident that GLOH, SIFT, and SURF, are superior to the original approach of PCA-SIFT in terms of repeatability, the length of these descriptor types is also up to three times (two for SURF) higher. Moreover, using our keypoint-reduced approach, the number of local descriptors for an image is reduced considerably — down to one-tenth the original approach — while maintaining effectiveness at a competitive level.

Retrieval effectiveness and efficiency

Figure 4.3 shows the average recall and precision of 200 queries on 20K image collection. We use the original image as a query to retrieve all of its relevant alterations; we apply the same variation of threshold values (1,000, 100, and 10) to gauge the effectiveness of each threshold level for retrieval tasks. Using a small threshold value of 100, we observe a pleasingly high precision of 99.3% with only a slight drop in average recall of 90.7% from the original recall of 97.1%. This implies that using approximately 130 local descriptors, we retrieve 45 instead of 48 out of 50 relevant answers, on average. The average precision values for this threshold also indicate that, when using fewer than 10% of the default number of local descriptors, most of the relevant answers are retrieved with near-perfect accuracy. However, a threshold value of 10 results in a low average recall of 51.7%, indicating that too severe a reduction
Figure 4.3: Average recall and precision (%) (top) of retrieved answers using original images as queries on collection 20K using original number of local descriptors (default), and threshold values of 1000, 100, and 10. The average query run-time (secs) and index size (GB) are also shown; all results are averaged over 200 queries.
results in considerable loss in effectiveness, as almost 45% of the relevant answers are missed. Nevertheless, this also reflects an interesting finding; that is, approximately 50% of the near-duplicate alterations require no more than approximately 10 local descriptors per image for reliable matching.

In Figure 4.3, we also show the effect of varying threshold values on query run-time and memory requirements for the LSH index structure. With default settings, a single query evaluation on the 20K collection takes 152.4 seconds on average (over 200 queries). Using our keypoint-reduction strategy with thresholds of 1000, 100, and 10, we observe timings of 69.4, 2.2, and 0.4 seconds on average for a single query. This is a significant result, given that a threshold value of 100 also yields high effectiveness (average recall and precision of 90.7 and 99.3, respectively), with only a seventieth of the time of the original approach. We also observe that the difference in the recall and precision levels between the original approach and the keypoint-reduced approach are statistically significant (p-value < 0.05); a two-tailed paired t-test — with a confidence level of 95% — is performed using the 200 queries as samples to determine the level of significance.

The on-disk memory requirements for the index (size) of the same 20K collection are also considerably reduced from 15.3 GB — that of the original approach — to 8.5 GB, 1.6 GB, and 0.2 GB, for threshold values of 1000, 100, and 10, respectively. Note that the index size includes the Keypoint Table (KT) — wherein each entry is 92 bytes in length — and the LSH index, as described in Chapter 2 (page 55); we do not include the File Table (FT) as this remains unchanged.

We do not place emphasis on the timing patterns that are observed in our experiments as the existing implementation of the LSH index is not optimised in-memory; the timings are merely an indication of savings by the current framework. We believe that an investigation of efficient in-memory data structures can further improve scalability.

We observe that the threshold value of 100 provides the best trade-off between effectiveness and efficiency, and so we apply this threshold value in further experiments to compare it against the original approach.

**Further studies**

To further investigate the effects of our keypoint-reduction strategy on each image alteration when posed as a query, we use each altered version to retrieve its respective original image. We experiment with a threshold value of 100 on collection 20K and compare it to the original
approach. We henceforth refer to the keypoint-reduction that uses a threshold value of 100 as the \textit{T100} method. We use the entire seed collection, which consists of 200 different examples of each alteration with a total of 10,000 queries. For this experiment, since we intend to evaluate the average recall and precision of both approaches in retrieving only the original images, we remove the altered images that are used as queries from the index, and replace them with an equivalent number of random images from the SPIRIT collection. We do not consider other altered images (derived from the same source) that are relevant since this experiment is designed to evaluate the effectiveness of each altered image as a query in retrieving its corresponding original.

Figure 4.4 depicts the average recall and precision of the T100 method and the original approach, using 200 queries of each alteration as a collection 20K to retrieve their corresponding original images. The 50 alterations are categorised into 18 groups as listed in Chapter 3 (page 79).

We observe a very high effectiveness level — relative to the original approach — in terms of average recall and precision across all alteration groups. For most alteration groups, the average recall is lossless, indicating that the original image is found with the altered query image using both the T100 method and the original approach. An even more surprising result as shown in Figure 4.4 is that the same threshold value achieves an overall higher average precision than the original approach across almost all alterations, except for scale-down, rotate+scale, and shear. Thus, our pruning strategy does not adversely affect precision, but rather improves it. This can be explained by the fact that our pruning strategy reduces the number of keypoints (and local descriptors) that are computed with lower confidence, and therefore also eliminates the number of potential false matches during matching.

Figure 4.5 shows the effectiveness — average recall and precision — of each image alteration as a query in retrieving all other alterations on collection 20K. While we observe a lower average recall for the T100 method when compared to the original approach, we note that this is a rather stringent evaluation measure given that a miss in one image alteration will be reflected in two groups. For instance, if a query image of alteration \textit{A} does not retrieve an image of alteration \textit{B}, a query the other way around would inevitably fail. However, for most alterations the differences between them are relatively uniform, with the only exceptions of scale-down, contrast (sev), and shear, where we observe greater differences.

This observation is within our expectation as our keypoint-pruning strategy reduces the cardinality of a set of local descriptors considerably, resulting in a reduced number of local descriptors by which to compare every other relevant image. Thus, given a set of local de-
Figure 4.4: Average recall (%) and precision (%) of the original approach and the T100 method, using 200 queries of each alteration on collection 20K (total of 10,000 queries) to retrieve their corresponding original images. The 50 individual alterations are described in Chapter 3 (page 79); for ease of representation, they are grouped into 18 categories.
Figure 4.5: Average recall (%) and precision (%) as observed with 10,000 queries (200 sets of each alteration) on collection 20K to retrieve other altered images using the original approach and the T100 method; each of the 18 alteration groups are described in Chapter 3 (page 79), where there are 50 individual alterations in total.
4.1. IMPROVING RETRIEVAL EFFICIENCY BY PRUNING SIFT

<table>
<thead>
<tr>
<th>Index</th>
<th>Query</th>
<th>Recall (%)</th>
<th>Precision (%)</th>
<th>run-time (secs)</th>
<th>candidate pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Default</td>
<td>97.1</td>
<td>93.1</td>
<td>152.4</td>
<td>62549</td>
</tr>
<tr>
<td>100</td>
<td>Default</td>
<td>94.5</td>
<td>98.4</td>
<td>4.7</td>
<td>7836</td>
</tr>
<tr>
<td>1000</td>
<td>94.5</td>
<td>98.5</td>
<td>2.8</td>
<td>5279</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>90.7</td>
<td>99.3</td>
<td>2.2</td>
<td>2647</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>56.2</td>
<td>100.0</td>
<td>0.5</td>
<td>321</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Average recall and precision (%) of 200 queries on collection 20K using three threshold values of 1000, 100, and 10; default indicates the original number of keypoints (1500 on average). Column one indicates the number of indexed local descriptors, and column 2 indicates the number of local descriptors used for the query images. The last column shows the average number of candidate descriptor matches from the index.

Table 4.2: Average recall and precision (%) of 200 queries on collection 20K using three threshold values of 1000, 100, and 10; default indicates the original number of keypoints (1500 on average). Column one indicates the number of indexed local descriptors, and column 2 indicates the number of local descriptors used for the query images. The last column shows the average number of candidate descriptor matches from the index.

scripts $A$ that belong to the original image, and two sets of local descriptors ($B$ and $C$) from its near-duplicates, the number of matching descriptors between $A$ and $B$ is different from those between $B$ and $C$. This means that, even without keypoint-reduction (the original approach) there is no guarantee that $|A \cap B| = |B \cap C|$, but only that there is a high probability that $B$ also has matches to $C$, given the large number of descriptors that are extracted from each image. Since our keypoint-pruning strategy further reduces the cardinality of each set of local descriptors (denoted by $A'$, $B'$, and $C'$), it is conceivable that for our keypoint-reduction strategy further amplifies this behaviour, that is:

$$\Pr(|A' \cap B'| = |B' \cap C'|) \leq \Pr(|A \cap B| = |B \cap C|) \quad (4.1)$$

Consequently, when an image with a severe alteration is posed as query, it is to be expected that some relevant images are missed. Figure 4.5 also shows a higher precision for all alterations using the T100 method, indicating that our keypoint reduction strategy achieves the level of precision of the original approach.

To further study the effects of keypoint reduction, we vary the threshold levels between the query and indexed images, thereby varying the quantity of local descriptors in the indexed images and that of the queries. This experiment allows us to determine whether the effectiveness of keypoint-reduction approach can be improved by using more keypoints in the
query; that is, using more local descriptors to find matches in a smaller indexed set of local descriptors. Table 4.2 shows the average retrieval effectiveness of 200 queries on collection 20K using three different variations for thresholding the query and test images, namely 1000, 100, and 10; default denotes the original number of keypoints (1500 on average). Columns 1 and 2 indicate the threshold value used for the test (indexed) and query images, respectively. The last column indicates the average number of candidate keypoints and local descriptors that are matched and retrieved from the LSH index during query evaluation.

As shown in this table, using query images with a default number of keypoints to retrieve images that are thresholded with a value of 100 yields a high average recall and precision, of 94.5% and 98.4% respectively, in a little under 5 seconds; this is similarly observed for the use of threshold value of 1000 for the query image, taking on average close to 3 seconds, whereas the original approach has a query run-time of over 2 minutes. We also observe that the number of candidate local descriptors matched in the index is proportional to the number of local descriptors used for querying. Clearly, the number of local descriptors used in the query image can improve the effectiveness of retrieval, but at the expense of a large candidate pool size; this translates to more disk reads from the keypoint table — which contains the local descriptor and keypoint information of all images.

Nevertheless, we observe that the number of candidate local descriptors that are examined for a set of local descriptors in a query image is reduced considerably with a threshold value of 100 from 62,549 to 2,647 on collection 20K while maintaining a competitive level of effectiveness; using the default number of local descriptors on the reduced indexed local descriptor also yields a substantially lower number (7,836) of candidate local descriptors.

This is an important finding given that with this threshold value, the loss of average recall is a small tradeoff when compared to the considerable savings in terms of index size, query run-time, and the number of local descriptors that are examined.

**Retrieval effectiveness on large collections**

To further test the effectiveness of our approach on large collections, we experiment with the T100 method (threshold value of 100) on image collections 150K, 300K, and 1M; we also include results from collection 20K for comparison. We do not perform any experimentation of the original approach on these larger collections as the query run-time and on-disk memory requirements are impractical. For instance, without keypoint-reduction, the size of the index for a collection of one million images is estimated to be close to half a terabyte. We do not
4.1. IMPROVING RETRIEVAL EFFICIENCY BY PRUNING SIFT

<table>
<thead>
<tr>
<th>Collection</th>
<th>Avg. recall(%)</th>
<th>Avg. precision(%)</th>
<th>Avg. query run-time(sec)</th>
<th>Index size(GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20K</td>
<td>91.0</td>
<td>99.3</td>
<td>2.2</td>
<td>1.6</td>
</tr>
<tr>
<td>150K</td>
<td>90.9</td>
<td>99.3</td>
<td>12.2</td>
<td>7.9</td>
</tr>
<tr>
<td>300K</td>
<td>90.7</td>
<td>99.3</td>
<td>21.6</td>
<td>17.4</td>
</tr>
<tr>
<td>1M</td>
<td>90.9</td>
<td>99.3</td>
<td>27.2</td>
<td>57.0</td>
</tr>
</tbody>
</table>

Table 4.3: Average recall (%) and precision (%) of 200 queries on collections of 20K, 150K, 300K, and 1M images using the T100 method. Columns 4 and 5 indicate the average run-time (over 200 queries) and the index size, respectively.

make further predictions pertaining to the query run-time of the original approach.

As shown in Table 4.3, the average recall and precision — of our keypoint-reduction approach — are highly stable across all collections from 20K to 1M, suggesting that the high level of effectiveness can be maintained even in large collections. We observe the average query run-time to be proportional to the size of the image collections. We also find that the differences between the recall and precision values produced on collection of varying sizes are not statistically significant (p-value > 0.05). The query evaluation run-time rises from 2.2 seconds for 20K images to a little under half a minute for collection 1M (one million images). Indeed, a preliminary observation of the trends here indicates that the results conform to our expectation of sublinear time complexity [Gionis et al., 1999] when using the LSH index.

Table 4.4 shows the results for 200 queries of each alteration to retrieve all other relevant images on collections 150K, 300K, and 1M; we also include results from collection 20K for comparison. We observe that the average recall and precision as observed in the small collection of 20K images is reproduced across all larger collections, indicating that the level of effectiveness when increasing the size of image collection is almost lossless. This shows that the effectiveness of our keypoint-reduction scheme is not affected by changes in collection size.

The overall findings for our experiments on the large collections indicate that our keypoint-reduction scheme can scale well to even a million images, where efficiency has been improved considerably from the baseline approach.
### Table 4.4: Average recall (%) and precision (%) of 200 queries on all collections using the T100 method for each alteration. Column 1 indicates each alteration. Recall and precision values are presented in pairs.

<table>
<thead>
<tr>
<th>Alterations</th>
<th>20K images</th>
<th>150K images</th>
<th>300K images</th>
<th>1M images</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorise</td>
<td>91.5 / 95.3</td>
<td>91.5 / 95.1</td>
<td>91.5 / 95.1</td>
<td>91.5 / 94.9</td>
</tr>
<tr>
<td>contrast</td>
<td>89.7 / 95.3</td>
<td>89.7 / 95.2</td>
<td>89.7 / 95.2</td>
<td>89.7 / 95.0</td>
</tr>
<tr>
<td>crop (5-30%)</td>
<td>88.7 / 95.1</td>
<td>88.7 / 95.0</td>
<td>88.7 / 95.0</td>
<td>88.7 / 94.9</td>
</tr>
<tr>
<td>despeckle</td>
<td>92.1 / 95.3</td>
<td>92.0 / 95.2</td>
<td>92.0 / 95.2</td>
<td>92.0 / 95.1</td>
</tr>
<tr>
<td>jpeg_to_gif</td>
<td>91.1 / 95.5</td>
<td>91.3 / 95.4</td>
<td>91.3 / 95.3</td>
<td>91.3 / 95.2</td>
</tr>
<tr>
<td>frame</td>
<td>88.0 / 94.2</td>
<td>88.1 / 94.1</td>
<td>88.1 / 94.2</td>
<td>88.0 / 93.9</td>
</tr>
<tr>
<td>rotation</td>
<td>89.1 / 95.4</td>
<td>89.1 / 95.4</td>
<td>89.1 / 95.3</td>
<td>89.1 / 95.3</td>
</tr>
<tr>
<td>scale-down</td>
<td>57.6 / 98.4</td>
<td>57.7 / 98.4</td>
<td>57.7 / 98.3</td>
<td>57.7 / 98.3</td>
</tr>
<tr>
<td>scale-up</td>
<td>92.4 / 95.4</td>
<td>92.5 / 95.4</td>
<td>92.5 / 95.3</td>
<td>92.5 / 95.1</td>
</tr>
<tr>
<td>saturation</td>
<td>91.8 / 95.4</td>
<td>91.9 / 95.3</td>
<td>91.9 / 95.3</td>
<td>91.9 / 95.1</td>
</tr>
<tr>
<td>saturation (sev)</td>
<td>91.3 / 95.4</td>
<td>91.2 / 95.3</td>
<td>91.2 / 95.3</td>
<td>91.2 / 95.1</td>
</tr>
<tr>
<td>intensity</td>
<td>91.5 / 95.3</td>
<td>91.5 / 95.2</td>
<td>91.5 / 95.2</td>
<td>91.5 / 95.0</td>
</tr>
<tr>
<td>crop (40-90%)</td>
<td>55.5 / 96.8</td>
<td>55.5 / 96.8</td>
<td>55.5 / 96.8</td>
<td>55.5 / 96.6</td>
</tr>
<tr>
<td>intensity (sev)</td>
<td>85.7 / 96.1</td>
<td>85.8 / 96.1</td>
<td>85.8 / 96.0</td>
<td>85.8 / 95.8</td>
</tr>
<tr>
<td>contrast (sev)</td>
<td>72.9 / 97.5</td>
<td>72.9 / 97.5</td>
<td>72.9 / 97.4</td>
<td>72.9 / 97.3</td>
</tr>
<tr>
<td>rotate+crop</td>
<td>74.9 / 95.4</td>
<td>74.9 / 95.2</td>
<td>74.9 / 95.3</td>
<td>74.9 / 95.2</td>
</tr>
<tr>
<td>rotate+scale</td>
<td>81.0 / 96.4</td>
<td>81.0 / 96.3</td>
<td>81.0 / 96.3</td>
<td>81.0 / 96.1</td>
</tr>
<tr>
<td>shear</td>
<td>52.0 / 66.9</td>
<td>52.0 / 65.6</td>
<td>52.0 / 65.8</td>
<td>52.0 / 65.8</td>
</tr>
</tbody>
</table>
4.1. IMPROVING RETRIEVAL EFFICIENCY BY PRUNING SIFT

Comparative evaluation against the DPF method

Finally, to gauge the effectiveness of our keypoint-reduction approach, we compare our keypoint-reduced T100 method, against a previous approach proposed by Qamra et al. [2005] — known as the DPF method — that uses standard colour and texture features, as described in Chapter 2 (page 11). For this particular collection, we use only the 200 queries on image collection 20K; we also include the results for the original PCA-SIFT method for comparison.

The DPF method is somewhat different from the PCA-SIFT method in that each image is quite efficiently represented by one 279-dimensional feature vector; each vector contains image features that are computed using perceptual colour filters and Daubechies wavelets (see Chapter 2, page 64). The application of an approximate indexing scheme — such as the LSH index used for the PCA-SIFT method — have only been preliminarily explored in the work of Qamra et al. [2005]; the impact of false positive or false negative matches as a result of using such schemes on the DPF method is unknown.

We use a simple linear search for the DPF method where every feature vector is computed against a query using the DPF function. This is essentially a brute-force nearest neighbour search, where every image within the collection can be ranked by their computed score to the specified query. In contrast, the results produced by the PCA-SIFT methods are not ranked, as they are only approximated by the LSH index structure. While the list of answers returned by the PCA-SIFT methods can be further ranked, we do not perform such ranking at this stage as the complete list of answers returned by the PCA-SIFT methods has already been shown to be highly accurate with average precision values ≥ 93%. The difference between the two approaches has a minimal impact on the overall comparison, as the list of computed answers (ranked or not), are observed using a pair of recall and precision values. Hence, to fairly evaluate the PCA-SIFT methods against the DPF method, we observe the average precision of each method at the same recall level.

We do not further compare the efficiency of the DPF method against the PCA-SIFT method at this stage, as we expect that even without an approximate indexing structure, the DPF method will outperform the PCA-SIFT methods (even the T100 method) by a wide margin; the DPF method uses only a single feature vector for each image, whereas the PCA-SIFT method uses a number of feature vectors generally in the order of ≥ 10^2.

Before we begin our experiments with the DPF method, we must find an optimal number of bins or dimensions (from the 279-dimensional vector) — denoted by $m$ — for a given image collection. We described in Chapter 2 (page 31) that $m$ is the number of dimensions
Figure 4.6: Precision (%) at every recall level (%) — using the DPF method — averaged over 200 queries for collection 20K. The $m$ parameter is empirically tested using values ranging from 30 to 279, with increments of 30; $m = 279$ is also equivalent to a conventional Euclidean distance comparison.
4.1. IMPROVING RETRIEVAL EFFICIENCY BY PRUNING SIFT

<table>
<thead>
<tr>
<th></th>
<th>PCA-SIFT (Orig.)</th>
<th>DPF (m=240)</th>
<th>PCA-SIFT (T100)</th>
<th>DPF (m=240)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Recall (%)</td>
<td>97.1</td>
<td>97.0</td>
<td>91.0</td>
<td>91.0</td>
</tr>
<tr>
<td>Average Precision (%)</td>
<td>93.1</td>
<td>9.0</td>
<td>99.3</td>
<td>22.0</td>
</tr>
</tbody>
</table>

Table 4.5: Average recall (%) and precision (%) of 200 queries on collections of 20K, using the original PCA-SIFT method, the T100 (keypoint-reduction) method, and the DPF method with the optimal settings of $m = 240$.

with the smallest distance to the query, where setting $m = 279$ is equivalent to an exact $L_2$ distance computation. While Meng et al. [2003] and Qamra et al. [2005] have shown that $m = 180$ yields high effectiveness, they have also noted that this parameter is somewhat collection-dependent; thus we empirically test the $m$ parameter for our collection.

Figure 4.6 shows the precision of the DPF method — using a series of $m$ values (with increments of 30 units) — at every level of recall that is averaged over 200 queries on collection 20K; each query corresponds to 50 image alterations, as described in Chapter 3 (page 79). While we observe that the value of $m = 240$ yields the highest overall effectiveness in terms of the levels of average recall and precision, we also find that using other settings of $m \geq 180$ yield similar levels of effectiveness. More importantly, the results produced by the value of $m = 279$ (standard $L_2$ distance measure) also supports the argument that the standard distance measures computed over typical colour and texture image features are subject to higher sensitivity against image variations and photometric changes.

The overall results in this figure show the level of average precision to be generally poor ($\leq 50\%$) at average recall levels of 80% and above, where a gradual downward trend in average precision is observed beginning at average recall level of about 65%. While we also note that the results produced for our collection are not as good as those in the work of Meng et al. [2003] and Qamra et al. [2005], we observe that the levels of effectiveness are similar. The discrepancies can be explained by our wider variety and more severe image alterations (50 alterations instead of only 40 in their work). Nevertheless, our results show that using the DPF method or a simple $L_2$ distance measure, conventional colour and texture features, approximately 50% of the relevant answers (about 25 of the 50 altered images) can be retrieved with relatively high accuracy.
Using the optimal settings of \( m = 240 \), we also compare the DPF method against the PCA-SIFT approaches (both the original and the T100 method), as shown in Table 4.5. At an average recall level of 97% (as achieved by the original PCA-SIFT method), the average precision is 93.1% and 9% for the PCA-SIFT method and the DPF method, respectively. Similarly the T100 method yields an average precision of 99.3% at 91% recall, whereas the DPF method produces only 22% at this average recall level. Overall, by comparing each approach using the average precision at their respective levels of average recall, we find the PCA-SIFT methods to outperform the DPF method by a wide margin. We also observe that difference in the recall and precision levels to be statistically significant (p-value < 0.05); the results produced by each of the 200 queries is used as a sample.

To further evaluate the effects of individual image alterations on the performance of the DPF method and the PCA-SIFT approaches, we examine the rank of each altered image — corresponding to a specified query — relative to the number of irrelevant image that is returned ahead of it. For instance, using our collection for a particular query, an ideal result should yield 50 relevant altered images that all ranked one, since there should be no other images that are returned ahead of the altered images. Thus any rank (for an altered image) that is lower than one reflects the number of irrelevant images that are falsely returned ahead of the altered image.

For the DPF method, this is straightforward as the ranking is already produced for an entire collection of images. Since the PCA-SIFT methods do not return a complete list of answers, a situation may occur in which the altered image being examined is not returned at all within the result set. Thus, for the PCA-SIFT methods, we penalise these images that are not found by assigning the highest possible rank of 20000, which is also the last position in the collection. While this is a somewhat stringent evaluation — given that any relevant answers that are missed by the approximate indexing scheme can cause an abrupt drop in ranking — it also serves as an indicator as to the percentage of queries in which a given image alteration is not returned.

Figures 4.7, and 4.8 show the relative ranks of images from 10 alteration groups for all 200 queries, using all three methods: the DPF method, the original PCA-SIFT method, and the T100 method. The results are observed using collection 20K, where the 50 alterations arranged into 18 groups (described in Chapter 3, page 79); as in earlier experiments, the results are averaged over the number of distinct alterations in each group. For simplicity, we make an arbitrary decision in selecting only 10 alteration groups to be presented in this section; the results for the remaining 8 groups for all three methods are shown in Figures B.2
Figure 4.7: Average relative rank of the images from each of the 10 alteration groups as produced by the DPF method (using the optimal value of $m=240$). There is a total of 18 groups of 50 image alterations; the remaining 8 groups are further shown in Figure B.2 (page 200).
Not surprisingly, the average relative ranks produced by the DPF method for each of the 10 alteration groups are unsatisfactory, given the relatively low average recall and precision as observed earlier. We observe that only images of alteration `jpeg_to_gif` are top ranked for 70% of all queries. We find that images in the majority of these alteration groups have relative ranks between 3 and 100, which means that using the DPF method, most of the altered images in these 10 groups are returned along with a number of irrelevant images.

Compared to our T100 method and the original PCA-SIFT approach — as shown in Figure 4.8 — the average relative ranks of the images from each alteration group are relatively high. We observe that for the T100 method, the majority of the image alterations in these 10 groups are top-ranked, with only a few rare cases of failure. We find that images of group `scale-down` are top-ranked only a little over 25% of all queries; images from this group are ranked above 6000 most of the time. The other alteration groups for which the T100 method occasionally fails include `rotate+crop` and `rotate+scale`, where images from these groups are only top-ranked about 80% of all queries. We also observe that the average relative ranks produced by the original PCA-SIFT method very rarely fails, where near-perfect average relative ranks are observed for almost all alteration groups. Similar results were observed for the remaining 8 alteration groups for both PCA-SIFT methods, as shown in Figure B.3.

The sudden spikes in the average relative ranks as observed for the PCA-SIFT methods — for certain alteration groups — can be explained by the stringent penalty for alterations that are not returned. For instance, we confirm that the results for the `scale-down` alteration group is mainly affected by the most severe variation of the `scale-down` alteration; that is, the alteration in which an image is scaled-down to one-fourth its original size. Given that the average relative rank averages the distinct alterations within each group, the results for this alteration group are skewed towards low relative ranks.

Overall, the results here clearly indicate that the original PCA-SIFT method and the T100 method considerably outperform the standard DPF method. The results also show that although the DPF method that uses colour and texture features are highly efficient, they are not specifically designed for accurate detection of near-duplicate images. Nevertheless, we believe that given the DPF method uses simple features that can be computed quite efficiently and that approximately 50% of the tested altered images were accurately identified, there is potential for this method to be used as a probable complementary method to the PCA-SIFT methods; this warrants further investigation.
Figure 4.8: Average relative ranks of the images from each of the 10 alteration groups as produced by the PCA-SIFT methods (both the T100 method and the original approach). There is a total of 18 groups of 50 image alterations; the remaining 8 groups are further shown in Figure B.3 (page 200).
4.1.3 Discussion

We have shown that a similar level of effectiveness can be achieved by using approximately 10% (on average) of the SIFT keypoints — which are characterised by PCA-SIFT local descriptors — as compared to those of the original approach. While we have shown competitive levels of effectiveness with only a subset of the features, we note that this observation is limited to the set of alterations as described in Chapter 3 (page 79), and in related prior work [Ke et al., 2004; Meng et al., 2003; Qamra et al., 2005]; these alterations are believed to be common digital alterations amongst near-duplicate images.

Overall, we have observed that the slight decline in recall when using such a severe pruning approach is an inevitable consequence, but that, without such strategies, retrieval on large collections would have been impractical. The scalability issues were apparent in the work of Ke et al. [2004], as the retrieval efficiency was severely limited by the substantial number of local descriptors indexed for each image.

Here, we have made considerable reductions to the bag-of-vectors approach in terms of its spatial and time complexity, while retaining the high level of effectiveness that is achieved in such schemes. We have shown that our proposed scheme has a considerable positive impact on query evaluation. Although we have yet to achieve interactive (sub-seconds) querying evaluation time for large collections, with our proposed improvements to the bag-of-vectors scheme, we have shown that relevant answers can be effectively and efficiently retrieved from a million images in a little under 30 seconds.

4.2 Indexing of local descriptors using a modified RBV

Motivated by the high memory requirements of the LSH indexing scheme, we explore an alternative indexing method known as Redundant Bit Vectors, as described in Chapter 2 (page 57). In this section, we investigate this indexing scheme for high-dimensional local descriptors; it promises a considerable reduction in memory consumption as compared to the LSH index. We also propose some improvements to the original approach.

The application of RBV for local descriptors is straightforward since each local descriptor is treated as a single point within a database. For convenience, we use the terms vector and point interchangeably for discussion of RBV, both referring to a local descriptor of an image. Using a similar scheme to LSH, described in Chapter 2 (page 55), all identifiers to the indexed images are stored in a file table (FT), and each PCA-SIFT local descriptor is mapped to a keypoint table (KT) where each entry contains the location, scale, orientation,
4.2. INDEXING OF LOCAL DESCRIPTORS USING A MODIFIED RBV

and local descriptor of a keypoint. For every local descriptor in a query image — that is, query descriptor — we approximate the potential matching local descriptors using the RBV index and verify the short-listed matched pairs using geometric verification (RANSAC). The key differences are the indexing technique employed (RBV instead of LSH). All query local descriptors and keypoint information is read into memory during query evaluation, and all matched descriptors are fetched from disk.

4.2.1 Limitations of Redundant Bit Vectors (RBV)

The original RBV method was proposed as an indexing technique that allows negative queries that do not correspond to any database items to be quickly searched in a high-dimensional space [Goldstein et al., 2004]. While the proposed indexing method is highly efficient in terms of memory consumption, one of the drawbacks of this technique is the ambiguous nature of negative queries; the role of negative queries for retrieval applications is unclear. Goldstein et al. [2004] defined negative queries as those that are not likely to have a match in the database; they have also shown that RBV can be used to efficiently search the database when posed with negative queries. However, RBV has not been shown to work for positive queries, where a large number of matches are expected.

In the original indexing scheme that we described in Chapter 2 (page 57), Goldstein et al. [2004] show that the best efficiency is achieved when the data points are sorted in ascending order of the most selective dimension (smallest amount of overlap) prior to constructing the bit vectors; here, the term dimension refers to the 36-dimensional space of the PCA-SIFT local descriptor. This organises the RBV index such that the first dimension will have data hyper-rectangles closely located along the axis of its dimension, resulting in tightly packed bits of 1s between the range of the first 1-bit (lo-bit) and the last 1-bit (hi-bit), with all the bits preceding the lo-bit and trailing the hi-bit, are respectively 0s. Since the number of bitwise AND operations can be reduced to the number of integers in this range, the most selective dimension is used as the dimension first queried. The ordering is important since the first dimension always dictates the resultant list of matching vectors (descriptors) within the database — using the bitwise AND operation. Querying the most selective dimension first will quickly narrow the number of potential matching vectors. However, given that each dimension is a very coarse approximation of the distance in the hyper-rectangular space, retrieval effectiveness suffers. In the next section, we show that this RBV index can be modified for positive queries.
4.2.2 Indexing using a modified RBV

We index the local descriptors extracted for each image in the collection, where each local descriptor is a 36-dimensional feature vector. Each vector can be represented by \( k_i(x_1...x_D), i = 1,\ldots,N \), where \( N \) is the total number of local descriptors in the collection, and \( x_D \) is the coordinate of dimension \( D \). During RBV index construction, each point \( k_i \) is mapped to the smallest hyper-rectangle \( c \) that encompasses the hyper-sphere centered on \( k_i \) with a radius of \( \epsilon \) — for \( \epsilon \)-range search. For two PCA-SIFT local descriptors to be deemed a match, an \( L_2 \) norm ranging from 2200 to 3000 (\( \epsilon \)) yields high effectiveness [Ke et al., 2004; Ke and Sukthankar, 2004]. Hence, each high-dimensional descriptor is converted into a hyper-rectangle (or a hyper-cube if given equal sides) with a half-side-length (HCS) of \( \epsilon \) where \( c = 2\epsilon \). To improve the effectiveness of the original RBV indexing scheme, we propose simple modifications to the algorithm to ensure that queries on the modified RBV index maximise recall. We show via experimentation (in later sections) that our modification yields a high level of effectiveness, while retaining the compactness of the original RBV indexing scheme.

Instead of performing the partitioning with a user-defined number of \( Q \) partitions — as with the original approach — we use \( Q_i = (x_{i\text{range}})/\epsilon \) partitions to create the desired number of disjoint intervals that cover the entire axis of a single dimension, where \( x_{i\text{range}} = x_{i\text{max}} - x_{i\text{min}} \) for dimension \( i \). The range of the values within a particular dimension can be used to define the most suitable number of partitions. For instance, using the original approach, where the number of user-defined partition is fixed for every dimension, there is a possibility that every element within that dimension falls within the \( \epsilon \) distance to the query vector. Thus, regardless of the number of partitions, every bit within the \( Q \) bit-vectors will be set to 1 as they overlap. In that case, it would be unreasonable to divide that dimension to more than a single partition.

The choice of value for HCS is critical given that it determines the granularity of our partitioning scheme. To create the partitions, we first sort the boundaries of all data hyper-cubes in a given dimension along its axis; each partition is then represented using a bit-vector where each bit reflects the index position of a given descriptor in a keypoint table (KT as described earlier). Following Goldstein et al. [2004], we then select \( Q - 1 \) dividers from the sorted hyper-cube boundary values, and partition the dimension using the overlap test for each interval, where each bit in a bit-vector reflects the predicate (1 or 0).

As with the original approach, each bit vector is represented using an array of integers, where each integer can store up to 32 (or 64) vectors depending on the system architecture.
Algorithm 5 A summary of the process for building the modified RBV.

Require: Database of \(N\) \(D\)-dimensional descriptor \(x\), \(\epsilon = \frac{\epsilon}{2}\), \(Q_i\) partitions for each dimension \(i\).

Create and initialise boundary array \(B[D \times Q_i]\), and temporary array \(T[2N]\).

\[
\text{for } i = 0 \text{ to } D - 1 \text{ (every dimension in a descriptor) do}
\]

\[
\text{for } k = 0 \text{ to } N - 1 \text{ (every descriptor in the database) do}
\]

\[
\begin{align*}
T_{2k} &= x_{ki} + \epsilon, \\
T_{2k+1} &= x_{ki} - \epsilon.
\end{align*}
\]

\[
\text{end for}
\]

Sort \(T\).

\[
\text{for } k = 0 \text{ to } Q_i - 2 \text{ do}
\]

\[
B_{ik} = T_{\left\lceil \frac{2kN}{Q_i} \right\rceil} \text{ (select } Q_i - 1 \text{ dividers from sorted boundaries)}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

Create \(Q_i\) bit-vectors \(BV\) one dimension at a time using overlap tests as follows:

\[
\text{for } i = 0 \text{ to } D - 1 \text{ (every dimension) do}
\]

\[
\text{Initialise and set all bits in } BV[Q_i \times \left(\frac{N}{32}\right) \text{ machine words}] \text{ to 1 for dimension } i
\]

\[
\text{for } k = 0 \text{ to } N - 1 \text{ (every descriptor) do}
\]

\[
\text{for } j = 0 \text{ to } Q_i - 2 \text{ do}
\]

\[
\text{if } x_{ki} + \epsilon \leq B_{ij} \text{ then}
\]

\[
BV_{j+1,k} = 0
\]

\[
\text{end if}
\]

\[
\text{if } x_{ki} - \epsilon \geq B_{ij} \text{ then}
\]

\[
BV_{jk} = 0
\]

\[
\text{end if}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

Store boundary array \(B_i\) and \(BV\) bit-vectors for each dimension \(i\) to disk.

Clear \(BV\) for subsequent dimensions.
CHAPTER 4. EFFICIENT NEAR-DUPLICATE IMAGE RETRIEVAL

(machine word size). The bit vectors are constructed in memory, and written to disk one dimension at a time. Each bit vector is stored using an array of $N/32$ integers using four bytes each, where $N$ is the total number of local descriptors in our collection. We also store $Q-1$ dividers (each represented with 4 bytes) for the axes of each dimension that is used for query evaluation. Thus, the size of the index for the entire image collection is approximately:

$$\sum_{i=1}^{D} Q_i \left( 4 \times \frac{N}{32} + 4 \times (Q_i - 1) \right) \text{ bytes}$$

where $D$ is the number of dimensions; in our application $D$ is 36.

As our aim is to maximise the number of positive matches, the index is modified to be less restrictive for this application. In our scheme, a further modification to the original approach is that the most selective dimension is not pre-determined during indexing, and requires no prior sorting of the data points; consequently, since there are no trailing or leading zeroes to keep track of, we do not apply the lo-bit and hi-bit for bit-vector processing. Our approach simplifies index construction of the original RBV scheme considerably. The algorithm for the process for constructing our proposed modified RBV index is provided in Algorithm 5.

4.2.3 Querying the modified RBV index

Instead of querying with the most selective dimension during index construction, we determine the order of dimensions dynamically during query evaluation, thereby eliminating the need to pre-process the data points. For each element $x_i$ of a query descriptor, we determine the normalised distance to mean using $|x_\mu - x_i|/x_\mu$, where $i$ is a specific dimension of the descriptor and $x_\mu$ is the mean value of that dimension in all descriptors. We sort the distances in ascending order, and use the sorted order of dimensions for query evaluation. The dimensions are thus dynamically selected to maximise the potential descriptor matches to the query coordinates. In our approach, the search space is not immediately pruned with the dimension first queried — as was the case with the original approach of using the most selective dimension — but is instead narrowed progressively by processing each subsequent dimension.

During query evaluation, the required partition for each dimension can be calculated in memory by using the $Q_i - 1$ dividers of each dimension $i$ to determine which partitions to retrieve from disk. Given that each bit-vector is bitwise ANDed one dimension at a time, and that the ordering of dimensions can be pre-processed, we can bulk-process the query descriptors simultaneously. This is achieved by using a temporary resultant bit-vector in
Algorithm 6  A summary of the process for querying the modified RBV index.

Require: Database of $N$ $D$-dimensional descriptors $x$, $M$ query descriptors $q$ from query image, boundary arrays $B[D \times Q_i - 1]$ for all $i$ dimensions, completed bit-vectors $BV[D \times Q_i \times \frac{N}{32}]$ for all dimensions $i$.

Create and initialise $M$ (one for each $q$) temporary resultant bit-vectors $R[M \times \frac{N}{32}]$, and temporary container $T[D]$.

for $k = 0$ to $M - 1$ (every query descriptor) do
  for $i = 0$ to $D - 1$ (every dimension of the query descriptor) do
    $T_i = |q_{k\mu} - q_{ki}|/q_{k\mu}$ (calculate normalised distance to mean)
  end for
  Sort $T$ in ascending order and determine the order of dimensions.
  for $i = 0$ to $D - 1$ (or $< D - 1$ if pruned) do
    $j = \text{smallest index such that } q_{ki} < B_{ij}$
    for $n = 0$ to $\frac{N}{32} - 1$ (once for every machine word) do
      $R_{kn} = R_{kn} \& BV_{ijn}$ (bitwise AND operation)
    end for
  end for
  for $n = 0$ to $N - 1$ (for every descriptor in database) do
    if $R_{kn}$th-bit is 1 then
      Perform $L_2$ distance verification between descriptors $x_n$ and $q_k$
    end if
  end for
end for
memory for each query descriptor; hence, the order in which the required partitions are read can also be sorted to allow sequential access to disk. We can further enhance the efficiency of this approach by bulk-processing the vectors (descriptors) in memory due to our keypoint-pruning strategy — as described in Section 4.1 — as the number of local descriptors of the query images are pruned; without which the memory requirement would be considerably higher. A summary of the process for querying the modified RBV index is detailed in 6.

Compared to the original RBV indexing scheme [Goldstein et al., 2004], we trade speed for effectiveness, in that we maximise the number of potential matches (the candidate pool) and bitwise AND the entire bit-vector. This results in a larger number of false positives in the pool of candidate descriptor matches, and consequently results in more computation. To further reduce the computational cost of the bitwise operations, we prune the number of processed dimensions during query evaluation to narrow the search space gradually while minimising the number of false negatives. We conjecture that there is a cut-off point at which the processing can be halted; that is, we predict that only a subset of dimensions is needed for bit-wise processing to return the points corresponding to a specified query.

The number of dimensions to prune depends on the partition granularity (HCS) since these two parameters are coupled: a change in one parameter will inevitably affect the other. In Section 4.2.5, we empirically evaluate the effects of varying HCS and the number of dimensions pruned on retrieval speed and effectiveness in

4.2.4 Experimental setup

Here, we compare our approach — keypoint-reduced local descriptors with the modified RBV index — against the approach of Ke et al. [2004] and our keypoint-reduced approach with the LSH index. We use the modified RBV index on only local descriptors with a threshold of 100 (the T100 method). We evaluate both efficiency and effectiveness of these approaches on collection 20K. For our queries, we select 200 images from the seed collection, as described in Chapter 2 (see page 74). We also investigate the number of vectors (or descriptors) that are processed during query evaluation. We use a similar framework to that of Ke et al. [2004], as described previously in Section 4.2. We similarly use the keypoint table (KT) to store all local descriptors and a file table (FT) to map the descriptors to their corresponding images, where geometric verification (RANSAC) is performed when descriptors are returned from the modified RBV index structure. The main difference is the index structure and the amount of PCA-SIFT features used.
4.2.5 Results

Here, we present our results of the experiments on retrieval effectiveness of the RBV index on our dataset. We further discuss the effects on query performance of varying the RBV parameters.

Retrieval accuracy

In Figure 4.9, the effectiveness of the RBV index is measured using average recall and precision; all measured values are averaged over 200 queries. We experiment with varying the HCS parameter — from 1 000 to 3 000 — and the number of pruned dimensions; each increment of four dimensions is shown. We use the LSH approach — using both the original number of local descriptors, and our keypoint-reduction approach (T100 method) — as baselines with average recall of 97% and 90%, respectively; the average precision at these recall levels are 93% and 99%, respectively. The highest observed average recall and precision with RBV is 94% and 97% respectively, with an HCS of 2 000 only after eight dimensions are processed. As expected, using an HCS of 1 000, we observe a dramatic drop in recall and precision if more than one dimension is processed, which suggests that the boundaries in the hyper-cube space are too tight, resulting in high partition granularity.

With an HCS of 1 500, recall remains at 91% and precision at 98% after only 4 dimensions have been processed. This is interesting since we expected the HCS value that would yield good effectiveness to be closer to the PCA-SIFT $L_2$ distance threshold of 3 000. We do not further prune the number of dimensions for higher ranges of HCS — from 2 000 to 3 000 — as we observe the level of effectiveness to remain relatively stable even without pruning. For instance, an HCS of 2 500 achieves average recall and precision of 86% and 99% respectively, even after processing 32 dimensions; we also observe that after processing all 36 dimensions using HCS of 3 000, average recall and precision remain at 85%, and 99%, respectively.

We observe that given a large enough HCS, the drop in recall and precision is less abrupt, since the majority of the answers are still within the hyper-cube boundary of a single dimension, resulting in fewer eliminated matches. The results here show that our proposed modification for the RBV index is indeed effective; the findings also corroborate our hypothesis, that all number of dimensions do not have to be fully processed to yield high effectiveness. The observations here suggest that the number of dimensions to be processed can be pruned to yield higher efficiency — given that fewer bit-vectors have to be fetched from disk and bitwise ANDed — and at the same time produce satisfactory results. It also clearly shows
Figure 4.9: Average recall (%) and precision (%) (over 200 queries) of the RBV index for variations of HCS and number of dimensions; dimensions refer to the 36-dimensional space of the PCA-SIFT. LSH is the baseline.
4.2. INDEXING OF LOCAL DESCRIPTORS USING A MODIFIED RBV

Figure 4.10: Average run-time for querying the RBV index (over 200 queries) for variations of HCS and number of dimensions. LSH is the baseline.

that the choice of HCS is critical for the effectiveness of the modified RBV index.

Retrieval efficiency

To further analyse the efficiency of the modified RBV index, we measure the average elapsed run-time for evaluating a single query. It is also instructive to examine the reduction in the search space, that is, the average number of examined local descriptors under different settings of the HCS values, and the number of pruned dimensions. All numbers are reported as an average over 200 queries, where all disk reads are linearised to minimise random disk seeks.

We compare these against the LSH baselines, which we observe to have average running times of approximately 150 and 2 seconds respectively for the original approach and that of the keypoint-reduction approach. Since the implementation of LSH in both approaches, and our RBV implementation, are yet to be optimised — in terms of in-memory data structures — we do not have high confidence in the degree of improvement over the baselines. We elaborate on this issue later in Section 4.2.5. The timing results for query evaluation using
the RBV index are presented in Figure 4.10; Figure 4.11 shows the total number of local descriptors that are fetched from disk when querying from the modified RBV index. The 20K collection produces a total of almost 2.6 million local descriptors.

Overall, we observe that the modified RBV index is clearly more efficient than the LSH index using the original number of local descriptors — the approach of Ke et al. [2004] — but it is less efficient than the LSH index using the reduced set of keypoints, as there is a considerable difference in the average query-run time. This is also within our expectation, given that Ke et al. [2004] use considerably more local descriptors than we do in our keypoint-reduced approach.

A smaller candidate pool translates to higher efficiency, since fewer descriptors need to be fetched from disk and examined. We observe that the run-time generally reduces as more dimensions are processed, and the pool of candidate matches becomes smaller, requiring fewer local descriptors to be examined. This is evident from the much higher running time of 150 seconds with an HCS of 1500 and processing of only one dimension, where it is effectively reduced to an on-disk sequential scan. The sequential scan always requires the worst-case
number of local descriptors, as it performs a brute-force search to find \( k \)-nearest-neighbours. Using an HCS of 1 000, we observe that there is a slight increase in running time from 7 to 13 seconds when the number of dimensions is more than 12. This can be explained by the increased computational cost of processing — including CPU operations required for bitwise ANDing and fetching bit vectors from disk — more dimensions without a corresponding decrease in the number of local descriptor pairs.

This behaviour is evident in the comparable run-time between an HCS of 3 000 for processing all 36 dimensions, and an HCS of 1 500 after processing only four dimensions. When processing four dimensions using an HCS of 1 500, we observe that approximately two million local descriptors are fetched from disk, whereas only an estimated half a million descriptors are fetched using an HCS of 3 000 after processing 36 dimensions. This shows that while processing more dimensions can more effectively reduce the search space, it also has a negative impact on query run-time. Clearly, it is critical to find an optimal setting that balances between effectiveness and efficiency when using the modified RBV index.

As we can see in Figure 4.11, using an HCS of 1 000, the candidate pool is dramatically reduced to less than 80 000 after processing only eight dimensions; but the same settings were also shown to not return any descriptor matches, with 0% for both average recall and precision. Indeed, this corroborates our earlier observation that an HCS of 1 000 is too small, as it indicates that only a few dimensions of local descriptors can be matched within a hypercube side-length of 1 000. For the range of HCS values of 1 500 to 3 000, we observe that a gradual decline in the number of examined local descriptors as we process more dimensions. However, we also observe that using a larger HCS value, more dimensions need to be processed for the search space to be reduced. For instance, with an HCS of 3 000, the number of local descriptors that are fetched from disk remains at approximately 1.9 million even after processing 32 dimensions. This indicates that a large HCS value is counter-productive, since pruning cannot be effectively applied when individual dimensions do not discriminate against the false positive descriptor matches. These results suggest that the modified RBV index cannot efficiently reduce the search space without sacrificing effectiveness.

Figure 4.11 further shows that the modified RBV indexing method is not as effective as the LSH method in reducing the search space of a given query; the LSH approach using the original number of local descriptors, and the keypoint-reduced variant require, 62 549 and 2 647 local descriptors to be examined, respectively. This indicates that the candidate pool size is considerably smaller than that of the modified RBV index; at best, our modified RBV indexing method improves upon only the sequential search method in terms of search
space reduction. We find that, overall, using an HCS of 1500 after processing only eight dimensions provides the best balance between effectiveness and efficiency, with average recall and precision of 87% and 99%, respectively, and an average run-time of 23 seconds. We observe that the difference in the observed level of recall and precision between the RBV index and that of the LSH index is statistically significant (p-value < 0.01).

Related studies

To gain further insights into the modified RBV indexing scheme, we study the effects on query run-time when the local descriptors are stored in main memory, where no disk reads are required. For this experiment, all keypoint-reduced PCA-SIFT local descriptors are read into memory; only the RBV index structure is stored on disk. Since, the local descriptors are now memory-resident, the disk activity is restricted to fetching only the bit-vectors. To gain a better understanding of the computational cost of query processing, we compare the query run-time between the on-disk and in-memory implementations of the modified RBV index; we do not make further comparisons against the LSH indexing scheme. The reported

Figure 4.12: Average query run-time (over 200 queries) for the RBV index using memory-resident local descriptors. Sequential scan is the baseline.
results are averaged over 200 queries.

As shown in Figure 4.12, similar query timings to the on-disk approach are observed for all HCS parameters; on average, only a 33% reduction in query run-time is observed. This is an interesting finding, implying that the time spent for local descriptor processing in main memory constitutes a large fraction of the total cost; only about 33% of the time is spent on retrieving the local descriptors from disk. This corroborates our earlier observation that a large proportion of the time is spent on fetching the bit-vectors from disk, and the bitwise processing of each bit vector. This is a pleasing result as it indicates that there is ample room for improvement, as the time spent on fetching the bit-vectors from disk can be further reduced or eliminated altogether by storing the modified RBV index in memory; this warrants further investigation in future work.

Table 4.2.5 shows the sizes of the modified RBV index along with its HCS value for collection 20K. This clearly shows that relatively modest memory requirements of the modified RBV index. It also shows that HCS dictates the number of partitions, which determines the number of bit-vectors that are stored on disk. It is also interesting to note that using the baseline method as described by Ke et al. [2004], and without our keypoint-reduction approach, the size of the index is considerably larger than that of RBV.

As a final test to gauge the performance of our modifications to the RBV indexing scheme, on a larger collection of 40 000 images (doubling the 20K collection with images from the SPIRIT collection), we measure the factors of growth in the number of candidate vectors, and the query run-time. We observe that the level of effectiveness and query run-time for the larger collection remain stable across all HCS values. As shown in Figure 4.13, we observe that the candidate pool size increases by a factor of only 1.4 (by processing 16 dimensions), even though the collection size increases by a factor of two from 20 000 to 40 000 images; all observations are based upon an HCS of 1 500. As expected, processing only a single dimension results in double the number of candidate local descriptors and query run-time.
Interestingly, this figure also indicates that while the growth factor of the candidate pool size shows a downward trend, the query run-time shows an opposite trend. This means that although the decline in the candidate pool size has reduced the number of local descriptors that need to be fetched from disk, the query run-time is not proportional. This behaviour suggests that, if a large number of dimensions are processed, the query run-time is largely dictated by the amount of in-memory processing that is required, which implies that there is potential for optimizing the in-memory data structures.

Discussion

As mentioned previously, neither our RBV implementation nor the LSH approaches are optimised in terms of in-memory data structures; we believe that the efficiency of these approaches can be further improved. Furthermore, we conjecture that later implementations of LSH [Bawa et al., 2005; Datar et al., 2004] can readily be applied to this application and could further improve performance in terms of effectiveness and efficiency. We have shown
that our proposed modification on the RBV index yields high effectiveness for retrieval of near-duplicate images, given suitable parameters. We also note that our proposed scheme is not limited to our application of near-duplicates, and that this scheme can be similarly applied for any high-dimensional feature vectors.

While we have also demonstrated that the RBV index is highly compact compared to the LSH indexing scheme, we have also clearly shown that — even with our proposed modification — the RBV indexing scheme is not as efficient as the LSH indexing scheme when used with the keypoint-reduction strategy, as described in Section 4.1. This is not surprising given that the main drawback of the RBV index is its guaranteed linear query run-time and memory complexity [Goldstein et al., 2004].

Nevertheless, our proposed modification to the RBV index, is the first to show the effectiveness of RBV for positive queries; we have also shown that using the RBV indexing scheme for the bag-of-vectors approach — with our keypoint-reduction strategy — yields a considerable speed-up over the original approach of Ke et al. [2004]. An in-depth study of the RBV index is warranted to further improve the efficiency of this approach on larger collections; we do not experiment with this indexing scheme further in this thesis.

4.3 Summary

We have presented a pruning approach to near-duplicate image retrieval using reduced SIFT keypoints (on difference-of-Gaussian regions) that are characterised by PCA-SIFT local descriptors. We have shown that our pruning strategy allows images to be compared with considerably higher efficiency, achieving comparable effectiveness in terms of recall and precision. We have shown that our keypoint-reduction method outperforms previous approaches for near-duplicate detection, providing a good balance between effectiveness and efficiency. With our proposed keypoint-reduction strategy, scalability issues with application of bag-of-vectors (local descriptors) for near-duplicate image detection can be made less apparent, scaling well to even large collections of one million images.

We have also described a more compact indexing structure — the modified RBV index — that serves as an effective alternative method of indexing and retrieval of near-duplicates in moderate-sized image collections. We have demonstrated that even though the RBV index was not initially designed for positive queries, it can be modified to cater for such tasks. We have shown that this indexing scheme performs as well as the LSH index — given suitable parameters — in terms of effectiveness and runs in a little under ten seconds on average for a
single query, on a moderate sized collection of images. The RBV index is also very compact in terms of index storage. In the next chapter, we describe detection of near-duplicates using a different approach in which a query is not required, known as the non-query-based detection.
Chapter 5

Automated Discovery of Near-duplicate Images

Thus far, we have discussed only the detection and matching of near-duplicate images from the perspective of a query image, where images are retrieved based on their likelihood of being near-duplicates with respect to a given example image. In the previous chapter, we proposed a keypoint-pruning strategy for retrieval of near-duplicate images in response to a query, and described an alternative method for indexing moderate-sized image collections using the RBV index. For some applications, we may wish to find all near-duplicate image pairs in a given collection, as we believe reliable detection of near-duplicates without a query example is valuable for search efficiency, collection management, and copyright protection. This presents a challenging many-to-many discovery problem, as compared to the one-to-many search problem of query-based methods.

In this chapter, we investigate a non-query-based approach of detecting near-duplicates in a collection of images, with the aim of identifying the near-duplicate relationships between images in a collection without any query examples. We propose a novel method for automatically identifying the near-duplicate images in a large collection. Our approach is based on analysis of the LSH index generated for the PCA-SIFT local descriptors, which we have previously shown to be effective and compact. We also note that, using this approach, it is possible to adapt various interest point detectors and local descriptors — as described in Chapter 2 (pp. 41—46) — for our approach; but we do not investigate these adaptations at this stage.
5.1 Related work

Little prior work addresses how to sift a collection to find all near-duplicate pairs of images. As described in Chapter 2 (page 64), the Replicate IMagE Detector (RIME) proposed by Chang et al. [1998] is designed to address this issue using a cluster-based approach. However, the severity of the image alterations they study is limited. Zhang and Chang [2004] propose a framework to identify near-duplicate images using machine learning by graph matching (see Chapter 2, page 64). They present an application of this detection for topic and semantic association, where they observe low effectiveness on a small collection of six hundred TRECVID video keyframes; efficiency and scalability remains an issue. Moreover, the approach we describe in this chapter is not a clustering approach, but a pairwise detection method, since we do not investigate cluster formation heuristics; we discuss a clustering approach in Chapter 6 (page 160).

5.2 The discovery problem

The problem of automatic identification of pairs of near-duplicate images can be conceptualized using a relationship graph [Bernstein and Zobel, 2004], where each node represents an image, and an undirected edge between two nodes reflects a near-duplicate relationship between the images. Discovery of the relationships in a given collection is a challenging task due to the quadratic number of potential edges (image pairs) [Zhang and Chang, 2004]. The goal of the discovery process is to find an efficient solution for discriminating between near-duplicate and unique images within a given collection.

As described in Chapter 2 (page 11), instances of near-duplication in text documents can be detected by parsing them into representative units of words or characters that can be indexed in an inverted file [Zobel and Moffat, 2006]; each entry contains the postings list of documents identifiers (IDs) in which each particular unit occurs, along with any auxiliary information. Near-duplicate text-document detection algorithms exploit the postings list to generate the relationship graph; the principal differences between algorithms reported in the literature lies in their unit selection heuristics [Bernstein and Zobel, 2004]. Here, we borrow the concepts of near-duplicate text-document detection [Bernstein and Zobel, 2004; Broder et al., 1997; Shivakumar and Garcia-Molina, 1998], and adapt them for our application.

To generate the relationship graph of an image collection, we adopt the refine-and-filter scheme as proposed by Shivakumar and Garcia-Molina [1998]. Their approach uses hash
tables to quickly discard most of the unrelated pairs of text documents, and further processes
the pruned collection to discard remaining false positives. This avoids the inefficiency inherent
in quadratic-cost comparison of every pair of text document. This approach is also known as
hash-based probabilistic counting, which we adapt for identification of pairwise near-duplicate
images.

5.3 Deriving the relationship graph using local descriptors

Hash-based probabilistic counting is suited for our application domain because of the charac-
teristics of local descriptors. In contrast to colour and texture features, which are commonly
represented as a single descriptor (feature vector) — as described in Chapter 2 (page 16)
— each image usually generates a large number of local descriptors, which are highly dis-
tinctive, and are likely to be found within images that share certain geometric properties.
Although the approach we describe in this section is not limited to a particular type of local
descr iptor, we have chosen to use the PCA-SIFT local descriptors as they strike a balance
between effectiveness and efficiency, as shown in Chapter 4 (page 92).

Using the LSH indexing method, the post-indexing phase of local descriptors of an image
are mapped to their corresponding hash-keys across a series of LSH indexes (or hash tables)
such that two features sharing identical hash-keys are, with high confidence, close to each
other in a high-dimensional space. As described in Chapter 2 (page 55), all local descriptors
(or points) that collide within a given LSH hash table are approximated, by the $L_1$ distance
(embedded in the Hamming space); thus, colliding points are estimated to be closer to each
other (than other points) within the $L_1$ norm. For retrieval tasks — such as those described
in Chapter 4 (page 92) — an additional $L_2$ distance verification stage is used to discard false
positive matches under the $L_1$ norm, where matching pairs are further processed using the
RANSAC geometric verification technique. For pairwise detection, however, any additional
verification imposes processing overhead, which can be substantial and impractical when
applied with a large number of images. Thus, we posit that matches within the $L_1$ norm are
sufficient, and that some false positives can be tolerated.

Once the local descriptors of all images are indexed, we analyse the LSH indexes to de-
termine the number of co-occurring hash values for any two images. As shown in Figure 5.1,
each hash-key can be treated as a unit, where an image is transformed into a series of repre-
sentative units that can be stored in a postings list of an inverted file, akin to words in text
documents. Thus, each entry consists of a list of images that contain local descriptors with
Figure 5.1: Process of mapping a bag-of-points (local descriptors) from an image to a series of representative units. Each high-dimensional descriptor is mapped to the same number of units as the number of LSH indexes; where each unit essentially represents a hash-bucket of one LSH index, or the neighbourhood within the high-dimensional space. Hash-based probabilistic counting can be used on these generated units to approximate the number of matching local descriptors between two images efficiently.
common (identical) hash-keys. Near-duplicate images are likely to produce local descriptors that will generate identical hash values in a given index; we hypothesise that processing the hash table provides an efficient mechanism for identifying candidate near-duplicate image pairs.

As in the relationship graph generation approach of Shivakumar and Garcia-Molina [1998], all possible image pairs in every entry of the inverted list can be rapidly hashed to an array $A$ of $m$ counters using a universal hash function $h$. For a pair of images with identifiers (ID) of $\text{imgID}_1$ and $\text{imgID}_2$ that share co-occurring entries (units) in the inverted list, $A[h(\text{imgID}_1, \text{imgID}_2)]$ is incremented; this is essentially a *probabilistic* hash-counter.

Due to hash collisions, this hash-counter can occasionally generate spurious edges (false positive pairs) due to hash collisions between image pairs with no co-occurring descriptors, given that they share identical hash locations within an LSH index. This phenomenon is typically observed with high probability when incommensurate hash-counters are used; that is, when the number of images is considerably larger than the number of hash-counters, the probability of two unrelated images being hashed to the same location is higher. However, given that we can discard image pairs without a minimum number of matching descriptors, we are only concerned with values lower than or equal to a specified threshold $T$; thus, a small range of threshold values ($T$) can be implemented such that the width of a hash-counter is upper-bounded to reduce the effects of hash-collisions. The number of hash-counters can be considerably increased by using byte-sized hash fields, or smaller multiples of bits, rather than typical 32-bit machine word. For instance, a 500 MB of main memory can accommodate approximately 520 million byte-sized hash-counters instead of an estimated 130 million counters for an integer-sized hash field. Thus, the rate of hash collisions can be reduced with limited impact on memory usage, and substantial numbers of image pairs can be processed with relatively small amounts of memory, and therefore avoid potential exhaustion in memory resources.

After the image pairs have been approximately identified using probabilistic hash-counters, matching features from a pool of reduced image pairs — that are estimated to share at least a number of matching descriptors equivalent to the threshold — can be counted exactly; this is done to further minimise the number of spurious edges. We accumulate the matching features between two images using exact counting as similarly described in the work of Shivakumar and Garcia-Molina [1998], that is, we keep an integer counter for every unique image pair using a static structure with no possibility of collision, thereby produce a list of $<\text{imgID}_1, \text{imgID}_2, \text{counter}>$ triplets. As such, each counter reflects the actual number of local descrip-
tor matches between any two images in the series of LSH hash tables.

Even with the keypoint-reduction strategy we presented in Chapter 4, the potential number of matching descriptors between two near-duplicate images can reach tens or hundreds, so we apply the same thresholding ($T$) as the probabilistic hashing described earlier, to discard edges without a minimum number of matching descriptors. Thus the first-pass hash-based counters can efficiently approximate the probable local descriptor matches between the images, and a more precise estimation can be obtained by processing a smaller pool of images that have matching local descriptors above a certain threshold. The selection of this threshold value is critical, as it dictates the number of false positive and false negative edges that are detected in the two stages; too small a threshold value could result in a potentially large number of edges, whereas a large threshold value could be overly stringent. We empirically investigate the optimal threshold for identifying all near-duplicate images, while omitting as many false matches as possible.

The complexity of this process, is in the order of $O(l \times M \times B^2)$; where $l$, $M$, and $B$ are respectively, the number of LSH indexes (or hash tables), the size of each hash table, and the maximum number of hash-buckets in each entry. Each entry in the LSH indexes is examined first for approximate hashing, and then for exact counting of descriptors. As we discussed in Chapter 2 (page 51), $l$ is the user-defined number of hash tables, and $M$ is dynamically determined using $n$ number of points (or local descriptors) by $M = \frac{n}{\alpha B}$; $\alpha$ and $B$ are user-specified variables. Here, we use $B = 20$ and $\alpha = 0.5$ as shown by Gionis et al. [1999] and Ke et al. [2004] to yield high effectiveness, and we empirically test different $l$ values. Although the cost of resolving all image pairs within a single entry is quadratic $O(B^2)$, this is not prohibitive in practice, as variables $B$ and $l$ have been empirically observed to yield high effectiveness for near-duplicate detection at relatively small values [Ke et al., 2004]. This implies that, while $l$, $B$, and $M$ should grow proportional to the size of the collection, variables $l \ll M$ and $B \ll M$ grow at a considerably slower rate than that of $M$, and therefore the impact of these variables is less severe.

5.4 Experimental setup

For our image collection, we use all four image collections (20K, 150K, 300K, and 1M), and the collection of web images with manually evaluated near-duplicates, as described in Chapter 3 (page 82). Throughout this chapter, we also use a 40K collection created by adding an additional 20,000 images from the SPIRIT collection to the 20K collection. This particular
collection is used to study the effects of doubling the moderate-sized collection. We use the coverage and average precision measures, described in Chapter 2 (page 71), to evaluate the effectiveness of the discovery experiments using hash-based probabilistic counting.

The measures of coverage and average precision require a complete reference graph, against which to compare the edges generated by the algorithm. Using the seed collection, we can generate reference edges using the predefined near-duplicate relationship of each image pair. The 200 groups in our seed collection, each with 50 near-duplicate images will generate \(200 \times \frac{50 \times 49}{2} = 245,000\) edges. We can then evaluate the coverage of the returned relationship graph as a ratio of pre-determined edges that are identified in the reference graph. We also test the effectiveness of the adapted hash-based probabilistic algorithm for each individual alteration using the ratio of pre-defined edges that are identified in the reference graph. The seeded images are essential for evaluating the effectiveness of our approach on identifying severely altered near-duplicates; moreover, they serve as a suitable testbed for testing the limits of various threshold values to determine an optimal setting. For our experiments, we use byte-sized hash fields, and empirically test seven threshold values \(T\) doubling progressively from 4 to 255 and measure the performance of the algorithm using coverage and average precision for each collection. This way, we can meaningfully evaluate the accuracy of our algorithm-generated graph in identifying the seed collection.

To evaluate the effectiveness and efficiency of the probabilistic counting algorithm, we experiment with the PCA-SIFT local descriptors for both the original approach and the keypoint-reduced approach we described in Chapter 4. We also evaluate timing results, and observe the number of algorithm-identified edges using the adapted hash-based probabilistic counting approach; we henceforth refer to this approach as the HPC algorithm.

5.5 Results

Here, we discuss our the experimental results of the HPC algorithm on various image collections using the testbed as described previously.

5.5.1 Effectiveness of automated discovery

As our first study, we use the original approach of PCA-SIFT local descriptors; we discuss the same study for our keypoint-reduction approach later in this section. We explore the effectiveness of these approaches using seven threshold values \((T)\), doubling progressively from 4 to 255 for each collection of 20K and 40K as described earlier. As shown in Figure 5.2,
using the HPC algorithm with the original number of PCA-SIFT local descriptors, the seeded
near-duplicate images are detected with high overall effectiveness for both collections.

As anticipated, we find that coverage favours low threshold values whereas precision
favours higher threshold values. With low threshold values, the minimum matching local
descriptors for two images is insufficient to rise above the noise level, resulting in low average
precision. For instance, using a threshold $T = 4$ for collection 40K, the coverage and average
precision are observed to be approximately 93% and 17%, respectively; whereas a threshold
of $T = 32$ yields coverage of approximately 85% and 91%, respectively. Similar coverage and
average precision values were observed across two collections.

Evidently, a small number of local descriptor matches between two images is insufficient
to discard non-near-duplicate pairs, due to spurious edges that are generated by images
that share similar image regions, corners, or image objects. However, as the threshold value
increases, the overall coverage exhibits a downward trend, indicating that some near-duplicate
images are missed during the detection. This implies that, occasionally, the number of shared
local descriptors between some near-duplicate images (after severe alterations) does not rise
above the noise level. This observation was expected given that the digital alterations are
automatically generated without knowledge of image content; hence, operations such as severe
cropping, shearing, or scaling may cause images to appear unrelated even to the human
observer; this was manually confirmed upon further investigation. The detailed results of
coverage and average precision for our experiments on each individual alteration using both
collections 20K and 40K can be found in Tables C.1 (page 203) and C.2 (page 205). Indeed, a
pleasing result from this study, in terms of effectiveness, is the relatively consistent coverage
and average precision of the HPC algorithm over two image collections of varying sizes; it
appears that doubling the collection size has little effect on the overall coverage and precision.

We observe some interesting trends in relation to the run-time of the HPC algorithm, and
the number of HPC-identified edges, for each collection. Figure 5.2 also shows the number
of identified near-duplicate edges, and the run-time of the HPC algorithm for each threshold
value. We observe that the number of edges is reduced dramatically when the threshold value
is increased from $T = 4$ to $T = 255$; this especially pronounced from 4 to 8, with
reductions in edges from approximately 15 million to 2.7 million. This implies that a small
threshold yields good coverage — observed earlier — at the expense of a large increase in
the number of identified edges, considering that there are only 245,500 edges in the reference
graph, as discussed in Section 5.4. This also explains the poor average precision observed for
Figure 5.2: Coverage (%) and average precision (%) of the HPC algorithm (top) using the original (non-keypoint-reduced) number of PCA-SIFT local descriptors on collections 20K and 40K of varying sizes; the run-time and the number of identified edges are also shown (bottom). We use seven threshold values $T$, ranging from 4 to 255 for the HPC algorithm to observe their effects on coverage and precision.
low threshold values, as shown earlier in Figure 5.2; nevertheless, the observed coverage as shown in Figure 5.2 relative to the number of identified edges, indicates that our approach is indeed effective for narrowing the search space for the candidate edges.

Figure 5.2 further shows that for all threshold values, the number of identified near-duplicate edges grows linearly with the collection size, even though the growth of the number of total edges is quadratic. Following this pattern, a collection of over 100,000 images will generate an estimated half a million algorithm-identified edges with a threshold of $T = 32$. This indicates that the growth of an index structure used to maintain these identified near-duplicate edges should also grow somewhat linearly, which makes this a scalable approach for only moderate-sized image collections.

In Figure 5.2, we also observe a considerable decrease in run-time from $T = 4$ to $T = 16$; run-times remains relatively stable from $T = 32$ onwards for all three collections. This phenomenon is unsurprising as we expect there to be a lower bound on run-time given that a constant number of accesses to the LSH indexes is required regardless of the threshold value. This lower-bound run-time depends largely on the number of local descriptors, and the number of LSH indexes; we study these parameters further in Section 5.5.3. We also note that using threshold values of $T = 16$ and $T = 32$, the HPC algorithm takes approximately 16 minutes to identify the entire collection of 40,000 images. With the smallest threshold value of $T = 4$, the run-time is observed to be around 30 minutes.

Overall, using the original approach of the PCA-SIFT local descriptors with the HPC algorithm, we observe that a threshold value of $T = 16$ or $T = 32$ leads to large gains in precision, and reductions in evaluation time, without great loss of coverage, whereas a threshold value below $T = 16$ results in better coverage at the expense of poor average precision.

5.5.2 Efficient automated discovery

We have shown that the HPC algorithm is highly effective, and uses only relatively modest processing time when coupled with a large number of PCA-SIFT local descriptors, and a suitable threshold value. However, a disadvantage of this approach is the potential growth in the number of identified edges. Given the effectiveness of our keypoint-reduction strategy as observed in Chapter 4 (page 96), we believe that keypoint-reduced local descriptors can improve the efficiency of the HPC algorithm, and at the same time, yield similar levels of effectiveness.
Figure 5.3: Coverage (%) and average precision (%) of the HPC algorithm (top) using the keypoint-reduced PCA-SIFT local descriptors, on two collections 20K and 40K of varying sizes; the total run-time and number of identified edges are also shown. We use seven threshold values, ranging from 4 to 255 for the HPC algorithm to observe their effects on coverage and precision.
Using the keypoint-reduced PCA-SIFT local descriptors with a keypoint rejection threshold set to 100 (or the T100 method) — as described in Chapter 4 (page 92) — we use the same experiments as described earlier to compare the effectiveness of both approaches, and observe the advantages of a reduced set of local descriptors for the HPC algorithm. We use the same 20K and 40K collections; we expect the trends for variation in collection sizes to remain the same as compared to those observed using the original number of the PCA-SIFT local descriptors.

Figure 5.3 shows the coverage and average precision of the HPC algorithm using the keypoint-reduced PCA-SIFT local descriptors. We observe that the overall coverage of the keypoint-reduced local descriptors is lower for all threshold values, with an average relative difference of 37% across all threshold values; the degradation in coverage is amplified when the threshold value increases. However, average precision is considerably higher than the original approach for lower threshold values ($T < 64$), with an the average relative difference of approximately 69%. With lower threshold values, we also observe the relative difference in coverage to be no more than 30%. These trends reflect those observed in Chapter 4 (page 96): keypoint-reduction boosts average precision at the expense of a lower coverage overall. We also note that the differences in the coverage and average precision levels between the HPC methods — using the original approach and the keypoint-reduced PCA-SIFT local descriptors — using all threshold values are statistically significant ($p$-values $< 0.05$).

Figure 5.3 further illustrates that by using the keypoint-reduced local descriptors, the number of identified near-duplicate edges and the run-time\(^1\) for the HPC algorithm have been dramatically reduced across all threshold levels, compared to using the original number of local descriptors. Comparing two threshold levels that yield comparable effectiveness ($T = 4$ in the keypoint-reduced approach and $T = 32$ in the original approach), we find a huge reduction in run-time — from approximately 15 minutes to a little over a minute — for processing collection 40K. This is a good result given the considerable improvement in efficiency. On average, we observe approximately 78% relative difference in the number of identified edges, and 96% relative difference in the run-time, over all threshold values. Using the reduced set of local descriptors, the HPC algorithm can process the collection of 40,000 images in approximately 60 seconds, for all threshold values; using the original number of local descriptors, we observe average run-time to be approximately 18 minutes.

Table 5.1 lists the detailed differences of coverage, average precision, run-time and the

\(^1\)Run-time is averaged over three runs.
### Table 5.1: Difference in effectiveness and efficiency of the HPC algorithm on collection 40K, between the original PCA-SIFT local descriptors, and that of our keypoint-reduced variant. The table shows the threshold values $T$ from 4 to 255.

<table>
<thead>
<tr>
<th>$T_4$</th>
<th>$T_8$</th>
<th>$T_{16}$</th>
<th>$T_{32}$</th>
<th>$T_{64}$</th>
<th>$T_{128}$</th>
<th>$T_{255}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>93.6</td>
<td>91.5</td>
<td>88.8</td>
<td>84.7</td>
<td>78.2</td>
<td>69.2</td>
</tr>
<tr>
<td>Original</td>
<td>Avg. prec.</td>
<td>16.8</td>
<td>48.8</td>
<td>79.4</td>
<td>90.8</td>
<td>92.9</td>
</tr>
<tr>
<td>Approach</td>
<td>Edges</td>
<td>15 559 155</td>
<td>2 499 650</td>
<td>551 458</td>
<td>226 994</td>
<td>137 399</td>
</tr>
<tr>
<td>minutes</td>
<td>28.3</td>
<td>18.2</td>
<td>16.0</td>
<td>15.6</td>
<td>15.6</td>
<td>15.5</td>
</tr>
<tr>
<td>Coverage</td>
<td>83.9</td>
<td>78.8</td>
<td>70.5</td>
<td>57.9</td>
<td>40.3</td>
<td>25.6</td>
</tr>
<tr>
<td>Keypoint-</td>
<td>Avg. Prec.</td>
<td>88.0</td>
<td>92.0</td>
<td>92.7</td>
<td>91.9</td>
<td>87.0</td>
</tr>
<tr>
<td>Reduced</td>
<td>Edges</td>
<td>201 107</td>
<td>130 859</td>
<td>100 470</td>
<td>76 851</td>
<td>51 946</td>
</tr>
<tr>
<td>minutes</td>
<td>1.1</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
</tr>
</tbody>
</table>

number of algorithm-identified edges of the HPC algorithm, between the original number of local descriptors, and the keypoint-reduced local descriptors on collection 40K. An interesting observation is that the HPC algorithm favours lower threshold values for the keypoint-reduced local descriptors, whereas the same threshold values on the original number of local descriptors yield low average precision; a threshold value of $T = 4$ yields similar levels of coverage and average precision for $T = 32$ in the original approach. This is an expected consequence of the keypoint-reduction as the reduced set of local descriptors also suggests a generally lower noise level. Thus, the threshold value of $T = 4$ is sufficient to sift the collection of the majority of non-near-duplicate images, at the expense of a slight drop in coverage. This clearly shows that our adaptation of the HPC algorithm requires a careful selection of threshold values using the local descriptors, and that there is an observed tradeoff between effectiveness and scalability. Based on the observed results, we believe that the keypoint-reduced local descriptors will scale to much larger collections at the expense of slightly lower coverage overall.

#### 5.5.3 Effects of the $l$ and $k$ parameters on the HPC algorithm

In our experiments using the LSH index, we have used the default parameters as determined by Ke et al. [2004]. In Chapter 4 (page 106), we showed that these parameters yield high recall and precision levels for even much larger collections.
As described in Section 5.3, our proposed discovery method using the HPC algorithm does not use any additional verification — such as $L_2$ distance verification and RANSAC geometric verification — for matching local descriptors due to the additional computational cost. Our keypoint-reduction approach also uses a considerably smaller number of local descriptors per image compared to the original approach; we conjecture that the number of $l$ LSH indexes, and the $k$ random bits (as described in Chapter 2, page 51) could possibly be reduced. Consequently, we believe that the choice of the LSH parameters warrants further investigation to derive optimal settings for the HPC algorithm. This is especially the case for large image collections, as these parameters directly dictate the efficiency of the algorithm.

We experiment with the $l$ and $k$ parameters to observe their effects on the HPC algorithm. For this study, we empirically test a range of threshold values from $T = 4$ to $T = 255$ for the HPC algorithm, with the keypoint-reduced local descriptors. However, we present results for only threshold value $T = 8$ as this setting best balances effectiveness and efficiency in our previous experiments, as discussed in Section 5.5.2. Instead of using collections 20K or 40K, we use collection 150K as we believe a larger collection yields better reliability for empirical experimentation. Detailed results for different $l$ and $k$ parameters using a range of threshold values from $T = 4$ to $T = 255$ on this collection are shown in C.3 (page 207).

Figure 5.4 shows the coverage and average precision of the HPC algorithm for variations of $l$ and $k$; $l$ ranges from 20 to 5 with stepwise decrements of 5, and $k$ ranges from 450 to 250 with stepwise decrements of 100. Overall, we observe that coverage declines as we reduce the number of LSH indexes ($l$), while average precision remains relatively stable over the range of $l$ values. While we observe a gradual decline in effectiveness — as measured by coverage — when the number of $l$ indexes are reduced, there appears to be only a limited impact on average precision. By reducing the $k$ random bits used for the LSH hash tables, we see an average of 5% relative improvement in coverage across all $l$ values, accompanied by a slight decrease in average precision. This is unsurprising given that the $k$ parameter dictates the rate of collision within a hash table; a smaller $k$ value results in more local descriptors colliding within a hash-bucket, which in turn, increases the likelihood of potential descriptor matches.

To further analyse these parameters, we study their effect on the number of identified edges and the run-time. Again, we present results for only $T = 8$ from a series of experiments with all threshold values from $T = 4$ to $T = 255$. As shown in Figure 5.5, the reduction in the number of LSH indexes causes a large reduction in the number of identified edges and the run-time of the HPC algorithm, but a reduction in the $k$ parameter has an opposite effect.
Figure 5.4: Coverage (%) and average precision (%) of the HPC algorithm on collection 150K using different values of $l$ and $k$ for the LSH index; $l$ ranges from 5 to 20 and $k$ ranges from 250 to 450. The results are observed for the keypoint-reduced PCA-SIFT local descriptors (using keypoint thresholding of 100); we use a threshold value of $T = 8$ for the HPC algorithm.
Figure 5.5: Run-time for processing 150,000 images using the HPC algorithm with different values of $l$ and $k$ for the LSH index; $l$ ranges from 5 to 20 and $k$ ranges from 250 to 450. We also report the number of identified edges. The results are observed for the keypoint-reduced PCA-SIFT local descriptors (using keypoint thresholding of 100); we use a threshold value of $T = 8$ for the HPC algorithm.

For $l = 20$, we observe that a reduction in $k$ from 450 to 250 causes the number of edges to grow by around a factor of three, from approximately 300,000 to one million. We believe that this is due to the increased number of local descriptors that are hashed to the same bucket, leading to overall longer postings list; a longer postings list inevitably generates a larger pool of potential edges. In terms of run-time, the HPC algorithm can process an image collection of 150,000 images under 13 minutes in the worst case. The overall results indicate that both the $l$ and $k$ parameters directly dictate the effectiveness of the HPC algorithm; this study has also shown that the original settings of $l = 20$ and $k = 450$ do not yield the optimal results, and that the settings of $l = 10$ and $k = 250$ best balances between effectiveness and efficiency of the HPC algorithm. While we have only presented the results for threshold value $T = 8$, all the observed trends for this particular study were similarly evident across other threshold values; we refer to Figure C.3 (page 207) for detailed results using all parameters.
Table 5.2: Effectiveness and efficiency of the HPC algorithm on large collections of 300K, and 1M; results for 150K are also included for comparison purposes. The table reports the coverage, average precision, total elapsed time, and the number of algorithm identified edges above the threshold $T = 8$; the optimal LSH parameters $l = 10$ and $k = 250$ are used. The results are observed for keypoint-reduced PCA-SIFT local descriptors using keypoint thresholding of 100.

5.5.4 Effectiveness of HPC on large collections

The level of effectiveness as observed in the previous section suggests that the HPC algorithm would yield high effectiveness for large collections. To confirm our results, we test the HPC algorithm on collections 300K and 1M, to observe any obvious trends pertaining to the scalability of the HPC algorithm. The implementation of the HPC algorithm for the (considerably larger) collection 1M is different from the other collections, as it is largely disk-based; that is, when performing a hash-based probabilistic counting of the local descriptors between two images (the first phase of HPC algorithm), we use a disk-based structure to maintain the approximate score between two images. A disk-based approach is necessary since for collections of $\leq 300\,000$ images, the total number of edges that are approximated using the hash-counter between any two images can be entirely contained and processed within approximately 3 GB of memory; but this is not possible for a collection of 1 000 000 images; it can easily cause an exhaustion in memory. This is especially the case when there is a large number of unknown images in the noise collection (from SPIRIT), where large number of false positive matches that seemingly share local descriptors could be present. Thus, we use the disk-based structure for the 1M collection, and report the run-time based on this framework.

We perform the same experiments using the optimal settings — as empirically observed in the previous section — of $l = 10$ and $k = 250$ using a threshold value of $T = 8$ on both collections. As shown in Table 5.2, we find that the HPC algorithm remains highly effective for one million images, producing competitive levels of approximately 80% coverage and 91%
average precision; this level of effectiveness is similarly observed for collections 300K and 150K. We also note that the difference in the observed coverage and average precision values across the three collections are not statistically significant (p-value > 0.05). In terms of the run-time and the number of algorithm-identified edges, we observe that the HPC algorithm can process a collection of a million images in less than an hour, with approximately 500,000 in the number of edges that are identified to be above the threshold value. This shows that the HPC algorithm can effectively discard large numbers of image pairs that are not near-duplicates while retaining the near-duplicate pairs, even within substantial collection sizes.

We also note the increase in the run-time and the number of edges when the collection size is increased from 150K to 1M. The trend as observed for our particular case implies that the number of identified edges follows a sublinear pattern; this shows that although the number of artificial (or known) near-duplicate edges (245,000) in the reference graph is expected to remain constant — given that our seed collection does not increase — and the number of identified edges above the given threshold is not growing at a rate proportional to the size of the image collection. Moreover, given that the noise collection is gathered from the Web, there could naturally be potential near-duplicates that are unknown to us.

Although we expected the time complexity of this algorithm to be quadratic, we observe that, in practice, its behaviour conforms to our initial expectation; that is, when variables $B$ and $l$ are considerably smaller than $M$, (see Section 5.3), the quadratic time complexity $O(l \times M \times B^2)$ of this approach is not apparent. The overall results are satisfactory, as they suggest that the HPC algorithm is sufficiently effective for even large collections of a million images, while its spatial complexity for such collection sizes is also relatively low.

We conjecture that this behaviour can probably be replicated if an image collection is expected to have a relatively stable proportion of near-duplicate images to unique ones. This means that, we expect the HPC algorithm to work in image repositories where the growth in the number of near-duplicate images is substantially lower than the overall growth in the number of unique images. Thus, we can expect a minimal growth in the number of edges that are below a given threshold. Based on our previous observation and analysis of web images in Chapter 3 (page 82), we suspect that this is the case for real image collections, where the proportion of near-duplicate images is generally small.

To further observe the effectiveness of the HPC algorithm, we measure the coverage and average precision for each individual alteration, as described in Chapter 3 (page 79). As shown in Table 5.3, we observe that the overall coverage is lower across all alterations,
<table>
<thead>
<tr>
<th>Alteration</th>
<th>150K</th>
<th>300K</th>
<th>1M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>coverage/avg. prec</td>
<td>coverage/avg. prec</td>
<td>coverage/avg. prec.</td>
</tr>
<tr>
<td>colorise</td>
<td>92.2 / 95.6</td>
<td>92.3 / 95.4</td>
<td>92.3 / 95.3</td>
</tr>
<tr>
<td>contrast</td>
<td>91.4 / 95.5</td>
<td>91.5 / 95.3</td>
<td>91.5 / 95.1</td>
</tr>
<tr>
<td>crop (5-30%)</td>
<td>90.4 / 94.8</td>
<td>90.4 / 94.5</td>
<td>90.4 / 94.4</td>
</tr>
<tr>
<td>despeckle</td>
<td>93.1 / 96.2</td>
<td>93.1 / 96.4</td>
<td>93.1 / 96.1</td>
</tr>
<tr>
<td>jpeg_to_gif</td>
<td>92.5 / 96.5</td>
<td>92.6 / 96.2</td>
<td>92.6 / 96.1</td>
</tr>
<tr>
<td>frame</td>
<td>88.9 / 93.1</td>
<td>89.1 / 92.8</td>
<td>89.1 / 92.7</td>
</tr>
<tr>
<td>rotation</td>
<td>91.5 / 95.5</td>
<td>91.5 / 95.3</td>
<td>91.6 / 95.2</td>
</tr>
<tr>
<td>scale-down</td>
<td>62.2 / 92.2</td>
<td>62.4 / 92.2</td>
<td>62.4 / 92.2</td>
</tr>
<tr>
<td>scale-up</td>
<td>93.3 / 96.6</td>
<td>93.3 / 96.3</td>
<td>93.3 / 96.4</td>
</tr>
<tr>
<td>saturation</td>
<td>92.4 / 97.4</td>
<td>92.5 / 97.3</td>
<td>92.5 / 97.2</td>
</tr>
<tr>
<td>saturation (sev)</td>
<td>92.2 / 98.0</td>
<td>92.3 / 97.8</td>
<td>92.3 / 97.6</td>
</tr>
<tr>
<td>intensity</td>
<td>91.8 / 97.6</td>
<td>91.9 / 97.5</td>
<td>91.9 / 97.3</td>
</tr>
<tr>
<td>crop (40-90%)</td>
<td>54.9 / 67.4</td>
<td>55.1 / 67.6</td>
<td>55.1 / 67.4</td>
</tr>
<tr>
<td>intensity (sev)</td>
<td>89.0 / 96.4</td>
<td>89.2 / 96.2</td>
<td>89.2 / 95.9</td>
</tr>
<tr>
<td>contrast (sev)</td>
<td>77.0 / 90.5</td>
<td>77.1 / 90.3</td>
<td>77.1 / 90.0</td>
</tr>
<tr>
<td>rotate+crop</td>
<td>75.4 / 95.6</td>
<td>75.6 / 95.3</td>
<td>75.7 / 95.3</td>
</tr>
<tr>
<td>rotate+scale</td>
<td>80.8 / 96.3</td>
<td>81.2 / 96.0</td>
<td>81.2 / 95.9</td>
</tr>
<tr>
<td>shear</td>
<td>54.0 / 45.0</td>
<td>54.3 / 44.6</td>
<td>54.3 / 44.3</td>
</tr>
</tbody>
</table>

Table 5.3: Coverage and average precision of the HPC algorithm for each alteration on large collections of 150K, 300K, and 1M; results are categorised into 18 alteration groups for ease of representation. The results are observed for threshold $T = 8$ for the HPC algorithm, using the keypoint-reduced PCA-SIFT local descriptors with keypoint thresholding of 100; the optimal LSH parameters $l = 10$ and $k = 250$ are used.
compared to those obtained for the retrieval experiments in Chapter 4 (page 96), but average precision is unaffected. The observed result is an expected outcome of the trade-off we made when selecting the LSH parameters, sacrificing effectiveness in the interest of better efficiency.

We find that the level of effectiveness is generally stable across three collections of varying sizes, indicating that the high level of coverage and average precision can be maintained for up to one million images. As expected, the HPC algorithm yields low coverage for some alterations such as scale-down, crop (40-90%), and shear, which is similarly to the results for retrieval experiments in Chapter 4 (page 96).

Overall, our experiments suggest that the HPC algorithm can automatically identify all near-duplicate instances within large image collections with relatively high effectiveness, while using modest computational resources and processing time.

5.5.5 Effectiveness on the web images

All our experiments up to this point use controlled collections where evaluation is based on the ground truth generated using seeded images. Even though the majority of our collection was gathered from the Web, the ground truth was created specifically for the purpose of testing the effectiveness and robustness of our proposed methods. To further gauge the performance of the HPC algorithm, we use our previously sampled web image collection, as described in Chapter 3 (page 82). For this experiment, we compare the HPC algorithm against the PCA-SIFT query-based approach, as described in Chapter 4 (page 92), using both the keypoint-reduced version and the original approach; we use the PCA-SIFT query-based approaches as baselines.

Table 5.4 shows the effectiveness of the original PCA-SIFT query-based approach, and the PCA-SIFT query-based approach using keypoint-reduced local descriptors, against the HPC algorithm (using keypoint-reduced local descriptors) for detecting all instances of near-duplicate images within the web collection. We henceforth refer to the original and keypoint-reduced PCA-SIFT query-based approach, as the query-based and keypoint-reduced query-based approaches, respectively. The first column shows the 23 subjects used for this experiment; “Disney” does not have any usable clusters, and for “BMW” no near-duplicate images were found (all clusters are singletons). As can be seen from column 4, the query-based approach yields an average recall and precision of 65% and 72%, and the keypoint-reduced query-based approach (column 6) produces average recall and precision of 58% and 70%; column 8 indicates that the HPC algorithm achieves the slightly lower average recall and
<table>
<thead>
<tr>
<th>Query Subject</th>
<th>No. of queries</th>
<th>rel. ans.</th>
<th>query-based (orig)</th>
<th>query-based (red.)</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>average rec/prec</td>
<td>average run-time</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(%)</td>
<td>(sec)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>average rec/prec</td>
<td>average run-time</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(%)</td>
<td>(sec)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>average rec/prec</td>
<td>average run-time</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(%)</td>
<td>(sec)</td>
<td></td>
</tr>
<tr>
<td>50 cent</td>
<td>14</td>
<td>6</td>
<td>64/74</td>
<td>1.8</td>
<td>48/73</td>
</tr>
<tr>
<td>Aaliyah</td>
<td>11</td>
<td>9</td>
<td>60/79</td>
<td>1.4</td>
<td>43/78</td>
</tr>
<tr>
<td>Jolie</td>
<td>10</td>
<td>3</td>
<td>69/80</td>
<td>1.4</td>
<td>63/75</td>
</tr>
<tr>
<td>Batman</td>
<td>8</td>
<td>5</td>
<td>51/66</td>
<td>2.1</td>
<td>35/62</td>
</tr>
<tr>
<td>Marley</td>
<td>14</td>
<td>8</td>
<td>31/44</td>
<td>1.8</td>
<td>29/45</td>
</tr>
<tr>
<td>Spears</td>
<td>6</td>
<td>2</td>
<td>58/59</td>
<td>1.4</td>
<td>54/42</td>
</tr>
<tr>
<td>Electra</td>
<td>11</td>
<td>3</td>
<td>63/80</td>
<td>1.2</td>
<td>54/62</td>
</tr>
<tr>
<td>Beckham</td>
<td>9</td>
<td>2</td>
<td>83/88</td>
<td>1.5</td>
<td>83/89</td>
</tr>
<tr>
<td>Ferrari</td>
<td>7</td>
<td>2</td>
<td>90/79</td>
<td>0.9</td>
<td>86/83</td>
</tr>
<tr>
<td>Garfield</td>
<td>7</td>
<td>2</td>
<td>65/66</td>
<td>1.8</td>
<td>60/63</td>
</tr>
<tr>
<td>Potter</td>
<td>11</td>
<td>6</td>
<td>44/51</td>
<td>2.1</td>
<td>38/35</td>
</tr>
<tr>
<td>Cobain</td>
<td>12</td>
<td>9</td>
<td>75/77</td>
<td>2.1</td>
<td>72/78</td>
</tr>
<tr>
<td>Lambo</td>
<td>7</td>
<td>4</td>
<td>79/93</td>
<td>0.8</td>
<td>72/93</td>
</tr>
<tr>
<td>Mustang</td>
<td>2</td>
<td>1</td>
<td>100/100</td>
<td>2.7</td>
<td>100/100</td>
</tr>
<tr>
<td>Porsche</td>
<td>4</td>
<td>3</td>
<td>28/50</td>
<td>1.0</td>
<td>25/55</td>
</tr>
<tr>
<td>Snoopy</td>
<td>10</td>
<td>1</td>
<td>65/59</td>
<td>1.6</td>
<td>55/57</td>
</tr>
<tr>
<td>Park</td>
<td>10</td>
<td>5</td>
<td>70/76</td>
<td>1.8</td>
<td>61/86</td>
</tr>
<tr>
<td>Spiderman</td>
<td>7</td>
<td>9</td>
<td>57/72</td>
<td>2.4</td>
<td>42/79</td>
</tr>
<tr>
<td>Superman</td>
<td>7</td>
<td>7</td>
<td>57/75</td>
<td>1.1</td>
<td>50/73</td>
</tr>
<tr>
<td>Schiavo</td>
<td>10</td>
<td>14</td>
<td>77/88</td>
<td>2.2</td>
<td>79/88</td>
</tr>
<tr>
<td>Simpsons</td>
<td>8</td>
<td>2</td>
<td>78/75</td>
<td>2.9</td>
<td>68/72</td>
</tr>
<tr>
<td>Tupac</td>
<td>10</td>
<td>7</td>
<td>54/63</td>
<td>2.5</td>
<td>41/63</td>
</tr>
<tr>
<td>Yoda</td>
<td>11</td>
<td>7</td>
<td>73/71</td>
<td>2.7</td>
<td>68/70</td>
</tr>
</tbody>
</table>

Table 5.4: Columns 2 and 3 indicate, respectively, the average number of usable queries (clusters) and the average number of relevant answers for each query. Columns 4 and 6 show the coupled average recall and precision (rec/prec) for each subject (averaged over the number of queries) using PCA-SIFT query-based approach; and our HPC algorithm. We also report the average time it takes to evaluate each query on the collection of approximately 750 images for that topic in columns 5 and 7; the latter (a non-query-based approach) is approximated by averaging the total processing time by the size of each set of images.
precision values of 57% and 63%. These results clearly show that the original query-based approach yields the highest effectiveness, followed by the keypoint-reduced query-based approach. The lower effectiveness level reflects the trading of effectiveness for higher efficiency.

For the query-based approach, the average evaluation time for a single query is approximately 1.8 seconds, whereas for the keypoint-reduced query-based approach it is approximately 0.4 seconds. Each query is timed for retrieval of near-duplicate images within the result set of its corresponding subject as returned by GISE (750 images on average). Column 7 shows that the HPC algorithm takes, on average, approximately 0.05 seconds for each image. The average processing time on the full collection of each subject is 35.6 seconds. Based on our observation, using a naïve query-by-example scheme, where each image within the collection is used as a query-example in turn, the query-based method would take over 29,843 seconds (approximately 8 hours) to identify all near-duplicate instances within this collection of 23 result sets with a total of approximately 17,250 images. The keypoint-reduced query-based method would take 6,900 seconds (approximately two hours) to perform the same task. The query-based approaches are clearly computationally expensive for such tasks; the speed advantage of the HPC algorithm over the PCA-SIFT query-based approaches is also apparent.

There is considerable difference between the level of effectiveness observed here for the PCA-SIFT query-based approaches and the HPC algorithm, and those reported for the controlled collections used in Chapter 4 and in earlier experiments. The original and keypoint-reduced PCA-SIFT query-based approach on the controlled collections produce average recall and precision values of 97% and 95%, 91% and 99%, respectively; whereas the HPC algorithm yields average recall and precision of 80% and 92% respectively. The discrepancies can be partly explained by the considerably smaller number of relevant answers in each human-evaluated cluster, where the distribution of difficult alterations remains about the same. In the controlled collections, the number of relevant answers is considerably larger; for instance, each seed image has 50 alterations, and so retrieving 48 of the 50 relevant answers (with 2 difficult alterations not retrieved) yields a recall of 96%. In contrast, in the web collection, there are on average only five relevant answers for each query, as shown in Table 5.4; thus, if 3 of the 5 relevant images are retrieved, recall remains at 60%. This causes the observed recall values to fluctuate sharply; it also partly explains the greater discrepancies in the average recall levels observed between the original and keypoint-reduced query-based approaches. We observe that images of alteration combination — including many artistic alterations — are missed by both PCA-SIFT query-based approaches and the HPC algorithm, resulting in the
low average recall and precision values. Nevertheless, both approaches prove to be effective in retrieving images of many alterations, and our HPC algorithm is more efficient, albeit less accurate, than the query-based approaches.

**Efficiency of the HPC method against DPF**

To evaluate the efficiency of the HPC method against a previous query-based method that was shown — in Chapter 4 (page 109) — to be highly efficient but less accurate than the PCA-SIFT query-based methods (both our proposed keypoint-reduced version and the original approach by Ke et al. [2004]), we compare the HPC method with the DPF method, which was first proposed by Meng et al. [2003]. Qamra et al. [2005] also reported this method to be effective for proprietary large image collections. As discussed in Chapter 4, the DPF method is inherently more efficient than the PCA-SIFT query-based methods as each image is only represented by a single feature vector, instead of a bag-of-vectors.

For this study, we use the standard DPF method, as used for our experiments in Chapter 4 (page 96) where a ranked list of images are returned for a specified query example; this is essentially an exact nearest neighbour approach, rather than the approximate nearest neighbour approaches used in the LSH scheme. As with our previous experiments, we do not use an index structure for the DPF approach as the methodology for applying an approximate nearest neighbour and the levels of efficiency and effectiveness produced by such methods are unclear; these methods have only been preliminarily reported in the works of Qamra et al. [2005]. We use the settings we determined to be optimal from our experiments in Chapter 4 (page 109).

As shown in Figure 5.6, using the collection of web images, the DPF method with colour and texture features is less effective than the both the original and keypoint-reduced PCA-SIFT query-based methods, as well as our HPC algorithm. This is not surprising given that the effectiveness was previously shown to be inferior to those of the PCA-SIFT methods for controlled collections. On average, precision at 100% recall is observed to be 27%, which shows that the majority of the queries return the near-duplicate images within the top 18 answers. The average number of relevant answers for each cluster is only five, and so, most relevant answers would typically be returned within the first page of answers shown to the user by a web search engine.

The HPC algorithm, and the original and keypoint-reduced PCA-SIFT query-based methods produce an average precision of 72%, 70%, and 63% respectively, at 57%, 58%, and 65%
Figure 5.6: Retrieval effectiveness of colour and texture features using DPF on the images of 23 Web topics (Set 1) from GISE; average recall and precision (%) over the number of queries are reported. The example images from the first 20 images are used as queries to retrieve all near-duplicates from the result sets.
average recall points. In contrast, the DPF query-based method yields only 39%, 39%, and 37% average precision at these average recall points. Nevertheless, the advantage of the DPF method lies in the fast query evaluation time, with an average of 0.06 seconds per image. This translates to approximately 30 times faster than the original PCA-SIFT query-based method, and about 7 times faster than the keypoint-reduced query-based method, even without the use of an indexing structure. Even so, the observed level of efficiency is still lower when compared to our HPC algorithm, and the overall level of effectiveness is lowest of all tested methods.

The results observed here provide only an estimated comparison of effectiveness and efficiency of the HPC method. In large collections, we believe that the computational complexity of such an approach makes it implausible to use a naïve query-based method for the pairwise detection of near-duplicate images. Moreover, we note that our method of comparison undermines the level of improvements in efficiency achieved by the HPC method; the effectiveness and efficiency of the query-based methods were evaluated for 23 individual sets of image results that were obtained from GISE, where each set of approximately 750 images were evaluated separately. In contrast, the HPC algorithm performs the detection on the entire set of 17,250 all at once. Thus, the estimated query evaluation time for the query-based methods can potentially be higher than those observed in our results. Nevertheless, the results here show that even with evaluations that favour query-based methods, our HPC algorithm yields comparable levels of effectiveness.

5.6 Discussion

We have shown through experimentation that our HPC algorithm is effective and efficient in sifting near-duplicates within a large collection of images. We have also demonstrated that the HPC algorithm can boost the efficiency of query-based retrieval considerably; an image collection can be first processed with this method, such that the query evaluation and retrieval is rapidly narrowed to a much smaller number of near-duplicate instances.

While we have shown that our method scales well, a potential drawback of this approach is the scalability of the HPC algorithm for web-scale collections in the order of $\geq 10^7$. Even though our keypoint-reduction strategy has improved the scalability of the bag-of-vectors approach by a wide margin, its scalability for web-scale image collections remains to be tested.

To this end, object-recognition approaches such as the work of Chum et al. [2007] use
k-means to quantise the large number of local descriptors into clusters that are used as visual words [Sivic and Zisserman, 2003], where a visual word is essentially a neighbourhood of local descriptors that are closer to each other than to other descriptors according to the \(L_1\) (Manhattan) distance. They have shown that the scalability issues of the bag-of-vectors approach can be eliminated to a large extent during query evaluation. We also note that their approach is similar to our HPC algorithm, where — as described in section 5.3 — only one LSH index is used for the mapping of local descriptors.

We have observed that effectiveness of this approach generally deteriorates when \(l\) (the number of LSH indexes) is severely reduced. This implies that the HPC algorithm is dependent on the number of LSH indexes, and consequently on the number of hash-keys; this also directly relates to the number of local descriptors that are used for each image. This behaviour also corroborates our hypothesis, that providing a sufficiently large number of matching hash-keys, using an appropriate \(l\) value and number of local descriptors will enable near-duplicate matches to rise above the noise level. However, our experiments have also shown that, for large collections, a trade-off is required as using large \(l\) values, and large numbers of local descriptors are generally counter-productive, due to severely impaired efficiency when coupled with such combinations. Thus a balance of these factors are required for effective and efficient operation of the HPC algorithm.

While it is also conceivable that the approach of Chum et al. [2007] in quantising each bag-of-vectors into a single vector may be more efficient, their observed experimental results for pairwise detection of near-duplicate video keyframes using such a mapping was reported to be comparable to only a colour histogram. The colour histogram was also not specifically designed and tested for robustness against variations in imaging parameters common in near-duplicates, which indicates the relatively low effectiveness of their approach [Chum et al., 2007]. Their definition of near-duplication is also ambiguous; they define a near-duplicate pair as being two images (keyframes) with a calculated distance between their histogram less than a certain threshold. More importantly, their definition also encompasses the domain of object-recognition, where images are matched if they share identical objects. This is counter-productive for our application as the definition of near-duplication in this thesis is different (see Chapter 3, page 78); we do not consider images that have variations of background and foreground image objects, a common phenomenon in video keyframes that are taken a few seconds apart. Nevertheless, we believe the adaptation of vector quantisation of the bag-of-vectors shows promise in addressing the scalability issues of the HPC algorithm, and that it warrants further investigation.
5.7 Summary

In this chapter, we have demonstrated that the hash-based probabilistic counting approach — originally a near-duplicate text-document detection technique — can be effectively adapted for images indexed using PCA-SIFT local descriptors and the LSH indexing scheme. We have developed a method that automatically sifts an image collection of near-duplicates with high effectiveness and efficiency; we believe that this is the first work to show such high levels of performance for this application. Our findings here corroborate our hypothesis that near-duplicate images can be automatically and effectively identified using a refine-and-filter scheme: a first-pass strategy coarsely estimates all potential near-duplicate image pairs, allowing us to process a smaller image set further for near-duplicate pairs.

Our experiments show that the approach is feasible and scalable for moderate-sized real-world collections; we have also shown that it scales well to collections of up to a million images. Accuracy is high, especially for less severe image alterations, and the computational costs are moderate. Our method provides effective discovery of duplicates and near-duplicates, and thus is a practical approach to collection management and protection of copyright. In the next chapter, we discuss the clustering approach that extends the HPC algorithm.
Chapter 6

Clustering Near-duplicate Images

The discovery task discussed in the previous chapter, aims to identify the pairwise relationships of every image in a given collection. We adapted a near-duplicate text document detection method to the image domain, and presented an effective method of identifying near-duplicate image pairs in relatively large collections. In the previous chapter, we have shown that the automatic identification of the pairwise relationship of every image in a collection is useful from a retrieval perspective, where retrieval can be limited to images with associated near-duplicate relationships; thus substantially speeding up the evaluation process.

Another interesting problem associated with the non-query-based approach is the distinct grouping of the near-duplicate images within a given collection. Consider the example in Figure 6.1, which shows the first 28 image results returned by the Google image search engine\(^1\) for the image query “Edvard Munch Madonna”. From the perspective of a user, it would be useful to be able to group these duplicate and near-duplicate instances that appear in the image answers into a set, so that a greater variety of relevant images can be presented more effectively. This has the added benefit that users can avoid viewing re-occurrences of the same image (or variants of it) in the image results. The task of categorising each image into its respective near-duplicate group can be seen as a clustering problem, where each cluster represents a distinct group of images that are (or suspected to be) near-duplicates of one another.

In this chapter, we show that standard text-document clustering approaches — such as those of Broder et al. [1997] and Haveliwala et al. [2000] as described in the Chapter 2 (page 11) — can be adapted for near-duplicate images; we show that our previous HPC

\(^1\)http://images.google.com
Figure 6.1: First two pages of image answers returned from Google Image Search for query “Edvard Munch Madonna”.
method of near-duplicate pairwise image identification, as described in Chapter 5 (page 133), can be used as part of this clustering process. We provide empirical evidence to show that the proposed adaptation is highly effective for clustering near-duplicates in collections of up to a million images, and demonstrate that it manages such tasks using only modest processing time. We show that this approach offers an effective solution that is practical for web images, and report high effectiveness in clustering real-world near-duplicate examples gathered from the Web.

6.1 Related work

Recently, some research groups showcased a video copy detection evaluation benchmark at CIVR 2007 [Sebe and Worrying, 2007], for which the goal is to identify matching random keyframe sequences (images) of videos that have been randomly selected and post-processed using some samples of artificial transformations, such as cropping, borders, and insertion of texts, among others. We believe that the featured task can be addressed very efficiently given an effective method of near-duplicate image clustering; we observe that the task is almost identical to that of near-duplicate image matching, where the only difference is that we adopt an adversarial approach to test the robustness of our proposed methods using individual images instead.

As we discussed in Chapter 2 (see page 69), there is little prior work on clustering of near-duplicate images. The Replicate IMagE Detector (RIME) by Chang et al. [1999] is arguably the only system thus far that is designed specifically for clustering near-duplicate images. However, the robustness of this system has only been tested against ten minor image alterations on a single query image, and the scalability of the system for a wider range of imaging conditions, and large datasets is not reported [Chang et al., 1999]. This system was also designed using images that are indexed using simple colour and texture features that have been shown to be sensitive against imaging conditions such as cropping and scaling [Meng and Chang, 2003].

Standard clustering techniques in general image retrieval — such as the k-means technique described in Chapter 2 (page 60) — that employ colour and texture features, are not suitable for near-duplicate images as the features and comparison methods used are not specifically designed for this purpose. Standard distance measures such as the $L_1$ or $L_2$ norm are typically used to compute the distance between two feature vectors during the $k$-means computation, and the problems associated with using these distance measures and typical colour and
6.2. CLUSTERING NEAR-DUPLICATE IMAGES

texture features for near-duplicate images are well documented in the work of Meng et al. [2003] and Qamra et al. [2005].

We believe that some work on clustering near-duplicate text documents can be adapted for near-duplicate images. Broder et al. [1997] describe a simple technique to cluster text-documents by examining the pairwise similarity. They first compute the pairwise similarity of the entire document collection using a shingling approach, as we have described in Chapter 2 (page 11). Haveliwala et al. [2000] show that, using a similar approach, the same clustering technique can be used to efficiently gather web documents for effective identification of near-duplication. We describe the adaptation of the clustering method in the next section.

6.2 Clustering near-duplicate images

As described in Chapter 5 (page 135), the near-duplicate image pairwise relationship of an entire image collection can be visualised as a graph; each node represents an image, and the presence of an undirected edge between two nodes reflects a near-duplicate relationship. As such, a collection of $N$ images can be represented as a weighted graph $G = (V, E)$ where a set of nodes $V = \{1, \ldots, N\}$ represents the images, and $E = \{m(i, j) : i, j \in V\}$ represents the set of edges between every pair of nodes in the graph. The expression $m(i, j)$ denotes the similarity between two images, or the approximated local descriptor matches between two nodes of $i$ and $j$ within the LSH indexes. Thus, the similarity measure reflects the matching descriptors between two images $i$ and $j$ as estimated by the HPC algorithm, with a high value indicating a likely near-duplicate relationship. Once an image collection is processed by the HPC algorithm, each triplet $\langle$imgID$_1$, imgID$_2$, counter$\rangle$ can be seen as a weighted edge between two nodes, where the number of approximated local descriptor matches (henceforth referred to as the similarity) is reflected by the edge weight. Thus, the aim is to find an efficient solution for discriminating between unique images and their near-duplicates, and also to accurately form non-overlapping clusters for each near-duplicate set. The method that we describe in this section is akin to a graph partitioning method, which is also considered to be an agglomerative and partitional approach, as described in Chapter 2 (page 60). All nodes are initially considered to be distinct clusters that are progressively merged (based on a criterion) to form larger clusters, and the algorithm-created (flat) clusters do not form a hierarchy. A summary of the entire process of the adapted clustering method is provided in Algorithm 7.

We adapt the clustering approach proposed by Broder et al. [1997] and Haveliwala et al.
Algorithm 7 The process of clustering near-duplicate images.

Require: A collection of $N$ images, clustering threshold value $CT$, a set of image IDs ($idSet$) from 0 to $N - 1$.

Generate relationship graph — a list of $\langle \text{imgID}_x, \text{imgID}_y, \text{COUNT}\rangle$ triplets using the HPC algorithm, where $\text{COUNT} \geq CT$.

Initialise first cluster set ($CID_i = \emptyset$, where $i = 0$)

repeat

Initialise temporary unique image ID set ($tidSet$) by taking any element in $idSet$ not already processed.

for all image $ID \in tidSet$ (process each unique image ID once) do

Identify all unique IDs (not in $tidSet$) in list of triplets such that $\text{imgID}_x = ID$ or $\text{imgID}_y = ID$.

Append $x$ and/or $y$ to $tidSet$.

end for

for all image $ID \in idSet$ do

Add $ID$ to $CID_i$ (add image $ID$ to cluster set $CID$).

Remove $ID$ from $idSet$.

end for

Set $tidSet$ to $\emptyset$.

Create new cluster set $CID_{i+1}$ and set to $\emptyset$

until $idSet = \emptyset$

All clusters are found when $idSet$ is empty, and all entries are populated.
6.2. CLUSTERING NEAR-DUPLICATE IMAGES

[2000] originally designed for text documents for near-duplicate images; this approach is also known as the CENTER algorithm. They show that using triplets of $(\text{docID}, \text{docID}, \text{count})$ that are generated from text documents can be used to effectively identify cluster groups. By examining each triplet, they unite (add an edge between) two nodes (documents) from a union-find algorithm if the pair of the examined triplet has a count of common shingles exceeding a certain threshold [Broder et al., 1997; Haveliwala et al., 2000]. A detailed process for this method adapted for near-duplicate images is shown in Algorithm 7. When all triplets have been examined and all appropriate pairs are connected, clusters are formed by identifying all the disjoint sets of connected nodes.

A shingle, as described in Chapter 2, is simply the smallest unit by which a text document can be partially represented. For our application, they are the representative units of hash-keys or tokens used in the HPC algorithm.

We hypothesise that near-duplicate images form natural clusters — that is, clusters that consist mainly of images that are near-duplicates of one another — and that they can be accurately identified by this clustering approach. Based on the high effectiveness of pairwise detection of near-duplicates that we observed in Chapter 5 (page 133), we posit that a high similarity between two near-duplicate images is often indicated by a large number of matching local descriptors, with the exception of some occasional false positive matches. We believe that by using a suitable threshold, we can effectively gather near-duplicates into their respective clusters. The adaptation is straightforward, since we can examine the triplets generated by the HPC algorithm to identify cluster groups. We can then create $m$ clusters of $\{C_1, \ldots, C_m\}$ where two non-overlapping clusters can be defined as: $C_a \cap C_b = \emptyset$, $a \neq b$ and $\cup_{a=1}^{m} C_a = V'$. Using this method, clusters are formed for only the set of nodes ($V'$) with non-zero weighted edges; image pairs without any matching local descriptors (zero-weighted edges) are not considered during clustering. Thus, clustering is performed on $G'(V', E')$, for $V' \subseteq V$ where $E' = \{s(i, j) \geq T : i, j \in V\}$, and $T$ is the inclusion threshold for the nodes in each triplet.

The threshold $T$ used for clustering is essentially the same as that of the HPC algorithm for pairwise near-duplicate identification, as discussed in Chapter 5. In the HPC algorithm, the threshold serves as a cut-off point for image pairs that do not have descriptor matches (edge weight) at least the amount equivalent to the threshold. For clustering, we conjecture that the best value of $T$ is considerably different due to the transitive property of the near-duplicate relationship. Consider three near-duplicate images of $A$, $B$, and $C$, where $A$ shares high similarity with $B$ but not with $C$ (due to its indirect derivation from $B$), images $A$ and
CHAPTER 6. CLUSTERING NEAR-DUPLICATE IMAGES

C remain associated due to the approximated matches between images B and C that exceed the threshold. A pair of nodes within a cluster may have an edge weight below a certain threshold \( T \) but are indirectly associated via another node within the same cluster. Thus, we empirically determine a suitable threshold for clustering near-duplicates by experimenting with the cluster threshold (denoted by \( CT \)) in later sections. We also henceforth refer to this clustering method as the \textit{ND-CENTER} algorithm.

6.3 Experimental setup

For our clustering experiments, we use image collections 150K, 300K, 1M, and the web collection as described in Chapter 2 (page 74); we do not experiment further on the smaller collection of 20K.

In our experiments, the seed collection is used to form the predefined near-duplicate sets; a near-duplicate set is a reference by which to measure the quality of the algorithm-identified clusters. The seed collection consists of 200 images with 50 altered versions each, and therefore generates 200 unique non-overlapping clusters, where each non-overlapping cluster is used as a predefined near-duplicate set. Hence, with the near-duplicate sets as a reference, we use measures of purity, entropy, recall, and precision, defined in Chapter 2 (page 73). We also measure the number of identified edges, and report timing\(^2\) of the clustering method for all collections tested, and vary the threshold value \( CT \) to study its effects on the cluster effectiveness and efficiency.

To further test our clustering approach on near-duplicate examples on the Web, we also use the web collection manually clustered by a human evaluator, as discussed in Chapter 3. We use the same 23 subjects from the web collection as in the previous chapter: \textit{50 cent, Angelina Jolie, David Beckham, Carmen Electra, Britney Spears, The Simpsons, South Park, Garfield, Ferrari, Lamboerghini, Batman, Harry Potter, Yoda, Spiderman, Superman, Bob Marley, Tupac, Kurt Cobain, Aaliyah, and Terri Schiavo}. There is a total of 205 human-evaluated clusters from this collection; we discard singleton clusters and those that contain images that are unusable.
Figure 6.2: The number of non-overlapping clusters identified by the ND-CENTER algorithm for collection 150K, at each clustering threshold $CT$ (from 8 to 100) using LSH parameters of $l = 10$ and $k = 250$. 
CHAPTER 6. CLUSTERING NEAR-DUPLICATE IMAGES

6.4 Results

Here, we present results for the clustering method (ND-CENTER) on all three collection of varying sizes, and the web collection that contains predefined human-evaluated image clusters. In Chapter 5, we observed empirically that using \( l = 10 \) indexes and \( k = 250 \) for the LSH parameters best balances efficiency and effectiveness, that is, the number of identified edges, and the level of coverage and average precision, respectively. Thus, we use this setting for our clustering experiments.

Figure 6.2 shows that the number of clusters in collection 150K that are identified by the ND-CENTER algorithm fluctuates with lower cluster thresholds \( CT \), but gradually increases with larger threshold values; we varied the cluster threshold from 8 to 200 in increments of two. This generally increasing trend is unsurprising as the increased cluster threshold causes more images (nodes) with lower edge weights to lose their association with their respective clusters, resulting in the formation of additional clusters. As discussed in Section 6.3, an ideal algorithm would form \( m \) clusters such that \( m = c \), where \( c = 200 \) artificial groups. But we observe that, on average, there are approximately 1200 non-overlapping clusters across all threshold values, exceeding the number of seeded groups by a wide margin. We believe that this can be partially attributed to the rest of the non-seeded images in the collection that serve as noise; the majority of the images are gathered from the Web, and therefore we expect the clusters to be formed from duplicates or near-duplicates in the collection that are unknown to us.

Figure 6.3 shows the distribution of images within the clusters identified by the ND-CENTER algorithm for a range of cluster thresholds; this is observed using the same collection (150K). For simplicity, we show only selected cluster threshold values. We observe a skewed distribution in the clusters, where the majority of images are grouped into a small number of clusters, with the remaining clusters containing only a few images each. The highly skewed distribution of images in the clusters produced by cluster thresholds of 8 to 32, indicates that small threshold values are counter-productive for clustering; this contrasts with results observed for pairwise detection of near-duplicates — using the HPC algorithm — in Chapter 5. An extreme case is observed using a cluster threshold of \( CT = 8 \), where there are 30,072 images in the largest cluster, and the second largest cluster contains fewer than 20 images. This result clearly shows that this threshold value is below the noise level; a low threshold value causes images to be less selectively linked to one another, and is therefore

\(^2\)All timing results are averaged over three runs.
Figure 6.3: Distribution of the number of images for the ND-CENTER-identified clusters for collection 150K using LSH parameters of $l = 10$ and $k = 250$; each line in the graph denotes a different clustering threshold $CT$ (from 8 to 100).
unable to find a connection between the majority of image pairs, resulting in the formation of large clusters. This also explains the fluctuation in the number of formed clusters for the lower range (from 8 to 40) of cluster threshold values, as observed in Figure 6.2.

However, we observe that the number of images in the largest cluster declines steeply as the clustering threshold value increases; for instance, the difference between cluster thresholds 8 and 16, with a variation of close to 20,000 images is evident in the sizes of the largest clusters. The distribution also becomes more uniform as the cluster threshold is increased; using cluster threshold $CT = 100$, there are only 153 images in the largest cluster, while the second largest cluster consists of 80 images.

Figure 6.3 shows that for clustering threshold values from 64 to 100, the number of images (approximately 40 to 50) remains consistent for approximately 200 clusters. The observed results also indicate that the majority of the clusters contain only a single pair of near-duplicate images. This is a pleasing result as Collection 150K contains the seed collection with 200 artificial near-duplicate sets, each containing 50 near-duplicate images. Even though these clusters do not explicitly reflect the 200 artificial groups — that is, whether the algorithm-identified clusters contain images from the respective artificial near-duplicate sets — the distribution of the images provide an indication of cluster formation. We corroborate this observation with further results in Section 6.4.1.

To analyze the effect of clustering granularity, we show in Figure 6.4 the average number of clusters formed for each of the 200 artificial near-duplicate sets; these results are observed for cluster threshold ranging from 8 to 200. We observe that applying large cluster threshold values tends to over-cluster; the average number of clusters for each group (denoted by $m_k$) approaches 4, where the ideal case for $m_k$ is 1. This means that, on average, each near-duplicate set is divided into four clusters, using high threshold values. We believe that this result can be explained by the various alterations of near-duplicate images present within each artificial group; the more severe alterations lead to the formation of individual clusters due to low-weighted edges that do not rise above the threshold. Nevertheless, the result indicates the effectiveness of the ND-CENTER algorithm considering the relatively low number of algorithm-formed clusters per group ($m_k$) against the number and variation of near-duplicate images in each group.
Figure 6.4: Average number of clusters identified by the ND-CENTER algorithm on collection 150K, for each of the 200 artificial groups, for different clustering threshold $T$ (from 8 to 100) using LSH parameters of $l = 10$ and $k = 250$. 
Figure 6.5: Average purity $p(k)$ and average entropy $h(k)$ over 200 near-duplicate sets on collection 150K, for different clustering thresholds $CT$ (from 8 to 100) of the ND-CENTER algorithm using LSH parameters of $l = 10$ and $k = 250$. 
6.4. RESULTS

Figure 6.6: Average recall and precision over 200 near-duplicate sets from Collection 150K, for different cluster thresholds CT (from 8 to 100) of the ND-CENTER algorithm using LSH parameters of $l = 10$ and $k = 250$.

6.4.1 Cluster quality and accuracy

Figure 6.5 shows the average purity and entropy of all 200 near-duplicate sets, and Figure 6.6 shows the average recall and precision of these sets. The results in these two figures are observed over a range of cluster threshold values CT from 8 to 200 on Collection 150K; the results are averaged over 200 artificial near-duplicate sets. As shown in Figure 6.5, we observe that as we increase the threshold value, the average purity of the clusters approaches 1 whereas average entropy approaches 0. For instance, with cluster threshold $CT = 200$, the average purity and average entropy, are observed to be 1 and 0, respectively. These results also indicate that the ND-CENTER algorithm using small cluster thresholds does not produce quality clusters; we observe higher cluster quality with threshold values of approximately 70 and above.

Figure 6.6 shows that small clustering thresholds yield low average precision and high average recall, and that large threshold values yield the opposite effect. Evidently, a high average recall point that is coupled with a low average precision translates to low average
purity and high entropy; this is indicative of the generally high mixture of unrelated images within each cluster. We observe that average recall falls gradually, and remains above 60% with the largest threshold value of 200, with average precision close to 100%.

The overall results from both figures indicate that the ND-CENTER clustering method favours large threshold values that yield high average recall and precision, with high purity and low entropy. Given reasonably large threshold values, this clustering method excels at forming high-quality clusters. Taking into consideration the distribution of images in the clusters, and the number of algorithm-identified clusters — as shown in earlier experiments — we find that a cluster threshold of 100 provides a good balance of these factors.

The results presented so far only indicate the average cluster quality over 200 artificial sets. To further evaluate the quality of the clusters, we report the same measures of average entropy, average purity, recall, and precision for each of the 200 near-duplicate sets. This is different from the earlier experiments in that the results are observed for each near-duplicate set. The results presented in Figure 6.7 and Figure 6.8, are observed using a cluster threshold $CT = 100$; as shown earlier, this value yields the best tradeoff amongst all measures.
6.4. RESULTS

Figure 6.8: Recall and precision (%) (sorted by precision) for each of the 200 near-duplicate groups using cluster threshold of $T = 100$, with LSH parameters of $l = 10$ and $k = 250$.

As indicated in Figure 6.7, the majority of identified clusters for each of the 200 near-duplicate sets contains a high concentration of correctly identified images; this is indicated by generally high average purity and low average entropy. We also observe that the majority of the near-duplicate groups have average purity over 0.8 and average entropy below 0.2, which indicates high clustering effectiveness.

Figure 6.8 shows that the recall and precision for clusters in the majority of the near-duplicate groups are high; for all groups with 100% precision — approximately 135 of 200 groups — the level of recall ranges from 60% to 90%. The drop in precision of clusters in some of the near-duplicate groups indicates that our clustering algorithm occasionally fails to correctly categorise images into their respective clusters, resulting in a mixture of images within each cluster. This yields a relatively low level of precision, albeit with a slightly higher level of recall. The two distinct levels of precision can be attributed to the precision measure used for cluster evaluation (see Chapter 2, page 73), which measures the ratio of correctly identified images in all algorithm-formed clusters for each of the 200 near-duplicate sets. This results in a more stringent measure, as a low precision level of a single cluster —
which contains images from a near-duplicate set — can lead to an abrupt drop in the overall precision.

Nevertheless, the overall results indicate that the ND-CENTER algorithm is effective for accurately gathering relevant near-duplicate images into their respective clusters. On collection 150K, it has an average of approximately 10 misses for every 50 near-duplicate images considered during clustering.

6.4.2 Effectiveness and efficiency on large collections

To study the effectiveness of our clustering approach on an even larger collection, we use the same experiments on collections 300K and 1M, using the best settings (LSH parameters $l = 10$, $k = 250$, and cluster threshold $CT = 100$) determined from earlier experiments. We find the overall level of effectiveness for these two collections to be comparable to that observed on the collection of 150K images.

Using cluster threshold $CT = 100$ for image Collection 300K, we observe that the average number of clusters formed for each of the 200 near-duplicate sets remains at 2.89; this was similarly observed for the collection 1M, with an average of 2.88 clusters for each near-duplicate set. This indicates that although the collection size is increased, the number of identified clusters for each artificial near-duplicate group remains relatively constant. As with collection 150K, our clustering approach for collection 300K achieves similarly high levels of effectiveness. The average recall and average precision of 80.7% and 76.4%, respectively, for the clusters within the 200 near-duplicate sets are nearly identical to those for collection 150K (with 80.6% and 74.4%, respectively). This level of effectiveness, again, was similarly observed for image collection 1M, with average recall and average precision of 80.8% and 76.3%, respectively.

The average purity and average entropy for collection 300K were observed to be 0.89 and 0.09, respectively; the corresponding values for collection 150K are respectively 0.91, and 0.07. Finally, we observe that a comparable level of effectiveness — average purity and average entropy of 0.89 and 0.09, respectively — is achieved for collection 1M; this shows that the level of effectiveness of ND-CENTER algorithm can be maintained for large image collections.

These results demonstrate our proposed clustering method to be indeed effective; they also further show that the level of effectiveness produced by our clustering approach is not strongly dependent on the size of the collections.
Figure 6.9: Efficiency of identification and clustering of all near-duplicate images for three image collections: 150K, 300K, and 1M; the figure shows the total run-time for the entire process of image pairwise identification and clustering. It also shows the number of identified edges using cluster threshold of $CT = 8$ and $CT = 100$ for the ND-CENTER algorithm, with LSH parameters of $l = 10$ and $k = 250$. 
To gain further insight into the efficiency of our clustering technique with regards to the larger collections 300K and 1M, we compare the total run-time required to identify the edges — near-duplicate image pairs — and for our algorithm to cluster the near-duplicate images for both collections; we also include the results for collection 150K for comparison purposes. The time required to cluster a collection of images depends largely on the HPC algorithm — described in Chapter 5 (page 133) — as the number of algorithm-identified edges dictates the number of image pairs to be considered for the clustering process.

Figure 6.9 shows the total run-time and the number of identified edges for all three collections (150K, 300K, and 1M) using the cluster thresholds \( CT = 8 \) and \( CT = 100 \). We use two different threshold values to observe the impact of small and large threshold values on efficiency. As shown in Figure 6.9, using cluster threshold \( CT = 8 \), the time to process collections 300K and 1M are approximately 11 minutes and 45 minutes, respectively; the corresponding time for collection 150K is approximately 6 minutes. Using a cluster threshold \( CT = 100 \), the time to process collections 150K, 300K, and 1M, are approximately 6 minutes, 10 minutes, and 43 minutes, respectively. These timing results indicates that the clustering process has modest run-time even for substantial collection sizes.

Although these times include the run-time of the HPC algorithm, they are similar to those reported in earlier experiments of the HPC algorithm for pairwise detection of near-duplicate images, as discussed in Chapter 5 (page 133). This is expected as identification of the pairwise relationships of all images within a collection — the HPC algorithm — is more computationally expensive than the clustering process. Our proposed clustering method is used only to process the number of identified edges, that is, image pairs that are identified to be near-duplicates of each other. Thus, the computational resources required for clustering near-duplicate images is low, compared to that required for the detection of pairwise relationships of all potential near-duplicate images. The relatively unchanged run-time across cluster threshold values is also within our expectation; the hash-counters used during pairwise identification (HPC algorithm) of all image pairs has a cost of \( O(lMB^2) \), as described in Chapter 5 (page 135). It requires the matched local descriptors from all image pair to be approximately counted from each of the \( M \) entries for the series of \( l \) LSH indexes, and then to be verified if the matches are within the given threshold. Hence, the run-time is only slightly affected by the cluster threshold value, since this value only determines the pool of image pairs that fit the criterion. Nevertheless, the relatively modest run-time for such a substantial number (approximately \( 130 \times 1000000 = 130 \) million) of local descriptors indicates that a large collection can be processed and clustered efficiently using the ND-
Figure 6.10: The ratio between seed images and all images that are identified to be within a given cluster; the ratio between seed images that are identified within clusters, and the total number of 10,000 seed images in the collection are also shown. The results are observed for every cluster threshold ($CT$ ranges from 8 to 100) using LSH parameters of $l = 10$ and $k = 250$ on one million images (collection 1M).

Figure 6.9 further shows that using a cluster threshold $CT = 8$, the number of identified edges doubles from 338,798 to 507,405, when the collection size is increased (from collection 300K to 1M). However, the cluster threshold $CT = 100$ substantially reduces the number of edges to 69,132 and 75,779, respectively, for collections 300K and 1M. We also find that, using $CT = 8$, the number of identified clusters grows linearly as the collection size increases; the algorithm-identified clusters in collections 150K, 300K, and 1M, are respectively 1,060, 3,059, and 9,763.

The growth in the number of clusters is unavoidable, as we expect that unaccounted duplicates and near-duplicates in the noise collection may form additional clusters. Nevertheless, we observe that the growth can be curbed to an extent, using $CT = 100$, as the number of identified clusters for collections 150K, 300K, and 1M, are respectively 1,178, 2,610 and 4,024. This is an important result, as it shows that the ND-CENTER algorithm using a
large cluster threshold value can restrict the number of edges, and the number of clusters as
the collection size is increased. Based on earlier observations on cluster quality, it also shows
that a high level of clustering effectiveness can be maintained in large collections.

To further analyse the relationship between the near-duplicate images within the noise
collection — the rest of the non-seeded images that are unknown to us — and those within
our seed collection, we measure the ratio of seed images that are identified amongst the
algorithm-formed clusters. As shown in Figure 6.10, the ratio between the number of seed
images that are distributed amongst clusters and the number of images identified within
the clusters increases with cluster threshold $CT$, whereas the ratio of the total seed images
within clusters to the total number of seed images (analogous to recall) in the collection falls
gradually with increasing cluster threshold. This shows that the pool of unknown images —
not within our seed collection — that are distributed amongst the clusters can be removed
gradually with little impact on the relevant near-duplicate images in the seed collection. At
clustering threshold $CT = 100$, we observe that the ratio between the seed images and the
total seed images remains at 80.8%. At the same threshold level, we observe 45% of seed
images amongst all images within the clusters, which means that about 55% of the images
within the pool are unaccounted for. This is expected given that the majority of the images
are crawled from the Web, and it is infeasible to ascertain that all images in the collection
are unique; we believe that near-duplicates unknown to us that are correctly identified by
the ND-CENTER algorithm constitute a portion of the unaccounted pool of images.

To corroborate this hypothesis, we perform a manual evaluation (where the author is
the human assessor) of the images within 100 clusters, selected using 100 randomly sampled
images from the unknown pool of images that are included in the clusters; we then tally the
total number of images within these clusters to determine the percentage of near-duplicates.
We observe that from the 100 randomly selected algorithm-identified clusters (with a total
of 231 images), there are 98 near-duplicate images present, and consequently, 133 (or 58%
of the 231 sampled images) non-near-duplicates. Therefore, on the assumption that the 231
random images are accurate samples of the image population, this result implies that among
the 55% of the images that are unaccounted for within the identified clusters, approximately
42% of the images are near-duplicates unknown to us, and only a relatively small percentage
are false positives.

Nevertheless, it is evident that the level of effectiveness can be further improved; when
100% of the total number of seed images are identified, about 80% of the images in all
identified clusters are unaccounted for. This also implies that we cannot expect the formation
of high quality clusters if we require high recall. This observation also corroborates our earlier experimental results in Section 6.4.1 on cluster quality. Finally, we observe that using cluster threshold $CT = 8$ for collection 1M — where the recall of the seed images is close to 100% — there is a total of 116,241 images in all identified clusters; this means that less than 12% of one million images are clustered. This means that, even in an extreme situation where 100% recall is required, the ND-CENTER algorithm can discard close to 90% of the image collection while retaining close to 100% of all (seeded) near-duplicate images. While further work is warranted to improve the effectiveness of the ND-CENTER clustering method, the results strongly indicate that our approach remains effective even in large collections of up to a million images.

### 6.4.3 Clustering effectiveness on web images

To further validate the results of the ND-CENTER algorithm, we perform clustering experiments on the web collection. Instead of the using the 200 artificially seeded clusters, as used in previous experiments, we use 205 human-evaluated clusters, as described earlier in Section 6.3.

Table 6.1 shows results for the clustering approach on this collection for each of the 23 subject queries. Columns two and three list the number of human-evaluated clusters and the number of algorithm-formed clusters for each subject query, respectively. The measures of average recall, average precision, average purity, and average entropy for the algorithm-formed clusters within each of the 23 subjects are shown in columns four to seven. The results in this table show that the average number of algorithm-formed clusters for each human-evaluated cluster is approximately 1.7. This means that on average, for every human-evaluated cluster, the ND-CENTER clustering method can identify all images belonging to this cluster in no more than 2 algorithm-identified groups. This implies that the amounts of over-clustering is relatively low. Although the average recall is not as high as those observed for collections 150K, 300K, and 1M, it remains at an acceptable average level of 70% recall and 80% precision.

We expect this level of effectiveness for this collection, considering that there are some very severe alterations within each of the subject groups in the collection, as discussed in Chapter 3 (page 79). The relatively low recall value can also be explained by the smaller number of near-duplicate images in each cluster as compared to the artificial collections; hence, the fluctuation in recall level is more abrupt. This was similarly observed in Chapter 5.
### Table 6.1: Clustering results for the web collection for 23 different subjects; each subject consists of \( c \) human-evaluated clusters. Column three shows the total number of algorithm-identified clusters that contain images from the \( c \) clusters of each category. Columns four and five report the average recall and precision (%) of the identified clusters, respectively. Average purity and average entropy are also reported, respectively, in the final two columns.

<table>
<thead>
<tr>
<th>Query Subject</th>
<th>( c )</th>
<th>( m_k )</th>
<th>Average</th>
<th>Average</th>
<th>Average</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Recall (%)</td>
<td>Precision (%)</td>
<td>( p(k) )</td>
<td>( h(k) )</td>
</tr>
<tr>
<td>50 cent</td>
<td>50</td>
<td>25</td>
<td>77.57</td>
<td>68.67</td>
<td>0.71</td>
<td>0.04</td>
</tr>
<tr>
<td>aaliyah</td>
<td>46</td>
<td>24</td>
<td>74.27</td>
<td>72.32</td>
<td>0.78</td>
<td>0.04</td>
</tr>
<tr>
<td>angelina</td>
<td>12</td>
<td>6</td>
<td>86.11</td>
<td>100.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>batman</td>
<td>22</td>
<td>11</td>
<td>65.83</td>
<td>84.64</td>
<td>0.78</td>
<td>0.03</td>
</tr>
<tr>
<td>marley</td>
<td>40</td>
<td>20</td>
<td>54.58</td>
<td>80.48</td>
<td>0.78</td>
<td>0.03</td>
</tr>
<tr>
<td>spears</td>
<td>6</td>
<td>3</td>
<td>63.33</td>
<td>68.69</td>
<td>0.70</td>
<td>0.04</td>
</tr>
<tr>
<td>electra</td>
<td>16</td>
<td>8</td>
<td>70.79</td>
<td>88.89</td>
<td>0.89</td>
<td>0.02</td>
</tr>
<tr>
<td>beckham</td>
<td>12</td>
<td>6</td>
<td>94.44</td>
<td>100.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>ferrari</td>
<td>14</td>
<td>7</td>
<td>80.95</td>
<td>89.29</td>
<td>0.89</td>
<td>0.02</td>
</tr>
<tr>
<td>garfield</td>
<td>10</td>
<td>5</td>
<td>82.50</td>
<td>79.55</td>
<td>0.60</td>
<td>0.06</td>
</tr>
<tr>
<td>potter</td>
<td>22</td>
<td>11</td>
<td>54.81</td>
<td>76.07</td>
<td>0.85</td>
<td>0.02</td>
</tr>
<tr>
<td>cobain</td>
<td>46</td>
<td>23</td>
<td>68.34</td>
<td>72.86</td>
<td>0.76</td>
<td>0.03</td>
</tr>
<tr>
<td>lamborghini</td>
<td>16</td>
<td>8</td>
<td>75.00</td>
<td>92.06</td>
<td>0.92</td>
<td>0.01</td>
</tr>
<tr>
<td>mustang</td>
<td>4</td>
<td>11</td>
<td>100.00</td>
<td>100.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>porsche</td>
<td>2</td>
<td>23</td>
<td>66.67</td>
<td>100.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>snoopy</td>
<td>12</td>
<td>8</td>
<td>94.44</td>
<td>95.83</td>
<td>0.96</td>
<td>0.01</td>
</tr>
<tr>
<td>south park</td>
<td>24</td>
<td>13</td>
<td>70.36</td>
<td>67.74</td>
<td>0.65</td>
<td>0.06</td>
</tr>
<tr>
<td>spiderman</td>
<td>20</td>
<td>11</td>
<td>47.47</td>
<td>63.52</td>
<td>0.73</td>
<td>0.03</td>
</tr>
<tr>
<td>superman</td>
<td>22</td>
<td>10</td>
<td>73.49</td>
<td>78.92</td>
<td>0.78</td>
<td>0.03</td>
</tr>
<tr>
<td>schiavo</td>
<td>44</td>
<td>22</td>
<td>68.37</td>
<td>80.78</td>
<td>0.77</td>
<td>0.04</td>
</tr>
<tr>
<td>simpsons</td>
<td>8</td>
<td>5</td>
<td>62.78</td>
<td>79.17</td>
<td>0.79</td>
<td>0.03</td>
</tr>
<tr>
<td>tupac</td>
<td>28</td>
<td>16</td>
<td>60.67</td>
<td>86.02</td>
<td>0.89</td>
<td>0.02</td>
</tr>
<tr>
<td>yoda</td>
<td>26</td>
<td>11</td>
<td>68.88</td>
<td>73.35</td>
<td>0.76</td>
<td>0.04</td>
</tr>
</tbody>
</table>
6.5. DISCUSSION

for retrieval tasks. Table 6.1 also shows that the quality of the clusters in each subject group remains high, at an observed average of 0.82 and 0.03 for purity and entropy, respectively.

Figure 6.11 shows some examples of the algorithm-identified clusters using the web images. We observe, from these examples, that some images within certain clusters do not belong; images from clusters 199, 235, 919, and 1210 are examples of failed cases. These observations are consistent with the analysis and results obtained from the seeded clusters in Section 6.4.1, where the algorithm occasionally generates clusters with a high mixture of different images. Upon further analysis, we find that this anomaly is a direct result of the sharing of multiple identical local structures — such as embedded textual information, boxes, and lines — amongst non-near-duplicate images. This causes the algorithm to falsely identify these images as being near-duplicates given that there is substantial correspondences amongst the images. Another example can be drawn from clusters 351 and 363, where there is an instance of over-clustering. This is partially attributable to the textual information at the bottom and top portions of the images in cluster 363. Clearly, the effectiveness of this algorithm is highly dependent on the local image structures; it is also one of the limitations of this approach. Nevertheless, the observed level of effectiveness suggests that this algorithm can be applied for even uncontrolled collections of web images, given that the rate of false positive clusters is considerably lower than that of the true positive ones.

Overall, the results reflect the high effectiveness of the ND-CENTER clustering algorithm for both large generated image collections and real world examples on the Web. More examples of algorithm-identified clusters are shown in Figure D.1 (page 210).

6.5 Discussion

We have described an effective method of clustering near-duplicate images in large collections by adapting a standard text-document clustering approach. This method exploits the pairwise relationship of near-duplicate images that are computed by the HPC algorithm. Our main contribution is the adaptation of a text-document clustering approach for the domain of near-duplicate image clustering; we have also shown empirically that it performs well for collections of up to a million images.

Our choice of the CENTER algorithm for this adaptation is justified by the fact that it is a standard yet simple near-duplicate text document detection technique, where clustering techniques in the text-document domain mainly differ in their distinct representation of text-shingles [Broder et al., 1997; Haveliwala et al., 2000; Hoad and Zobel, 2003; Yang and Callan,
Figure 6.11: Examples of some clusters that are identified by the ND-CENTER algorithm — using the same settings as those reported in our experiments — on images retrieved from the Web (Google retrieved images). The numbers on the left column denote the cluster IDs.
While it is conceivable that different methods of text-document clustering can also be applied for this domain, we do not as yet address the comparison of these various methods, where our focus is on the evaluation and experimentation of the adaptation of a standard technique. Our contribution in this chapter lies in the fact that we have demonstrated — via thorough experimental evaluation — that a standard near-duplicate text-document clustering method can indeed be adapted to yield high effectiveness in this domain. We have also empirically shown that such a technique can be applied for image collections of substantial sizes, provided that the appropriate image features are used.

The approach that we have presented here is mainly a memory-based structure, where the algorithm performs the majority of processing in-memory, with the exception of our slightly different implementation for collection 1M, as discussed in Chapter 5 (page 149). It is conceivable that our approach can work well for collections in the order of $10^7$, but we suspect collections of such scale would have an adverse impact on the efficiency of this approach. We conjecture that a truly scalable approach for web-scale collections (millions or even billions of images) is one that uses a more efficient representation method such as those of CBIR features; where each image is typically represented compactly using a single vector. For scalable clustering of near-duplicate images, object-recognition approaches such as those of Philbin et al. [2007] and Sivic and Zisserman [2003] are promising, and can be adapted for near-duplicate image detection.

While we note that potentially suitable candidates for comparison include the work of Chang et al. [1999] and Qamra et al. [2005], we also note that these methods are not directly comparable without further work, and so we are disadvantaged by the absence of a suitable baseline. This is within our expectation given that clustering of near-duplicate images in large collections is not a well explored domain.

Nevertheless, we have presented the first approach that demonstrates such a high level of effectiveness for clustering of near-duplicate images in large collections. Our experiments demonstrate the applicability of our approach — the adaptation of a standard text-document clustering technique — for near-duplicate images, and corroborates our hypothesis that we can use the pairwise similarity between two images computed by the HPC algorithm to form natural clusters of near-duplicate images.
6.6 Summary

We have demonstrated the first effective clustering approach for near-duplicate images that combines techniques from computer vision and text-document clustering methods. We have successfully extended the HPC algorithm, and adapted a standard text-document clustering algorithm for near-duplicate images. The experimental results indicate that our approach yields high effectiveness even for large collections of a million images. Our clustering algorithm has been shown to effectively generate non-overlapping clusters containing large concentrations of relevant near-duplicate images belonging to the same set. We have demonstrated effectiveness in both controlled collections and real examples of near-duplicates that are gathered from the Web.

Although much work is needed to improve the scalability of our approach for web-scale image collections, this clustering method offers a promising approach for organising near-duplicate images in large collections, and presents a practical solution to the challenges of image redundancy.
Chapter 7

Conclusion

We have developed methods for efficient and effective detection of near-duplicate images in large collections. Reliable means of managing near-duplicate images in large collections are vital, given the growing prevalence of digital images in private repositories and on the Web; they allow image redundancy and violation of image copyright to be effectively detected.

Our investigation of previous approaches for near-duplicate image detection has shown that they are of limited practicality for such tasks; methods that are highly effective are computationally expensive, whereas fast methods lack robustness against the variations of photometric conditions and imaging parameters that are common in near-duplicate images. In contrast, our methods have been demonstrated to provide a practical solution for detection of near-duplicate images in large collections, striking a good balance between effectiveness and efficiency; our methods have also extended the limitations of the state-of-the-art approaches in this domain.

7.1 Research contributions

We have described methods for detecting near-duplicate images that address the limitations of previous approaches. We have investigated the nature of near-duplicate images, and the kinds of near-duplicate images on the Web. Our method improves the efficiency of the previous query-based approach based on PCA-SIFT local descriptors, while maintaining comparable effectiveness. In addition, we have adapted a text-document near-duplicate detection approach to automatically detect near-duplicate images — characterised by local descriptors — without a query image. We have extended this approach for clustering near-duplicate images, and showed empirically that our method clusters near-duplicates with high
effectiveness using only modest resources.

We have shown that our proposed methods are robust against a wide range of digital image alterations, photometric variations, and image degradations. Through experimentation on large-scale image collections, we have demonstrated our methods to be more efficient than previously proposed methods in detecting near-duplicate images, and that the methods proposed in this thesis can scale well for large image collections.

Motivated by the need for realistic data and the unavailability of such data in this domain, we have collated a moderate-sized collection of web images, where we manually evaluated the near-duplicates present to establish groundtruth. We have also performed in-depth analysis of the kinds of near-duplicates prevalent on the Web. We have tested all our proposed methods on the web data to gauge their performance on realistic data. We believe this to be the first study of near-duplicate image detection using real web data.

Our experiments on near-duplicate images that are manually gathered from the Web have also shown that the results produced by our proposed methods are not as good as those produced on the controlled collections; this indicates that the results on controlled collections cannot serve as reliable indicators as to the performance of near-duplicate detection schemes on the Web. Nevertheless, the performance of our proposed methods are satisfactory, considering that our methods have outperformed baseline methods — in terms of a balance between efficiency and effectiveness — when tested on the same web data.

**Fast and effective near-duplicate detection in large collections**

We have demonstrated that a simple pruning approach to reduce PCA-SIFT local descriptors, computed from SIFT keypoints (or difference-of-Gaussian regions), can substantially improve the efficiency of near-duplicate image retrieval. We have shown that our pruning strategy allows near-duplicate images to be identified with considerably higher efficiency than previous approaches, and at the same time achieve comparable effectiveness in terms of recall and precision. With our proposed keypoint-reduction strategy, scalability with application of bag-of-vectors (local descriptors) for near-duplicate image detection are substantially improved. For a small collection of 20,000 images, our experiments have shown that our method produces results comparable to those of Ke et al. [2004] at slightly lower levels of effectiveness. In experiments using our reduction strategy, where only 10% of the features of the original approach are used, we observe recall and precision of 91% and 99% respectively, compared to 97% recall and 93% precision produced by the previous approach of Ke et al. [2004]. Our
scheme requires an average of only 2.2 seconds per query, compared to the original average of 152.4 seconds. This translates to a saving in run-time by a factor of 70. We have also shown that, by substantially reducing the number of local descriptors, our method achieves an 80-fold reduction in memory requirements.

We have shown our method to scale well for large collections, for which the previous method would have been infeasible; we have demonstrated that as many as 50 variants of near-duplicate images can be effectively searched and returned from a collection of a million images within approximately 30 seconds in total, yielding recall and precision of 91% and 99% respectively. We have also additionally demonstrated that our approach is more effective than the DPF method [Qamra et al., 2005] — one of the previous approaches that applies standard colour and texture features. We have also shown our method (and also the original PCA-SIFT method) to be considerably more robust than the DPF method.

Motivated by the considerable index sizes generated by the LSH index used in the framework of Ke et al. [2004], we have investigated a more compact indexing structure using Redundant Bit Vectors (RBV). We have demonstrated that, even though the RBV index was not initially designed for typical queries, where a large number of positive matches is expected, it can be adapted for this purpose. We have shown that this indexing scheme can be extended to identify near-duplicate images. In our experiments, a query on the RBV index, for 50 image alterations in a collection of 20,000 images, can (on average) be resolved with 87% recall and 99% precision. While the average search time of 23 seconds is higher than the 2.2 seconds for the LSH index, the RBV index is far more compact. The RBV structure indexed a collection of 20,000 images in less than 150 MB of memory, whereas the latter used 1.2 GB. While it yields a satisfactory level of effectiveness and is more compact than the LSH index, it does not achieve the same level of effectiveness or efficiency as the approach of Ke et al. [2004] or our pruning scheme that employs the LSH index.

We have also described an approach to automatically detect near-duplicate images using the PCA-SIFT local descriptors without a specified query. We have adapted a method that was originally used for near-duplicate text-document detection to create this non-query-based approach — which we refer to as the HPC method — and demonstrated empirically that our method can effectively identify the pairwise near-duplicate relationship between any two images within a collection. We have shown that the number of matching local descriptors between two images can be efficiently approximated using this method to reflect the estimated near-duplicate relationship; therefore, the collection of images can be effectively and efficiently preprocessed, such that search and retrieval of near-duplicate images corresponding
to a query can be substantially expedited. We have demonstrated that our proposed method is capable of achieving a level of effectiveness that, while not as good as those achieved by the baseline query-based approaches of our pruning scheme and that of Ke et al. [2004], is satisfactory. The HPC method accurately identified 81% of all 10,000 known near-duplicate images with 91% precision within a million images, with the entire process completed within an hour on a typical workstation. Our non-query-based method was also shown to be more efficient than the approach of Qamra et al. [2005], while achieving a higher level of effectiveness. Our experiments have also shown that our non-query-based approach can scale well for a large collection of a million images.

Finally, by extending the non-query-based detection approach, we have discovered that standard near-duplicate text-document clustering approaches can be easily modified to effectively and accurately cluster near-duplicate images. We have shown that the estimated near-duplicate relationship as computed by our non-query-based approach can be exploited using a thresholding scheme to empirically form near-duplicate clusters. Our experimental results have indicated that the ND-CENTER algorithm we developed is effective for clustering near-duplicate images using modest resources; the clustering process for a million images can be completed within an hour. We have observed that 135 non-overlapping clusters were effectively created out of 200 known near-duplicate clusters planted within a collection of a million images; precision levels for all 135 clusters were recorded at 100%, whereas recall levels varied between 60% and 90%. While we have found that a large number of unknown images — from the non-seeded (noise) collection — were also identified as additional near-duplicate clusters, our manual evaluation of a small sample indicated that only a relatively small percentage were false positives; the rest were near-duplicates present in the noise collection that were not known in our groundtruth. Arguably, we have demonstrated the first effective clustering approach for near-duplicate images.

**Detection of near-duplicates in web images**

In addition to controlled image collections where near-duplicate instances were artificially seeded, we have also gathered a moderate-sized collection of approximately 19,000 images that are returned from a text-based web image search engine using 25 popular queries. We have analysed the characteristics of near-duplicate images on the Web, and also examined the different types of near-duplicates common among these images. We have made some general observations regarding interesting links between near-duplication and image content,
and noted that the characteristics of near-duplicates on the Web do not always coincide with those of the controlled collections.

The manually identified near-duplicate instances have also been used for experimentation with our proposed methods and the appropriate baselines to observe the effectiveness of these methods on real-world data. Our experiments using web images showed that the PCA-SIFT query-based approach by Ke et al. [2004] yields an average recall of 65% and average precision of 72%, whereas the standard DPF approach using colour and texture features as proposed by Qamra et al. [2005] produces considerably lower effectiveness, with average precision of 38% at the same recall level. Our keypoint-reduced PCA-SIFT query-based approach yields a level of effectiveness comparable to that of the original PCA-SIFT query-based approach, with 58% average recall and 70% average precision. While our proposed method is not as effective as the former approach, it is considerably more efficient in query evaluation; it is also more effective than the standard DPF approach.

Using the web images, our proposed non-query-based method — the HPC algorithm — produced results with 57% average recall and 63% average precision, which indicates comparable effectiveness for even uncontrolled collections. Results from our comparative assessment, of both the query-based and non-query-based approaches, for web images showed that using the query-based approach to identify the pairwise near-duplicate relationship of image collections is implausible given the computational complexity, and that the HPC algorithm offers a practical solution for this task.

Testing of the ND-CENTER clustering method — which adapts standard text-document clustering approaches — on web images showed that our clustering method yields satisfactory results. By evaluating the algorithm-identified clusters using all human-evaluated clusters from the web images, we observed 70% recall and 80% precision on average; this means that, approximately 70% of the images that were identified within the human-evaluated clusters were also correctly identified to their respective groups, where no more than 20% of the images within each algorithm-identified cluster were false positives.

Overall, while the results of all our proposed methods were not as good as those observed in the controlled collections, they were comparable to previous methods of Ke et al. [2004] and Qamra et al. [2005], while producing generally higher efficiency.
7.2 Future research

We have shown that the level of effectiveness of our methods to be comparable to previous approaches, while exhibiting considerably higher efficiency on collections of up to a million images. Nevertheless, it is conceivable that these methods can be improved with further research.

We have shown that pruning the computationally expensive PCA-SIFT local descriptors improves the scalability of the bag-of-vectors approach by a wide margin, but the scalability for web-scale collections of hundreds of millions (or billions) of images has not been investigated. It is probable that, for approaches that are truly scalable for collections of billions of images, more efficient means of image representation are required. Some recent work — such as that of Nistér and Stewénius [2006], and Philbin et al. [2007] — has focused on improving the scalability of distinctive features — such as local descriptors for object-recognition and video keyframe detection. We believe that their techniques show promise, and that they can be adapted for highly scalable near-duplicate image detection. More scalable image features that retain the distinctive characteristics of the bag-of-vectors approach can also positively impact the methods proposed in this thesis.

As previously discussed, we have shown that alternative indexing structures such as the modified RBV indexing scheme can reduce memory requirements, while producing results comparable to predominant indexing schemes for high-dimensional data structures. While the advantages in memory requirements of this scheme are evident, the scalability of query evaluation using this scheme remains an issue. We believe that this approach has ample room for improvement, where the factors that hinder scalability need to be further researched.

We have demonstrated that a non-query-based approach — the HPC algorithm — that we developed can be used to efficiently identify the pairwise near-duplicate similarity of an entire collection. While the effectiveness of this method was shown to be high, it was not as good as those observed for query-based detection. This is expected, as we made a conscious decision to omit the verification stage for improved speed, at the cost of more false positives. We believe that further research to incorporate efficient verification methods could improve the quality of results to be comparable to that of the query-based method, while retaining its advantages in efficiency.

The clustering method that we have described in this thesis is the first to show a high level of effectiveness and efficiency in grouping near-duplicate images. This was shown for controlled images, and for near-duplicate examples that were sampled from the Web, even
7.3. SUMMARY

though our clustering method — the ND-CENTER algorithm — uses a relatively simple and standard clustering technique that incorporates the HPC algorithm. With further research and experiments, we believe that the effectiveness and efficiency of this clustering method can be further improved.

All the proposed approaches in this thesis treat images within a collection as equals, where our methods are tested using an adversarial approach. We do not take advantage of some general image classification and categorisation techniques to apply a two-pass filter: first to classify image groups into sub-categories (such as images of scenery, or images with faces), then to apply near-duplicate detection on each sub-category. This approach is worth investigating as such a partitioning scheme could further improve the scalability of the algorithms — especially the clustering method — proposed in this thesis.

In light of the results that were obtained from our experiments using web images, it is evident that controlled collections with artificially seeded near-duplicate images are not ideal for testing approaches that are proposed for the Web. Our investigation has also revealed that, to date, there has been no prior published study of near-duplicate images on the Web. To this end, it is also worthwhile to pursue efficient means of deriving a large collection of near-duplicate web samples that can be used for further experiments. A larger collection of web images also means that algorithms and experiments can be designed with more confidence to cater specifically for the kinds of near-duplicates on the Web.

7.3 Summary

In this thesis, we have presented a detailed study of the nature of near-duplicate images, and explored the problem of near-duplication on large image collections. We have also shown that our proposed approaches are efficient and effective for detecting near-duplicate images. While the methods presented here can undoubtedly be improved with further research, our experiments have indicated that they are sufficiently reliable for near-duplicate detection in large collections — even when near-duplicate instances are severely altered or suffer from degradation in image quality.

Near-duplicate image detection is a relatively new field of research that is nonetheless of practical value due to the prevalence of digital image data in private and public repositories. Previous approaches in the field of near-duplicate image detection have either been limited to small data sets, or suffer from poor effectiveness against the wide variation of image alterations and imaging conditions common in near-duplicate images. While the approaches
presented in this thesis do not present final solutions to the research problems in this domain, we have shown — through extensive experiments on large data sets and near-duplicate examples on the Web — that they address the limitations, and substantially extend, the state-of-art approaches. The methods presented in this thesis are the first to demonstrate efficient and effective approaches to identifying near-duplicate images in large collections.
Appendix A

Taylor’s quadratic expansion
(matrix form)

Equation A.1 shows the quadratic Taylor’s expansion on the scale-space extremum using the notation of Lowe [2004].

\[ DoG(x) = DoG + \left( \frac{\partial DoG}{\partial x} \right)^T x + \frac{1}{2} x^T \left( \frac{\partial^2 DoG}{\partial x^2} \right) x \]  

\hspace{1cm} \text{(A.1)}

is equivalent to:

\[ DoG(x) = DoG + \left( \begin{array}{c} \frac{\partial DoG}{\partial x} \\ \frac{\partial DoG}{\partial y} \\ \frac{\partial DoG}{\partial \sigma} \end{array} \right)^T \left( \begin{array}{c} x \\ y \\ \sigma \end{array} \right) + \frac{1}{2} \left( \begin{array}{c} x \\ y \\ \sigma \end{array} \right)^T \left( \begin{array}{ccc} \frac{\partial^2 DoG}{\partial x^2} & \frac{\partial^2 DoG}{\partial x \partial y} & \frac{\partial^2 DoG}{\partial x \partial \sigma} \\ \frac{\partial^2 DoG}{\partial y \partial x} & \frac{\partial^2 DoG}{\partial y^2} & \frac{\partial^2 DoG}{\partial y \partial \sigma} \\ \frac{\partial^2 DoG}{\partial \sigma \partial x} & \frac{\partial^2 DoG}{\partial \sigma \partial y} & \frac{\partial^2 DoG}{\partial \sigma^2} \end{array} \right) \left( \begin{array}{c} x \\ y \\ \sigma \end{array} \right) \]  

\hspace{1cm} \text{(A.2)}

where \( x = [x, y, \sigma]^T \) is a 3 element column vector and \( DoG \) maps \( x \) to a scalar value (or the initial sample point).
Appendix B

Evaluation of Local descriptors

The repeatability of the each local descriptors for all 50 image alterations are detailed in Table B.1, Table B.2, and Figure B.1. Figure 4.7 and Figure B.3 show the average relative ranks of various altered images using the DPF method and PCA-SIFT query-based methods, respectively.
<table>
<thead>
<tr>
<th>Alteration</th>
<th>Def. 1000</th>
<th>100</th>
<th>10</th>
<th>Alteration</th>
<th>Def. 1000</th>
<th>100</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>coloriseB</td>
<td>84.4</td>
<td>87.2</td>
<td>90.2</td>
<td>88.9</td>
<td>satr_80</td>
<td>82.4</td>
<td>85.5</td>
</tr>
<tr>
<td>coloriseG</td>
<td>78.2</td>
<td>81.3</td>
<td>85.1</td>
<td>83.8</td>
<td>contr_mns</td>
<td>68.5</td>
<td>72.8</td>
</tr>
<tr>
<td>coloriseR</td>
<td>81.0</td>
<td>84.1</td>
<td>87.1</td>
<td>85.7</td>
<td>crop_5</td>
<td>40.5</td>
<td>45.7</td>
</tr>
<tr>
<td>contr_mns</td>
<td>68.5</td>
<td>72.8</td>
<td>76.5</td>
<td>73.5</td>
<td>crop_20</td>
<td>31.6</td>
<td>37.3</td>
</tr>
<tr>
<td>contr_pls</td>
<td>67.9</td>
<td>69.9</td>
<td>73.6</td>
<td>71.1</td>
<td>despeckle</td>
<td>59.0</td>
<td>59.8</td>
</tr>
<tr>
<td>crop_5</td>
<td>40.5</td>
<td>45.7</td>
<td>49.6</td>
<td>42.7</td>
<td>border_0</td>
<td>38.8</td>
<td>37.0</td>
</tr>
<tr>
<td>crop_10</td>
<td>36.3</td>
<td>42.1</td>
<td>47.0</td>
<td>38.2</td>
<td>border_3</td>
<td>36.1</td>
<td>35.0</td>
</tr>
<tr>
<td>crop_20</td>
<td>31.6</td>
<td>37.3</td>
<td>42.8</td>
<td>35.2</td>
<td>border_2</td>
<td>31.8</td>
<td>30.5</td>
</tr>
<tr>
<td>crop_30</td>
<td>26.4</td>
<td>31.0</td>
<td>36.4</td>
<td>32.3</td>
<td>rotate_90</td>
<td>46.6</td>
<td>51.4</td>
</tr>
<tr>
<td>despeckle</td>
<td>59.0</td>
<td>59.8</td>
<td>71.4</td>
<td>70.4</td>
<td>rotate_270</td>
<td>46.3</td>
<td>50.5</td>
</tr>
<tr>
<td>jpeg_to_gif</td>
<td>73.2</td>
<td>78.5</td>
<td>84.5</td>
<td>84.4</td>
<td>resize_12</td>
<td>4.6</td>
<td>3.7</td>
</tr>
<tr>
<td>border_1</td>
<td>32.7</td>
<td>31.6</td>
<td>36.2</td>
<td>30.4</td>
<td>resize_12</td>
<td>4.6</td>
<td>3.7</td>
</tr>
<tr>
<td>border_1</td>
<td>32.7</td>
<td>31.6</td>
<td>36.2</td>
<td>30.4</td>
<td>resize_12</td>
<td>4.6</td>
<td>3.7</td>
</tr>
<tr>
<td>border_2</td>
<td>31.8</td>
<td>30.5</td>
<td>33.3</td>
<td>14.7</td>
<td>resize_12</td>
<td>4.6</td>
<td>3.7</td>
</tr>
<tr>
<td>border_3</td>
<td>36.1</td>
<td>35.0</td>
<td>41.6</td>
<td>38.3</td>
<td>resize_400</td>
<td>60.2</td>
<td>62.8</td>
</tr>
<tr>
<td>rotate_90</td>
<td>46.6</td>
<td>51.4</td>
<td>57.6</td>
<td>53.7</td>
<td>resize_800</td>
<td>59.2</td>
<td>62.0</td>
</tr>
</tbody>
</table>

Table B.1: Average repeatability (%) of descriptor within $L_2$ norm threshold at every level of reduction. Columns 1, and 6 indicate the different alterations. Keypoint thresholds of 1000, 100, and 10 are used. Default indicates the original number of keypoints per image (average of 1594). The results are averaged over 100 sets of images, with 50 alterations each.
## APPENDIX B. EVALUATION OF LOCAL DESCRIPTORS

<table>
<thead>
<tr>
<th>Alteration</th>
<th>surf</th>
<th>sift</th>
<th>gloh</th>
<th>Alteration</th>
<th>surf</th>
<th>sift</th>
<th>gloh</th>
</tr>
</thead>
<tbody>
<tr>
<td>coloriseB</td>
<td>86.82</td>
<td>87.16</td>
<td>92.07</td>
<td>satr_80</td>
<td>84.53</td>
<td>85.59</td>
<td>88.50</td>
</tr>
<tr>
<td>coloriseG</td>
<td>83.57</td>
<td>81.27</td>
<td>89.24</td>
<td>satr_90</td>
<td>87.10</td>
<td>87.17</td>
<td>91.53</td>
</tr>
<tr>
<td>coloriseR</td>
<td>84.86</td>
<td>84.29</td>
<td>90.30</td>
<td>satr_110</td>
<td>86.30</td>
<td>86.80</td>
<td>91.45</td>
</tr>
<tr>
<td>contr_mns</td>
<td>77.37</td>
<td>73.93</td>
<td>80.73</td>
<td>satr_120</td>
<td>84.32</td>
<td>85.41</td>
<td>88.88</td>
</tr>
<tr>
<td>contr_pls</td>
<td>74.85</td>
<td>71.20</td>
<td>75.92</td>
<td>int_80</td>
<td>79.86</td>
<td>72.13</td>
<td>88.15</td>
</tr>
<tr>
<td>crop_5</td>
<td>55.66</td>
<td>49.72</td>
<td>47.42</td>
<td>int_90</td>
<td>82.39</td>
<td>77.93</td>
<td>89.13</td>
</tr>
<tr>
<td>crop_10</td>
<td>48.80</td>
<td>49.34</td>
<td>42.95</td>
<td>int_110</td>
<td>81.98</td>
<td>79.39</td>
<td>90.05</td>
</tr>
<tr>
<td>crop_20</td>
<td>38.75</td>
<td>47.65</td>
<td>39.19</td>
<td>int_120</td>
<td>78.84</td>
<td>74.32</td>
<td>87.76</td>
</tr>
<tr>
<td>crop_30</td>
<td>30.07</td>
<td>42.52</td>
<td>32.92</td>
<td>crops_40</td>
<td>20.77</td>
<td>38.16</td>
<td>24.05</td>
</tr>
<tr>
<td>despeckle</td>
<td>72.99</td>
<td>63.45</td>
<td>79.03</td>
<td>crops_50</td>
<td>14.01</td>
<td>33.25</td>
<td>16.29</td>
</tr>
<tr>
<td>jpeg_to_gif</td>
<td>81.59</td>
<td>79.69</td>
<td>88.33</td>
<td>crops_90</td>
<td>0.79</td>
<td>11.30</td>
<td>0.36</td>
</tr>
<tr>
<td>border_0</td>
<td>44.70</td>
<td>48.05</td>
<td>39.24</td>
<td>ints_50</td>
<td>72.55</td>
<td>56.67</td>
<td>82.88</td>
</tr>
<tr>
<td>border_1</td>
<td>38.89</td>
<td>45.49</td>
<td>35.55</td>
<td>ints_150</td>
<td>66.39</td>
<td>58.00</td>
<td>68.56</td>
</tr>
<tr>
<td>border_2</td>
<td>33.27</td>
<td>44.10</td>
<td>35.75</td>
<td>contrs_mns</td>
<td>57.02</td>
<td>49.66</td>
<td>50.41</td>
</tr>
<tr>
<td>border_3</td>
<td>42.96</td>
<td>47.36</td>
<td>37.92</td>
<td>contrs_pls</td>
<td>48.71</td>
<td>41.34</td>
<td>38.96</td>
</tr>
<tr>
<td>rotate_90</td>
<td>67.35</td>
<td>67.82</td>
<td>62.73</td>
<td>rotcrop_90</td>
<td>14.18</td>
<td>39.31</td>
<td>16.41</td>
</tr>
<tr>
<td>rotate_180</td>
<td>61.88</td>
<td>68.91</td>
<td>52.58</td>
<td>rotcrop_180</td>
<td>14.24</td>
<td>43.23</td>
<td>16.45</td>
</tr>
<tr>
<td>rotate_270</td>
<td>68.53</td>
<td>69.45</td>
<td>63.06</td>
<td>rotcrop_270</td>
<td>14.05</td>
<td>39.25</td>
<td>16.17</td>
</tr>
<tr>
<td>resize_50</td>
<td>54.67</td>
<td>27.59</td>
<td>57.83</td>
<td>rotscale_90</td>
<td>51.74</td>
<td>31.54</td>
<td>37.76</td>
</tr>
<tr>
<td>resize_25</td>
<td>28.91</td>
<td>11.80</td>
<td>15.54</td>
<td>rotscale_180</td>
<td>50.53</td>
<td>31.19</td>
<td>31.71</td>
</tr>
<tr>
<td>resize_12</td>
<td>6.11</td>
<td>9.32</td>
<td>2.95</td>
<td>rotscale_270</td>
<td>52.08</td>
<td>31.85</td>
<td>36.55</td>
</tr>
<tr>
<td>resize_200</td>
<td>79.29</td>
<td>64.21</td>
<td>85.86</td>
<td>shear_x5</td>
<td>50.51</td>
<td>52.82</td>
<td>48.89</td>
</tr>
<tr>
<td>resize_400</td>
<td>79.97</td>
<td>64.79</td>
<td>85.53</td>
<td>shear_x15</td>
<td>23.87</td>
<td>27.42</td>
<td>30.61</td>
</tr>
<tr>
<td>resize_800</td>
<td>79.36</td>
<td>63.98</td>
<td>85.46</td>
<td>shear_x5y5</td>
<td>37.77</td>
<td>49.28</td>
<td>35.74</td>
</tr>
<tr>
<td>satr_70</td>
<td>82.77</td>
<td>84.34</td>
<td>86.36</td>
<td>shear_x15y15</td>
<td>19.64</td>
<td>38.30</td>
<td>27.04</td>
</tr>
</tbody>
</table>

Table B.2: Average repeatability (%) of all descriptors within $L_2$ norm threshold. Columns 1, and 6 indicate the different alterations. The results are averaged over 100 sets of images, with 50 alterations each. All local descriptors are computed using their original interest region detectors.
Figure B.1: Average repeatability (%) of all generated local descriptors — including PCA-SIFT (original and thresholds of 10, 100, and 1000), SURF, SIFT, and GLOH — are computed from the Difference-of-Gaussian (DoG), Hessian-Affine, and Fast-Hessian regions (see pp. 41—47) — in various altered images; the numbers in the parentheses for dog denote the applied threshold value. The results are averaged over 100 sets of images, with 50 alterations each.
Figure B.2: Average relative rank of the images from each of the (rest of the) 8 alteration groups as produced by the DPF method (using the optimal value of $m = 240$). There is a total of 18 groups of 50 image alterations; the rest of the 10 groups were earlier shown in Figure 4.7 (page 113).
Figure B.3: Average relative ranks of the images from each of the (rest of the) 8 alteration groups as produced by the PCA-SIFT methods (both the T100 method and the original approach). There is a total of 18 groups of 50 image alterations; the rest of the 10 groups were earlier shown in Figure 4.8 (page 115).
Appendix C

Effectiveness of the HPC on image alterations

Tables C.1, C.2, and C.3 show the detailed results of the HPC algorithm on all 50 image alteration using various parameters and collections.
Table C.1: Estimated coverage (%) and average precision (%) in pairs for minimum inclusion threshold ranging from $T=4$ to $T=255$ on image collection 20K for the list of image alterations on column one.

<table>
<thead>
<tr>
<th>Alt</th>
<th>$T_4$</th>
<th>$T_8$</th>
<th>$T_{16}$</th>
<th>$T_{32}$</th>
<th>$T_{64}$</th>
<th>$T_{128}$</th>
<th>$T_{255}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>coloriseB</td>
<td>98.0/11.2</td>
<td>97.2/40.4</td>
<td>96.3/78.4</td>
<td>94.9/94.8</td>
<td>92.4/98.2</td>
<td>88.3/98.9</td>
<td>80.5/99.8</td>
</tr>
<tr>
<td>coloriseG</td>
<td>97.9/11.0</td>
<td>97.0/38.8</td>
<td>96.2/78.0</td>
<td>94.8/94.5</td>
<td>92.1/98.3</td>
<td>87.7/99.0</td>
<td>79.5/99.8</td>
</tr>
<tr>
<td>coloriseR</td>
<td>97.9/10.8</td>
<td>97.1/39.5</td>
<td>96.3/78.1</td>
<td>95.0/95.1</td>
<td>92.3/98.2</td>
<td>87.9/99.0</td>
<td>80.0/99.8</td>
</tr>
<tr>
<td>contr_muns</td>
<td>97.9/11.0</td>
<td>97.0/40.9</td>
<td>95.6/78.7</td>
<td>93.7/94.0</td>
<td>90.0/98.0</td>
<td>85.0/98.8</td>
<td>74.8/98.7</td>
</tr>
<tr>
<td>contr_pls</td>
<td>97.7/9.3</td>
<td>96.7/36.7</td>
<td>95.3/76.8</td>
<td>93.1/94.7</td>
<td>89.1/98.3</td>
<td>82.6/99.3</td>
<td>68.9/97.8</td>
</tr>
<tr>
<td>crop_5</td>
<td>97.4/15.1</td>
<td>96.2/49.9</td>
<td>94.4/82.3</td>
<td>91.6/94.9</td>
<td>88.1/96.4</td>
<td>80.5/96.0</td>
<td>67.9/95.8</td>
</tr>
<tr>
<td>crop_10</td>
<td>96.7/14.9</td>
<td>94.8/49.4</td>
<td>93.0/82.2</td>
<td>90.2/94.9</td>
<td>85.2/96.2</td>
<td>75.5/94.8</td>
<td>60.4/94.7</td>
</tr>
<tr>
<td>crop_20</td>
<td>96.3/15.0</td>
<td>94.8/50.4</td>
<td>92.9/82.2</td>
<td>90.4/95.3</td>
<td>85.4/97.2</td>
<td>76.7/94.7</td>
<td>63.3/94.5</td>
</tr>
<tr>
<td>crop_30</td>
<td>95.3/15.8</td>
<td>93.4/52.6</td>
<td>91.4/85.9</td>
<td>87.2/95.8</td>
<td>80.6/95.3</td>
<td>71.1/94.9</td>
<td>53.0/89.7</td>
</tr>
<tr>
<td>despeckle</td>
<td>97.3/24.9</td>
<td>96.7/65.8</td>
<td>95.4/90.3</td>
<td>93.8/97.3</td>
<td>90.2/98.9</td>
<td>84.2/99.4</td>
<td>73.2/98.0</td>
</tr>
<tr>
<td>jpeg_to_gif</td>
<td>97.9/9.9</td>
<td>97.2/38.0</td>
<td>96.2/78.4</td>
<td>94.7/95.1</td>
<td>92.2/98.3</td>
<td>88.0/98.9</td>
<td>79.1/98.8</td>
</tr>
<tr>
<td>border_0</td>
<td>96.7/22.6</td>
<td>95.5/63.8</td>
<td>93.3/91.0</td>
<td>89.4/98.1</td>
<td>82.5/99.2</td>
<td>70.4/99.7</td>
<td>54.1/100.0</td>
</tr>
<tr>
<td>border_1</td>
<td>95.8/20.3</td>
<td>94.3/60.8</td>
<td>92.2/89.1</td>
<td>88.3/97.6</td>
<td>79.6/99.1</td>
<td>67.4/99.4</td>
<td>47.8/100.0</td>
</tr>
<tr>
<td>border_2</td>
<td>95.7/17.5</td>
<td>94.3/52.8</td>
<td>92.5/84.4</td>
<td>88.2/96.1</td>
<td>79.4/98.6</td>
<td>67.0/99.5</td>
<td>47.6/100.0</td>
</tr>
<tr>
<td>border_3</td>
<td>96.3/23.7</td>
<td>94.7/65.5</td>
<td>92.5/91.1</td>
<td>88.5/98.0</td>
<td>80.6/99.3</td>
<td>68.3/99.7</td>
<td>49.7/100.0</td>
</tr>
<tr>
<td>rotate_90</td>
<td>97.6/9.5</td>
<td>96.7/37.7</td>
<td>95.0/77.9</td>
<td>92.9/95.0</td>
<td>90.1/98.2</td>
<td>85.0/99.4</td>
<td>74.1/98.8</td>
</tr>
<tr>
<td>rotate_180</td>
<td>97.5/10.2</td>
<td>96.6/39.4</td>
<td>94.9/79.3</td>
<td>92.6/95.2</td>
<td>90.2/98.2</td>
<td>85.5/99.4</td>
<td>75.2/97.8</td>
</tr>
<tr>
<td>rotate_270</td>
<td>97.5/9.6</td>
<td>96.4/37.1</td>
<td>94.9/77.4</td>
<td>92.5/95.1</td>
<td>89.3/98.3</td>
<td>84.3/98.4</td>
<td>73.6/98.8</td>
</tr>
<tr>
<td>resize_50</td>
<td>96.6/23.7</td>
<td>95.4/65.7</td>
<td>93.2/92.5</td>
<td>89.5/97.8</td>
<td>81.8/99.5</td>
<td>70.3/99.0</td>
<td>57.2/96.0</td>
</tr>
<tr>
<td>resize_25</td>
<td>89.5/37.4</td>
<td>84.8/82.7</td>
<td>78.2/94.9</td>
<td>69.2/98.1</td>
<td>56.0/98.0</td>
<td>41.4/92.0</td>
<td>25.1/76.0</td>
</tr>
<tr>
<td>resize_12</td>
<td>65.1/84.9</td>
<td>56.6/96.5</td>
<td>45.9/94.2</td>
<td>32.9/88.0</td>
<td>15.4/61.0</td>
<td>1.7/9.0</td>
<td>0.0/0.0</td>
</tr>
<tr>
<td>resize_200</td>
<td>97.7/14.2</td>
<td>97.0/49.8</td>
<td>96.1/84.7</td>
<td>94.8/95.9</td>
<td>91.8/98.6</td>
<td>87.6/99.2</td>
<td>78.5/99.9</td>
</tr>
<tr>
<td>resize_400</td>
<td>97.6/14.4</td>
<td>97.0/51.1</td>
<td>96.1/85.0</td>
<td>94.4/96.1</td>
<td>92.0/98.7</td>
<td>87.6/99.2</td>
<td>78.6/99.9</td>
</tr>
<tr>
<td>resize_800</td>
<td>97.5/14.9</td>
<td>97.0/51.3</td>
<td>96.0/84.9</td>
<td>94.7/96.1</td>
<td>91.8/98.6</td>
<td>87.5/99.2</td>
<td>78.6/99.9</td>
</tr>
<tr>
<td>resize_70</td>
<td>98.0/12.0</td>
<td>97.0/42.9</td>
<td>95.9/82.1</td>
<td>94.3/96.0</td>
<td>91.4/98.5</td>
<td>86.7/99.1</td>
<td>77.1/99.8</td>
</tr>
<tr>
<td>resize_80</td>
<td>98.0/12.2</td>
<td>97.2/44.3</td>
<td>96.1/82.1</td>
<td>94.5/95.7</td>
<td>91.8/98.4</td>
<td>87.5/99.0</td>
<td>78.8/99.8</td>
</tr>
<tr>
<td>resize_90</td>
<td>98.0/12.3</td>
<td>97.3/45.3</td>
<td>96.2/82.4</td>
<td>94.7/95.9</td>
<td>91.9/98.4</td>
<td>87.9/99.1</td>
<td>79.7/99.8</td>
</tr>
<tr>
<td>resize_110</td>
<td>98.1/12.7</td>
<td>97.1/45.6</td>
<td>96.2/83.2</td>
<td>94.7/95.8</td>
<td>91.8/98.5</td>
<td>87.7/99.0</td>
<td>79.4/99.8</td>
</tr>
</tbody>
</table>

continued on next page
### APPENDIX C. EFFECTIVENESS OF THE HPC ON IMAGE ALTERATIONS

<table>
<thead>
<tr>
<th>Alt</th>
<th>$T_4$</th>
<th>$T_8$</th>
<th>$T_{16}$</th>
<th>$T_{32}$</th>
<th>$T_{64}$</th>
<th>$T_{128}$</th>
<th>$T_{255}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>satr_120</td>
<td>97.8/12.6</td>
<td>97.0/44.7</td>
<td>96.1/83.5</td>
<td>94.2/96.2</td>
<td>91.5/98.4</td>
<td>87.2/99.1</td>
<td>77.8/99.8</td>
</tr>
<tr>
<td>int_80</td>
<td>97.8/13.1</td>
<td>96.9/46.2</td>
<td>95.9/83.6</td>
<td>94.3/96.0</td>
<td>91.0/98.6</td>
<td>85.6/99.5</td>
<td>74.0/98.8</td>
</tr>
<tr>
<td>int_90</td>
<td>97.8/12.0</td>
<td>97.0/43.7</td>
<td>96.2/81.9</td>
<td>94.5/95.7</td>
<td>91.3/98.4</td>
<td>86.7/99.1</td>
<td>77.0/98.8</td>
</tr>
<tr>
<td>int_110</td>
<td>98.0/12.7</td>
<td>97.2/45.0</td>
<td>96.3/82.3</td>
<td>94.5/95.8</td>
<td>91.8/98.3</td>
<td>88.0/98.9</td>
<td>79.0/99.7</td>
</tr>
<tr>
<td>int_120</td>
<td>97.9/11.0</td>
<td>97.1/41.2</td>
<td>96.1/79.5</td>
<td>94.3/95.1</td>
<td>91.2/98.1</td>
<td>87.3/98.9</td>
<td>76.7/99.7</td>
</tr>
<tr>
<td>crops_40</td>
<td>95.1/20.2</td>
<td>93.3/61.0</td>
<td>90.0/89.8</td>
<td>85.5/97.0</td>
<td>77.0/96.3</td>
<td>68.2/92.8</td>
<td>51.8/85.8</td>
</tr>
<tr>
<td>crops_50</td>
<td>94.2/23.8</td>
<td>91.7/66.9</td>
<td>88.6/91.9</td>
<td>83.2/97.0</td>
<td>74.7/97.1</td>
<td>63.2/96.1</td>
<td>52.2/93.7</td>
</tr>
<tr>
<td>crops_90</td>
<td>12.9/52.9</td>
<td>3.5/45.1</td>
<td>0.6/18.5</td>
<td>0.0/1.0</td>
<td>0.0/0.0</td>
<td>0.0/0.0</td>
<td>0.0/0.0</td>
</tr>
<tr>
<td>ints_50</td>
<td>96.7/28.8</td>
<td>95.0/67.5</td>
<td>92.7/91.6</td>
<td>88.9/96.8</td>
<td>82.4/98.2</td>
<td>70.6/95.9</td>
<td>53.0/93.0</td>
</tr>
<tr>
<td>ints_150</td>
<td>97.4/7.7</td>
<td>96.6/31.7</td>
<td>94.7/71.3</td>
<td>91.8/93.5</td>
<td>86.8/98.1</td>
<td>76.9/99.3</td>
<td>60.1/98.7</td>
</tr>
<tr>
<td>contrs_nms</td>
<td>95.4/12.2</td>
<td>93.0/40.3</td>
<td>89.3/78.9</td>
<td>84.5/93.2</td>
<td>75.0/97.5</td>
<td>60.1/96.3</td>
<td>40.0/93.7</td>
</tr>
<tr>
<td>contrs_pls</td>
<td>93.6/10.1</td>
<td>89.6/38.2</td>
<td>82.9/80.1</td>
<td>71.8/94.7</td>
<td>53.2/97.5</td>
<td>32.9/90.0</td>
<td>12.2/82.0</td>
</tr>
<tr>
<td>rotcrop_90</td>
<td>92.8/30.1</td>
<td>90.4/74.6</td>
<td>87.0/93.7</td>
<td>81.6/97.7</td>
<td>69.7/98.5</td>
<td>57.4/99.4</td>
<td>44.2/97.8</td>
</tr>
<tr>
<td>rotcrop_180</td>
<td>92.5/33.2</td>
<td>90.1/77.3</td>
<td>87.1/94.7</td>
<td>81.4/98.0</td>
<td>69.9/98.6</td>
<td>58.1/98.4</td>
<td>45.2/95.8</td>
</tr>
<tr>
<td>rotcrop_270</td>
<td>93.0/29.4</td>
<td>89.9/73.9</td>
<td>86.6/93.7</td>
<td>80.4/97.8</td>
<td>69.0/98.6</td>
<td>56.8/99.5</td>
<td>42.8/97.8</td>
</tr>
<tr>
<td>rotscale_90</td>
<td>95.5/32.7</td>
<td>93.3/78.1</td>
<td>89.4/95.4</td>
<td>82.0/98.9</td>
<td>70.3/99.9</td>
<td>51.8/99.0</td>
<td>29.5/97.0</td>
</tr>
<tr>
<td>rotscale_180</td>
<td>95.1/33.2</td>
<td>92.6/78.3</td>
<td>88.3/95.3</td>
<td>81.0/98.8</td>
<td>68.8/99.9</td>
<td>48.7/98.0</td>
<td>26.8/95.0</td>
</tr>
<tr>
<td>rotscale_270</td>
<td>95.1/31.6</td>
<td>92.6/76.7</td>
<td>87.9/95.4</td>
<td>81.1/99.0</td>
<td>68.3/99.9</td>
<td>49.6/98.0</td>
<td>28.0/97.0</td>
</tr>
<tr>
<td>shear_x5</td>
<td>96.0/10.1</td>
<td>94.6/38.3</td>
<td>92.7/77.0</td>
<td>88.7/94.6</td>
<td>78.4/97.4</td>
<td>60.3/95.5</td>
<td>34.6/80.5</td>
</tr>
<tr>
<td>shear_x15</td>
<td>85.4/5.7</td>
<td>73.2/16.6</td>
<td>52.6/28.3</td>
<td>30.1/33.0</td>
<td>11.5/34.6</td>
<td>2.7/28.7</td>
<td>0.4/12.6</td>
</tr>
<tr>
<td>shear_x5y5</td>
<td>94.9/10.4</td>
<td>92.8/34.6</td>
<td>89.6/68.4</td>
<td>83.3/88.6</td>
<td>71.5/93.3</td>
<td>55.6/91.9</td>
<td>39.3/80.0</td>
</tr>
<tr>
<td>shear_x15y15</td>
<td>79.5/5.8</td>
<td>69.6/13.7</td>
<td>63.3/22.8</td>
<td>57.8/32.0</td>
<td>49.3/40.0</td>
<td>35.1/42.3</td>
<td>14.0/39.2</td>
</tr>
</tbody>
</table>
Table C.2: Estimated coverage (%) and average precision (%) in pairs for minimum inclusion threshold ranging from $T=4$ to $T=255$ on image collection 40K for the list of image alterations on column one.

<table>
<thead>
<tr>
<th>Alt</th>
<th>$T_4$</th>
<th>$T_8$</th>
<th>$T_{16}$</th>
<th>$T_{32}$</th>
<th>$T_{64}$</th>
<th>$T_{128}$</th>
<th>$T_{255}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>coloriseB</td>
<td>98.1/9.7</td>
<td>97.3/37.8</td>
<td>96.4/76.6</td>
<td>95.0/94.1</td>
<td>92.5/98.1</td>
<td>88.6/98.9</td>
<td>81.0/99.7</td>
</tr>
<tr>
<td>coloriseG</td>
<td>97.9/9.6</td>
<td>97.1/36.3</td>
<td>96.2/76.0</td>
<td>94.8/93.9</td>
<td>92.2/98.0</td>
<td>87.9/99.0</td>
<td>79.8/99.8</td>
</tr>
<tr>
<td>coloriseR</td>
<td>97.9/9.4</td>
<td>97.2/36.6</td>
<td>96.4/76.0</td>
<td>94.9/94.4</td>
<td>92.4/98.0</td>
<td>88.1/98.9</td>
<td>80.3/99.8</td>
</tr>
<tr>
<td>contr_muns</td>
<td>98.0/9.4</td>
<td>97.0/38.0</td>
<td>95.7/76.5</td>
<td>93.8/93.4</td>
<td>90.1/97.9</td>
<td>85.1/98.7</td>
<td>75.2/98.7</td>
</tr>
<tr>
<td>contr_pls</td>
<td>97.7/7.9</td>
<td>96.8/34.1</td>
<td>95.4/74.6</td>
<td>93.2/94.2</td>
<td>89.3/98.2</td>
<td>82.9/99.2</td>
<td>69.2/97.8</td>
</tr>
<tr>
<td>crop_5</td>
<td>97.4/13.1</td>
<td>96.3/47.1</td>
<td>94.5/80.8</td>
<td>91.7/94.3</td>
<td>88.2/96.2</td>
<td>80.6/95.9</td>
<td>68.2/95.8</td>
</tr>
<tr>
<td>crop_10</td>
<td>96.7/12.9</td>
<td>94.9/46.5</td>
<td>93.0/80.7</td>
<td>90.3/94.2</td>
<td>85.3/96.2</td>
<td>75.8/94.7</td>
<td>60.7/94.7</td>
</tr>
<tr>
<td>crop_20</td>
<td>96.3/12.9</td>
<td>94.8/47.7</td>
<td>93.0/80.7</td>
<td>90.5/94.8</td>
<td>85.6/97.1</td>
<td>76.8/94.6</td>
<td>63.7/94.4</td>
</tr>
<tr>
<td>crop_30</td>
<td>95.3/13.7</td>
<td>93.4/49.9</td>
<td>91.5/84.3</td>
<td>87.3/95.5</td>
<td>80.8/95.1</td>
<td>71.3/94.9</td>
<td>53.3/90.6</td>
</tr>
<tr>
<td>despeckle</td>
<td>97.3/22.2</td>
<td>96.7/62.9</td>
<td>95.5/89.2</td>
<td>93.9/96.9</td>
<td>90.3/98.9</td>
<td>84.4/99.4</td>
<td>73.7/98.0</td>
</tr>
<tr>
<td>jpeg_to_gif</td>
<td>97.9/8.5</td>
<td>97.2/35.3</td>
<td>96.3/76.1</td>
<td>94.9/94.5</td>
<td>92.3/98.2</td>
<td>88.1/98.9</td>
<td>79.7/98.7</td>
</tr>
<tr>
<td>border_0</td>
<td>96.7/20.0</td>
<td>95.6/61.1</td>
<td>93.4/89.8</td>
<td>89.6/97.9</td>
<td>82.8/99.2</td>
<td>70.7/99.7</td>
<td>54.4/100.0</td>
</tr>
<tr>
<td>border_1</td>
<td>95.8/17.5</td>
<td>94.4/57.3</td>
<td>92.4/87.8</td>
<td>88.3/97.2</td>
<td>80.0/99.1</td>
<td>67.8/99.4</td>
<td>48.4/99.9</td>
</tr>
<tr>
<td>border_2</td>
<td>95.7/15.2</td>
<td>94.3/49.9</td>
<td>92.7/83.1</td>
<td>88.3/95.8</td>
<td>79.8/98.6</td>
<td>67.4/99.5</td>
<td>48.0/100.0</td>
</tr>
<tr>
<td>border_3</td>
<td>96.3/20.8</td>
<td>94.8/62.7</td>
<td>92.6/90.1</td>
<td>88.6/97.8</td>
<td>80.9/99.3</td>
<td>68.6/99.7</td>
<td>50.2/100.0</td>
</tr>
<tr>
<td>rotate_90</td>
<td>97.7/8.3</td>
<td>96.7/35.1</td>
<td>95.0/75.9</td>
<td>93.0/94.1</td>
<td>90.2/98.0</td>
<td>85.2/99.3</td>
<td>74.4/98.8</td>
</tr>
<tr>
<td>rotate_180</td>
<td>97.6/8.6</td>
<td>96.6/36.4</td>
<td>94.9/77.3</td>
<td>92.6/94.5</td>
<td>90.2/98.1</td>
<td>85.5/99.3</td>
<td>75.5/98.7</td>
</tr>
<tr>
<td>rotate_270</td>
<td>97.6/8.3</td>
<td>96.5/34.4</td>
<td>95.0/75.3</td>
<td>92.6/94.4</td>
<td>89.4/98.1</td>
<td>84.5/98.3</td>
<td>73.8/98.8</td>
</tr>
<tr>
<td>resize_50</td>
<td>96.6/21.3</td>
<td>95.5/63.0</td>
<td>93.3/91.8</td>
<td>89.6/97.7</td>
<td>82.1/99.5</td>
<td>70.7/99.0</td>
<td>57.6/96.0</td>
</tr>
<tr>
<td>resize_25</td>
<td>89.7/34.7</td>
<td>85.0/81.5</td>
<td>78.3/94.4</td>
<td>69.5/98.0</td>
<td>56.4/98.0</td>
<td>41.8/92.0</td>
<td>25.9/76.0</td>
</tr>
<tr>
<td>resize_12</td>
<td>65.4/83.2</td>
<td>56.8/96.1</td>
<td>46.4/94.2</td>
<td>33.2/88.0</td>
<td>16.1/62.0</td>
<td>1.8/9.0</td>
<td>0.0/0.0</td>
</tr>
<tr>
<td>resize_200</td>
<td>97.7/12.3</td>
<td>97.0/46.6</td>
<td>96.2/83.1</td>
<td>94.9/95.3</td>
<td>91.9/98.5</td>
<td>87.8/99.1</td>
<td>79.0/99.9</td>
</tr>
<tr>
<td>resize_400</td>
<td>97.6/12.5</td>
<td>97.0/48.0</td>
<td>96.1/83.7</td>
<td>94.5/95.4</td>
<td>92.1/98.6</td>
<td>88.7/99.2</td>
<td>79.0/99.9</td>
</tr>
<tr>
<td>resize_800</td>
<td>97.5/12.9</td>
<td>97.0/48.0</td>
<td>96.1/83.4</td>
<td>94.8/95.5</td>
<td>91.9/98.5</td>
<td>87.6/99.2</td>
<td>79.0/99.9</td>
</tr>
<tr>
<td>satr_70</td>
<td>98.0/10.3</td>
<td>97.1/39.9</td>
<td>95.9/79.9</td>
<td>94.5/95.4</td>
<td>91.5/98.3</td>
<td>86.8/99.0</td>
<td>77.7/99.7</td>
</tr>
<tr>
<td>satr_80</td>
<td>98.0/10.5</td>
<td>97.2/40.8</td>
<td>96.2/80.2</td>
<td>94.6/95.1</td>
<td>91.9/98.2</td>
<td>87.7/99.0</td>
<td>79.4/99.8</td>
</tr>
<tr>
<td>satr_90</td>
<td>98.1/10.5</td>
<td>97.3/41.9</td>
<td>96.2/80.3</td>
<td>94.8/95.3</td>
<td>92.0/98.3</td>
<td>88.2/98.9</td>
<td>80.2/99.8</td>
</tr>
<tr>
<td>satr_110</td>
<td>98.1/10.8</td>
<td>97.1/42.2</td>
<td>96.3/81.2</td>
<td>94.8/95.3</td>
<td>92.1/98.3</td>
<td>87.9/99.0</td>
<td>80.0/99.8</td>
</tr>
</tbody>
</table>

continued on next page
### Appendix C. Effectiveness of the HPC on Image Alterations

<table>
<thead>
<tr>
<th>Alt</th>
<th>$T_4$</th>
<th>$T_8$</th>
<th>$T_{16}$</th>
<th>$T_{32}$</th>
<th>$T_{64}$</th>
<th>$T_{128}$</th>
<th>$T_{256}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>satr_120</td>
<td>97.8/10.7</td>
<td>97.1/14.1</td>
<td>96.2/81.5</td>
<td>94.3/95.5</td>
<td>91.6/98.2</td>
<td>87.4/99.0</td>
<td>78.3/99.7</td>
</tr>
<tr>
<td>int_80</td>
<td>97.8/11.4</td>
<td>97.0/43.2</td>
<td>95.9/81.6</td>
<td>94.3/95.6</td>
<td>91.2/98.5</td>
<td>85.7/99.5</td>
<td>74.7/98.8</td>
</tr>
<tr>
<td>int_90</td>
<td>97.8/10.3</td>
<td>97.1/40.4</td>
<td>96.2/79.7</td>
<td>94.7/94.9</td>
<td>91.6/98.2</td>
<td>87.2/99.0</td>
<td>77.5/98.8</td>
</tr>
<tr>
<td>int_110</td>
<td>98.0/10.9</td>
<td>97.3/41.8</td>
<td>96.3/80.4</td>
<td>94.5/95.0</td>
<td>92.0/98.0</td>
<td>88.2/98.8</td>
<td>79.6/99.7</td>
</tr>
<tr>
<td>int_120</td>
<td>97.9/9.3</td>
<td>97.1/38.0</td>
<td>96.2/77.5</td>
<td>94.4/94.4</td>
<td>91.4/97.9</td>
<td>87.5/98.8</td>
<td>77.2/99.6</td>
</tr>
<tr>
<td>crops_40</td>
<td>95.2/17.8</td>
<td>93.3/58.4</td>
<td>90.1/88.4</td>
<td>85.5/96.6</td>
<td>77.4/96.2</td>
<td>68.5/92.7</td>
<td>52.1/85.7</td>
</tr>
<tr>
<td>crops_50</td>
<td>94.2/21.1</td>
<td>91.9/63.9</td>
<td>88.7/91.1</td>
<td>83.4/96.6</td>
<td>75.2/97.1</td>
<td>63.5/96.0</td>
<td>52.6/93.7</td>
</tr>
<tr>
<td>crops_90</td>
<td>13.2/52.5</td>
<td>3.7/45.7</td>
<td>0.7/18.5</td>
<td>0.0/1.0</td>
<td>0.0/0.0</td>
<td>0.0/0.0</td>
<td>0.0/0.0</td>
</tr>
<tr>
<td>ints_50</td>
<td>96.7/26.3</td>
<td>95.0/64.3</td>
<td>92.9/90.5</td>
<td>89.1/96.5</td>
<td>82.7/98.2</td>
<td>71.2/95.9</td>
<td>53.6/94.0</td>
</tr>
<tr>
<td>ints_150</td>
<td>97.4/6.4</td>
<td>96.5/29.1</td>
<td>94.8/68.8</td>
<td>91.9/92.5</td>
<td>87.1/97.9</td>
<td>77.2/99.3</td>
<td>60.6/98.7</td>
</tr>
<tr>
<td>contrs_mns</td>
<td>95.5/10.3</td>
<td>93.1/37.3</td>
<td>89.5/77.0</td>
<td>84.8/92.5</td>
<td>75.3/97.2</td>
<td>60.8/96.3</td>
<td>40.5/93.7</td>
</tr>
<tr>
<td>contrs_pls</td>
<td>93.7/8.7</td>
<td>89.7/35.5</td>
<td>82.9/77.9</td>
<td>72.1/93.9</td>
<td>53.7/97.3</td>
<td>33.4/90.0</td>
<td>12.7/82.0</td>
</tr>
<tr>
<td>rotcrop_90</td>
<td>92.9/26.8</td>
<td>90.6/72.2</td>
<td>87.3/93.0</td>
<td>81.6/97.6</td>
<td>70.2/98.4</td>
<td>57.7/99.9</td>
<td>44.7/97.8</td>
</tr>
<tr>
<td>rotcrop_180</td>
<td>92.6/29.8</td>
<td>90.1/74.8</td>
<td>87.3/94.1</td>
<td>81.6/97.7</td>
<td>70.4/98.5</td>
<td>58.4/98.3</td>
<td>45.6/95.8</td>
</tr>
<tr>
<td>rotcrop_270</td>
<td>93.1/26.1</td>
<td>90.1/71.3</td>
<td>86.8/93.0</td>
<td>80.8/97.5</td>
<td>69.4/98.5</td>
<td>57.0/99.4</td>
<td>43.0/97.8</td>
</tr>
<tr>
<td>rotscale_90</td>
<td>95.5/29.7</td>
<td>93.4/76.3</td>
<td>89.5/94.8</td>
<td>82.4/98.7</td>
<td>70.8/99.9</td>
<td>52.7/99.0</td>
<td>30.8/97.0</td>
</tr>
<tr>
<td>rotscale_180</td>
<td>95.2/30.4</td>
<td>92.7/76.4</td>
<td>88.7/94.9</td>
<td>81.2/98.7</td>
<td>69.2/99.9</td>
<td>49.7/98.0</td>
<td>27.8/95.0</td>
</tr>
<tr>
<td>rotscale_270</td>
<td>95.1/28.9</td>
<td>92.7/74.7</td>
<td>88.3/95.0</td>
<td>81.5/98.8</td>
<td>68.9/99.8</td>
<td>50.2/98.0</td>
<td>29.6/97.0</td>
</tr>
<tr>
<td>shear_x5</td>
<td>96.1/8.6</td>
<td>94.7/35.8</td>
<td>92.8/75.1</td>
<td>88.9/93.9</td>
<td>78.7/97.1</td>
<td>60.7/96.5</td>
<td>35.1/80.5</td>
</tr>
<tr>
<td>shear_x15</td>
<td>85.6/4.9</td>
<td>73.7/15.3</td>
<td>53.2/27.3</td>
<td>30.6/32.6</td>
<td>12.0/34.3</td>
<td>2.8/27.9</td>
<td>0.5/13.4</td>
</tr>
<tr>
<td>shear_x5y5</td>
<td>95.0/9.0</td>
<td>93.0/32.6</td>
<td>89.7/66.9</td>
<td>83.4/87.9</td>
<td>72.0/93.4</td>
<td>56.0/90.8</td>
<td>39.8/79.9</td>
</tr>
<tr>
<td>shear_x15y15</td>
<td>79.9/4.8</td>
<td>69.6/12.4</td>
<td>63.4/21.8</td>
<td>57.9/31.2</td>
<td>49.5/39.6</td>
<td>35.5/41.7</td>
<td>14.6/38.5</td>
</tr>
</tbody>
</table>
Table C.3: Estimated coverage (%), average precision (%) and number of identified edges of
the near-duplicate relationship graph generated from 150000 images (collection 150K) using
the HPC algorithm with a threshold of 100 keypoints; seven threshold values ranging from
T=4 to T=255 are tested on a range of LSH indexes (from 5 to 20) and k parameters that
range from 250 to 450.

<table>
<thead>
<tr>
<th></th>
<th>T_4</th>
<th>T_8</th>
<th>T_16</th>
<th>T_32</th>
<th>T_64</th>
<th>T_128</th>
<th>T_255</th>
</tr>
</thead>
<tbody>
<tr>
<td>l=5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k=250</td>
<td>Coverage 80.6</td>
<td>70.7</td>
<td>55.2</td>
<td>35.9</td>
<td>21.0</td>
<td>9.0</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Average Precision 86.0</td>
<td>93.1</td>
<td>92.7</td>
<td>86.9</td>
<td>74.6</td>
<td>54.3</td>
<td>22.6</td>
</tr>
<tr>
<td></td>
<td>Identified Edges 453404</td>
<td>208297</td>
<td>144866</td>
<td>91455</td>
<td>53213</td>
<td>23230</td>
<td>4469</td>
</tr>
<tr>
<td></td>
<td>Run Time (mins) 3.3</td>
<td>3.0</td>
<td>2.9</td>
<td>2.9</td>
<td>2.8</td>
<td>2.8</td>
<td>2.8</td>
</tr>
<tr>
<td>k=350</td>
<td>Coverage 75.8</td>
<td>64.7</td>
<td>48.1</td>
<td>30.2</td>
<td>18.7</td>
<td>8.0</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Average Precision 92.0</td>
<td>93.0</td>
<td>90.5</td>
<td>82.8</td>
<td>70.4</td>
<td>50.6</td>
<td>22.2</td>
</tr>
<tr>
<td></td>
<td>Identified Edges 312858</td>
<td>188121</td>
<td>127886</td>
<td>78123</td>
<td>47853</td>
<td>20880</td>
<td>4613</td>
</tr>
<tr>
<td></td>
<td>Run time (mins) 1.7</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>k=450</td>
<td>Coverage 70.3</td>
<td>57.5</td>
<td>40.3</td>
<td>25.4</td>
<td>15.8</td>
<td>6.1</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Average Precision 92.5</td>
<td>92.2</td>
<td>87.7</td>
<td>78.8</td>
<td>65.7</td>
<td>44.4</td>
<td>17.0</td>
</tr>
<tr>
<td></td>
<td>Identified Edges 250892</td>
<td>164277</td>
<td>107401</td>
<td>66240</td>
<td>40780</td>
<td>16354</td>
<td>3489</td>
</tr>
<tr>
<td></td>
<td>Run Time (mins) 0.9</td>
<td>0.9</td>
<td>0.9</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l=10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k=250</td>
<td>Coverage 86.4</td>
<td>80.5</td>
<td>70.7</td>
<td>55.3</td>
<td>36.0</td>
<td>21.1</td>
<td>9.3</td>
</tr>
<tr>
<td></td>
<td>Average Precision 70.7</td>
<td>91.5</td>
<td>93.5</td>
<td>92.8</td>
<td>87.0</td>
<td>74.6</td>
<td>54.7</td>
</tr>
<tr>
<td></td>
<td>Identified Edges 1298352</td>
<td>289296</td>
<td>198488</td>
<td>143741</td>
<td>91577</td>
<td>53568</td>
<td>23812</td>
</tr>
<tr>
<td></td>
<td>Run Time (mins) 8.6</td>
<td>6.0</td>
<td>5.9</td>
<td>5.8</td>
<td>5.8</td>
<td>5.7</td>
<td>5.7</td>
</tr>
<tr>
<td>k=350</td>
<td>Coverage 82.8</td>
<td>76.2</td>
<td>65.4</td>
<td>49.0</td>
<td>30.9</td>
<td>19.1</td>
<td>8.4</td>
</tr>
<tr>
<td></td>
<td>Average Precision 89.0</td>
<td>92.8</td>
<td>93.2</td>
<td>90.9</td>
<td>83.1</td>
<td>70.7</td>
<td>51.5</td>
</tr>
<tr>
<td></td>
<td>Identified Edges 568199</td>
<td>259631</td>
<td>184261</td>
<td>129208</td>
<td>79714</td>
<td>48831</td>
<td>21791</td>
</tr>
<tr>
<td></td>
<td>Run time (mins) 3.8</td>
<td>3.5</td>
<td>3.4</td>
<td>3.3</td>
<td>3.3</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>k=450</td>
<td>Coverage 77.4</td>
<td>68.7</td>
<td>55.5</td>
<td>38.1</td>
<td>24.4</td>
<td>15.1</td>
<td>5.8</td>
</tr>
<tr>
<td></td>
<td>Average Precision 91.6</td>
<td>93.0</td>
<td>91.9</td>
<td>86.4</td>
<td>77.3</td>
<td>64.3</td>
<td>43.0</td>
</tr>
<tr>
<td></td>
<td>Identified Edges 351268</td>
<td>217571</td>
<td>154526</td>
<td>101283</td>
<td>63444</td>
<td>39057</td>
<td>15490</td>
</tr>
<tr>
<td></td>
<td>Run Time (mins) 1.8</td>
<td>1.8</td>
<td>1.7</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
</tr>
</tbody>
</table>

continued on next page
### APPENDIX C. EFFECTIVENESS OF THE HPC ON IMAGE ALTERATIONS

<table>
<thead>
<tr>
<th>l=15</th>
<th>(T_4)</th>
<th>(T_8)</th>
<th>(T_{16})</th>
<th>(T_{32})</th>
<th>(T_{64})</th>
<th>(T_{128})</th>
<th>(T_{255})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>88.4</td>
<td>84.0</td>
<td>76.6</td>
<td>64.7</td>
<td>46.8</td>
<td>28.8</td>
<td>16.1</td>
</tr>
<tr>
<td>k=250 Average Precision</td>
<td>55.8</td>
<td>89.0</td>
<td>93.2</td>
<td>93.5</td>
<td>91.1</td>
<td>82.1</td>
<td>67.0</td>
</tr>
<tr>
<td>Identified Edges</td>
<td>3025543</td>
<td>367146</td>
<td>230808</td>
<td>173138</td>
<td>120013</td>
<td>72978</td>
<td>40824</td>
</tr>
<tr>
<td>Run Time (mins)</td>
<td>16.9</td>
<td>9.0</td>
<td>8.8</td>
<td>8.7</td>
<td>8.7</td>
<td>8.6</td>
<td>8.6</td>
</tr>
<tr>
<td>Coverage</td>
<td>85.3</td>
<td>80.3</td>
<td>72.1</td>
<td>59.0</td>
<td>40.6</td>
<td>25.0</td>
<td>14.6</td>
</tr>
<tr>
<td>Identified Edges</td>
<td>929205</td>
<td>320700</td>
<td>216505</td>
<td>159577</td>
<td>105754</td>
<td>61185</td>
<td>37565</td>
</tr>
<tr>
<td>Run Time (mins)</td>
<td>6.2</td>
<td>5.3</td>
<td>5.2</td>
<td>5.1</td>
<td>5.1</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Coverage</td>
<td>81.6</td>
<td>75.2</td>
<td>65.0</td>
<td>49.8</td>
<td>32.4</td>
<td>21.1</td>
<td>11.4</td>
</tr>
<tr>
<td>Identified Edges</td>
<td>1572083</td>
<td>387980</td>
<td>244202</td>
<td>181821</td>
<td>128484</td>
<td>79244</td>
<td>48822</td>
</tr>
<tr>
<td>Run Time (mins)</td>
<td>10.0</td>
<td>7.4</td>
<td>7.3</td>
<td>7.2</td>
<td>7.2</td>
<td>7.2</td>
<td>7.2</td>
</tr>
<tr>
<td>Coverage</td>
<td>83.4</td>
<td>78.2</td>
<td>69.7</td>
<td>56.8</td>
<td>39.3</td>
<td>25.0</td>
<td>15.7</td>
</tr>
<tr>
<td>Identified Edges</td>
<td>628288</td>
<td>310015</td>
<td>214526</td>
<td>157078</td>
<td>104070</td>
<td>65065</td>
<td>40585</td>
</tr>
<tr>
<td>Run Time (mins)</td>
<td>4.0</td>
<td>3.8</td>
<td>3.7</td>
<td>3.7</td>
<td>3.6</td>
<td>3.6</td>
<td>3.6</td>
</tr>
</tbody>
</table>

### l=20

| Coverage | 88.9 | 86.1 | 80.1 | 70.2 | 54.6 | 35.5 | 20.9 |
| k=250 Average Precision | 49.1 | 85.6 | 92.6 | 93.7 | 92.7 | 86.4 | 74.0 |
| Identified Edges | 7927789 | 470178 | 261796 | 193847 | 141770 | 90233 | 53061 |
| Run Time (mins) | 28.3 | 12.1 | 11.8 | 11.8 | 11.7 | 11.6 | 11.5 |
| Coverage | 86.9 | 82.8 | 76.1 | 65.4 | 48.9 | 30.8 | 19.1 |
| Identified Edges | 1572083 | 387980 | 244202 | 181821 | 128484 | 79244 | 48822 |
| Run Time (mins) | 10.0 | 7.4 | 7.3 | 7.2 | 7.2 | 7.2 | 7.2 |
| Coverage | 83.4 | 78.2 | 69.7 | 56.8 | 39.3 | 25.0 | 15.7 |
| Identified Edges | 628288 | 310015 | 214526 | 157078 | 104070 | 65065 | 40585 |
| Run Time (mins) | 4.0 | 3.8 | 3.7 | 3.7 | 3.6 | 3.6 | 3.6 |
Appendix D

Examples of algorithm-formed clusters

Figure D.1 shows more examples of near-duplicate clusters generated using the ND-CENTER algorithm.
APPENDIX D. EXAMPLES OF ALGORITHM-FORMED CLUSTERS

Figure D.1: Examples of selected clusters that are identified by the ND-CENTER algorithm — using the same settings as those reported in our experiments — on images retrieved from the Web (Google retrieved images). More examples of algorithm-identified clusters can be found in http://sico.cs.rmit.edu.au/ClusterEval_WebCol/.
Appendix E

Testbed specifications

The experiments reported in this work were implemented using a combination of C/C++ code — compiled with the GNU C++ compiler — and bash and awk wrapper scripts. The timing experiments were carried out on a workstation with the following specifications:

Processor(s): 2×Intel(R) Xeon(TM) CPU 3.00GHz
RAM: 4 GB (DDR)
Hard disk drive: 4× (300 GB Barracuda, SATA, 7200RPM, 8 MB cache)
Operating system: GNU/Linux (kernel: 2.6.9-42.0.2.ELsmp)
Bibliography


