Content-based multimedia retrieval: indexing and diversification

Cover design by Arno Kamphuis and Reinier H. van Leuken

Typeset in LaTeX
Printed in The Netherlands by Ponsen en Looijen (http://www.p-l.nl/)

Content-based multimedia retrieval: indexing and diversification

Inhoudsgebaseerd zoeken van multimedia: indexatie en diversificatie

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad
van doctor aan de Universiteit Utrecht
op gezag van de rector magnificus, prof.dr. J.C. Stoof,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen
op maandag 11 mei 2009 des middags te 2.30 uur

door

Reinier Henricus van Leuken

geboren op 18 september 1981
te Leidschendam

Promotor: prof. dr. R.C. Veltkamp

This work was partially financially supported by the IST Programme (6th framework) of the European Union project under contract No IST 05/450 4401 (PROFI - Perceptually Relevant Retrieval of Figurative Images)

Making the simple complicated is commonplace; making the complicated simple - awesomely simple - that's creativity.

– Charles Mingus (1922-1979) –

Contents

Chapter 1

Introduction

THE first issue of a 'magazine' called Aspen was published in 1965 by Roaring Fork Press, New York, and was named "The black box". It officially contained nine items, the first of them being a black hinged box, printed with a large letter A, that contained the other items. Among those are a 20-page booklet with three perspectives on jazz. First, Dixieland woodwind player Freddie Fisher criticises modern jazz for a lack of direction, musical sense and overuse of imagery like wearing sunglasses and growing beards. In response, pianist Jon Hendricks stands up for the modernists claiming that "they're not ahead, we're behind", just as "it took us 300 years to get Bach into our brain". Finally, as there is apparently so much to what the term "jazz" really comprises, Chuck Israels, bass player in Bill Evans' trio, takes it all apart in a scholarly essay.

What matters to this thesis is that one of the other items included in this first issue of Aspen, was a record with on one side a Dixieland version of St James Infirmary Blues, and on the other side a take off the tune Israel by the Bill Evans trio, featuring the same Chuck Israels. It said: "With whom do you agree? Play our record and see. One side is Dixieland; The other is wayoutland." Phyllis Johnson, Aspen's publisher, describes the magazine as "the first three-dimensional magazine", because each issue came in a special box filled with items in varying formats such as booklets, records, posters, postcards and super-8 footage. Each issue had a new designer and director, and with contributions from some of the most interesting artists of the 20th century, its ten issues now take an important place in art history.

Although Johnson described the magazine as three-dimensional, when the third issue came out, the term multimedia emerged with it as well. This issue was designed by Andy Warhol (see the figure at this chapter's title page) and contained a spin-off of his *Exploding Plastic Inevitable* series of multimedia events, which were musical performances by The Velvet Underground with simultaneous dance performances, screenings of Warhol's movies, light shows and interactive elements such as celebrities from Warhol's social environment running around with microphones asking confronting questions to the audience.

Of course, neither the Exploding Plastic Inevitable project nor Aspen magazine are the first occurrences of synthesis of multiple kinds of media. In 1928, Walt Disney created his famous 'Steamboat Willie', the first cartoon with a fully synchronised soundtrack. The idea of a soundtrack wasn't new at that time either: silent film often came with organ or piano scores to be played during the screening, and even the ancient Greeks composed music to accompany specific drama and theatre scenes. However, the *term* multimedia is commonly believed to stem from this pop-art period in the sixties, defining Andy Warhol's projects and his Aspen issue. Literally, it refers to multiple kinds of media that are to be perceived simultaneously. Since

media is already plural, it would be pleonastic to let multi refer to multiple instances of the same media, like multiple books or multiple photographs [1].

With the computer maturing in the second half of the twentieth century and with what can be called a digital revolution, more and more multimedia applications were developed. The first commercial computer game, Pong, was launched in 1972. Nowadays, computer games deploy a very sophisticated integration of animation, video, audio, text, still images and interactive components. In commercial and academic settings, slide show presentations that may contain any kind of media are given on a daily basis. In 1973 the first ideas for the Internet were presented, and the World Wide Web saw the light in 1990. The Web as we know it today easily contains the same plurality of media as mentioned before in the context of computer games.

Devices that are capable of keeping up with all these applications are being developed equally rapidly. Bell invented the telephone in 1876 and in 1979 Sony produced the first portable cassette player; a modern smartphone currently plays music, takes pictures, sends emails, displays webpages and contains an agenda. Apart from that, it can be used for making phone calls. The VHS video format, introduced by JVC in 1976 was followed by the DVD in 1996 and ten years later by the Blu-ray disc in 2006. These discs are usually not played in a monolithic, dedicated player anymore, but in versatile multimedia computers that can be used as gaming console, television or Internet PC at the same time. Just as multimedia is the synthesis of various forms of media, the current trend is to merge a large number of applications into a single device.

Data storage costs have dropped enormously over the years, and content creation, sharing and access is available cheaply to anyone nowadays. For example, with the introduction of the digital camera, there is practically no restriction to the number of pictures one can take. Moreover, with ever increasing bandwith, a huge amount of these pictures is uploaded to the Internet on a daily basis. How do we keep this manageable, how can we ever find anything we're looking for? This demand for search systems is not only raised from a personal, individual point of view. It is raised within a large number of professional application domains where the data heap continues to expand as well. Some examples include criminology (face and fingerprint recognition), musicology (music information retrieval), trademark registration (automatic trademark retrieval), medicine (DNA fingerprinting), engineering (3D model retrieval) and image or video retrieval on the web.

[1] In that respect, the title of this thesis is not entirely correct, because our experiments focus on indexing collections that contain only a single kind of media: a collection of photographs for instance, or a collection of 3D models. However, in chapter 3 we limit the melody length based on the lyrics, and in chapter 5 we base our search both on textual annotations of the photographs and their visual characteristics.

Luckily, it is bred in our bones to organise things, as we have learnt over the years that it is the only way of making progress without losing what we already possess. People have categorised and ordered texts in libraries for centuries, but the organisation of the multimedia objects is not so straightforward, let alone the automatic organisation. Yet it is key to unlocking media collections for both dedicated professionals and the general public.

In this thesis, we concentrate on how to retrieve relevant information or items from large collections, given an information need, a query, expressed by a user. One possible solution is to describe the objects in the collection textually, for instance to annotate each object with a set of keywords. Well known textual information retrieval techniques can be used to address a user query. Unfortunately, this approach has many limitations. First of all, it requires from the user to express his information need textually, and does not support query by example ("Give me some objects that look like this one"). Second, it is a very cumbersome and in some cases simply impossible task to annotate large datasets, that often require expert knowledge. Finally, the representational power of natural language is usually not strong enough: a photograph has to be seen, a melody has to be heard to understand its meaning, beauty or significance. The retrieval techniques discussed in this thesis are therefore *content-based*.

Content-based multimedia retrieval is a discipline within several fields of computer science that emerged in the early nineties of the twentieth century. In its entirety it is somewhere in between pattern recognition, artificial intelligence, database technology, signal processing and computer vision. It is concerned with capturing the characteristics of the objects in the collection (photographs, video footage, music, 3D models etc.) in a digital form, such that they can be evaluated for similarity: the system must compute to what extent two objects resemble each other and it must base this computation on the content of the objects and not on associated meta data like file names or descriptions. This process of similarity evaluation must coincide with what humans perceive as similarity within the specific application domains. Therein lies a serious challenge, that is often referred to as the semantic gap. Basing on experience and knowledge of our surrounding world and using the possibility to associate, humans are able to understand meaning that is perhaps not immediately apparent. For instance, we instantly recognise a house in a drawing of a house even in its simplest form. The semantic gap is defined in the context of image retrieval in [107] as "the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation". But there is more to our grasp of the things we see or hear. We understand that the olympic logo is constructed by overlaying five circles rather than connecting

individual curves. We distinguish the main melody from its variations in a trumpet concerto. All these abilities are extremely hard to teach a computer, but essential to a successful, generic retrieval system.

There is yet another challenge. Suppose we have succeeded in devising a system that evaluates similarity between objects adequately. Many applications treat enormous datasets, and comparing the query to every object in the collection would be infeasible. In these cases, an indexing strategy can be deployed to effectively prune the database: to make a preselection of candidate models with which to compare the query. Basically, an indexing system is a system to find things more rapidly. Some PhD-theses for instance contain an index at the very end, with all the important terms and their corresponding pages. The object collection is in that case a set of words, and their indexing signature consists of a series of page numbers. When an interested reader wants to learn about a certain concept, he or she only has to concentrate on the pages that compose the indexing signature and the rest of the thesis can be discarded. To make it even easier, the index itself is indexed too: the terms are sorted in alphabetical order.

We largely focus on the indexing part of the multimedia retrieval problem in this thesis. In chapter 2, we study a generic indexing approach that supports any dataset or collection, as long as there is an appropriate similarity or distance measure defined on the objects. In chapters 3 and 4, we specifically investigate how to index objects when they can be represented by graphs. Finally in chapter 5, we broaden our scope a little and present techniques to find results that are not only relevant to the query, but diverse as well. The diversification of search results is also performed content-based.

To motivate this work, we present in the following sections a more detailed description of the application domains that we study in this thesis. Furthermore, we define the indexing concept in more detail and discuss desirable properties. Several measures are used throughout this thesis to assess the quality of the proposed and existing techniques; these measures are also given in detail. Finally, we list the contributions of this thesis.

1.1 Application domains

As multimedia retrieval lies on the crossing of so many research disciplines, research papers can be found in many different journals and conferences. Luckily the field is establishing its own platforms as well. For instance, the annual ACM workshop on

multimedia information retrieval (ACM MIR) was upgraded to a conference on its tenth anniversary in 2008. In that year, the event had to be relocated last minute to a conference room with double capacity because of the overwhelming number of participants. The program contained blocks of presentations regarding various application domains, such as image retrieval, video retrieval and concept detection, audio retrieval, automatic summarization and multimedia browsing. We will study several of these applications in this thesis, and in the following subsections we present a short overview of them.

1.1.1 Image retrieval

Content-based image retrieval is perhaps the most researched and most famous application of multimedia retrieval. As papers on the subject from before 1990 are rare and of little impact today, it can be said that the work really started in 1992, when the US National Science Foundation sponsored a workshop on Visual Information Management Systems (VIMS) [65]. The goal of this workshop was "to identify major research areas that should be addressed by researchers for VIMS that would be useful in scientific, industrial, medical, environmental, educational, entertainment, and other applications." The workshop's major findings were that new techniques were required in all aspects of databases, computer vision and knowledge representation and management, and that these techniques were to be developed in interdisciplinary research groups in the context of concrete, practical applications. According to the workshop findings, computer vision researchers should identify features required for interactive image understanding, that must be stored in both symbolic and non-symbolic representations, and reasoning approaches that can deal with such representations should be developed.

Whether a direct result of the workshop or not, publication rates on content-based image retrieval increased quickly, and really took off after 1997. In 1998, an event called Challenge of Image Retrieval was organised in Newcastle, UK, to "bridge the gap between the different communities with an interest in image retrieval". It had successors in 1999 and 2000. The video domain was soon taken into consideration and the event was renamed Challenge of Image and Video retrieval. It 2003 it became the Conference on Image and Video retrieval, which is nowadays an important ACM conference where image and video retrieval practitioners exchange ideas on all related topics.

In 2002, many content-based image retrieval systems were surveyed by Veltkamp and Tanase in [129]. These systems rarely used shape, but favoured colour features instead. The authors conjecture that shape is found a too difficult feature to work

with at the time. The early years of the field are further extensively surveyed in 2000 by Smeulders et al. [107]. Following [106], they categorise the applications into two classes: those with a narrow scope and those with a broad scope. "A narrow domain has a limited and predictable variability in all relevant aspects of its appearance", they state, whereas "a broad domain has unlimited and unpredictable variability in its appearance even for he same semantic meaning". A set of photographs of cars, taken sideways from the same angle and with the same lighting in the same studio against a white background is an example of a narrow domain; even though the cars are dissimilar to some extent, there is strong expectancy about position, geometry and shape. The domain is therefore constrained. On the other hand, when the set consists of random car pictures taken outside, or provided by marketing departments, or even taken from video footage or other external sources, the domain is much broader. This generally poses extra challenges such as occlusion, noise, changes in illumination and difficult semantics. The broadest set of images possible is the entire collection of images on the Internet. Most content-based image retrieval applications are in between these two ends of the spectrum. The semantic gap is generally larger in broad domains than in narrow domains. Additionally, the *sensory gap* is usually wider in broad domains as well. The sensory gap is defined as "the gap between the object in the world and the information in a (computational) description derived from a recording of that scene" [107]. For instance, two different physical (3D) objects may have the same 2D view, resulting in a sensory gap between recording and object when recognition is based on the 2D views.

In the early years of content-based image retrieval, the emphasis has been on using low-level features such as colour, texture, edge information or orientation. Since 2000, the field has matured and the continuing interest is demonstrated by an extensive and recent survey by Datta et al. [34]. The status on narrowing the semantic gap is provided by Liu et al. in [71], where methods are classified according to the way they handle high-level semantics. Some examples are the use of object ontologies, the use of machine learning to associate high-level concepts with low-level features or the use of relevance feedback to learn the user's intention.

The World Wide Web is becoming an increasingly important application domain of content-based image retrieval. A specific example are social media sites where users have a network of friends and can upload their photos, such as Flickr or Picasa. These photos are often tagged with keywords and additional descriptions, which leads to an interesting fusion between textual and content-based search. Moreover, the social character of these sites provides new interesting features, such as distance in the friends network [142], geographic location, popularity or temporal data [136].

We perform several experiments with image data in this thesis. In chapter 2, both

colour histograms of photographs and curvature characteristics of silhouette images form the input for a generic indexing scheme. We use several low level image features of photographs in chapter 5 for content-based diversification of search results.

1.1.1.1 Trademark retrieval

An example of a rather narrow image retrieval domain is trademark retrieval [96]. Trademark infringement is one of the major issues in the intellectual property field. Strong or famous trademarks have to be monitored constantly to avoid the situation where a *confusingly similar* trademark is registered in the same market. Some examples of trademark infringement are given in figure 1.1. In these cases, the owner of the infringed trademark can start legal action (opposition) against the owners of the infringing trademarks. The monitoring of strong trademarks is sometimes called a *watch* operation. On the other hand, when a trademark is designed for a product or service, and the owner wants to avoid a costly legal dispute, he can order a *search* operation to check for registered trademarks that are possibly confusingly similar.

The aforementioned operations can be performed by large trademark research companies such as Thomson Compu-Mark (TCM), that have access to national trademark registers. These registers are expanding with hundreds of trademarks each day, for instance in 2006, there were 69,706 trademarks added to only the French register. A special requirement of search and watch operations is the zero tolerance for missed hits; e.g. Puma will not accept that one of the imitated trademarks of figure 1.1 is not retrieved. The main paradigm for performing trademark retrieval has been using textual coding schemes, that associate numerical codes to each trademark. The most commonly used codification is the Vienna classification developed by the World Intellectual Property Organisation. The codes reflect directly what the trademark image is composed of. For instance, when a trademark has the code 3.5.9, it contains hedgehogs or porcupines, such as the OrangeHedgehog trademark displayed in figure 1.2 (a). The Vienna classification also supports abstract shapes; 26.3.11 belongs for instance to "Triangles containing one or more quadrilaterals".

When a trademark search has to be performed, the Vienna codes are determined for the query, and all trademarks with one or more similar Vienna codes are retrieved from the trademark register. In the current trademark registration practice, all retrieved trademarks have to be inspected by a trademark expert for similarity: only a very few of them are probably close enough to the query to pose a threat. Unfortunately, this process of manual labelling and inspection is time consuming and error prone. It is therefore worthwhile to deploy an automatic trademark retrieval system for watch and search operations. Some of the first efforts in automatic trademark

Figure 1.1: Examples of trademark infringement

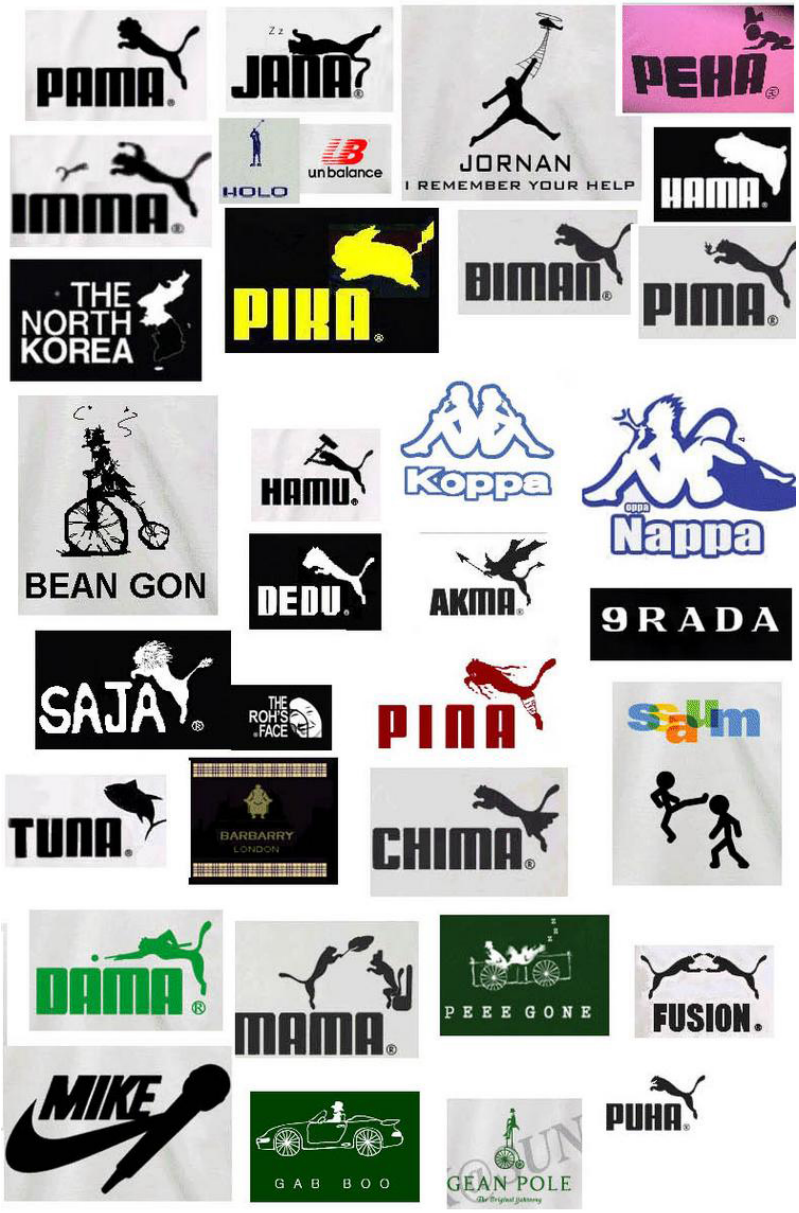


Figure 1.2: (a) OrangeHedgehog trademark (b) Gestalt psychology in trademarks

(a) (b)

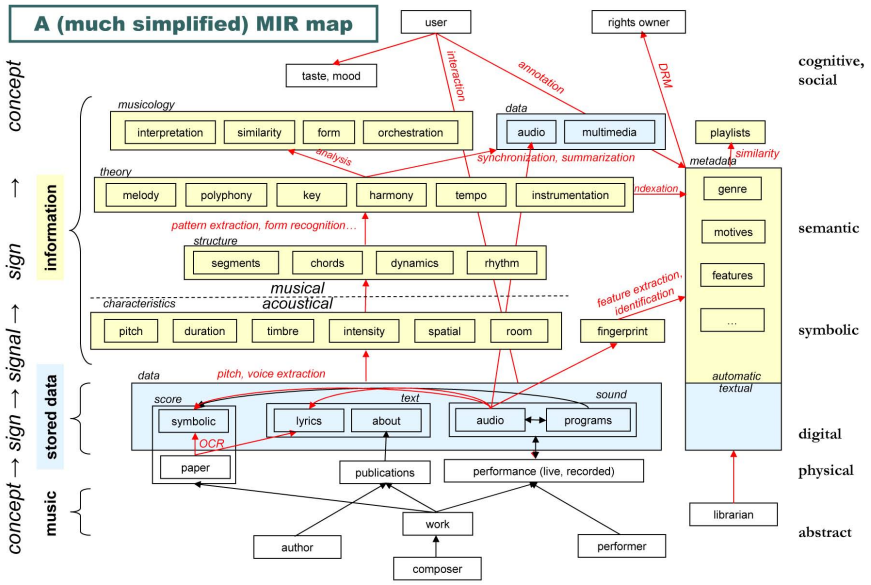
retrieval are due to Eakins et al. [41, 66] and Austin and Alwis [2, 3, 4].

Many difficulties of content-based image retrieval apply to automatic trademark retrieval as well. The semantic gap poses a serious challenge; it is easy for us to recognise the hedgehog in figure 1.2 (a), but hard for a computer. Moreover, even when automatic trademark retrieval is restricted to the class of geometric or abstract shapes, where semantics play no significant role, similarity may be implicitly present. See for instance figure 1.2 (b), where there are three Pacman-like shapes explicitly present. By their organisation however, a triangle is implicitly outlined. The way our brain is capable of such form-forming is studied in Gestalt psychology, where the key question is how we see in the whole something that is different from the sum of parts. In the case of trademark registration, the implicit triangle might be crucial to the recognizability of the trademark and should be taken into account by an automatic trademark retrieval system. An effective retrieval system should in fact generate all possible *views* of a trademark.

These challenges were studied in the Profi project [2], funded by the European Commission FP6 research program. The problem of applying Gestalt principles and other perception theory during segmentation and matching of trademarks proved to be particularly difficult indeed, and provides interesting research directions in both image processing and shape matching. We study in chapter 3 of this thesis a subproblem of trademark retrieval: trademark layout. The motivation for this work is that in some specific cases, the way shapes or components of a trademark image are laid out is more discriminative and representative for the trademark than the actual shapes this trademark is composed of.

[2] Perceptually-relevant Retrieval of Figurative Images, http://profi.cs.uu.nl/

Figure 1.3: A map of the music information retrieval field, due to Michael Fingerhut [45]



1.1.2 Music information retrieval

Like content-based image retrieval, music information retrieval emerged in the late nineties. Its major event, the International Conference on Music Information Retrieval ISMIR [3] was organised for the first time in 2000, its 11th edition will be co-hosted by Utrecht University in Utrecht in 2010. A schematic overview of the field is given in figure 1.3, a figure due to Michael Fingerhut [45]. Music is in essence an art form with sound organised in time as medium. With this medium, a composer or performer is able to convey his message to an audience, a fascinating and sometimes magical process. A musical message however is intricate and of complex nature, and contains many layers of information. Figure 1.3 unravels this process to some extent and shows several layers where musical information exists, when going from music creator to audience, or 'user'. The arrows between these layers indicate the actions required to extract or retrieve this information at intermediate stages: they map directly to the research directions within this field.

[3] The S in the acronym stands for Symposium, which is how the event started, and was kept in the acronym for sentimental reasons. On July 4th 2008 however, the International Society for Music Information Retrieval was incorporated, and the S became relevant again.

The first actors in this scheme (at the bottom in figure 1.3) are the creators of music: the composer, the performer, the author. Their output is stored in physical or digital form, such as in score, text or recordings: the above layer. The transition from paper to digital symbolic score can be made with optical character recognition. Pitch and voice extraction can bridge the gap between recordings and symbolic storage such as text and score. Plain data are not yet information; characteristics and features need to be extracted from the data to enter the next information layer. The simple characteristics such as pitch, duration and timbre need be further studied and aggregated into structural features to be meaningful. Not the notes themselves, but their co-occurrence, sequence and interplay form the music in the end. The structural features in turn reveal aspects that define the music from a music theory point of view: melody, harmony, key etc. Finally, by analysing and interpreting these aspects, we enter the level of musicology, before the message is considered to be delivered to the audience.

Musicology is actually concerned with all the above stages. Music information retrieval can be seen, among many things, as a toolbox for musicologists. It provides automatic methods for retrieving musical information at all the stages described above and supports musicologists in their research. The word retrieval here refers to revealing the message that is contained in the music. In this thesis however, the word retrieval generally refers to retrieving objects or documents from a large collection. Querying for music by measuring musical similarity is also a part of music information retrieval, as can be seen from the side-track on the right in figure 1.3. All the information that is revealed can be used to describe a piece of music, to form a fingerprint for it, and these representations can be used in querying or retrieval systems.

Among the many challenges music information retrieval poses, measuring musical similarity is a significant one. There are many possible ways in which two pieces of music may display similarity: melodic, harmonic, rhythmic, but also contextual, conceptual, historic etc. Even when constrained to melodic similarity, which is what we focus on in this thesis, many distance measures are possible. One of the main difficulties in devising a distance measure for melodic similarity is that the pure note-to-note similarity (as measured for instance with the edit distance on two melodic sequences) may not reflect the *perceived* similarity. The simplest example is a transposition of a melody into another key: all the notes are different, but the intervals remain the same so the two melodies are perceived highly similar. Melodic similarity should therefore take into account the framework of harmony, rhythm, polyphony and instrumentation, provided by music theory. However necessary, this contextual information also introduces many uncertainties and ambiguities. Despite its some-

times strict and formal grammar, the true art in music lies in the creative process of applying these rules to endless new combinations, that more often than not break the rules as if they never existed.

In this thesis we will perform two sets of retrieval experiments (i.e. where retrieval refers to finding relevant objects or documents in a collection given a query). In chapter 2 we use a distance measure for melodic similarity as proposed by Typke et al. [119]. This distance measure, that reflects the effort it takes to transform one melody into the other, is used to demonstrate the effectiveness of new algorithms regarding a generic indexing scheme. In chapter 3, we describe a new graph based representation of music that we use for retrieval with the in that chapter proposed framework.

1.1.3 3D object retrieval

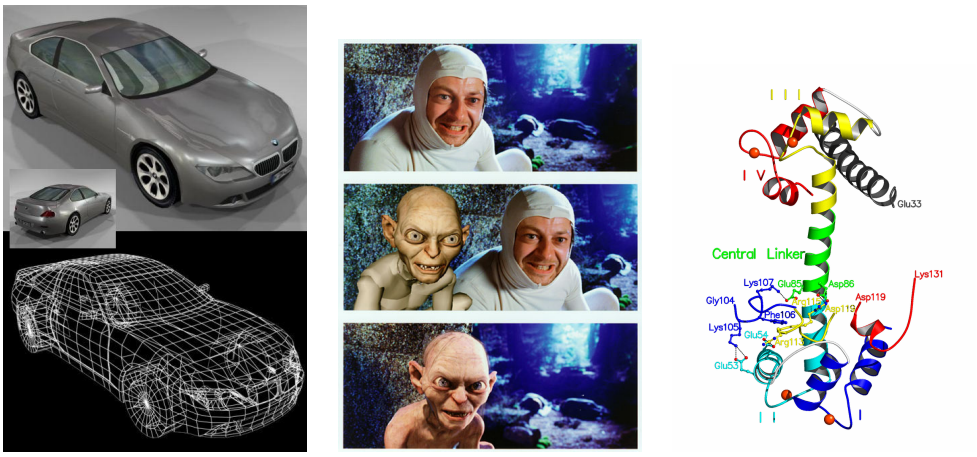
In modern computer graphics, 3D object representations play a central role. They are frequently used in computer aided design, the video game industry (see figure 1.4 (a)), the movie industry (figure 1.4 (b)), molecular biology (figure 1.4 (c)), medicine and archives of cultural heritage. The models are obtained through intricate modelling processes that may include scanning, alignment of these scans, reconstruction based on 2D images, surface sampling and many other techniques. The advancement in hardware (scanners, but also graphics processors and dedicated 3D hardware) has made the creation, acquisition and manipulation of models much easier and the body of available models continues to grow. It is therefore worthwhile to investigate content-based retrieval techniques for 3D models. Within computer vision and pattern recognition, there has been a growing interest in 3D shape description and invariant feature extraction for matching, retrieval and recognition. See [113] for an extensive and recent survey on 3D shape retrieval methods.

In this thesis we perform experiments with 3D models in the graph domain; the problem of 3D object retrieval is transformed into the problem of graph retrieval. These experiments are described in chapters 3 and 4.

1.2 Index properties

An index makes a preselection of the dataset for a given user query, and does so without inspecting the entire collection. For an (implementation of the) index or indexing strategy to be successful in making this preselection, it should possess a number of properties. First of all, as with many algorithms or software systems,

Figure 1.4: Examples of 3D models or object representations



(a) (b) (c)

it should preferably be *simple* where possible: simple to implement and simple to maintain. Furthermore, it must be *dynamic*. It should be able to adjust to changes in the collection at any time; objects have to be removed or added without affecting the retrieval quality. Consequently, the index has to be *scalable* in the sense that it should perform equally well for a small collection as for a large collection. Another design issue is *efficiency*: both in time and in space. Processing a query should not take too long, as speeding up the query process is the whole purpose of using the index. Generally more time is allowed to construct the index, sometimes referred to as the offline process. Efficiency in terms of storage space is important as well: the indexing signatures should be relatively compact.

The index implementation has to be *concurrent*: it must be possible to retrieve and store multiple objects at the same time. Furthermore, there must be a seamless transition between memory types, if for instance both main and secondary memory are used. One other important property is that the index has to be independent of the input data and insertion sequence: the order of inserting the objects should not influence the index quality.

Finally, the index should support a number of essential operations. It must allow for object insertion and deletion. The most important operation it should support is the search operation. A possible categorisation of search operations defines three types:

search by association, category search and aimed search. A *search by association* is similar to how the World Wide Web is often used and is based on iterative refinement of the search by inspecting search results and choosing a path to continue that seems most interesting or relevant. It requires a highly interactive system that can process different user commands and allows browsing through the collection. *Category search* on the other hand aims to find one or more representatives of a certain class of objects from the collection. The query formulation is often by example: the user provides one or more objects and wants to find more objects from the same class. Finally, with *aimed search* the user has a specific object in mind that should be retrieved from the collection. In many cases the goal of this search is beyond checking whether the object is present in the collection or not: the user aims to find objects that display similarity with the query. We focus largely on this search operation, and distinguish between two kinds: range query, where the number of retrieved objects may vary, and k-nearest neighbour query, where the number of answers to the query is fixed. For a collection of objects A with distance function d and a query object A_q, a range query with range ϵ is defined as

$$\text{retrieve } A_i \in A | d(A_q, A_i) \leq \epsilon.$$

A k-nearest neighbour query on the other hand is defined as

$$\text{retrieve } A' \text{ where } |A'| = k \text{ and } \forall A_i \in A, \forall A_j \in A - A' | d(A_q, A_i) \leq d(A_q, A_j).$$

1.3 Performance assessment

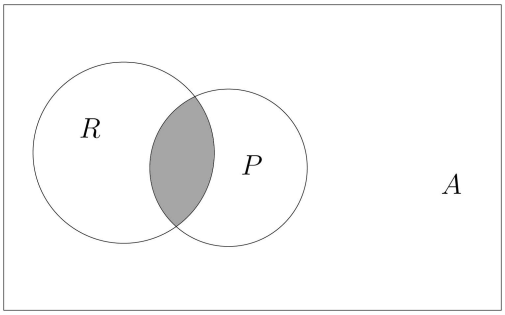
It is not trivial to assess the retrieval quality of an index by itself. Retrieval results are formed by a difficult interplay between the descriptiveness of the representation, the quality of the matching algorithm, the parameters of the query and the effectiveness of the indexing strategy. It is therefore common to assess the quality of this pipeline as a whole. Varying the indexing strategy and keeping all the other parts fixed provides insight in the indexing quality, but only relatively. All experiments in this thesis must therefore be considered in this light: the reported retrieval results are not the optimal results that can be obtained for the various data sets and applications. The focus is always on improving the indexing strategy with respect to existing indexing strategies within the boundaries of chosen algorithms for the rest of the pipeline.

To evaluate the quality of a retrieval result, it is necessary to know what the correct answer must be for that query. This is called a ground truth and has to be established by hand, a time consuming procedure that requires specific domain knowledge. The

ground truth must therefore be considered a subjective matter. Nevertheless it is essential in distinguishing between correct retrieved objects and incorrect ones, and to detect missing objects. Let A be the entire collection of objects and let P be the set of retrieved images (the positives) for a certain query, possibly ranked in a list in decreasing order of similarity to the query. The number of objects in P is determined by the scope; this can be a specific number of objects to retrieve, or a distance threshold. Finally, let R be the set of relevant objects in the collection for that query according to the ground truth. See figure 1.5 for an illustration of these sets of objects. In this scenario we can now distinguish between four kinds of objects:

- True Positives (TP): $P \cap R$
- False Positives (FP): $P - R$
- False Negatives (FN): $R - P$
- True Negatives (TN): $A - (P \cup R)$

Figure 1.5: Illustration of a retrieval result. A is the entire collection of objects, P is returned for a given user query: the positives. R are the relevant objects for this query in the collection. All objects within the intersection of R and P are true positives.



We will now provide some performance measures based on these four object types that are used throughout this thesis. These measures are usually evaluated for a large number of queries to obtain a good indicator of the real performance. The simplest measure is without a doubt the nearest neighbour check: is the object at the top rank a true positive or not? In some cases the query is also part of the collection, which makes the nearest neighbour measure an identity check. It is common to look for the second item in the ranking then as well.

Possibly the most famous performance measures are precision and recall, defined as follows:

$$\text{Precision} : \text{TP}/(\text{FP} + \text{TP})$$
$$\text{Recall} : \text{TP}/(\text{TP} + \text{FN}).$$

Recall is sometimes expressed with respect to tiers. Let c be the class size, or the number of relevant items for a certain query. The first tier result refers to the recall when the scope is set to $c-1$ when the query is part of the class, otherwise when the scope is set to c. The k-th tier refers to recall within the scope $kc-1$ or kc respectively. The second tier is sometimes referred to as the Bull's eye percentage. Furthermore, there exists a particular interest in the interplay between precision and recall, that is often visualised in a precision-recall graph. Two measures derived from recall and precision are Average Dynamic Recall (ADR) and Average Dynamic Precision (ADP) [120]:

$$\text{ADR} = \sum_{i=1}^{N} \frac{\text{Recall}_i}{i}$$
$$\text{ADP} = \sum_{i=1}^{N} \frac{\text{Precision}_i}{i},$$

where Recall_i and Precision_i are recall and precision values calculated when the scope is set to i. When the scope is smaller than the class size, recall of 100% can never obtained. In this however, Recall_i is still defined as 100% when all i retrieved objects are true positives. These measures take the ranking of the true positives into account as well and the highest possible score is 1. In case of ADR, the maximum score is obtained when all relevant items are retrieved and are on the top of the ranking as well. In case of ADP, it is obtained when those relevant items that are indeed retrieved, are on top of the ranking.

Sometimes some statistics on the ranking of the relevant items are provided. This is particularly common when evaluating a matching algorithm, and when the complete collection is ranked with respect to the query rather than retrieving a small preselection. The reported statistics may include average or median rank, highest rank, lowest rank etc. These rank statistics may be given directly, or transformed into a performance score. An example is Lowest Place Rank (LPR), defined as

$$LPR = 1 - \frac{\text{Rank} - c}{N - c}$$

where Rank is the rank of the lowest placed relevant item.

1.4 Contribution of the thesis

The results in this thesis are presented in four chapters. In chapter 2, we study a generic indexing scheme called vantage indexing. The performance of this scheme can be improved by selecting good vantage objects. We therefore present two new criteria to select vantage objects and show their soundness experimentally. We also present a heuristic algorithm that actually selects the vantage objects according to these criteria, and constructs the vantage index simultaneously. The performance increase is demonstrated in many application domains [124, 125].

In chapter 3, we introduce a new method for indexing graphs, based on their Laplacian spectra. Again, we show the effectiveness on a large number of domains [37, 36], and present new graph representations for trademarks [121] and melody scores [88]. By applying our framework to the music domain, we introduce spectral retrieval to the field of music information retrieval. Furthermore, we are, to our best knowledge, the first to deploy spectral integral variation in an indexing framework. We use this recently developed technique in graph theory to increase the efficiency of our method.

In chapter 4 we overcome one of the limitations of the in chapter 3 proposed approach. We extend our graph representation and store additional properties in a Hermitian matrix. We introduce the use of a specific eigenvector, the Fiedler vector, as indexing signature and present an efficient framework for this purpose [123]. We provide a graph representation in the domain of 3D object retrieval to instantiate the framework. This graph representation contains additional, domain specific properties, that are both relevant in extending the representation and obey constraints that are imposed by the spectral retrieval apparatus. A shape decomposition algorithm forms an alternative to spectral integral variation.

The goal of a retrieval system is extended in chapter 5. We no longer merely aim for relevant results, but demand that they are diverse as well. In the domain of content-based image retrieval we present a new framework to diversify the search results [122]. This method first determines dynamically which features are most discriminative for a given result set, and uses these features to present a diversified view of the results to the user. We show that the output of our method coincides consistently with the results of a user study.

Chapter 2

Selecting vantage objects for similarity indexing

Relevant publications:

R.H. van Leuken, R.C. Veltkamp
Selecting Vantage Objects for Similarity Indexing
UU-CS Technical Report, 2008-002

R.H. van Leuken, R.C. Veltkamp, R. Typke
Selecting Vantage Objects for Similarity Indexing
In: Proc. International Conference on Pattern Recognition, pp 453-456, 2006

2.1 Introduction

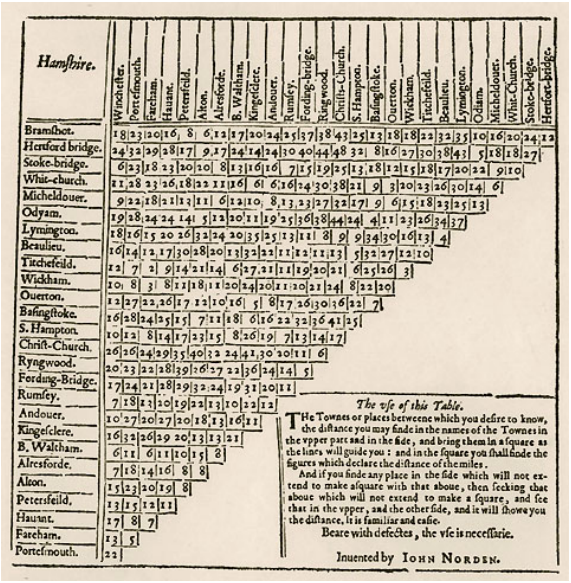
PROBABLY every major road atlas contains a triangular table with distances between major cities or landmarks. This device was invented in 1625 by the English topographer John Norden, who published the first distance table to facilitate navigation with a map of Hampshire he created 30 years before in 1595. It came with the following explanation (see also figure 2.1) [84]:

> The use of this Table.
> The Townes or places betweene which you desire to know, the distance you may finde in the names of the Townes in the upper part and in the side, and bring them in a square as the lines will guide you: and in the square you shall finde the figures which declare the distance of the miles.
> And if you finde any place in the side which will not extend to make a square with that above, then seeking that above which will not extend to make a square, and see that in the upper, and the other side, and it will showe you the distance. It is familiar and easie.
> Beare with defects, the use is necessarie.
>
> *– Invented by JOHN NORDEN. –*

Simply put, to find the distance between two cities a and b, just read the number in the table on position (a, b), or on position (b, a) should you end up outside the triangle. Norden's map of Hampshire did not have any roads yet, so the distances where measured 'as the crow flies'. As crow flights are known to obey metric properties (see for a definition section 2.2), it is possible that with this explanation, Norden provided the first algorithm for performing a search in a metric space. The algorithm was according to his own account not only 'necessarie', but 'familiar and easie' as well. However, in the light of modern algorithmic design, some criticism is in place. Its main limitation is its running time; it is basically performing a sequential search and thus its running time is linear in the number of cities. Suppose one wants to find the *closest* city to Portlemouth (see figure 2.1), he or she has to scan the entire second column to find that the answer is Fareham.

Of course, with current data structures and their corresponding point location algorithms we can perform a faster search. For instance, we can store all the cities in a

Figure 2.1: Triangular distance table of Hampshire by John Norden, 1625



Hamſhire.

Winchester. Portesmouth. Fareham. Hauant. Petersfeild. Alton. Alresforde. B. Waltham. Kingesclere. Andouer. Rumsey. Fording-bridge. Ryngwood. Christs-Church. S. Hampton. Basingstoke. Overton. Wickham. Titchefeild. Beaulieu. Lymington. Odyam. Micheldouer. Whit-Church. Stoke-bridge. Hereford-bridge.

Bramshot. 18 23 20 16 8 6 12 17 20 24 25 37 38 43 25 13 18 18 22 32 35 10 16 20 24 32
Hereford bridge. 24 32 29 28 17 9 17 24 14 24 30 40 44 48 32 8 16 27 30 38 43 5 18 28 27
Stoke-bridge. 6 23 18 23 20 20 8 13 16 16 7 15 19 25 13 28 12 15 28 17 20 22 9 10
Whit-church. 21 28 23 26 18 32 21 16 6 16 24 30 38 21 9 3 20 23 26 30 14 6
Micheldouer. 9 22 18 21 13 11 6 12 10 8 13 23 27 22 17 9 6 15 18 23 25 13
Odyam. 19 28 24 24 14 5 12 20 11 19 25 36 38 44 24 4 11 23 26 34 37
Lymington. 18 16 15 20 26 32 24 20 35 25 13 11 8 9 9 34 30 16 13 4
Beaulieu. 16 14 12 17 30 28 20 13 32 22 11 12 11 13 5 22 27 12 10
Titchefeild. 12 7 2 9 14 21 14 6 27 21 11 19 20 21 6 25 26 3
Wickham. 10 8 3 8 11 18 11 20 24 20 21 20 21 24 8 22 20
Ouerton. 22 27 22 26 17 22 10 16 5 8 17 26 30 16 22 7
Basingstoke. 16 28 24 25 15 7 11 18 6 16 22 32 36 41 25
S. Hampton. 10 12 8 14 17 23 15 8 26 19 7 13 14 17
Christ-Church. 26 26 24 29 35 40 22 24 42 30 20 11 6
Ryngwood. 20 23 22 28 39 26 27 22 36 24 14 5
Fording-Bridge. 17 24 21 28 29 32 24 19 31 20 11
Rumsey. 7 18 13 20 19 22 13 10 22 12
Andouer. 10 27 20 27 20 18 13 16 11
Kingesclere. 16 32 26 29 20 13 13 21
B. Waltham. 6 11 6 11 10 15 8
Alresforde. 7 18 14 16 8 8
Alton. 15 23 20 19 8
Petersfeild. 13 15 12 11
Hauant. 17 8 7
Fareham. 13 5
Portesmouth. 22

The vſe of this Table.

THe Townes or places betweene which you desire to know, the distance you may finde in the names of the Townes in the vpper part and in the side, and bring them in a square as the lines will guide you: and in the square you shall finde the figures which declare the distance of the miles.

And if you finde any place in the side which will not extend to make a square with that aboue, then seeking that aboue which will not extend to make a square, and see that in the vpper, and the other side, and it will showe you the distance, it is familiar and easie.

Beare with defectes, the vſe is neceſſarie.

Inuented by IOHN NORDEN.

well known kD-tree and report the closest city to Portlemouth in sublinear time [13]. Query time for retrieving all m cities within a certain range is then even bounded by $O(n^{1-1/d} + m)$, something John Norden would probably have liked. For him, the dimension d in this time bound would have been 2 (latitude and longitude), assuming he did not take elevation into account while creating the table. In the context of this chapter, we would say that he worked with a 2-dimensional *object space*. His metric distance measure 'as the crow flies', takes as input two objects that both have two features and outputs a distance between them. However, in many modern applications the dimensionality is larger, certainly in the field of multimedia retrieval. Suppose for instance that we want to create a distance table for colour photographs that are all represented by a 64-bins colour histogram and the distance between two photographs is calculated using the L_2 norm. In that case, the dimensionality of the object space is 64 and in practice searching using a kD-tree may take longer than a simple sequential search. A space partitioning approach is in these cases not successful and we need to turn our attention to different approaches, which is what we do in this chapter.

The contents of this chapter are further motivated by an even more difficult scenario. What if the dimensionality of the object space is unknown? Norden's cities have two

features in the above example, but what if the distance would be defined as the minimum number of horse and carriage trips it would take to relocate all the inhabitants and their furnishings from one city to the other and vice versa? In this case, there is no clear notion of the dimensionality of the object space. To give a multimedia retrieval example, consider a dataset containing fragments of music notation, and as distance between two of them the Proportional Transportation Distance [49], which is a pseudo-metric version of the Earth Mover's Distance [92]. All we have is the dataset with n objects and a way of calculating a distance between two objects. In a way, one could say that the features we have for each object are all its distances to the other objects; its row in the $n \times n$ distance matrix would then be the object's n-dimensional feature vector. But with a collection of millions of objects, this is a completely impractical structure to work with and again space partitioning approaches fail.

The music collection together with the proportional transportation distance is an example of a *general metric space*. Rather than partitioning this space, it can be embedded in a vector space for indexing purposes. Such an embedding is provided by the vantage indexing algorithm [132]. In turn, the resulting vector space can be partitioned by a space or data partitioning structure such as a kD-tree. In this chapter we improve the quality of search systems that use vantage indexing, by selecting good vantage objects.

2.1.1 Contributions

Firstly, we propose two criteria to assess the quality of vantage objects that are directly concerned with the retrieval performance, namely the reduction of the number of false positives in the returned sets. Secondly, we show how to select vantage objects according to these criteria in such a way that each object in the database is a candidate vantage object, no random pre-selection is made. Another attractive property of the approach is that the selection of the vantage objects and the actual construction of the index are handled at the same time. Thirdly, we have performed extensive experimentation using three data sets of different modality and size: the MPEG-7 CE-Shape-1 part B test set, consisting of 1,400 shape images, a set of the 50,000 colour photographs and a dataset containing 500,000 fragments of notated music, of 5 notes each. We have compared our method to three other methods: random selection, the loss-based selection method and the originally proposed MaxMin method, which are outperformed by the proposed approach. These experiments therefore show both the scalability and efficacy of the method [124, 125].

2.2 Vantage indexing

Vantage indexing [132] is an embedding technique, that is used to map a dissimilarity space (preferably metric) to a feature space in which querying takes place. To be more specific, it requires a dataset $A \subset \mathbb{U}$, where $\mathbb{U}$ is the universe of objects, and a distance measure $d : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$. The distance measure d is called metric if the following conditions hold:

1. *Self-identity* $\forall x \in \mathbb{U}, d(x, x) = 0$
2. *Positivity* $\forall x \neq y \in \mathbb{U}, d(x, y) > 0$
3. *Symmetry* $\forall x, y \in \mathbb{U}, d(x, y) = d(y, x)$
4. *Triangle inequality* $\forall x, y, z \in \mathbb{U}, d(x, z) \leq d(x, y) + d(y, z)$

To embed the metric space in a vector space, a set of m objects $A^* = \{A_1^*, ...A_m^*\}$ is selected, the so called vantage objects. The distance from each database object A_i to each vantage object is computed, thus creating a point $p_i = (x_1, ...x_m)$, such that $x_j = d(A_i, A_j^*)$. Each database object corresponds to a point in the m-dimensional vantage space, let $F(A_i)$ denote this mapping of an object A_i to a point in vantage space.

A query on the database now translates to a range-search or a nearest-neighbour search in this m-dimensional vantage space: compute the distance from the query object q to each vantage object (i.e. position q in the vantage space) and retrieve all objects within a certain range around q (in the case of a range query with range of size ϵ), or retrieve the k nearest neighbours to q (in case of a nearest neighbour query). The distance measure that is used to retrieve the nearest neighbours of q in the vantage space is called δ, and defined as L_∞. The complete process of retrieval through vantage objects is described in algorithm 2.1.

Using one of several known solutions to implement the querying algorithm (for instance [9], where approximate nearest neighbours are retrieved efficiently), a query time of $O(k \log n)$ can be achieved after $O(n \log n)$ preprocessing time, where n is the number of objects in the dataset and where k is the number of retrieved objects.

2.2.1 Performance assessment of a vantage index

A large variety of performance measures based on false and true positives and negatives can be used to assess the quality of a retrieval system that employs vantage in-

Algorithm 2.1 Retrieval through vantage indexing

Preprocessing:
1: **for** Each object $A_i \in A$ **do**
2: **for** Each vantage object $A_j^* \in A^*$ **do**
3: $x_j = d(A_j^*, A_i)$
4: $p_i = (x_1, \ldots, x_m)$

Querying:
5: let A_q be the query object
6: **for** Each vantage object $A_j^* \in A^*$ **do**
7: $y_j = d(A_j^*, A_q)$
8: $q = (y_1, \ldots, y_m)$
9: return $\{A_i \in A | \delta(p_i, q) < \epsilon\}$

dexing. However, a ground truth is generally required to classify candidate matches in false and true positives, and to detect whether there are still false negatives residing in the database. Establishing a ground truth for large databases is a time consuming and demanding task involving domain expertise, that can often only be performed for a small number of queries.

In the case of embedding or mapping methods, there are other ways to assess the index quality, such as distortion [70], stress [68] or the Cluster Preserving Ratio (CPR) [63] when a known clustering exists for the objects. The key idea behind these methods is the comparison between distances according to the original similarity measure (i.e., the distances in object space) and the distances in the embedding space (e.g. the vantage space). Distortion measures how much larger or smaller the distances are in the embedding space than in the object space, whereas stress measures the overall difference in distances. While these measures provide insight into how well (in terms of distance preserving properties) the original space has been embedded in a space more appropriate for querying, they are somewhat distant from the actual retrieval application. Therefore, we propose to stretch the definition of false and true positives beyond the borders of a ground truth toward the comparison of distances, in order to allow the use of performance measures designed for retrieval applications.

In the case of a range query, given $\epsilon > 0$ (the range) and query A_q, object A_i is included in the return set of A_q if and only if $\delta(F(A_q), F(A_i)) \leq \epsilon$. A false positive can now be defined as follows:

Definition 1 False positive *A_p is a false positive for query A_q if $\delta(F(A_q), F(A_p)) \leq \epsilon$ and $d(A_q, A_p) > \epsilon$.*

Note that this definition is not limited to assessing the quality of range searching, it can be applied to nearest neighbour or k-nearest neighbour searching as well. Although there is no predefined, fixed range ϵ in these cases, the distance between the query and the furthest of the nearest neighbours can be used as ϵ. This distance was exactly the required range to retrieve the requested number of nearest neighbours, and in that sense a correct yet strict threshold for determining whether the objects are true or false positives. Furthermore, it may seem awkward to use the same ϵ for both δ and d. However, recall that δ is L_∞. Since the maximum difference of all distances is taken, and not a combination, d and δ are in the same domain, and therefore the same ϵ can be used.

Along the same lines, we can define a false negative as follows:

Definition 2 False negative *A_n is a false negative for query A_q if $\delta(F(A_n), F(A_p)) > \epsilon$ and $d(A_q, A_p) \leq \epsilon$.*

However, if it is known that d is metric, 100 percent recall is guaranteed for a system using vantage indexing [132]. The metric properties assure that vantage indexing is a *contractive embedding* of the object space, i.e. $\forall A_1, A_2 \in \mathbb{U}, \delta(F(A_1), F(A_2)) \leq d(A_1, A_2)$. Contractive embeddings with respect to $\mathbb{U}$ always yield 100 percent recall in similarity searches [58]. As pointed out before however, the accuracy of a retrieval system is twofold; objects relevant to the query are to be included in the result, yet objects irrelevant to the query should be excluded from the result as much as possible. In other words, precision is important as well, the number or percentage of false positives must be kept small.

We claim that by choosing the right vantage objects, the precision can increase significantly. In the next section, we present a strategy of selecting vantage objects that is concerned with this issue of retrieval performance directly.

2.3 Related work

Given the enormous number of different indexing techniques that have been proposed, developed, tested and deployed over the last decades, it comes as no surprise that there exist many different *categorisations* of them as well. To correctly position

vantage indexing and vantage object selection strategies, we will shortly review a few of these categorisations.

Some strategies store their necessary data structures in main memory, while others operate on secondary storage and optimise IO-costs. This difference induces a memory-type categorisation, and vantage indexing falls under both categories. Another categorisation can be devised based on the *access method* that is deployed by the indexing approach. Primary key access methods retrieve the objects directly on their self-contained descriptor, which is the case for instance with hashing. When the objects are not self-explanatory and there are multiple attributes or features associated to them, a secondary key access method such as a kD-tree can organise the data. Spatial access methods, such as R-trees or space filling curves, locate objects based on a specified window or range in the data structure. The vector space embedding provided by vantage indexing can be implemented using a secondary key access method or a spatial access method.

Indexing techniques can also be divided based on the input of their construction: is it driven by data or by a hypothesis? A data driven index is built bottom up, it is a fixed data structure that enables searching in an existing data set. On the other hand, a hypothesis driven index can be seen as top down. A query operation is characterised in a certain domain, e.g. with a decision tree, and data with similar characterisations is returned. A vantage index is a data driven index.

The last categorisation we would like to make is most important when it comes to separating vantage indexing from other methods. We focus here on the difference between partitioning and mapping or embedding approaches. Note that vantage indexing is mainly an embedding approach. However, after the general metric space (or object space of very high dimensionality) has been embedded in a vector space, that vector space can be indexed again using a partitioning approach. We therefore briefly list some important partitioning approaches.

In the early years of content based multimedia retrieval, the main paradigm was based on feature extraction. Objects are characterised by vectors that are composed of numerical features, and distance between two objects is usually calculated as a Minkowski metric between their corresponding vectors. In these cases, the general type of indexing that is applied is a partitioning, whether it is a space based partitioning or a data based partitioning. Examples of these partitioning strategies, that are mostly stored in trees, are the kD-tree [13], R-tree [54] or variants such as the $R+$-tree [98] or $R*$-tree [12]. For a complete overview of these multidimensional access methods see the survey by Gaede and Günther [47] and Samet's extensive textbook [94]. In general, these methods either partition the data space into disjoint

cells of possibly varying size (kD-tree and related work), or associate a region with each object in the data space (R-tree family).

Basically, this type of retrieval system is an instance of a more generic paradigm, where a dataset of object models in whatever representation (feature vectors in the above mentioned case) are matched using a model matching algorithm. In many cases, objects are represented by other types of models than feature vectors, that don't allow a space or data partitioning so easily. Examples are weighted point sets (possibly matched with the Earth Mover's Distance [92]), polygonal curves (for instance matched with turning angle functions [6] or the Fréchet distance [1]) or any other type of higher level representation. All that is given in these cases is a dataset containing the object models and a similarity measure that outputs a distance given two models, that together span the object space. Tree-based space partitioning techniques can still be used for indexing purposes, but they have to be built on different grounds, since there is no feature space anymore. When the object space is metric, exemplar or pivot objects, as vantage objects are also sometimes called, can be stored in the tree nodes to guide the search. One of the first works in this field was by Yianilos [143]. All database objects are divided into concentric rings around one or multiple pivots and then stored in a tree. These example objects are often called vantage objects or vantage points. Other examples based on this strategy are the VP-tree [21], the M-Tree [29] and the MVP-Tree [22]. These and other techniques for searching in metric spaces are surveyed by Chavez et al. [42] and extensively described in the book by Samet [94].

A powerful alternative for storing an object space (possibly metric) in a tree, is the mapping or embedding approach. Here, the database objects are embedded again in a feature space, even though they reside originally in a featureless space. Instead of dividing the database objects in concentric rings around pivots and storing them in a tree, the pivots now function as feature descriptors; each database object is characterised by distances it has to a set of pivots. Examples of these embedding techniques are Vantage indexing [132], Fastmap [43], SparseMap [63] and MetricMap [135]. These methods are surveyed by Hjaltason and Samet [58]. A big advantage of these methods over tree-based indexing methods is that the required number of online distance calculations is reduced to the dimensionality of the embedding. Once the query has been positioned in the embedding space, all that is needed is a geometric range query or a nearest neighbour query where no more complex distance calculations are involved. To facilitate these searches, access methods as surveyed in [47, 94] can be used again.

In this chapter, we will focus on the last type of indexing strategy, the mapping approach. Although these methods may resemble dimensionality reductions such

as Principle Component Analysis (PCA) or Multi Dimensional Scaling (MDS), they have different starting points [68]. PCA and MDS reduce the dimensionality of a feature space, and actually make computations on this feature space. However, mapping approaches such as the ones mentioned before, don't assume such a feature space. The only possible feature space in this context is an n-dimensional space, where n is the number of objects in the dataset. In practice, this would mean that the distances to all objects in the database are used as feature descriptors, which would make any computation infeasible. The retrieval performance of these embedding techniques is influenced by the choice of the pivots, sometimes called reference objects, exemplars or vantage objects. We present an effective strategy to select good vantage objects to improve the retrieval performance.

Pękalska et al. have investigated a related problem [86]. Their aim is to select a proper set of prototype objects given a set of objects represented by dissimilarities as well, however the set of prototypes is used for classifying new objects into predefined classes rather than retrieving similar objects from the data set.

A strategy similar to the one proposed in this chapter, was proposed by Venkateswaran et al. [131]. Not every database element is a candidate vantage object (or *reference*, as they call them), in contrast to our method, where each element is a candidate. Furthermore, their selection criteria are based on variance of distance (relevance) and distance between vantage objects (redundancy), whereas our selection criteria are based on variance of spacing (relevance) and correlation between vantage objects (correlation). Finally, their criteria are evaluated over only a sample of the database.

BoostMap, as proposed by Athitsos et al. [7], is a fundamentally different approach. Using a popular machine learning technique, the AdaBoost framework, they combine many 1-dimensional classifiers into a high-dimensional classifier. In this case, a 1-dimensional classifier corresponds to a vantage object that for a particular triplet (q, p_1, p_2) decides whether p_1 is closer to q than p_2 or not. The goal is to provide a combined high-dimensional classifier that outputs a similarity ranking for q that reflects the true ordering of the database with respect to q. The nature of the algorithms (machine learning with intensive training rounds) and the fact that the embedding is designed to produce a ranking that is order preserving rather than distance preserving, make it fundamentally different from the proposed method.

The following methods have been implemented and compared to the proposed method, see for experimental results section 2.5.

Bustos et al. [25] propose to maximize the mean of distances d_μ between all objects in

the embedded space or vantage space, in order to disperse the objects evenly. They provide three algorithms implementing this criterion. As best performing algorithm, they report a greedy approach, that iteratively selects as next vantage object the one that produces the largest d_μ given the vantage objects that were already selected. We refer to this method as Maximum Mean Distance, (MMD). A large drawback of MMD is the underlying assumption that the distribution of distances is uniform. When this distribution is not uniform (e.g. when there are strong clusters), maximizing d_μ does not necessarily produce an even spread of the objects in vantage space. Moreover, only a small set of objects is candidate to be selected as vantage object, and the selection criterion is evaluated only over a sample of distances.

Brisaboa et al. [23] assume the distribution of distances to be uniform as well. They propose the following heuristic, called Sparse Spatial Selection (SSS): when a certain database object has a large enough distance to all the currently selected vantage objects, it is added to the set of vantage objects. A large advantage of this approach is that it doesn't require a predefined vantage space dimensionality. In their experiments, they show that the selected number of vantage objects reflects the *intrinsic dimensionality* of the dataset. However, a drawback of this method is that it is only concerned with the combined performance of vantage objects and not with their individual quality. Moreover, database objects that are inspected first have a larger chance of becoming a vantage object. Finally, their selection criterion is based on the assumption that the distribution of distances is uniform.

Henning and Latecki propose a loss-based strategy for selecting vantage objects [56]. The loss of a database object is defined as the real (object-space) distance between this object and its nearest neighbour in vantage space. To compute the loss of a complete vantage space, this distance is averaged over all database objects. The loss measure is minimized in greedily during the selection of vantage objects, by choosing a new vantage object such that the loss combined with other vantage objects is minimal. Due to the computationally expensive nature of the algorithm, the loss measure is evaluated over random subsamples of the database.

Originally, a MaxMin approach was proposed for the selection of vantage objects [132]. The first vantage object is chosen at random, all further vantage objects are chosen such that the minimum distance to the other vantage objects is maximized. As we will see however, this property does not necessarily guarantee the best choice of vantage objects.

2.4 Selecting vantage objects

In this section, we present a novel technique for selecting vantage objects that is based on two criteria that address the number of false positives (see Definition 1) in the retrieval results directly. The first criterion, *spacing*, concerns the relevance of a single vantage object. The second criterion, *correlation*, concerns the redundancy of a vantage object with respect to the other vantage objects. We propose a randomized incremental construction algorithm that selects the vantage objects according to these criteria, and builds the corresponding vantage space at the same time. We call this method Spacing-correlation based selection.

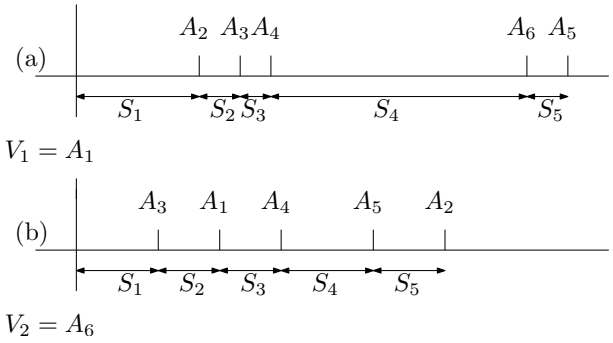
The main idea of the proposed approach is to keep the number of candidates that are returned for a query A_q and range ϵ as small as possible. Of course, a priori the query object A_q is unknown, so its location in vantage space is unknown as well. Furthermore, no prior knowledge is available on the size of the range query (ϵ), or the number of nearest neighbours that will be requested. Good retrieval performance (achieved by small return sets given a query A_q and range ϵ) should therefore be obtained over all possible queries and over all possible ranges ϵ.

Small return sets are a result of dispersing the objects over the vantage space, in order to avoid dense object clusters. This dispersion can only be achieved to a certain extent, since real object clusters cannot be taken apart, assuming d is metric (see section 2.2.1). Given the 100 percent recall guarantee, it is exactly the number of false positives within a range around the query that is reduced by spreading out the database over the vantage space as much as possible, since these are pushed outside the borders of the range ϵ.

Another way of looking at this dispersion of the objects over the vantage space is through the discriminative power of a set of vantage objects. In a vantage space, similarity between database objects is interpreted as similarity in distance to the vantage objects. In case many database objects have similar distances to the vantage objects, the vantage space is limited in its discriminative power over the database. The discriminative power of the vantage space is maximized by spreading out the objects as much as possible (i.e., within the boundaries as posed by the specific dataset that is to be embedded).

The following sections describe the two criteria (spacing and correlation) in more detail, provide some insight into finding a proper vantage space dimensionality, and present the selection algorithm and its computational complexity.

Figure 2.2: Schematic representation of a vantage axis with object clusters (a) and a vantage axis with dispersed objects (b).



2.4.1 Spacing

In this section we will define a criterion for the relevance of a single vantage object V_j. Suppose for one given vantage object, the distances to all items are marked on a vantage axis. The discriminative power can then be measured by calculating how evenly spaced the marks on this axis are. Our first criterion therefore concerns the *spacing* between objects on a single vantage axis, which is defined as follows:

Definition 3 *The spacing S_i between two consecutive objects A_a and A_b on the vantage axis of V_j is $|d(A_a, V_j) - d(A_b, V_j)|$.*

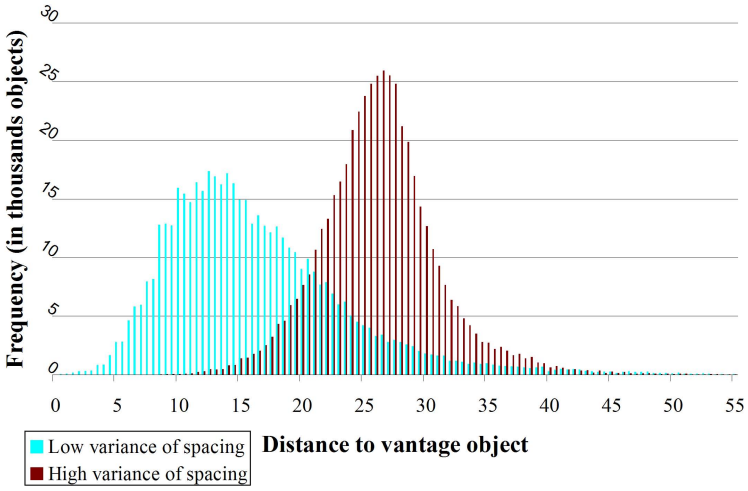
Let μ be the average spacing. The variance of spacing σ^2_{sp} is

$$\sigma^2_{\text{sp}} = \frac{1}{n-1} \sum_{i=1}^{n-1} ((|d(A_a, V_j) - d(A_b, V_j)|) - \mu)^2.$$

To ensure that the database objects are evenly spread in vantage space, the variance of spacing has to be as small as possible. A vantage object with a small variance of spacing has a high discriminative power over the database, and is said to be a relevant vantage object.

The spacing criterion is illustrated by figure 2.2, where axes of two vantage objects are displayed schematically. Figure 2.2 (a) displays a vantage axis with clustered objects, resulting in a large variance of spacing compared to figure 2.2 (b), where

Figure 2.3: Distance distributions for vantage objects with a high and low variance of spacing



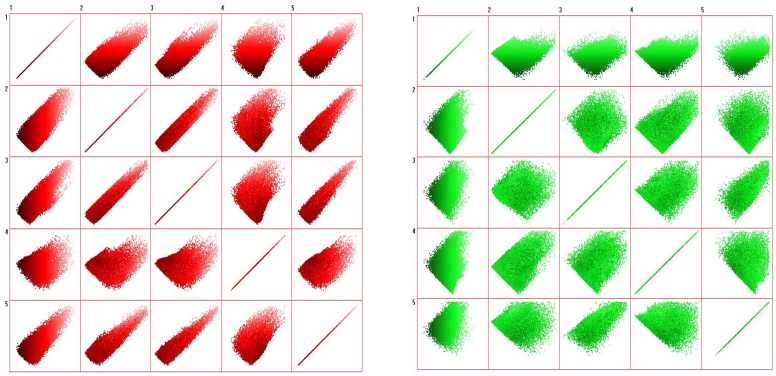
a vantage axis with dispersed objects is displayed. The spacing criterion is further illustrated by a real world example in figure 2.3, where distance distributions for two vantage objects are visualised in a histogram, one with a low and one with a high variance of spacing. It can be seen that the database objects have a wider variety of distances to the vantage object with a low variance of spacing than to the vantage object with a high variance of spacing.

In these histograms, the distances are binned; in practice, most of the distances are unique. A large bin (e.g. around 27.5 histogram of high variance) therefore means that there are a lot of distances within a certain range around this value. As a consequence, the spacings of the distances within this bin must be small. Distances in a smaller bin are 'less packed', and thus have larger spacings in between. If the bin heights vary a lot, there exist a lot of different spacing values, resulting in a higher variance of spacing values.

2.4.2 Correlation

It is not sufficient to just select relevant vantage objects, they also should be non-redundant. A low variance of spacing for all vantage objects does not guarantee that the database is well spread out in vantage space, since all the vantage objects may

Figure 2.4: Scatterplot matrices for 5 randomly selected vantage objects. Vantage objects were selected randomly in (a), and using the proposed method in (b)



(a) (b)

provide a similar view on the database. Two redundant vantage objects produce the same reduction of the return set, and there is no point in using one in combination with the other. This redundancy of a vantage object with respect to another can be estimated by computing the linear correlation coefficient between the distribution of database objects along their axes. Note that two vantage objects that have a large distance to each other can still be redundant, since the distribution of distances all objects have to these vantage objects may be similar.

To ensure no redundant vantage objects are selected, we compute linear correlation coefficients for all pairs of vantage objects and make sure these coefficients do not exceed a certain threshold. Figure 2.4 illustrates the correlation criterion in a real world example. These scatterplots show how the distances of all objects in a database of 500,000 elements to two vantage objects are correlated. The maximum correlation results in a simple diagonal line, as can be seen on the diagonal of the matrices, where each vantage object is compared to itself. The scatterplot matrix on the left displays pairwise correlations for five vantage objects that where selected randomly, whereas the five vantage objects in the scatterplot on the right were selected by the proposed method, i.e. such that the correlation coefficient for every pair of vantage objects is low. Clearly, random selection of vantage objects results in stronger correlated vantage objects than the proposed method, since the objects in the right matrix are dispersed over the space much better.

2.4.3 The number of vantage objects

The dimensionality of the vantage space (defined by the number of vantage objects) and the retrieval performance are closely related. In general, the more vantage objects, the smaller the number of false positives will be. A vantage object cannot degrade precision scores, at worst it won't influence the precision at all and thus be completely redundant. However, query times in a vantage space of higher dimensionality are longer. Therefore, the number of vantage objects should be set to an appropriate value given the needs of the application. In an interactive environment, the allowed dimensionality is limited, whereas in offline applications where precision is crucial, more vantage objects can be used. In our experimental work we evaluate the influence of the vantage space dimensionality on the retrieval results.

Although performance increases with a larger number of vantage objects, this increase is not necessarily gradual or unlimited; adding one vantage object to a very small set doesn't make a great difference if the set is still too small to obtain good results. On the other hand, at some point it will be hard to find more vantage objects that are non redundant and performance increase with extra vantage objects will slow down. The best strategy for finding an appropriate number of vantage objects given a specific dataset and considering the needs of the application therefore, is to perform some pilot selection runs to estimate the optimal vantage space dimensionality under these constraints.

2.4.4 Algorithm

Spacing-correlation based selection selects a set of vantage objects according to the criteria defined above with a randomised incremental algorithm. The key idea is to add the database objects one by one to the index while inspecting the variance of spacing and correlation properties of the vantage objects after each object has been added. As soon as either the variance of spacing of one object or the correlation of a pair of objects exceeds a certain threshold, a vantage object is replaced by a randomly chosen new vantage object. These repair steps are typically necessary only at early stages of execution of the algorithm. Since the database objects are added in random order to the index, intermediate spacing and correlation properties of vantage objects form a good estimator of the final properties once a sufficient number of database objects has been added to the index. Redundancy or small discriminative power of a vantage object can therefore be detected early on, keeping the amount of work that has to be redone (reposition all the objects that are already added with respect to the new vantage object) small. For details, see Algorithm 2.2.

Algorithm 2.2 Spacing-correlation based selection
Input: Database A with objects $A_1, ..., A_n$, $d(A, A) \rightarrow \mathbb{R}$, thresholds ϵ_{corr} and ϵ_{sp}
Output: Vantage Index with vantage objects $V_1, V_2, ..., V_m$

```
1: select initial V_1, V_2, ..., V_m randomly
2: for All objects A_i in random order do
3:     include F(A_i) in the vantage space
4:     for All vantage objects V_j do
5:         if σ²_sp(V_j) > ε_sp then
6:             remove V_j
7:             select new vantage object V_new randomly
8:             reposition already added objects w.r.t V_new
9:     if ∃{V_k, V_l| Corr(V_k, V_l)> ε_corr} then
10:        if σ²_sp(V_k) > σ²_sp(V_l) then
11:            remove V_k
12:        else
13:            remove V_l
14:        select new vantage object randomly
```

2.4.5 Complexity

The complexity of our algorithm is expressed in terms of distance calculations, since these are by far the most expensive part of the process. The running time complexity is then $O(\sum_{i=0}^{n} P_i \times i + (1 - P_i) \times m)$ where m is the (in our case constant) number of vantage objects and P_i is the probability that at iteration i a vantage object has to be replaced by a new one. This probability depends on the choice for ϵ_{spac} and ϵ_{corr}. There is a clear trade-off here: the stricter these threshold values are, the better the selected vantage objects will perform but also the higher the probability a vantage object has to be replaced, resulting in a longer running time. If we only look at spacing and set ϵ_{sp} such that for instance P_i is $(\log n)/i$, the running time would be $O(n \log n)$ since m is a small constant.

2.5 Experimental results

We implemented our algorithm and tested it on three data sets of different modality and size: one data set of 1,400 shape contour images, one collection of 50,000 colour photographs and a set of 500,000 fragments of music notation.

Figure 2.5: Examples of the MPEG-7 data set.

An advantage of defining a false positive as in Definition 1, is that evaluating the performance on these datasets does not require a ground truth. To measure performance, the matching process is applied to the candidate matches as returned by the range query on the index. After the exact distances between the query and all candidate matches have been computed, the percentage of false positives within the returned set can be calculated. This is our first evaluation criterion.

For some applications however, a shortcoming of just counting false positives is that it does not take into account the ranking of the true positives in the return sets. For this purpose, we have evaluated our results also by means of average precision: the mean of the precision scores obtained after each true positive is retrieved [24]. A maximum average precision score of 1.0 is obtained when all true positives are at the top of the retrieval ranking.

During the music retrieval experiment we evaluated the results with yet another performance measure, which we call the Average Distance Error (*ADE*), for the following reason. Recall that a false positive A_{fp} is an object that lies within a range of ϵ to the query A_q in vantage space, but has a real distance $d(A_{fp}, A_q)$to the query that is larger than ϵ. One may argue that a false positive with a real distance to the query slightly larger than ϵ is not as bad as a false positive with a real distance exceeding ϵ by orders of magnitude. Therefore, instead of just calculating the precision scores using this definition of a false positive, one may obtain more information by addressing a weight to each false positive. This weight is defined as $d(A_{fp}, A_q) - \epsilon$, i.e. the *extent* to which a false positive is actually a false positive. The Average Distance Error is average of all these false positive weights, taken over a large set of queries.

2.5.1 Shape retrieval

As a first dataset, we used the MPEG-7 test set CE-Shape-1 part B, consisting of 1,400 shape images, contained in 70 classes of 20 images per class. A few examples are given in figure 2.5.

The distance measure used to calculate the distance between two of these shape images is the Curvature Scale Space (CSS) [81]. This technique matches two shapes based on their CSS-image, which is constructed by iteratively convolving the contour with a Gaussian smoothing kernel, until the shape is completely convex. When at a certain iteration a curvature zero-crossing disappears due to the convolution process, a peak is created in the CSS-image. A CSS-image reflects in a way the effort it takes to smooth all the concavities in the curve until the image has become completely convex. Two shapes are now matched by comparing (the peaks in) their CSS-images.

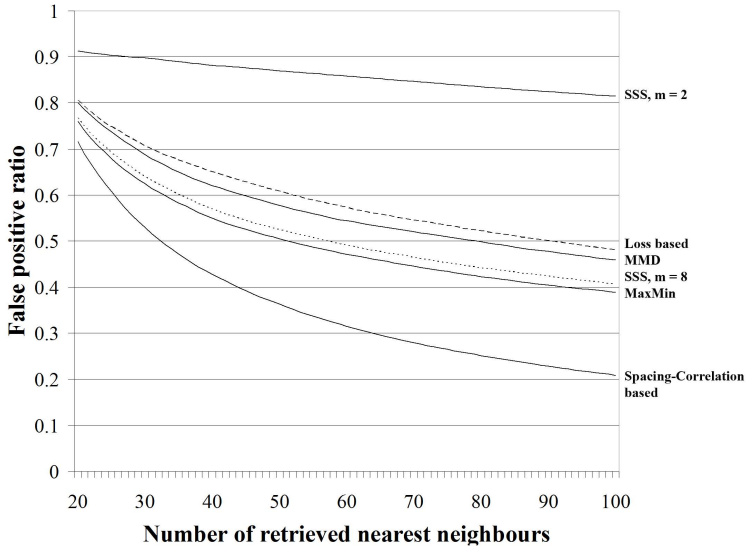
In this experiment, we compare Spacing-Correlation based selection to 4 existing methods that were all described in section 2.3. These methods are called Sparse Spatial Selection (SSS) [23], Maximum Mean Distance (MMD) [25], Loss based [56] and MaxMin [132]. All methods were set to select 8 vantage objects, except for SSS, that chooses its own number of vantage objects. Using the parameters reported in [23], the method selected for this dataset only 2 vantage objects. Because this leads to bad results, we reparametrized the method manually by trial and error such that it selects 8 vantage objects as well.

The performance of these selection methods was evaluated by querying in their vantage spaces with all 1,400 objects. The number of nearest neighbours that was retrieved for each query object ranges from 20 to 100. The distance of the furthest nearest neighbour functioned as ϵ, which was used to calculate the number of false positives among these nearest neighbours, see definition 1. For each vantage index, and all k-NN queries, $k = 20, ..., 100$, an average ratio of false positives was calculated over all 1,400 queries. The results are displayed in figure 2.6.

Although it may seem counter-intuitive that the ratio of false positives declines when more nearest neighbours are retrieved, it is a natural consequence of the definition of a false positive. This definition is dependent on ϵ, so the definition of a false positive may change when more nearest neighbours are retrieved. Specifically, since the distance between the query and the furthest nearest neighbour that is retrieved defines ϵ, the definition will become more tolerant when more nearest neighbours are retrieved. As stated before, the advantage of this definition of a false positive precludes the need of a ground truth and makes performance comparison independent of the quality of the matching algorithm.

This experiment clearly shows that Spacing-Correlation based selection outperforms the other selection techniques. This is mainly because the method achieves a better spread of the database objects in vantage space, since the selection criteria are not assuming that the distances in object space are uniformly distributed. For example,

Figure 2.6: Performance comparison of the proposed method with 4 existing methods: Sparse Spatial Selection (SSS), Loss based, Maximum Mean Distance (MMD) and MaxMin. The figure shows false positive ratios, averaged over all 1,400 queries from the MPEG-7 set (y-axis) with respect to number of retrieved nearest neighbours (x-axis). In all cases the number of vantage objects was 8, except for one automatic run of SSS (m=2). A lower false positive ratio indicates a better retrieval result.



MMD may select the vantage objects such that there exist clusters in vantage space. In this case, the spread is not even, but the mean distance between objects in vantage space may still be high. Furthermore, Spacing-Correlation based selection selects both relevant and non-redundant vantage objects, whereas SSS is only concerned with non-redundancy.

Table 2.1 shows similar results, concentrated on 100-nearest neighbour queries: on the left false positive ratios averaged over 1,400 queries, on the right average precision.

2.5.2 Colour based photo retrieval

The second dataset we used is significantly larger, it consists of 50,000 colour photographs. Colour histograms were constructed for these photographs, and normalised histogram matching was used as a distance measure. In this experiment, we com-

Table 2.1: False positive ratios and average precision for the MPEG-7 set

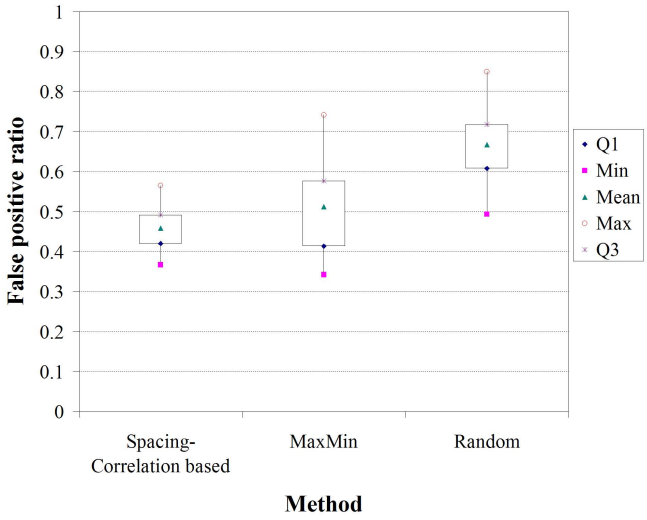
Method (100 NN)	false positive ratio	average precision
SSS (m=2)	0.81	0.27
Loss based	0.48	0.47
MMD	0.46	0.49
SSS (m=8)	0.41	0.52
MaxMin	0.39	0.53
Spacing-correlation based	0.21	0.65

pared our strategy for selecting vantage objects to randomly selecting the vantage objects and the MaxMin approach. Because the Loss-based method is computationally expensive to evaluate, this method has not been tested on a dataset of this size. All three strategies (Random, MaxMin and Spacing-correlation based) were applied multiple times (several runs), since randomness in the methods may influence the performance from one run to another. The performance for each run was measured over the same set of 1,000 randomly chosen query objects, and is expressed in terms of the average false positive ratio given a fixed range and dimensionality of the vantage space. Boxplots showing the results are presented in figure 2.7.

These results show that Spacing-correlation based selected vantage objects yield a lower false positive ratio (notice that both the median, represented by the line within the box, and the mean, represented by the diamond are lower). Furthermore, it shows that the variability over a large number of runs for Spacing-correlation based selection is lower. This means there is more reason to believe a specific set of Spacing-correlation based selected vantage objects performs well, than there is for the other selection methods, where random effects are influencing the performance wildly.

We also investigated the influence of the dimensionality of the vantage space on this dataset. Again, the range for all queries in this experiment was fixed, however the number of vantage objects that was used varied. For this experiment, we selected a typical run for each of the selection strategy and queried with 1,000 random queries on each index. The results are displayed in figure 2.8. These results show once more that false positive ratios are smaller for Spacing-correlation based selected vantage objects. In particular, they show that with well chosen objects, a vantage space of smaller dimensionality can yield the same performance as a vantage space of higher

Figure 2.7: Photographs: false positive ratios

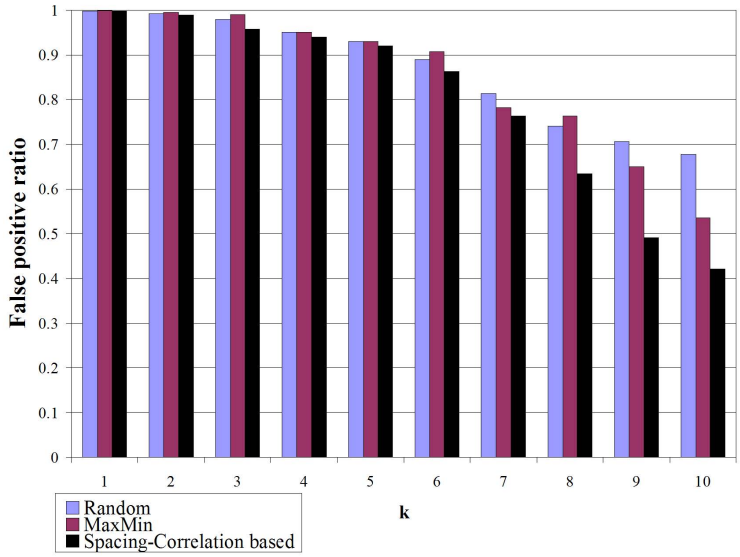


dimensionality with randomly selected vantage objects for instance. Furthermore, the higher the dimensionality of the vantage space, the larger the improvement in performance gets. This means that with Spacing-correlation based selection, more relevant and non-redundant objects can be found even though there are already a number of objects selected, whereas the other methods select at this point more redundant (and possibly less-relevant) vantage objects.

2.5.3 Music retrieval

We have compared Spacing-correlation based selection to random selection on a data set of 500,000 incipits of 5 notes from a collection of notated music. This collection is created by RISM, the Répertoire International des Sources Musicaux, and is called RISM A/II. The notes in these incipits are represented as weighted points in a space in which the pitch and onset time are the axes [119] and the duration denotes the weight. The distance between two fragments is computed using a transportation distance. These transportation distances measure minimum the effort it takes to transform one weighted pointset into the other. In the case of the Earth Mover's Distance, a metaphor of transporting piles of earth (one pointset) to empty holes (the other pointset) is used. The weight of the points corresponds to the mass of a

Figure 2.8: Photographs: false positive ratios for different vantage space dimensionalities (k)



pile, or the capacity of a hole. Both point sets can arbitrarily serve the role of receiver or supplier. A well known example of a transportation distance is the Earth Mover's Distance (EMD) [92]. However, for this EMD it is known that it does not obey the triangle inequality, so for this experiment the Proportional Transportation Distance [49] was used, which is a modified version of the EMD such that the triangle inequality holds.

The results for this experiment are shown in figure 2.9. In vantage spaces of different dimensionality k (multiple selection runs per dimensionality), false positive ratios were computed over 1,000 randomly chosen queries. Again, Spacing-correlation based selected vantage objects produce less false positives for all values of k than randomly selected vantage objects.

Figure 2.10 shows Average Distance Error (ADE) values for our music dataset. This experiment considers $k = 5$ and higher only, since smaller sets of vantage objects produce almost only false positives. These results show that Spacing-correlation based selection not only reduces the number of false positives; the extent to which the false positives are false is reduced as well. One may therefore say that pairwise distances are better preserved using Spacing-correlation based selection.

Figure 2.9: Music: false positive ratios for different vantage space dimensionalities (k)

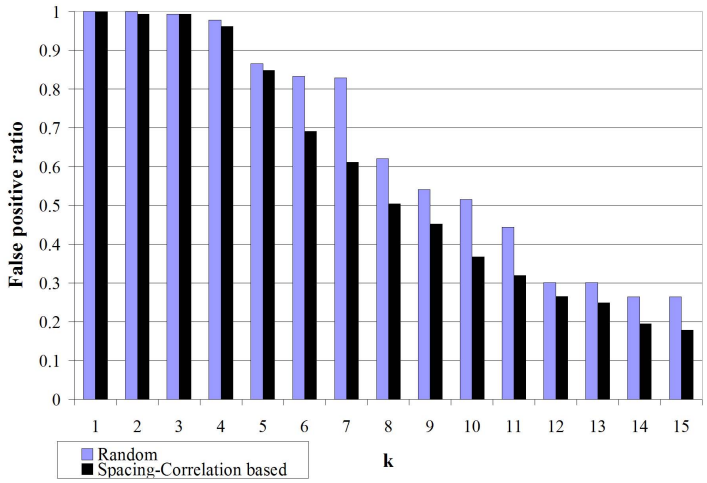
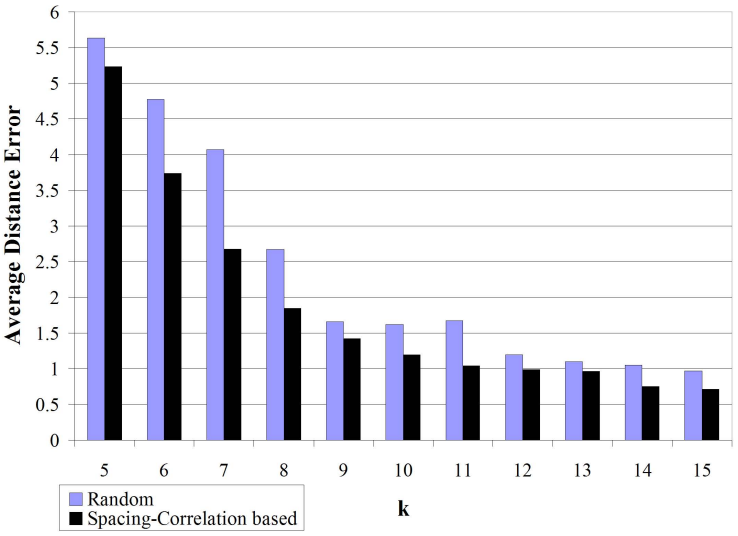


Figure 2.10: Music: average distance error for different vantage space dimensionalities (k)



2.6 Concluding remarks

An object space is spanned by a set of object models and a similarity measure that outputs a similarity or distance value for two given object models. Efficient querying of this object space requires indexing, since otherwise the query will have to be compared to every object in the dataset. When the object models are not feature vectors, a direct space or data partitioning of the object space is not trivial, and embedding or mapping methods are more appropriate. Every object model is mapped to a point in a new feature space, that in turn can be partitioned using well known access methods that facilitate range or nearest neighbour querying.

Vantage indexing is such a mapping technique, where the features of the mapped object models correspond to distances they have to reference objects, called vantage objects. The selection of these vantage objects is crucial to the performance of these systems. In this chapter, we have presented a new approach for selecting good vantage objects. The method, called Spacing-correlation based selection, uses two criteria that directly address the number of false positives. The first criterion is concerned with the relevance of the vantage objects. This individual performance of each vantage object is measured through the evenness of the distribution of distances all objects have to this vantage object. The second criterion is concerned with the redundancy of the vantage objects. The combined performance of a pair of two vantage objects is measured through the linear correlation coefficient of the distances all objects have to these two vantage objects. We have shown a selection strategy that chooses vantage objects according to these criteria, and at the same time constructs the actual index.

The approach has been tested on three real-life datasets of different size and modality: 1,400 silhouettes, 50,000 photographs and 500,000 musical incipits. On all datasets, Spacing-correlation based selected vantage objects produce significantly less false positives than random selection or other known selection techniques. In addition, we have shown that the variability in performance is smaller with Spacing-correlation based selection, and that the pairwise distances are better preserved.

Chapter 3

Indexing through Laplacian spectra

Relevant publications:

M.F. Demirci, R.H. van Leuken, R.C. Veltkamp
Indexing through Laplacian Spectra
In: Computer Vision and Image Understanding, Vol 110/3, pp 312-325, 2008

R.H. van Leuken, M.F. Demirci, V.J. Hodge, J. Austin, R.C. Veltkamp
Layout Indexing of Trademark Images
In: Proc. Conference on Video and Image Retrieval, pp 525-532, 2007

M.F. Demirci, R.H. van Leuken, R.C. Veltkamp
Shape Indexing through Laplacian Spectra
In: Proc. Visual and Multimedia Digital Libraries, pp 21-26, 2007

A. Pinto, R.H. van Leuken, M.F. Demirci, F. Wiering, R.C. Veltkamp
Indexing Music Collections through Graph Spectra
In: Proc. International Conference on Music Information Retrieval, pp 153-156, 2007

3.1 Introduction

FEATURE vectors that describe objects come in many different forms. For example, a vector with scalars for weight, roundness, curve and colour can be used to describe physical objects numerically. A photograph can be represented by a histogram that contains the frequency distribution of the colours that make up the image. Similarly, occurrence frequencies of terms can be used to capture a document's contents. All these feature vectors tell us something about the object they describe, they provide information that even *we* can understand. We can learn the size of an apple, the reddishness of a sunset picture or the relative occurrence of the word 'index' in this thesis. Although it can be a comforting thought to work with numerical representations of the objects that still make sense to us, it is by no means a necessary requirement for a computer when it needs to perform successful retrieval or recognition operations. All a computer needs is a digital representation that, when deployed in combination with a sensible matching operation, ensures that similar objects have similar representations and different objects have different representations, preferably with a gradual transition from one to the other that coincides with our notion of similarity. That this representation is unreadable to humans is of no importance. Rather, it provides us with more freedom to choose and calculate features, drawing on mathematical theorems to ensure their soundness instead.

In this and the following chapter, we derive index signatures or feature vectors that are quite distant from the original objects they describe. While going from original object to digital representation, we must ask ourselves at all stages what the consequences are for the descriptiveness of our representation. The first and probably most influential step we take in this process, is the representation of the object by a graph. A graph is a very flexible and intuitive structure, and its properties have been studied extensively in centuries of graph theory [4]. For recognition and retrieval purposes, it is common to represent objects by graphs whose nodes correspond to object features and whose edges indicate relations between these features. Both nodes and edges may be labelled with attributes. We will present graph representations for four different kinds of objects: 3D models, 2D shapes, fragments of music notation and trademark images. The graph representations express many significant object properties such as geometric or hierarchical structures. They come with a serious challenge however: we have to deal with the difficult problem of (sub)graph matching in order to design an indexing mechanism.

[4] Euler's article on the *Seven Bridges of Königsberg* from 1736 is commonly acknowledged as the first work in graph theory

3.1.1 Preliminaries

A graph G is a pair (V, E), where V is a finite set of vertices and E is a set of connections (edges) between the vertices. The size of a graph is defined as the number of vertices. An edge $e = (u, v)$ connects two vertices such that $u, v \in V$. A graph $G = (V, E)$ is called edge-weighted if each edge $e \in E$ has a weight $w(e) \in \mathbb{R}$. Unweighted graphs are a special case of weighted graphs, where each of the edges has weight 1. A graph is $simple$ if it does not contain self loops or multiple edges and thus its edge set consists of distinct pairs. All graphs considered in this chapter are simple. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are $isomorphic$, if there is a bijection $f : V_1 \rightarrow V_2$ such that for any vertex pair u and $v \in V_1$, $(u, v) \in E_1$ iff $(f(u), f(v)) \in E_2$.

The $adjacency\ matrix$ A of a graph G is a $|V| \times |V|$ matrix whose element with row index u and column index v is

$$A(u,v) = \begin{cases} w(e) & \text{if } (u,v) \in E \\ 0 & \text{Otherwise.} \end{cases}$$

The $degree\ matrix$ of a graph G is a diagonal matrix of vertex degrees with elements

$$D(u,u) = \sum_{v \in V} A(u,v)$$

The matrix $L(G) = D(G) - A(G)$ is called the $Laplacian\ matrix$ of G [5]. The Laplacian matrix is a positive semidefinite and symmetric matrix with at least one zero eigenvalue. The multiplicity of the eigenvalue zero of $L(G)$ is equal to the number of connected components in the graph. This implies that the second smallest eigenvalue, known as algebraic connectivity, is positive if and only if G is connected. There exist many important theorems about Laplacian matrices and in many problems in physics and chemistry they play a central role. The reader is referred to [79, 80, 75, 78] for surveys on this topic.

The spectrum of a graph's Laplacian matrix is obtained from its eigendecomposition. Specifically, the eigendecomposition of a Laplacian matrix is $L(G) = P\Lambda P^T$, where $\Lambda = \text{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_{|V|})$ is the diagonal matrix with the eigenvalues in increasing order and $P = (p_1|p_2|\ldots|p_{|V|})$ is the matrix with the ordered eigenvectors

[5] The Laplacian matrix is also called Kirchhoff matrix or the matrix of admittance in the literature.

as columns. The Laplacian spectrum is the set of eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_{|V|}\}$. The spectrum is permutation-invariant, i.e., two isomorphic graphs have the same set of sorted eigenvalues. However, the converse is not true, as two graphs that have the same spectra are not necessarily isomorphic. When two graphs have the same eigenvalues, they are called $cospectral$ or $isospectral$.

3.1.2 Related work

Graph matching problems are often formulated as largest isomorphic subgraph problems, for which a rich body of research exists in the literature, such as pattern recognition [72, 69], chemical structures [91], or computer vision [140, 62]. This problem has been studied for both theoretical and practical interests. While it is an open question whether the detection of graph isomorphism can be solved in polynomial time, the problem of subgraph isomorphism is known to be NP-complete [48]. Although (sub)graph isomorphism detection is computationally expensive, some graph isomorphism detection algorithms with polynomial-time complexity have been developed for specific graph classes, e.g., planar graphs [61]. It is also possible to derive polynomial-time complexity algorithms for graphs with certain restrictions [60].

When working with graph structures, indexing is formulated as the problem of efficiently selecting a small set of database graphs, which share a subgraph with the query. Several frameworks have been proposed to use (sub)graph isomorphism algorithms with indexing methods. Shapiro and Haralick [100] proposed a method to organise similar graphs in clusters where each cluster is indexed by a representative graph. Sossa and Horaud [110] used the coefficients of the d_2-polynomial of the Laplacian matrix of a graph to index into graph datasets. These coefficients, however, are only unique for small graphs with less than 12 vertices.

One important indexing method is a decision tree approach. Here, the goal is to hierarchically partition the database so that the query is first matched to the root. Depending on the result of this match, the query is then matched to either the right or the left child of the root. This process is repeated recursively until a match is found at an internal node (or leaf), or it exits with a failure indicating no database graphs are isomorphic to the query. Messmer and Bunke [76] use this approach to organise the set of all permutations of the adjacency matrix of database graphs in a decision tree. At run time, the (sub)graph isomorphism from the query to the database graphs is found by a decision tree traversal. A significant drawback of this method is its space requirement. All permutations of the adjacency matrix have to be encoded in decision trees, whose sizes grow exponentially with the size of the database graph. A set of pruning techniques is discussed to cut down the space complexity.

So far, we have only considered the problem of (sub)graph isomorphism. However, due to noise, occlusion, or object misinterpretation such as segmentation errors, (sub)graph isomorphism may not exist between the query and the database. Furthermore, only a certain degree of similarity between two objects and thus between their graphs may be present. The indexing problem, therefore, is reformulated as efficiently retrieving database graphs whose (sub)structure is similar to the query. Although considerable research has been devoted to the problem of inexact (or error-tolerant) graph matching, rather less attention has been paid to this type of indexing based on graph structures.

Costa and Shapiro [31] present a graph-based indexing method, where small relational subgraphs are used to efficiently retrieve similar graphs from a large database. An integrated framework related to the approach described in this chapter is that of Shokoufandeh et al. [102]. This framework is designed especially for tree structures in which the sum of the largest eigenvalues of the adjacency matrix for each subtree of the root form the component of its $\delta-$dimensional vector, where δ is the root degree. To account for occlusion and local deformation, these vectors are also computed for the root of each subtree. At indexing time, each non-leaf node of the query is represented as such a vector, and a nearest neighbour search is performed for each vector. Although effective, by summing up the largest eigenvalues one loses uniqueness, resulting in less representative graphs in the vector space. In addition, it is not clear how this approach can be extended to general graph structures.

One of the primary aspects of graph theory is to derive the principal properties and structure of graphs from their graph spectra. It is well known that eigenvalues are closely related to main invariants of a graph. In the computer vision and pattern recognition communities, eigenvalue-based frameworks have been applied to various problems including shape description and indexing. Sengupta and Boyer [99] used eigenvalue-based feature representation of CAD models to capture their gross characteristics. This representation is used to partition the database into structurally homogeneous groups. Shapiro and Brady [101] used eigenvectors of proximity graphs to compute the feature correspondences. Turk and Pentland [118] proposed an eigenface approach in which images were represented as linear combinations of a small set of images computed from a large database. The algorithm was applied to face recognition. Sclaroff and Pentland [97] computed the eigenmodels of 2D regions and used the model coefficients in a linear search of 2D shapes. However, since the characterisations in this approach are global, it is not clear how this method performs for retrieving models with local similarities. Some other eigenvalue-based methods consist of applications such as edge detection [115], motion estimation [52], and 3D object representation as 2D images [27].

3.1.3 Contributions

In this chapter, we propose a novel approach to the graph-based indexing problem [37]. Instead of using the adjacency matrix for graph characterisation as done some earlier work, we characterise our graphs based on the Laplacian spectrum, which is more natural, more important, and more informative about the input graphs [78]. The definition of a Laplacian matrix along with other graph-theoretical concepts used in this chapter are given in the next section. Given a graph $G = (V, E)$, the sorted eigenvalues of its Laplacian matrix become the components of its signature, a $|V|$-dimensional vector. Since the Laplacian spectrum is used as a graph signature without an approximation such as considering only largest eigenvalues, a high level of uniqueness is maintained. We will discuss techniques to reduce the cost we pay for computing such a signature in the framework.

Having established the signatures, the indexing now amounts to a nearest neighbour search in a vector space. For a query graph and a large graph dataset, we can, therefore, formulate the indexing problem as that of fast selection of candidate graphs whose signatures are close to the query signature. This formulation alone cannot support occlusion or segmentation errors as two graphs may share similar structures only up to some level. To perform indexing locally and thus to encode the topology of subgraphs in the framework, we adopt a technique analogous to that used in the decision tree approach [76]. Given the Laplacian spectrum of a principal submatrix B of matrix A, we draw on an important theorem from spectral graph theory to show that our graph characterization can be used to retrieve similar graphs or subgraphs from large database systems through a nearest neighbor search.

The local indexing method used in the framework is effective, but the signature of each subgraph of a graph is computed individually. To overcome this, we use recently-developed techniques in the domain of spectral integral variation. Specifically, given the Laplacian spectrum of graph G, we will explore an efficient method to generate the Laplacian spectrum of graph $G + e$, where $G + e$ is a graph obtained by adding edge e to graph G. To our knowledge, the proposed framework is the first framework that uses spectral integral variation in an indexing algorithm.

This approach is of particular interest to applications where the size of the database is large, but the size of each graph is relatively small (less than around 24 vertices). Although our method has a similar start-up to [76], it differs by a number of important factors. First, we use the Laplacian rather than the adjacency matrix for graph characterization. Second, since the permutation-similar matrices result in the same set of sorted eigenvalues, we consider such Laplacian matrices once, avoiding the need for a high-load compilation process described for this type of adjacency matrices

in [76]. Third, probably the most important difference is that our method is intended for retrieving similar database graphs, requiring no significant graph isomorphism, although the framework can easily be modified to isomorphism detection.

3.2 Graph representations of objects

The first step in the process from object to index signature is the representation of the object as a graph. The mapping from object to graph is of course different for each retrieval application or domain. In this section, we describe the graph representations that we used in our experiments, see section 3.5 for descriptions of the datasets and retrieval results. To underline the flexibility of the proposed framework, we used four different kinds of objects: melody scores [88], trademark images [121], 2D shapes (silhouette images) [36] and 3D models [37].

3.2.1 Melody graphs

Our goal is to provide a sufficiently abstract representation of a melodic line that actually makes sense from a musical point of view. With this aim, we concentrate on the pitches that make up the melody, ignoring the rhythm. Melodies are generally studied from a pitch sequence/contour point of view. Our approach is different: we take as a starting point the interval structure, by which we mean the network of connections between pitches. We remark that melodies use only a subset of all possible connections, and with different frequencies. To model these relationships between the pitches of the melody we use graphs. As such, the graph is a projection of the time-dependent concept of melody to a time-independent concept of intervallic structure. The next level of abstraction is to leave out pitch class information so that only the 'interval connectivity' of the melody remains, and this means that certain operations such as inversion, transposition, retrogradation, other kind of permutations in the pitch class set and (some) shifting of fragments does not affect the graph. In this perspective what we are modelling is a global, time-independent signature of the melody [89], [87]. Melodies that display a similar interval behaviour have similar graphs, for example melodies in which there are one or two central notes (with many connections) and a number of peripheral notes (few connections).

Let M be a melodic sequence of length $m = |M|$ and consider the sequence of pitches $\{p_j\}_{j \in I}$, $I = \{1, ..., m\}$. Then let $V = \mathbb{Z}_{12}$ be the (metric) space of pitches, or pitch classes, in the 12-tone system. We define the graph G with vertex set $V_G = V$ and

edge set whose elements are the edges a_j such that

$$a_j : \begin{cases} p_j \rightarrow p_{j+1} & \text{for every couple } (p_j, p_{j+1}) \subseteq M \\ p_m \rightarrow p_1 & \text{for the couple } (p_m, p_1) \end{cases}$$

where $j = 1, \ldots, m-1$.

The arrow $a_m : p_m \rightarrow p_1$ does not represent an actual interval in the melody in all cases, but it can been added for symmetry reasons and in order to take into account the relationship between the last and the first note as well, which otherwise would not have been reflected in the representation. In the experiments we conduct in this chapter, we consider single phrases of a melody. In many cases, these phrases are repeated, so the relation between the last and the first note is preferably encoded. The graph traversal that reproduces the original melody is not encoded, so the representation is invariant under the aforementioned transformations.

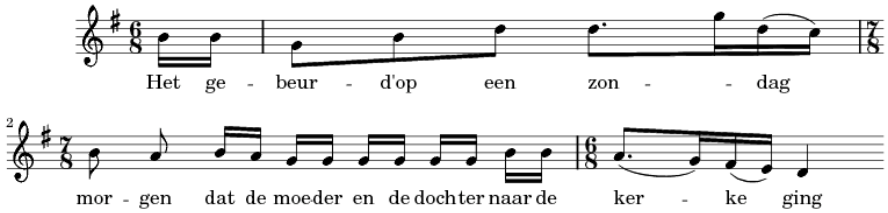
As an example, consider the two melodies and their graph representations in figure 3.1. To improve readability of the graphs, the pitch classes are drawn as if they were node labels, but these are not actually part of the representation. Furthermore, multiple occurrences of the same transition are grouped and their multiplicity is given as edge weights.

3.2.2 Trademark layout graphs

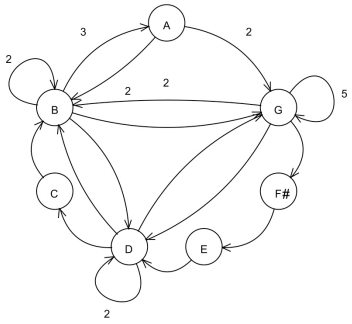
Given a query image, most automatic trademark retrieval systems aim to find images with similar shapes without taking into account the spatial layout of the shapes. Although retrieving images containing similar shapes may seem as the primary goal, there are many cases where the layout similarity plays a more important role for ensuring uniqueness. An example of this scenario is given in figure 3.2 in which the layout of the shapes reveals a strong figure in itself. The three trademarks resemble each other despite the dissimilarity between their individual shapes. In case these three trademarks are to be registered in the same or in a closely related product group or service category, a conflict of uniqueness arises.

Layout similarity between trademarks is also used to improve the quality of matching based on shape similarities. Consider figure 3.3, where two candidates are returned with the same similarity scores against a given query. Although they both contain the same shapes, the middle candidate should be assigned a stronger similarity value since its shapes are in a configuration similar to that of the query. Hence, one may observe that applying layout similarity improves the overall quality of a trademark retrieval system.

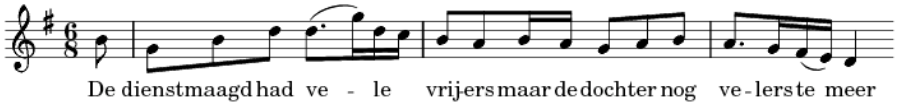
Figure 3.1: Two instances of the folk song "In Frankrijk buiten de poorten" (a and c), together with their melody graphs (b and d)



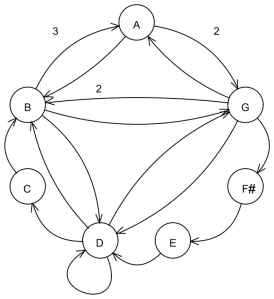
(a)



(b)



(c)



(d)

Figure 3.2: Three trademarks resemble each other based on the layout of their shapes despite the dissimilarity between their individual component shapes.

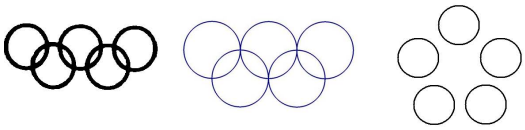


Figure 3.3: Three different configurations of 5 circles. Suppose the leftmost image, the Olympic logo, is used as a query. Because of a similarity in layout,the middle image should receive a higher vote than the rightmost image, despite the fact that pure shape similarity on components is the same.

We use the graph-based representation here to facilitate efficient trademark retrieval based on spatial relations between image shapes regardless of their mutual shape similarities. To construct a graph for a trademark, we follow the ideas presented in [59]. The trademark is first segmented into distinct shapes using a closed shape identification algorithm. A simple edge detector would not be sufficient as many images in our test set are noisy and this noise causes small gaps in the shape boundaries so these gaps need to be closed. In practice any closed shape identifier could be used here, such as region growing [148] or watershed [14].

We use a simplified version of the algorithm presented in [59]. The simplified version aims to find just the basic shapes present in an image, as these are the necessary building blocks for the layout graph. The closed shape algorithm requires an underlying technique to identify the line segments within an image and to detect the relationships between those line segments. It then uses this output to identify the closed shapes. Therefore, we initially find the edges in an image using the Sarkar & Boyer [95] edge detection algorithm and subdivide these into constant curvature segments using the Wuescher & Boyer [141] curve segmentation algorithm. The motivation for using these methods comes from their successful application in the trademark retrieval system developed by Alwis [2].

Each constant curvature segment that is obtained becomes a node in a temporary graph that is used in the closed shape finding algorithm. Two nodes in this graph

are connected if the endpoints of their respective segments are end-point proximal within some range. The closed shape algorithm overlays this graph: it follows Saund's approach in managing the search of possible path continuations through the segment graph [40]. From each endpoint (first and last) of each node, all possible paths are followed. This effectively forms a search tree with paths through the tree representing the paths of candidate shapes.

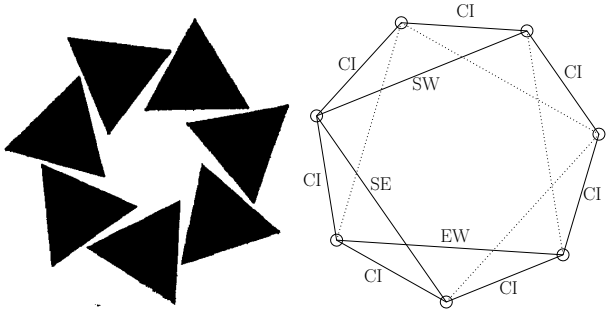
The search is managed through the use of local criteria (scores) for ranking possible paths through junctions (crossroads, T-junctions etc). Saund derived the scores from human observations. These scores prioritise which node to expand next. As each leaf node in the tree is expanded, any new child nodes are compared with child nodes in the opposite side of the tree. If they are end-point proximal then a closed path has been identified and its nodes and pixels are added to the list of candidate paths. To produce the set of shapes for the trademark image, we accept all candidate paths. However, closed paths that are subsumed by other closed paths with higher scores are discarded. Hence, each new closed path is compared to all existing stored paths. If the new path is equivalent to an existing path but has lower score then the new path is discarded. If the new path has higher score than the existing saved path then the saved path is discarded.

When all shapes have been found, the layout in which they form the trademark image is encoded in the final graph that we use in the indexing mechanism. The vertices in the layout graphs correspond to the shapes in the trademark image, and the edges between them carry relations between them. Foremost, we encode the directional information (in the form of both primary and secondary directions). Rotational invariance can be achieved on demand, by neglecting for instance the difference between a south-north edge and an east-west edge. Furthermore, we are interested in detecting certain basic layout configurations that often occur in trademarks, such as triangular, circular or square configurations. If one or more of these types of layout are present in a trademark, they are encoded in the appropriate edges too.

To extract the layout of the shapes from the trademark image, the centroids (centres of mass) of the shapes are used in the calculations. Each shape is connected to its n nearest neighbours, where n is a user defined parameter. The first step is to calculate the angle between the horizontal axis and a line connecting two centroids to determine the directional label for this edge. In principle there are eight possible directions (4 primary and 4 secondary, corresponding to the directions N, NE, E, SE, S, SW, W and NW), however with undirected edges there are four possible directions.

The next step is to detect the special pattern 'square'. This is done by performing a template match on the directional graph with a template representing a configu-

Figure 3.4: Simple example of a layout graph with its corresponding trademark. Some of the edges have dotted lines and their labels removed for readability



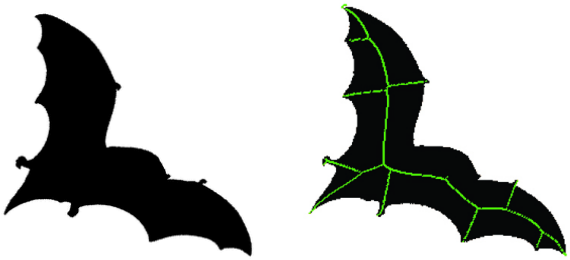
ration of four shapes in a 2×2 square. Whenever this template is found, the edge labels are updated accordingly from the directional information to the special edge type square. Note that the square needs to be isolated to a certain extent; e.g. a grid is not a large collection of squares according to this definition. The same kind of template matching is performed for triangular and circular configurations. The decision of triangularity depends on the angles between the possibly triangular edges. Since every triplet of objects forms a triangle by definition, only the edges of a triangle with three 60 degrees angles (or close to 60 degrees angles) are labelled with the special triangle edge type. To detect circular configurations, the following circularity criterion is evaluated on the convex hull of the shape centroids: $4\pi A \diagup \rho^2$, where A is the area and ρ is the perimeter of the convex hull. For a circle, the ratio is one; for a square, it is $\pi/4$; for an infinitely long and narrow shape, it is zero. A threshold is set on the outcome of this circularity criterion to determine whether the edges on the convex hull need to be labelled with the special circular type or not.

To summarise, our graph construction strategy in the case of layout graphs consists of the following steps: line segmentation, constant curvature segment aggregation, closed shape finding, centre of mass computation, nearest neighbour finding and layout feature extraction. An example layout graph is given with its corresponding trademark in figure 3.4.

3.2.3 Reeb and Shock graphs

To show that the proposed framework can also be applied for shape retrieval, we use it in combination with Shock graphs (for 2D shapes) and Reeb graphs (for 3D shapes). These shape descriptors are graph-based descriptors that capture geometri-

Figure 3.5: Left: a view of a bat. Right: the shock graph constructed from the medial axis and superimposed on the left image.

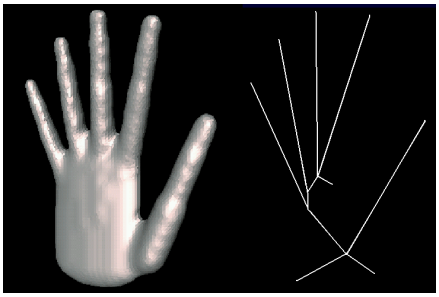


cal and/or topological characteristics of the shape and resemble the shape's skeleton or medial axis.

For a given 2D shape, its silhouette is represented by an undirected shock graph [104]. The graph is constructed from the discrete skeleton using the method described in [35]. To summarise this process, a shock point p on the discrete skeleton is labelled by a 3-dimensional vector $v(p) = (x, y, r)$, where (x, y) are the Euclidean coordinates and r is the radius of the maximal bitangent circle centred at the point. Each shock point becomes a node in the graph and edges connect nearby shock points. An illustration of this procedure is given in figure 3.5, where the left portion shows an input image taken from the database, while the right portion presents the constructed shock graph superimposed on top of the image.

We also conduct our experiments in the domain of 3D object recognition. Research in the field of shape description proposes different structures that can be used to represent a 3D model in the retrieval process. These structures, called shape descriptors, can capture different shape properties of a 3D model. For more information about shape descriptors, see [17, 113]. We use Reeb graphs to represent the 3D models, that were proposed in the context of shape description and retrieval in [19, 57, 73, 116, 117]. These graph representations allow for topological properties to be represented in a coarse sense. Let $f : S \rightarrow \mathbf{R}$ be a real-valued function on surface S. The Reeb quotient space is defined by the equivalence relation $\sim$ given by: $(\alpha, f(\alpha)) \sim (\gamma, f(\gamma))$ for $\alpha, \gamma \in S$ iff $\mathrm{f}(\alpha) = \mathrm{f}(\gamma)$ and α, γ are in the same connected component of $f^{-1}(f(\alpha))$. This means two points $(\alpha, f(\alpha))$ and $(\gamma, f(\gamma))$ are represented as the same node in the Reeb graph if values of f are the same and they belong to the same connected component of the inverse image of $f(\alpha)$ (or, equivalently $f(\gamma)$). The Reeb quotient space is coded in a Reeb graph such that the vertices

Figure 3.6: The Reeb graph constructed for the object on the left is shown on the right.



represent critical points of function f, while the edges show the connections between them. See [8, 57, 30, 16, 20] for details. The right of figure 3.6 shows a Reeb graph constructed for the image shown in the left.

3.3 Indexing the graphs

In the previous section we have shown how to represent objects of different kinds by graphs. However, this representation is still a geometric one. The next step in the process from object to index signature is to associate an algebraic structure to the graph. For a graph with vertex set V and edge set E, the most natural structure for this purpose is a $V \times V$ matrix. The spectrum of this matrix (its sorted set of eigenvalues), can then be used as index signature or final $|V|$-dimensional feature vector.

A common choice to represent a graph is to the adjacency matrix. However, Godsil and McKay [50] and more recently Haemers and Spence [55] have shown that the Laplacian matrix has more representational power than the adjacency matrix, in terms of cospectrality of non-isomorphic graphs. According to the results given in [55], of more than a billion graphs with 11 vertices characterised by the adjacency matrix, approximately 21% is cospectral, while this fraction is only 9% for the Laplacian matrix. As specific graph classes, trees were also investigated for cospectrality by Zhu and Wilson [139]. The authors report that out of more than two million trees with 21 vertices, 21.3% of them do not have a unique adjacency spectrum. With the Laplacian spectrum, this ratio decreases to 0.05%. Overall, these studies show that the Laplacian spectrum is more representative and more informative than the adjacency spectrum. Among other reasons, this motivates us to use the Laplacian

spectrum to represent the graphs.

In our framework therefore, we compute the similarity between two graphs as the Euclidean distance between their Laplacian spectra, which is inversely related to the structural similarity of the graphs. For a given query, retrieving similar graphs can then be reduced to a nearest neighbour search among a set of points. Note that it is important to construct the signatures using the sorted eigenvalues, as the kth smallest eigenvalue reflects specific information about the graph, e.g., the relation between the second smallest Laplacian eigenvalue λ_2 and the diameter, mean distance, minimum degree, and algebraic connectivity of the graph. See the papers [44, 85, 79, 80, 75, 78] for details about the relations between the Laplacian spectrum and the graph structure. Furthermore, by sorting the set of eigenvalues, the signature becomes invariant under reordering of the graph vertices.

Unfortunately, the above formulation cannot support occlusion or segmentation errors: two graphs may share similar structures up to only some level. Although adding or removing graph structure changes the Laplacian spectrum, the spectrum of the subgraphs that survive such alteration will not be affected. Therefore, our indexing mechanism cannot depend on the signature of the whole graph alone. Instead, we will combine the signatures of the subgraphs with signatures of the whole graph.

3.3.1 Local Indexing

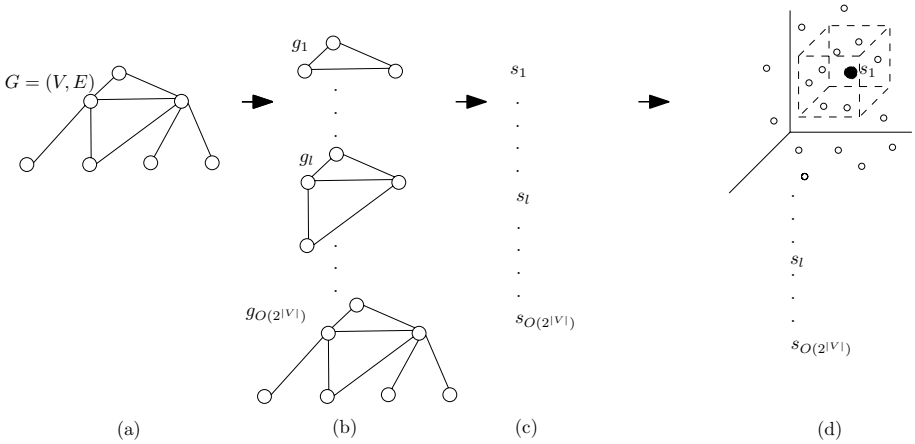
Let $G = (V, E)$ be a graph and let G' be a graph obtained from G by adding a new edge e' such that $e' \notin E$. Then the following theorem, known as the interlacing theorem, relates the Laplacian spectrum of both graphs. This theorem is obtained by Courant-Weyl [32], theorem 2.1, see also [53].

Theorem 1 *The eigenvalues of G and G' interlace:*

$$0 = \lambda_1(G) = \lambda_1(G') \leq \lambda_2(G) \leq \lambda_2(G') \leq \ldots \leq \lambda_n(G) \leq \lambda_n(G').$$

In addition, it is known that $\sum_{i=1}^{n}(\lambda_i(G') - \lambda_i(G)) = 2$, see [5]. Therefore, at least one inequality is strict. Overall this theorem implies the following. Assume that we are given a pair of isomorphic graphs g_1 and g_2. If we construct G_1 and G_2 out of g_1 and g_2 by adding different edges to each of them, one at a time, the Laplacian spectra of G_1 and G_2 become proportionally less similar. As a result, the similarity between the signatures of G_1 and G_2 may not reflect the similarity between the signatures of

Figure 3.7: Retrieving similar graphs. For graphs given in Part (a), its subgraphs are constructed in Part (b). A signature is computed for each subgraph in Part (c). Given a signature, retrieving its similar graphs from a large database is formulated as a nearest neighbour search as shown in Part (d).



their subgraphs g_1 and g_2. Therefore, constructing the indexing mechanism based on graph signatures is too weak. An ideal indexing framework should, in fact, select candidate database elements based on both local and global similarities. To account for local as well as global information in our framework, we will adopt the following method analogous to that used in the decision tree approach [76].

For a given database graph G, rather than storing its signature in the system only, we compute the signatures of each subgraph of G in our algorithm. In this process, we gradually increase the size of the subgraphs. Since the sorted eigenvalues are invariant under consistent re-orderings of the graph's vertices, it is sufficient to compute the spectrum of permutation-similar matrices once. This property avoids the need for a high-load compilation process described for adjacency matrices in the decision tree approach.

Associated with each signature in the system is a pointer to the corresponding graph or subgraph in the database. At runtime, we first generate the signature of each subgraph of the query. Given a query signature s_q, we then retrieve its nearest neighbours of the same size from the database through a nearest neighbour search (see figure 3.7). Each neighbour of s_q retrieved from the database gets a vote whose value is inversely proportional to the distance from s_q. Thus, as a result, each signature of the query generates a set of votes. Moreover, we weigh the votes according

to the size of the subgraphs corresponding to the signatures, i.e., the bigger the size, the more weight the vote receives. To collect these votes, we will use the following strategy.

Let $S_q = \{s_{q1}, \ldots, s_{qm}\}$ be the set of query signatures. For a particular signature $s_{qi} \in S_q$, let $N_{s_{qi}} = \{n_1, \ldots, n_k\}$ be the set of elements returned by the nearest neighbour search and let $|s_{qi}|$ denote the size of its corresponding subgraph. ($|s_{qi}| = |n_j|$ for $j = \{1, \ldots, k\}$). We compute the weight of the vote between s_{qi} and a signature s_{di} corresponding to a database (sub)graph as follows:

$$W_{s_{qi}s_{di}} = \begin{cases} \frac{|s_{qi}|}{1+||s_{qi}-s_{di}||_2} & \text{if } s_{di} \in N_{s_{qi}}, \\ 0 & \text{Otherwise.} \end{cases} \tag{3.1}$$

Given a query $G_q = (V_q, E_q)$ and a database graph $G_d = (V_d, E_d)$ of size $|V_q|$ and $|V_d|$ respectively, let S_q^k denote the set of signatures for subgraphs of size k of the query graph, and let S_d^k be this set for the database graph. For a certain size k, the weight of the votes based on S_q^k and S_d^k is computed using the directed Hausdorff distance as follows:

$$h(S_q^k, S_d^k) = \max_{s_{qi} \in S_q^k} \{ \min_{s_{di} \in S_d^k} \{W_{s_{qi}s_{di}}\}\}. \tag{3.2}$$

However, since $h(S_q^k, S_d^k)$ is not symmetric, the average of $h(S_q^k, S_d^k)$ and $h(S_d^k, S_q^k)$ is taken:

$$H(S_q^k, S_d^k) = (h(S_q^k, S_d^k) + h(S_d^k, S_q^k))/2. \tag{3.3}$$

The total weight of the votes accounting for both local and global similarities is then computed as:

$$W_{S_q S_d} = \sum_{j=1}^{\min(|V_q|,|V_d|)} H(S_q^j, S_d^j). \tag{3.4}$$

After performing a nearest neighbour search around the query signatures, we compute the weights of the votes between the query and the database graphs having at least one signature as the nearest neighbour of the query. We then sort the database graphs based on these weights. In this process, we only add the sufficiently high-support database graphs to the indexing hypothesis. Since a small number of structurally different graphs may also share the same Laplacian spectrum, each graph in the hypothesis should still be verified by some matching algorithm. Despite the

fact that such graphs may exist in the indexing hypothesis, the number of them is very small. In addition, based on Theorem 1, not only do isomorphic graphs share the same signature, non-isomorphic but similar graphs or subgraphs have close signatures in the vector space. The database, therefore, can be pruned without losing structurally similar graphs. The complexity of algorithm is presented in the next section.

3.3.2 Complexity Analysis

Let us first analyse the computational complexity of the signature generation for the database graphs, which is a preprocessing step performed offline. Let n_d denote the maximum number of vertices in a database graph. Given a single graph, the total number of its subgraphs of size k is $O\left(\binom{n_d}{k}\right)$. Assume that there are m graphs in the database, the total number of signatures generated by the framework is bounded by

$$m \times \sum_{k=0}^{n_d} \binom{n_d}{k} = O(m \times 2^{n_d}).$$

Notice, however, that during the signature generation for database graphs, the actual number of subgraphs for which we compute signatures is strictly less than $m \times \sum_{k=0}^{n_d} \binom{n_d}{k}$, since our signature is permutation invariant. In addition, subgraphs of size 2 and 3 are considered too small to represent a significant part of the original image. In the framework, we generate signatures of subgraphs starting from size 4.

On retrieval, we perform an approximate nearest neighbour search for each query signature, using a *balanced-box decomposition tree* (BBD-tree) as introduced by [9]. Given any positive real ϵ, a signature is an $(1+\epsilon)$-approximate nearest neighbor of the query signature s_q if its distance from s_q is within a factor of $(1+\epsilon)$ of the distance to the true nearest neighbor. In general, given an integer $k \geq 1$, $(1+\epsilon)$-approximations to the k nearest neighbours of s_q can be found using a BBD-tree in $O(kd \log n)$ time, where d is the dimension of the search space.

Thus, for a query subgraph of size n_q, the total running time T_{n_q} of a k-nearest neighbour search using is

$$T_{n_q} = O\left(kn_q \log\left(m \times \binom{n_d}{n_q}\right)\right).$$

3.4 Spectral integral variation

The local indexing procedure described above requires individual computation of the Laplacian spectrum for each subgraph. Although for database graphs known a priori this process is performed offline, in applications where new database entries are being inserted frequently, this step plays an important role in the efficiency of the whole system. In this section, we draw on recently developed techniques from the domain of spectral integral variation to avoid the individual computation of the Laplacian spectrum for each subgraph. Specifically, we will study the effect on the Laplacian spectrum when an edge is added into graph $G = (V, E)$. Let $G + e$ be a graph obtained by adding an edge $e = (u, v)$ into G such that $\{u, v\} \in V$ and $e \notin E$. Our interest in this topic is motivated by its ability to identify the changed eigenvalues of graph G, and therefore to generate the Laplacian spectrum of graph $G + e$ without computing them. Before we focus on this topic, let us first reconsider Theorem 1, which shows that when an edge is added into the graph, none of its Laplacian eigenvalues can decrease, while the trace of the Laplacian matrix increases by 2. This important observation implies that given the Laplacian spectrum of G, one can estimate the ranges of eigenvalues for $G+e$. The concept of spectral integral variation, on the other hand, provides more information.

It is shown in [108] that if an edge is added to a graph and the Laplacian spectrum changes by integer quantities, there can only be two possibilities: either one eigenvalue increases by 2 (and $n-1$ eigenvalues remain fixed) or two eigenvalues increase by 1 (and $n-2$ eigenvalues remain fixed). These two cases are called spectral integral variation in one place and spectral integral variation in two places, respectively. The following lemma characterises these two possible situations.

Lemma 1 *Let $G = (V, E)$ be a graph with $|V| = n$ vertices and $\Gamma_{(G)} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ be its Laplacian spectrum. The spectral integral variation of G by adding an edge $e \notin E$ occurs only in the following two cases:*

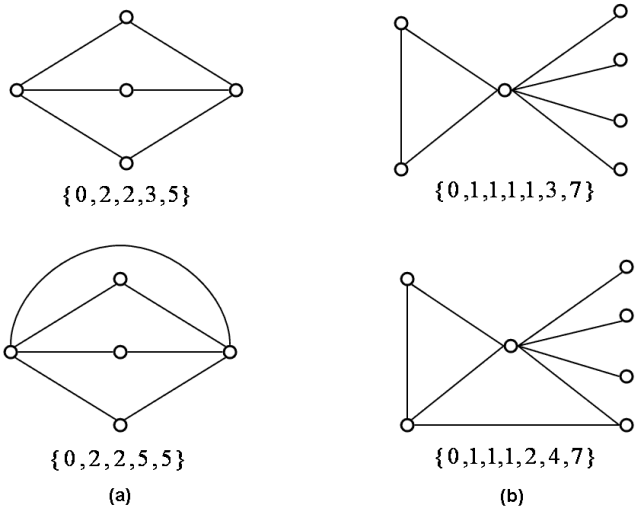
1. *The spectral integral variation occurs in one place, and thus*

$$\Gamma_{(G+e)} = (\Gamma_{(G)} \setminus \lambda_k) \cup \{\lambda_k + 2\}, \text{where } k \in \{1, 2, \ldots, n\}.$$

2. *The spectral integral variation occurs in two places, and thus*

$$\Gamma_{(G+e)} = (\Gamma_{(G)} \setminus \{\lambda_k, \lambda_l\}) \cup \{\lambda_k + 1, \lambda_l + 1\}, \text{where } k, l \in \{1, 2, \ldots, n\} \text{ and } k \neq l.$$

Figure 3.8: Spectral integral variation in one and two places are shown in Part (a) and (b), respectively. Bottom graphs are formed by adding one edge to the graphs shown at the top. The Laplacian spectrum is written below each graph. Observe that while only one eigenvalue increases by 2 in part (a), two eigenvalues increase by 1 in part (b).



The proof for this lemma is given by Yizheng [144]. Part (a) and (b) of figure 3.8 show two graphs where adding an edge results in spectral integral variation in one and two places, respectively.

In our framework, we will identify the changed eigenvalue(s) when spectral integral variation occurs. This, in turn, will allow us to generate the Laplacian spectrum of $G + e$ given that of G. In a somewhat related direction, there exists some work on characterizing graphs stating that when an edge is added, (one of) the changed eigenvalue is the algebraic connectivity [67, 11]. Recall that the algebraic connectivity of a graph is defined as the second smallest Laplacian eigenvalue.

Let $G = (V, E)$ be a graph with $|V| = n$ vertices. For $u \in V$, define $N(u) = \{v \in V : (u, v) \in E\}$. Assume that $e = (u, v)$ is added to $G = (V, E)$ such that $e \notin E$. The following theorems characterize and identify the changed Laplacian eigenvalue(s) when spectral integral variation occurs in one and two places. Theorem 2 appears in [108], while Theorem 3 is shown in [67].

Theorem 2 *$N(u) = N(v)$ if and only if the spectrum of $L(G)$ overlaps the spectrum of $L(G + e)$ in $n - 1$ places. Moreover, the Laplacian eigenvalue of G that increases by 2 is*

given by the degree of vertex u (or, that of vertex v) in this case.

For the following theorem, suppose that the degrees of vertices u and v are shown by d_u and d_v, respectively, and let t denote the number of vertices that are adjacent to both vertex u and vertex v. Without loss of generality, suppose also that $d_u \geq d_v$. Furthermore, let $1_x, 0_x$ denote the $x \times 1$ matrices whose entries are all 1,0, respectively, and let ${1^t}_x, {0^t}_x$ denote their transposes.

Theorem 3 *Let Laplacian matrix L of graph G be given by*

$$L = \begin{bmatrix} d_u & 0 & -{1^t}_x & {0^t}_x & -{1^t}_x & {0^t}_x \\ 0 & d_v & {0^t}_x & -{1^t}_x & -{1^t}_x & {0^t}_x \\ -1_x & 0_x & L_{11} & L_{12} & L_{13} & L_{14} \\ 0_x & -1_x & L_{21} & L_{22} & L_{23} & L_{24} \\ -1_x & -1_x & L_{31} & L_{32} & L_{33} & L_{34} \\ 0_x & 0_x & L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix},$$

where the blocks $l_{11}, l_{33}, l_{33}, L_{44}$ are of sizes $d_u - t, d_v - 1, t$, and $n - 2 - d_u - d_v + t$, respectively. Spectral integral variation occurs in two places if and only if the following conditions hold:

$$L_{11}1_x - L_{12}1_x = (d_v + 1)1_x, \tag{3.5}$$

$$L_{21}1_x - L_{22}1_x = -(d_u + 1)1_x, \tag{3.6}$$

$$L_{31}1_x - L_{32}1_x = -(d_u - d_v)1_x, \tag{3.7}$$

$$L_{41}1_x - L_{42}1_x = 0. \tag{3.8}$$

In the case that conditions (3.5)-(3.8) are satisfied, then the two eigenvalues of L that increase by 1 are

$$\lambda_1 = \frac{d_u + d_v + 1 - \sqrt{(d_u + d_v + 1)2 - 4(d_u d_v + t)}}{2} \tag{3.9}$$

and

$$\lambda_2 = \frac{d_u + d_v + 1 + \sqrt{(d_u + d_v + 1)2 - 4(d_u d_v + t)}}{2}. \tag{3.10}$$

After identifying the changed Laplacian eigenvalues when spectral integral variation occurs, we perform the following process for each given database graph offline. Suppose that the minimum number of edges in a subgraph for which we compute the signature is k. Given a database graph $G = (V, E)$, we first create its subgraph $\hat{G}$ with $|V|$ vertices and k edges and compute its Laplacian eigenvalues. Since the second smallest Laplacian eigenvalue is positive if and only if the graph is connected and the multiplicity of zero as a Laplacian eigenvalue reflects the number of connected components, only the positive eigenvalues of $\hat{G}$ are used as the signature. Next, when we add an edge to $\hat{G}$, we check whether spectral integral variation occurs and if so, we generate the eigenvalues using the theorems given above. We then repeat this process and consider all distinct subgraphs until the whole graph is constructed. At run time, we also apply the same procedure to construct the signatures for query graphs. The algorithm is shown in figure 3.1. Although the above formulation enables us to identify the changed Laplacian eigenvalues when they are increased by integer quantities, our empirical results show that it speeds up the signature generation step for database with 1440 graphs known a priori by 6.47%. We will present our report on this part in Section 3.5.

In retrospect, our encoding of a graph's structure captures its local topology, thus allowing for its use in the case of occlusion and segmentation errors. Furthermore, the signature of a graph is invariant under the reordering of its vertices. This, in turn, allows us to compare the signatures of a large number of graphs without solving the computationally expensive correspondence problem between their vertices. Since we generate signatures of each subgraph to account for the local topology, there is a high computational complexity associated with the generation of the signatures for the database graphs, as shown in the previous section. This complexity, however, can be reduced by considering subgraphs starting from some predefined size and in practice it is further lowered by employing the concept of spectral integral variation.

Algorithm 3.1 Generate spectra
Input: $G = (V, E)$, $\hat{G} = (V, \hat{E})$, and $\Gamma_{(\hat{G})}$
Output: The Laplacian spectrum $\Gamma_{(\hat{G})}$ for every distinct subgraph $\hat{G}$ of G

1: **if** $\hat{E} == E$ **then**
2: terminate
3: **for** every distinct subgraph $\hat{G}$ of G, where $\hat{E} = \hat{E} \cup \{e : e \in E, e \notin E'\}$ **do**
4: $\hat{E} = \hat{E} \cup \{e : e \in E, e \notin E'\}$
5: **if** spectral integral variation occurs in one place **then**
6: $\Gamma_{(\hat{G})}$ = generate $\Gamma_{(\hat{G})}$ according to Theorem 2
7: **else if** spectral integral variation occurs in two places **then**
8: $\Gamma_{(\hat{G})}$ = generate $\Gamma_{(\hat{G})}$ according to Theorem 3
9: **else**
10: $\Gamma_{(\hat{G})}$ = compute $\Gamma_{(\hat{G})}$
11: call Generate Spectra ($G = (V, E)$, $\hat{G} = (V, \hat{E})$, $\Gamma_{(\hat{G})}$)

3.5 Experimental results

In section 3.2, we have shown how different kinds of objects can be represented by graphs and thus form input to the proposed framework. We now present experimental results for each of them.

3.5.1 Music retrieval

A large number of Dutch folk songs is preserved in 'Onder de groene linde'[38]. This collection consists of more than 7300 songs recorded on tape. The songs in the collection have been transferred orally for generations, often changing over time and location. This resulted in the existence of many versions of the same songs, often displaying variations. A large part of these melodies and songs has recently been transcribed to music notation, documented and annotated in great detail.

We experimented on a subset of this resource that consists of 141 songs, of which we used the first phrase. These songs have been classified in 18 classes or *melody groups*, that relate to the concept of *melody norms*. At the Meertens Institute [6] (a research institute for Dutch language and culture) the concept of *melody norm*, equivalent with 'tune family' and 'Melodietyp', is used to group historically or 'genetically' related, orally transmitted melodies. Therefore, there exist some examples of melodies with

[6] www.meertens.knaw.nl

the same melody norm that are somewhat different from a musical similarity point of view as the process of oral transmission can change melodies beyond immediate recognition.

Because the contents of folk song collections such as OGL are highly fragmented, it is impossible to trace back the history of melodies and to find all variants that are derived from a common 'ancestor' melody. What can be done, is to find related groups of melodies within the collection, based on both melodic similarity and available meta data, and link them to melody norms. A search engine would speed up this process of relating melodies considerably. As a ground truth in our experiments, we used a classification of the melodies into *melody groups*, that serve as candidates for the melody norms to be assigned in a later stage.

For all the melodies in our test corpus, a graph has been constructed as described in section 3.2. It must be noted that we didn't generate any subgraphs for the melody graphs. Consequently, the spectral integral variation module has not been used in these experiments. For melody graphs it would be semantically incorrect to generate subgraphs directly from the main graph, because it would create a structure that doesn't contain a path that actually recreates part of the melody. To support partial matching in this domain, it would be more sensible to slide a window over the melody and create graphs for parts of the melody, but we didn't perform these experiments.

We evaluated retrieval performance with these graphs using both the adjacency and the Laplacian spectra. The results are summarised in table 3.1. For both experiments, we computed three simple retrieval statistics: nearest neighbour, first and second tiers, each averaged over all possible queries.

Although these performance figures show in general the efficacy of the method, there are some interesting cases in particular we would like to point out here. In figure 3.9 there is a special case of an 'almost false' positive: for query OGL19205 (belonging to "Heer Halewijn - 3^{rd} version"), the nearest neighbour is OGL19107, that belongs to the group "Heer Halewijn - 4^{th} version". However, the nearest neighbour is somehow related to the query; coincidentally they share the same graph representation, as is shown in figure 3.10. This example shows how two melodies can be identical from the interval connectivity point of view but can also be perceptually quite different. Ironically, they both belong to the tune family 'Heer Halewijn', albeit to different versions.

The second example (figure 3.11) shows the nearest neighbour for another query song, OGL19406. Both examples may suggest also that in the case of folk songs

CRITERIA	NN	1^{st} tier	2^{nd} tier
LAPLACIAN	66%	44%	63%
ADJACENCY	58%	28%	48%
OPTI3	40%	39%	56%
EMD	64%	33%	50%
PTD	64%	30%	46%

Table 3.1: Nearest neighbour (NN), first tier and second tier results on the *Onder de groene linde* collection, computed using Laplacian spectra (L) Adjacency spectra (A) of the graphs. Performance values are averages over all possible queries. The results are compared to the methods Opti3, EMD and PTD.

people tend to remember more the interval connectivity than the actual intervals of the melody.

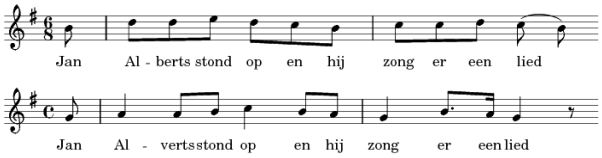


Figure 3.9: Example of false positive for the query song "Heer Halewijn" (3^{rd} version) OGL19205 (top chart) with its nearest neighbour, OGL19107 (bottom chart), instance of "Heer Halewijn" (4^{th} version).

Using the same test corpus and performance measures we compared our method to the optimal distance measure that was established by Müellensiefen and Frieler [83]. These results are also presented in Table 3.1, under the name Opti3. This distance measure is a weighted combination of three distance measures, each working on different feature sets. These measures are *harmcore* (using harmonic correlation), *rhythfuzz* (using fuzzified rhythm values) and *ngrukkon* (taking into account characteristic motives). This combined distance measure was established empirically out of 50 distance measure building blocks, by searching for a weighted combination whose performance best reflected the results of an extensive human listening experiment. Consequently, this method has been fitted to the data set at hand, probably explaining why the results are not optimal in our experiment. We also compared our method to the Earth Mover's Distance (EMD) an the Proportional Transportation Distance (PTD). These distance measures take two weighted point sets as input, and measure the minimum amount of work needed to transform one into the other

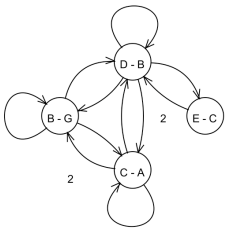


Figure 3.10: Graph representation of the folk songs OGL19205 and OGL19107 (see figure 1). The two letters in each circle represent the pitch classes respectively in the first and in the second song.

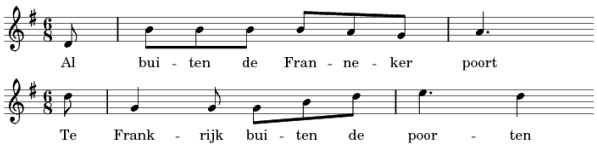


Figure 3.11: Example of true positive for the query song "In Frankrijk buiten de poorten" (2^{nd} version) OGL19406 (top chart) with its nearest neighbour, OGL41709 (bottom chart).

by moving weight. However, since our method only takes into account the global melodic structure, we projected the weighted points on the pitch axis prior to computing the transportation distances. Our framework, that uses a spectral representation of the intervallic structure clearly outperforms these methods.

3.5.2 Trademark retrieval based on layout

In this section we evaluate our framework in the context of a trademark retrieval experiment. We use a set of 450 trademark images from the UK PTO dataset used in the Artisan project [41]. Figure 3.12 shows some trademark images used in the experiments.

To check our segmentation and automatic graph construction procedures, the graph representation process was also performed manually in our experimental setup. Specifically, we manually selected shapes for each input trademark image, created a vertex for each shape, and connected two vertices by an edge if the layout of the corresponding shapes should be encoded in the graph based on the human perception. As a result, two datasets have been generated: one with manually constructed graphs and one with automatically constructed graphs. The performance of the proposed

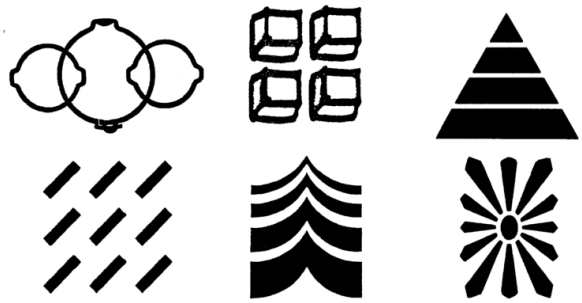


Figure 3.12: Some trademark images used in the experiments.

indexing algorithm was evaluated for both datasets.

In the experiments, the trademarks in our datasets are classified based on their layouts, not on the shapes they consist of. This classification was used to measure the retrieval performance. Since trademark retrieval and similarity are complex issues involving specific knowledge of perception and trademark logic, we presented our classification to a group of experts at Aktor Knowledge Technology, Antwerp, who examine trademark similarity in commercial surroundings on a daily basis. It was only after their concise inspection of our dataset and classification, that we could be sure conducting our experiments and measuring performance are in correspondence with the real trademark similarities.

According to the results (see also table 3.2), in 98.4% and 89.1% of the cases, the most similar database graph belongs to the correct layout class for manually constructed and automatically constructed datasets, respectively (nearest-neighbour performance). In addition, the worst position of the first correct graph is 5 for manual graphs, while this number is 9 for automatic graphs. These numbers show that 98% of the datasets can be pruned by the indexing mechanism to determine the correct layout class for a query. In the second experiment, the system's performance was evaluated by computing the total number of retrieved images that is necessary to retrieve the entire query class (maximum minimal scope). Our results show that the first 71 of the candidate return set always contains all the graphs belonging to the query class for manual graphs; this number is 80 for automatic graphs. This indicates that for this

METHOD	MANUAL GRAPHS	AUTOMATIC GRAPHS
NN	98.4%	89.1%
NN worst	5	9
FN worst	71	80
1^{st} tier	81.2%	86.3%
2^{nd} tier	98.1%	91.7%

Table 3.2: Trademark retrieval results for using the manually created graphs and the automatically created graphs. Criteria are nearest neighbour performance (NN), overall worst position of NN, overall worst position of furthest neighbour (FN), first tier and second tier.

task our framework prunes more than 84% and 82% of the manual and automatic graph datasets, respectively. In other words, the recall in each dataset is 100% if the scope is set to the first 16% and 18% of the sorted candidate models for manual and automatic graphs respectively. We also computed how many of the models in the query's class appear within the top $K-1$ matches, where K is the size of the query class (first tier). This number was 91.2% for manual graphs and 86.3% for automatic graphs. Repeating the same experiment but considering the top $2 \times K - 1$ matches (second tier) covers 98.1% and 91.7% members of the layout classes for manual and automatic graph datasets, respectively.

In figure 3.13, we have presented the matching results for a small subset of trademark images whose graphs were generated automatically using our approach. The first column of each row represents the query image; the remaining elements of each row show the top 10 closest database trademarks retrieved by our indexing algorithm. Squares are drawn around the wrong matches. In all but once case (row 5) the closest trademark image belongs to the same layout class as the query. Although the closest match for the query in row 5 was classified as a mismatch, one may notice that the query consists of three sets of small squares on top of each other and each set has the same layout as the mismatch. As another example, consider the query in row 8 of the left-right class and its first mismatch of the triangle class. Notice that three small triangles in the mismatch have the same layout as the query. Overall, rather than focusing on the mismatches that occur because of the result of a partial match, we observed that the wrong selections happen mainly due to the poor segmentation of the trademark. If different shapes in a trademark are connected, for instance, our segmentation algorithm detects them as one shape. Thus, the layout within these shapes are not encoded in the graphs.

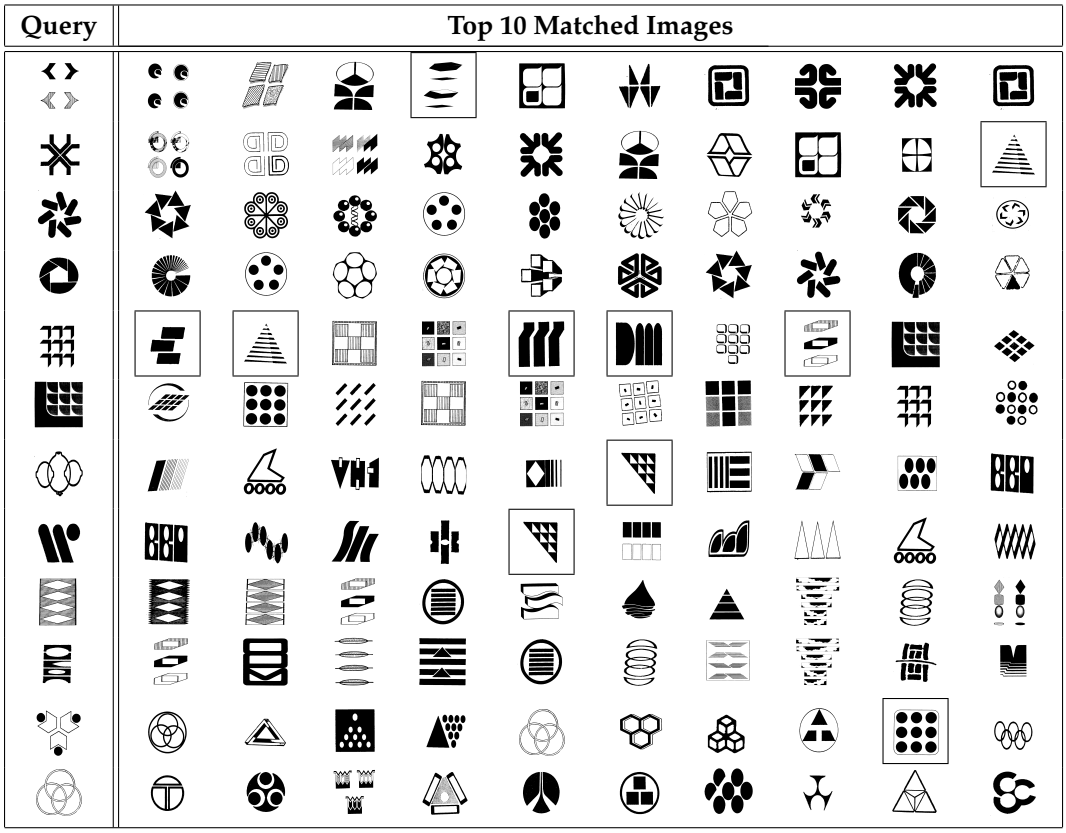
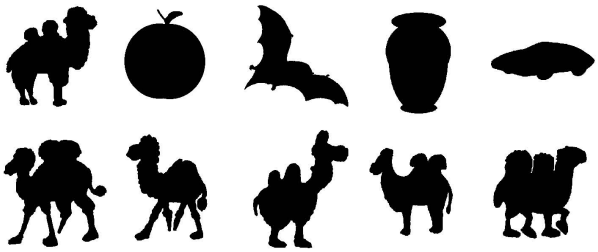


Figure 3.13: Top matched models are sorted by the similarity to the query.

Figure 3.14: Sample images of the MPEG-7 database are shown in the top row. The bottom row represents a set of different shapes of a particular class from the database.



3.5.3 2D and 3D shape retrieval

We used the MPEG-7 dataset CE-Shape-1 part B in our 2D shape retrieval experiments. This dataset consists of 70 classes and 20 shapes, silhouette images per class. The top of figure 3.14 shows a few sample classes, while the bottom of the figure presents different shapes taken from a particular class. A shock graph was generated for each image.

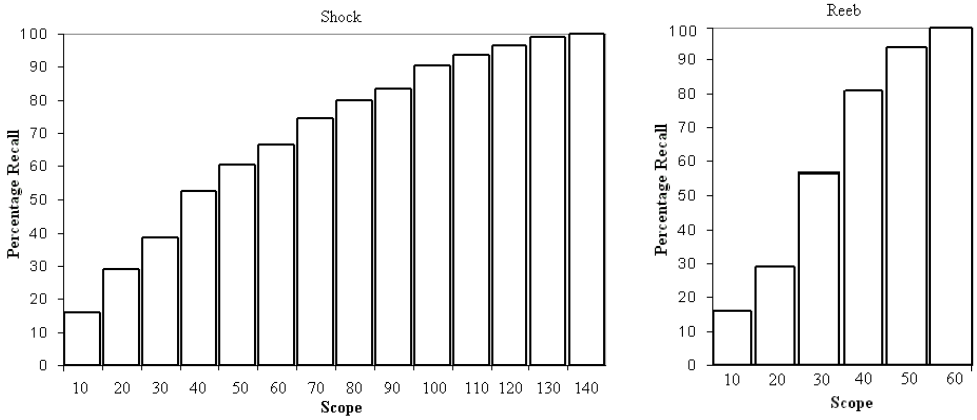
The 3D models for which Reeb graphs have been constructed, come from the McGill 3D Shape Benchmark [146]. It consists of 420 objects classified in 19 classes. Figure 3.15 shows representative views of objects from the database.

For each query, the database graphs are ranked in decreasing order of vote weights in these experiments. Here, we consider the top highest-weight candidates. We say that our indexing system is effective, if at least one graph belonging to the same class as the query is among such candidates. A qualitative measure, therefore, should be based on the smallest size of the candidate list containing one image from the query class. According to the results of this experiment, in 37.3% of the cases, the highest-weight database graph belongs to the correct shape class for shock graphs and this ratio is 29.6% for Reeb graphs. Moreover, the average position of the closest matching graph among the highest-weight candidates is 7.4 and 4.3 for shock and Reeb graphs, respectively. In addition, the worst position of the closest matching graph is 12 for shock graphs, while this number is 9 for Reeb graphs. These results present that to determine the correct class of the query, more than 99% and 97% of shock and Reeb graph datasets can be pruned by our indexing mechanism.

While the previous evaluation method is suitable for classification tasks, in some retrieval applications, however, it is a prerequisite to retrieve all images from the

Figure 3.15: Views of sample objects from the McGill 3D Shape Benchmark.

Figure 3.16: Percentage recall values for various top ranked highest-weight candidate graphs for shock graphs of MPEG-7 and Reeb graphs of McGill datasets.



database that belong to the query class. In the second experiment, the system's performance is evaluated by computing the total number of retrieved images that is necessary to retrieve the entire query class (maximum minimal scope). Our results show that the first 134 of the candidate return set always contains all the graphs belonging to the query class for shock graphs; this number is 54 for Reeb graphs. This indicates that for this task our framework prunes more than 90% and 87% of the shock and Reeb graph datasets, respectively. In other words, the recall in each dataset is 100% if the scope is set to the first 10% and 13% of the sorted candidate models for shock and Reeb graphs respectively. In figure 3.16, we show percentage recall values for various scopes for shock and Reeb graph datasets.

The experimental results presented above clearly show the efficiency of the proposed indexing framework using different graph-based object representations. We now compare the performance of our method to other indexing frameworks on the same datasets. For this purpose, we first compare our results to that of an indexing system in which the graph signatures are generated using the eigenvalues for adjacency matrices. The experiments are identical to the ones described above. The results along with our scores are shown in Table 3.3, and reveal that our indexing framework outperforms the indexing system with adjacency eigenvalues in all criteria mentioned above. These results confirm that graph characterisations through Laplacian matrices are more powerful and more informative than that of adjacency matrices. Representing graphs by Laplacian eigenvalues, therefore, is more effective than that by adjacency eigenvalues. Additionally, we also compare our indexing framework to the one presented in [102]. This indexing algorithm was used in [146] on a subset of the McGill dataset with the object parts represented by directed acyclic graphs (DAG) through medial surfaces [103]. The subset of the McGill dataset used in this experiment includes a total of 320 exemplars taken from several object classes (hands, humans, teddy bears, glasses, pliers, tables, chairs, cups, airplanes, birds, dolphins, dinosaurs, four-legged animals, and fish). To be consistent with the test in [146], we also merge the categories "four-legged" and "dinosaurs" into a broader class, "four-limbs". The results reported in [146] indicate that on average 70% of the models that are in the same class as the query are in the top 80 (25 % of 320). Moreover, for 9 out of these 13 object classes, all instances of the query are in the top 80. Our results, on the other hand, show that 100% of the query classes are always in the top 48 (15 % of 320). The improvement clearly demonstrates the better efficacy obtained by the proposed indexing framework. We believe that the improvement is due to 1) the more powerful representation of Laplacian matrices, 2) the more effective signature construction by our algorithm, i.e., low loss of uniqueness in the signature, and, 3) the better way of encoding local topology.

CRITERIA	HW(%)	AP	WP	MMS	PC(%)	PI(%)
SHOCK GRAPHS - LAPLACIAN	37.3%	7.4	12	137	99%	90%
REEB GRAPHS - LAPLACIAN	29.6%	4.3	9	54	97%	87%
SHOCK GRAPHS - ADJACENCY	18.02%	19.1	23	257	97%	82%
REEB GRAPHS - ADJACENCY	17.3%	10.2	22	92	94%	78%

Table 3.3: Results for indexing system constructed using eigenvalues for Laplacian and Adjacency matrices. HW: Percentage of highest-weighted graph belonging to the same class as the query, AP: Average position of the closest matching graph from the query class, WP: Worst position of the closest matching graph from the query class, MSS: Maximum minimal scope, PC: Percentage of database that can be pruned to determine the right query class, PI: Percentage of database that can be pruned to retrieve all instances of the query.

Our next set of experiments deals with measuring the time efficiency of the indexing framework when spectral integral variation is used during the signature generation for database graphs. Although these signatures are computed offline, for dynamic datasets where a new graph is likely to be added later, the time we spent in this process plays an important role. For each of the datasets, we generate graph signatures with and without spectral integral variation and measure the time taken in each case. We should note here that involving spectral integral variation in the framework does not change the effectiveness of the indexing system. According to the results, we observe that 47.3 and 115.8 minutes were spent to generate all subgraphs to be represented in the vector space for Reeb and Shock graph datasets on an Intel(R) Pentium(R) 4 CPU 3.00GHz computer. After involving spectral integral variation in the framework, the time we spent in this process decreases to 45.5 and 108.3 minutes, respectively. These results indicate that 1.8% and 6.47% time improvements are gained with spectral integral variation for these two datasets. One would expect that as the size of the database increases, the framework with spectral integral variation would yield an improved time efficiency. While the database size is an important factor, the number of Laplacian integral graphs (graphs whose Laplacian spectrum consists entirely of integers) in the database also effects the overall efficiency of the system using spectral integral variation. Balińska et al. [10] present an important survey on integral (graphs whose adjacency spectrum consists entirely of integers) and Laplacian integral graphs. The authors observe that such graphs can be found in all classes of graphs and among graphs of all orders.

Finally, to evaluate the fitness of our approach for dealing with occlusion in images, we generated 14 occluded scenes, each with two images selected from different classes in the MPEG-7 dataset. In each scene, one shape occludes the other to a

certain extent. The percentage of the occlusion varies from 2% to 16%, with on average 6.0%. Each of these 14 new scenes was used as a query against the complete database. We define our indexing schema to be effective if one of the shapes of the query appears in the highest vote-weight candidates. The results of this experiment is presented in figure 3.17, where the left column shows the occluded query images and the top ten candidates sorted by weight from left to right appear in each row. The average position of the closest shape belonging to the class of either of the query shapes was recorded as 1.7 in this experiment. We should point out that our signatures represent topological structures. Thus, images from different classes but with similar topology may be assigned high weights. To give an example, consider the top row, where the query consists of occluded shapes of a spoon and pencil. While neither a spoon nor a pencil is similar to a butterfly, the current combination of them in the query becomes similar to the butterfly retrieved as the highest rank. Shapes belonging to different classes, therefore, may be ranked high in the candidate list.

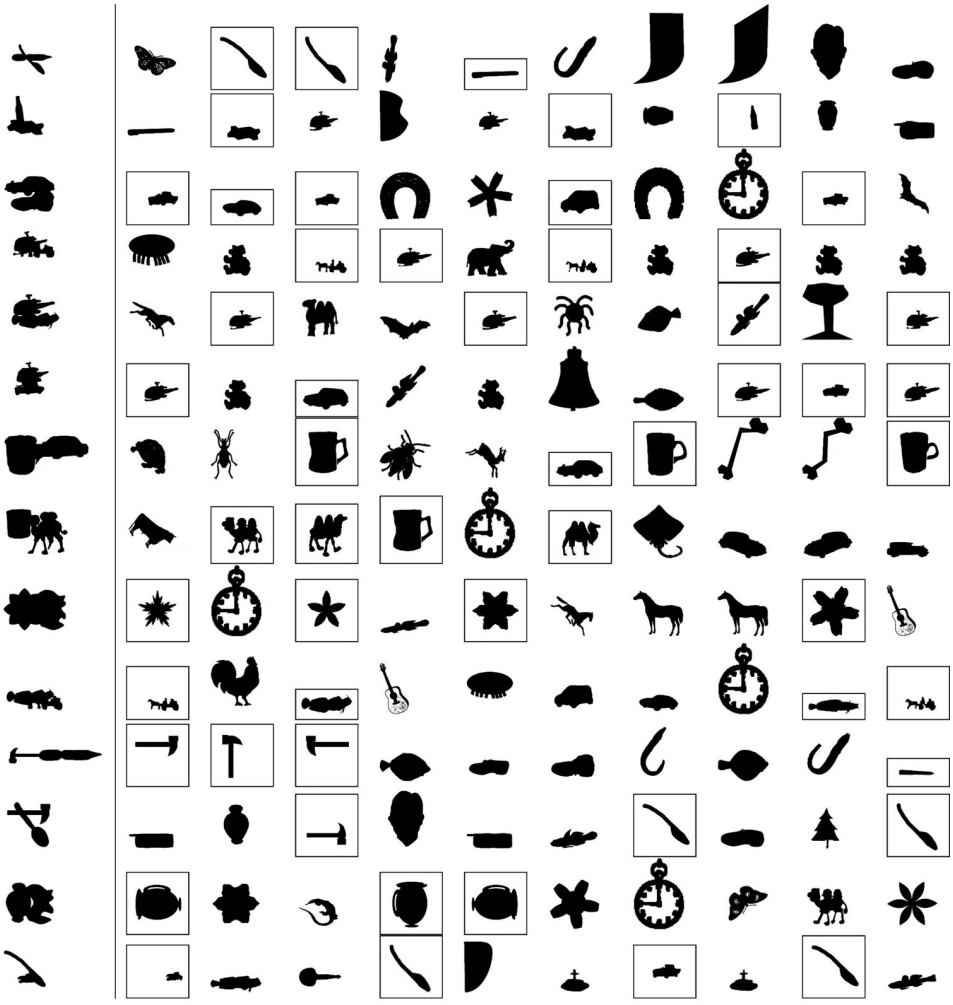
3.6 Concluding remarks

In this chapter, we have proposed a novel, graph-based indexing method using the eigenvalue characterisation of Laplacian matrices. The sorted eigenvalues of the Laplacian matrix of a graph $G = (V, E)$ become the components of an $O(|V|)$-dimensional vector. A nearest neighbour search around this vector returns graphs that are similar to G. This implies that no graph isomorphism is required; our method retrieves those graphs that are similar in terms of their topology. To account for partial similarities, we create signatures for subgraphs of G. We draw from recently-developed techniques in the field of spectral integral variations to overcome the problem of computing the Laplacian spectrum for every subgraph individually.

By using the Laplacian spectrum as a signature, we capture the graph topology to a large extent. The signature of a graph is invariant under the reordering of its vertices. This, in turn, allows us to compare the signatures of a large number of graphs without solving the computationally expensive correspondence problem between their vertices. Although determining graphs that can uniquely be defined by their graph spectra is a difficult problem [33], we showed in this chapter that representing graphs by their Laplacian spectra is more discriminating than that by adjacency spectra.

Our framework can index any multimedia database where objects can be represented as graphs. We have successfully evaluated the approach on several databases using four different graph-based object representations. Moreover, the approach

Figure 3.17: Results of occlusion experiments. The leftmost column shows the occluded query scenes for each row, while the top ten ranked candidate models are shown on the right, in the decreasing order of weight. An image inside a box indicates that it belongs to the class of one of the query shapes. Images belonging to different classes but are topologically similar to the query are ranked high in the candidate list.



1

compares favourably to a leading indexing algorithm. We also demonstrated the robustness of the proposed framework on a set of occlusion experiments.

Chapter 4

Fiedler vectors of Hermitian matrices for 3D object retrieval

Relevant publications:

R.H. van Leuken, O. Symonova, R.C. Veltkamp, R. de Amicis
Complex Fiedler vectors for shape retrieval
In Proc. of Structural and Syntactic Pattern Recognition (SSPR), pp 167-176 ,2008

4.1 Introduction

In the previous chapter, we have derived index signatures that are reflecting the topology of graphs that describe the objects. Although we have produced good retrieval results in several experimental setups, we have also seen some difficult cases in which dissimilar objects end up having similar or identical signatures. One reason for this is the existence of cospectrality in our graphs. Non-isomorphic graphs can have the same Laplacian spectrum, although it is much rarer than with the adjacency spectrum, as we have argued. Another reason for unwanted identical signatures, is simply that in some cases the graphs have similar topology, whilst describing different objects. We have given examples of this case in the context of music retrieval (two folk songs having the same intervallic structure) and 2D shape retrieval (different semantics, yet a similar shock graph). For Reeb graphs of 3D models we can observe similar behaviour. When the segmentation is carried out in such a way that small details are not revealed, it is for instance difficult to distinguish between a human and a Teddy bear based on merely the topology, see figure 4.1.

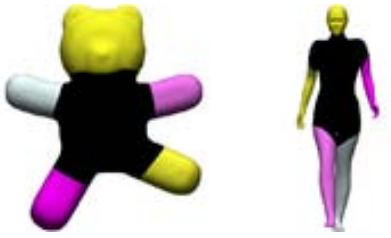


Figure 4.1: Teddy bear and human: similar topology of the graph in which a node represents a body part

The above examples call for a more discriminative representation, an indexing signature that captures more object properties than the topology of a graph representation. In this chapter, we will extend the graph representation with additional object measurements. We will focus on 3D shape retrieval, because it is a well suited domain to experiment with shape properties.

4.1.1 Contributions

First, we propose a new Reeb graph based shape descriptor that encodes both topological and geometrical features of the 3D model. Second, we propose to represent this descriptor by the complex-valued Fiedler vector of a Hermitian property matrix.

This specific eigenvector has not been used for indexing purposes before. The Hermitian property matrix is constructed to contain all the features that are stored in the graph, while mimicking the Laplacian matrix from a spectral point of view. Third, the calculated Fiedler vector is reused to partition the graph into non-overlapping, meaningful subgraphs that are used for partial similarity and to overcome the problem of graphs of different size. We show that this enriched representation has the ability to outperform adjacency matrix and Laplacian matrix based methods [123].

4.2 Segmentation and shape analysis

In this chapter we use an Extended Reeb Graph (ERG) to represent a 3D model. Depending on the function which is used to construct the graph, it can possess the properties of being invariant to the position of a model in space as well as to posture variation. One of the most important advantages of the ERG based descriptor, is that it represents the topological structure of a 3D model which can be additionally enriched with geometrical shape characteristics. As such, the ERG can give a complete shape description. We now present the enriched ERG construction in detail.

First, we define a mapping function f which is used to represent the manifold of a model; the results of the current work are obtained using the integral geodesic distance function [57]. The values of the mapping function are computed for each vertex of the triangular mesh of a model.

Second, we specify N levelsets $f_i^{-1} = \{x | f(x) = f_{min} + i\frac{f_{max}-f_{min}}{N+1}\}$ for $i = 1..N$, and the intervals of the mapping function $\bigcup_{i=0}^{N}[f_i, f_{i+1}]$. These intervals define the decomposition of the model into connected components (see figure 4.2*a*). All the components can be divided into three groups, having one, two or more boundaries. In this context, a boundary is a levelset f_i^{-1} shared by two adjacent components. The components with one boundary represent minimum or maximum regions, components with two boundaries are regular regions, and components with more than two boundaries correspond to saddle regions of the mapping function [18]. We call the components with one and two boundaries simple because their shape can be further characterised by several geometric features [82, 111].

Third, we merge the adjacent simple components into one segment. Figure 4.2*b* illustrates the result of merging. Finally, we construct the ERG. Each component obtained after the previous segmentation and merging phases is represented by a node in the ERG. There is an edge between two nodes if the corresponding components share a boundary. Hereafter, we refer to the nodes of the ERG representing simple and sad-

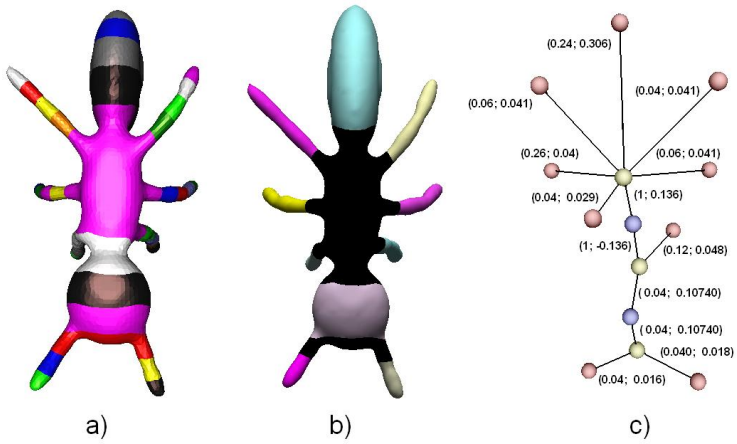


Figure 4.2: ERG construction. (a) Shape segmentation. (b) Merging simple adjacent components. (c) ERG with edge attributes: shape index and segment weight.

dle components as simple and saddle nodes correspondingly. If two saddle nodes are adjacent we insert an intermediate node between them. This node represents a body part of a model. Figure 4.2*c* illustrates the ERG with an intermediate node between two saddle nodes. Simple and intermediate nodes are connected only to saddle nodes and vice versa. To enrich the topological structure of the ERG with geometric characteristics, we use a recently proposed, detailed shape analysis for the components with one and two boundaries [111]. We now give a brief overview of this shape classification.

We call the components with one and two boundaries cone- and cylinder-like respectively. For these components, the barycenters of the boundaries define two unique reference points. For the cone-like components the second reference point is the tip point, where the mapping function reaches the maximum or minimum value. The two reference points define a line which we use as the main axis for further shape analysis. By intersecting a component with equally spaced planes perpendicular to the main axis, we obtain a set of contours. We analyse two shape criteria of the component while tracing the changes of these contours.

The first shape criterion is the change in area of the cross sections. We define seven different shape classes according to this criterion: constant cross areas, smooth/sharp increasing/decreasing, multiple cross area changes and flat components. If the area of the cross sections is constant to the extent of a predefined threshold, the compo-

nent belongs to the first type of the proposed classification. If the cross area changes (increases or decreases) we calculate the apex angle of the cones induced from each of the cross sections. The area of the cross sections is defined to change smoothly if the apex angle of each cone is less than $90°$, and sharply otherwise. The apex angle of a cone is less than $90°$ if $\frac{r_i}{h_i} < 1$, where r_i is the average radius of the i-th cross section and h_i is the height of the cone. If the alteration of cross area changes its behaviour, e.g. increasing and then decreasing, we define the component as having multiple cross area changes. If the main axis has almost zero length, the component is classified as flat. Figure 4.3 illustrates these classes by prototype.

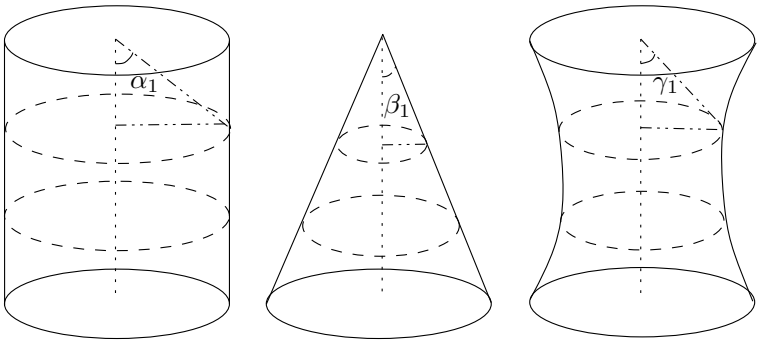


Figure 4.3: Examples of the first shape criterion: change in cross section area. From left to right: constant, single change (smooth/sharp increasing, smooth/sharp decreasing, multiple change

The second shape criterion we define is component bending. According to this criterion, we define three types of shape classification: zero, single and multiple bending. If the main axis passes through all the cross sections, the component is straight (with zero bending). If the axis passes through the cross sections in the vicinity of the reference point but is outside in between then the component has single bending. In the case when the axis is outside two or more sections while passing through the sections located in between, the component has multiple bending. See figure 4.4 for illustrations of this shape criterion.

The proposed shape classification defines 19 different shape types, that can be used for rough shape description and approximation. We reserve an individual shape type for the intermediate nodes that connect two saddle regions, in total we therefore have 20 shape types. The indices that represent the shape types are mapped onto the interval [0.04,1].

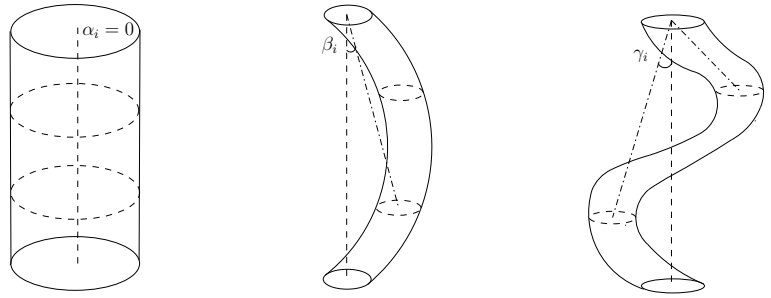


Figure 4.4: Examples of the second shape criterion: segment bending. From left to right: straight, single bending, multiple bending

Figure 4.5: Segmentation of a model without saddle regions and its ERG.

Additionally, we calculate the relative size of each segment. This size is defined as the ratio of the surface area of the segment to the surface area of the whole model

$$\text{Size}_{\text{Segm}} = \frac{\text{Area}_{Segm}}{\text{Area}_{Total}} \tag{4.1}$$

This value is used to adequately scale the influence of the segment on the whole graph matching process. The index of the shape classification together with the segment size can be stored as attributes of the edge that connects the corresponding simple node with a saddle node.

If there is no saddle region revealed after segmentation, there is a minimum area that is directly connected to a maximum area. In this case we insert an intermediate node to the ERG which interconnects two simple nodes. The shape index and segment weight of the maximum and minimum areas are stored as the attributes of the edges incident to the corresponding nodes. Figure 4.5 illustrates the segmentation of a 3D model without saddle regions and its ERG.

4.3 Graph property matrices

After segmentation, each model is represented by a graph as described in section 4.2. To perform efficient computations and similarity evaluations, it is necessary to associate an algebraic structure with each graph. In this section, details are given on our graph property matrix, which is a Hermitian matrix. A more popular and well-studied matrix, however, is the Laplacian matrix. We therefore take a weighted Laplacian as our starting point, and show how this matrix can be extended to a Hermitian matrix to contain more information, without losing the validity of known theorems.

4.3.1 Weighted Laplacian matrices

In chapter 3, we have associated to each graph a Laplacian matrix, defined as $L(G) = D(G) - A(G)$, where $D(G)$ is the diagonal matrix containing node degrees, and $A(G)$ is the adjacency matrix; the entry $A_{i,j}$ is the number of edges that connect i and j. As a result, for all rows in $L(G)$ the sum of the entries is 0. The spectrum of the Laplacian matrix can be used as a signature representation for the graph, and thus for the model that is represented by the graph. As we have seen, this signature representation can be used for efficient retrieval purposes (indexing). One of the main reasons for this is that many graph properties and invariants are reflected by the Laplacian spectrum [78]. Moreover, cospectrality for non-isomorphic graphs tends to be rare [139] and similar Laplacian matrices have similar spectra due to the interlacing theorem for two graphs where one is a slightly modified version of the other [75]. See section 3.3 for more details on these properties.

In the case of a weighted graph, $L_w(G) = D_w(G) - A_w(G)$ can be obtained. Here, $D_w(G)$ is a diagonal matrix containing for each node the sum of the weights of its incident edges. Correspondingly, in the adjacency matrix the entry $A_{i,j}$ represents the weight associated with the edge between nodes i and j. $A_{i,j}$ is 0 if there is no connecting edge between i and j. Therefore, all information about the graph's connectivity is still present in $L_w(G)$, since every non-zero entry indicates the existence of an edge between the corresponding nodes. In order to preserve the useful properties of a normal Laplacian spectrum, every edge weight $w_{a,b}$ should satisfy the following conditions:

$$W_{a,b} = W_{b,a}, \text{ where } a, b \in V, \tag{4.2}$$

$$W_{a,b} \geq 0, \text{ where } a, b \in V, \tag{4.3}$$

$$W_{a,b} \neq 0, \text{ iff } a \text{ and } b \text{ are adjacent in } G. \tag{4.4}$$

Equation (4.2) ensures a symmetric matrix, whereas equation (4.4) ensures that the connectivity of the graph remains unchanged after weighting the edges. It can be observed that using the number of connections between two adjacent nodes as weights, as we have done in chapter 3, obeys these conditions.

Unfortunately, it is not possible to store more information in a Laplacian matrix than the graph's connectivity together with the edge weights. Consequently, a spectral representation using this matrix will suffer in most cases from significant information loss, since other graph characteristics such as node locations (planar graphs, 3D graphs) or additional edge measurements are not encoded.

4.3.2 Hermitian matrices

In order to encode more information about the graph in a matrix, we follow the ideas of [138] and use a Hermitian matrix to store graph characteristics. By imposing several constraints on the elements of the Hermitian matrix, the authors of [138] claim to mimic the spectral behaviour of the Laplacian matrix. However, in order to make a Hermitian matrix to possess all spectral properties of the Laplacian, we add two more constraints which we discuss after a brief theoretical introduction to Hermitian matrices.

A Hermitian matrix H (or self-adjoint matrix) is a square matrix with complex entries that is equal to its own conjugate transpose. In other words, $H_{i,j}$ is equal to the complex conjugate of $H_{j,i}$. Every Hermitian matrix has a real valued spectrum. The corresponding eigenvectors however contain complex entries. By adding several additional constraints to the construction of H, we can mimic the spectral behaviour of a Laplacian matrix.

Therefore, off-diagonal elements of H are complex numbers written in polar form using Euler's formula, defined as:

$$H_{a,b} = -W_{a,b} e^{i y_{a,b}}, \tag{4.5}$$

where each edge has the pair of properties $(W_{a,b}, y_{a,b})$. The first property, $W_{a,b}$, is used as the magnitude of the entry and should satisfy conditions (4.2)-(4.4). The

second property, used as the phase of the complex matrix entry, must satisfy the following conditions:

$$y_{a,b} = -y_{b,a}, \tag{4.6}$$

$$-\pi < y_{a,b} < \pi. \tag{4.7}$$

The first condition (4.6) ensures that H is equal to its own conjugate transposed matrix. By obeying the second constraint (4.7), phase wrapping can be avoided.

The real-valued on-diagonal entries are defined as

$$H_{aa} = \sum_{b \neq a} W_{a,b}. \tag{4.8}$$

In this way, the entries in each row of the matrix sum up to zero. We would like to stress that this is a necessary property to correctly mimic the spectral behaviour of Laplacian matrices, contrary to the Hermitian matrix that is used in [138] (where additional node measurements on the diagonal are allowed). Furthermore, edge weights (magnitudes of the complex entries) should be calculated in such a way that an edge between two nodes can never be weighted 0, for it would destroy the connectivity of the graph.

As was described in section 4.2, an edge in the graph connects a saddle node and a simple node. As a consequence, the calculated shape index and the weight of the simple segment can be stored as the attributes of the edge incident to the corresponding simple node.

Mapping the shape indices to the interval $(0; 1]$ allows their use as magnitudes of complex entries of the Hermitian matrix. The positive shape index of the segment is invariant to edge direction thus satisfying the constraints (4.2)-(4.4).

We choose the values of shape indices based on visual shape similarity as well as on the alteration of one or both criteria used for shape analysis. Table 4.1 illustrates the list of all possible shape types and corresponding indices. As it can be seen from the table, the variation of the cross area criterion has a greater influence on the difference in shape index than the bending criterion. This choice can be explained by the fact that semantically equivalent sub parts of articulated models frequently have different bending properties, e.g. straight and bent arms.

The relative size of a segment (equation 4.1) can be used as the phase value y_{ab} for the entries of the Hermitian matrix. To obey the anti symmetric condition (4.6) we

Cross Area	Bending	Shape Index
constant	zero	0.04
constant	single	0.06
constant	multiple	0.08
alteration	zero	0.12
alteration	single	0.14
alteration	multiple	0.16
smoothly decreasing	zero	0.24
smoothly decreasing	single	0.26
smoothly decreasing	multiple	0.28
smoothly increasing	zero	0.32
smoothly increasing	single	0.34
smoothly increasing	multiple	0.36
flat		0.64
sharply decreasing	zero	0.68
sharply decreasing	single	0.7
sharply decreasing	multiple	0.72
sharply increasing	zero	0.76
sharply increasing	single	0.78
sharply increasing	multiple	0.8
body		1

Table 4.1: Shape indices.

set y_{ab} as:

$$y_{ab} = \begin{cases} \text{Size}_{S_{ab}}, & \text{if } degree(a) > degree(b) \\ -\text{Size}_{S_{ab}}, & \text{if } degree(a) < degree(b). \end{cases}$$

The value of $\text{Size}_{S_{ab}}$ is bounded by the interval $(0; 1]$. Scaling this interval up to $(0; \pi]$ and using the above definition of y_{ab} we satisfy the constraint (4.7) for the phase value of the complex entries of the Hermitian matrix.

4.3.3 Fiedler vectors

The Hermitian matrix for a graph, with edge attributes calculated in this way, mimics spectral behaviour of a Laplacian. More specifically, the spectrum is equal to the spectrum of a weighted Laplacian matrix, for which the edge weights are equal to the magnitudes of the entries in the Hermitian matrix (shape classifications). Unfortunately, in the real valued spectrum of this Hermitian matrix, the influence of the phase (segment weight) is lost. However, the eigenvectors of the matrix contain complex entries, where phase influence is preserved. We therefore propose to base retrieval not on spectra, but on eigenvectors of the Hermitian matrix. Specifically, we propose to use the eigenvector associated to the second smallest eigenvalue as a signature for the graphs. The second smallest Laplacian eigenvalue (i.e., the first non-zero eigenvalue) carries probably the most important information contained in the spectrum of a graph. It is also known as the Fiedler value, its corresponding eigenvector is called the Fiedler vector. The Fiedler vector and the Fiedler value are very well studied concepts, that reflect many graph properties and invariants, for which we refer to an extensive survey [78]. Known applications of these concepts include graph partitioning and optimal labelling problems (e.g. bandwidth problems, min-sum problem, vertex ordering).

Therefore, by first mimicking spectral behaviour of Laplacian matrices through the use of Hermitian matrices, and by extracting their Fiedler vectors, we obtain a strong and representative signature that can be used for retrieval purposes.

4.4 Retrieval through Fiedler vectors

The Fiedler vector of the Hermitian property matrix is used as a signature representation for each model. This vector is of dimension n, where n is the number of nodes in the graph. There exists a correspondence between the graph nodes and the entries of the Fiedler vector. Specifically, the node represented by the first column of the Hermitian matrix corresponds to the first entry in the Fiedler vector. This means that, in order to preserve permutation invariance of the representation, the entries of the Fiedler vector need to be sorted. Therefore, before evaluating the similarity between two n-dimensional Fiedler vectors, they are first sorted lexicographically (first based on the real part, then on the imaginary part) and interlaced in the following manner:

$$\begin{pmatrix} a_k + ib_k \\ a_l + ib_l \\ \vdots \\ a_n + ib_n \end{pmatrix} \rightarrow \begin{pmatrix} a_l \\ b_l \\ a_k \\ b_k \\ \vdots \\ a_n \\ b_n \end{pmatrix},$$

if $a_l < a_k$, or $a_k = a_l$ and $b_l \leq b_k$.

The similarity between two models that are represented by graphs with n nodes, can then be defined as the Euclidean distance between their interlaced and sorted Fiedler vectors. In practice however, models are often represented by graphs of different sizes. Furthermore, partial similarity is not taken into account by this approach. We propose to base retrieval both on complete and partial graphs. The following sections provide details on how to match complete graphs of different size, and how to match them partially using small, meaningful subgraphs.

4.4.1 Retrieval of complete graphs

One of the challenges of retrieval using spectra or eigenvectors, is handling graphs of different sizes. A common solution that has been proposed in the domain of spectral matching is padding with zeros: enlarge the smaller spectrum to the size of the larger spectrum by inserting zeros. This is a semantically sensible approach, since it means that the smaller graph is enlarged by inserting isolated nodes. In the context of a given database, the model with the largest graph of size n needs to be found. All the other graphs are brought up to this size by inserting isolated dummy nodes. When all Fiedler vectors are of the same dimension, a simple range search or nearest neighbour search among these vectors produces a set of similar objects to a query. The distance between two graphs g_1 and g_2 (based on the complete graphs) is then defined as the Euclidean distance

$$d_{\text{full}}(g_1, g_2) = \sqrt{\sum_{i=1}^{2n} (F_1(i) - F_2(i))^2}, \tag{4.9}$$

where F_1 and F_2 are the sorted, interlaced complex Fiedler vectors of graphs g_1 and g_2 respectively.

Although this is semantically correct, it can possibly distort the retrieval results. Suppose two similar graphs are of different size as a consequence of a different level of detail that was revealed during segmentation. When dummy nodes are inserted in the coarsely segmented graph simply to enlarge the dimension of its Fiedler vector, a larger distance to the finer segmented graph is found as a consequence of matching entries related to dummy nodes with entries related to real nodes. Despite this phenomenon, the approach still produces good results in some of the cases. We therefore propose to use it, but in combination with a method for retrieving database models based on parts of the graphs. This method is described in the following section.

4.4.2 Graph decomposition

Evaluating similarity of subgraphs is used to account for partial similarity. It also helps in solving the problem of handling graphs of different sizes, because combining similarity of parts may reduce the influence of a finer segmented region. Only subgraphs of the same size are used for similarity evaluation. The problem that remains, is to find subgraphs that are appropriate for this purpose. In chapter 3, we proposed to extract all possible subgraphs of all possible sizes, and match only subgraphs of the same size. We proposed exploiting spectral integral variation to decrease the computational complexity.

Here we present an alternative to this approach: instead of generating all possible subgraphs, we decompose the original graph into small, non-overlapping meaningful subgraphs, following ideas from [90]. An attractive feature of the proposed partitioning approach is that it reuses valuable information that is already calculated, i.e. the complex Fiedler vector. The partitioning approach is given in detail by algorithm 4.1. The main idea is to select nodes that become the centre node of a subgraph; a node can be a centre node if it has a higher *score* than its neighbours. This score of a node is based on its degree and on its position in the sorted Fiedler vector. It is possible to keep track of the position of a node in the Fiedler vector, since the order in which the nodes form the Hermitian matrix induces a relation between entries of the Fiedler vector and the nodes. A subgraph then consists of a centre node and (some of) its adjacent nodes. This decomposition is given in detail by lines 1-12 of algorithm 4.1.

For an example, see figure 4.6, where a segmented model of a goat is displayed, together with a view of its 3D graph and extracted subgraphs. The first extracted subgraph corresponds to the back (tail and back legs), the second subgraph corresponds to the middle part (body, neck and fore legs) and the third subgraph corresponds to the front part (head, nose and ears).

To be even more robust against small alterations in the graph structure that are caused by segmentation and/or analysis differences, the subgraphs are decomposed even further by lines 13-15 of algorithm 4.1.

Algorithm 4.1 Decompose graph
Input: $G = (V, E)$, Fiedler vector F of G, balancing factors α and β
Output: collection $\mathbb{G}$ with subgraphs of G

1: Sort F in decreasing order$^{(*)}$
2: **for** each node $i \in V$ **do**
3: let rank(i) be the corresponding position of i in the sorted Fiedler vector
4: calculate $score(i) = \alpha \times \text{degree}(i) + \beta/\text{rank}(i)$
5: **while** V, E are not empty **do**
6: **for** each node $i \in V$ **do**
7: **if** $score(i) > score(j)$ for all nodes j that i is connected to **then**
8: let $G' = (V', E')$ be a new subgraph with centre $c(G') = i$, add i to V'
9: **for** all $(i, j) \in E$ **do**
10: add (i, j) to E', remove (i, j) from E
11: add j to V', remove j from V
12: add G' to $\mathbb{G}$
13: **for** every subgraph $G' = (V', E')$ resulting from lines 1-12 **do**
14: **for** s is 2 to $|V'|$ **do**
15: add to $\mathbb{G}$ each subgraph $G'' = (V'', E'')$ of G' of size s where $c(G') \in V''$

(*) the sort function defined on the entries is lexicographical: first based on the real parts, if the real parts are equal, based on the imaginary parts.

To calculate the subgraph-based distance between two graphs g_1 and g_2, all their subgraphs are represented by their Fiedler vectors as described in the beginning of section 4.4. The distance between g_1 and g_2, based on their subgraphs, is then defined as the normalized and weighted sum of all pairwise subgraph distances, where the distance between two subgraphs is calculated only if they are of the same size. For a given subgraph size d, the average is taken of all pairwise distances of subgraphs of this size:

$$d_{\text{sg\_d}} = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} d_{\text{full}}(g_1^i, g_2^j) \tag{4.10}$$

where g_1^i and g_2^j are the $i-th$ and $j-th$ subgraphs of size d of graphs g_1 and g_2, and where n and m are the number of subgraphs of size d that were extracted from g_1 and g_2. For a definition of d_{full}, see equation 4.9.

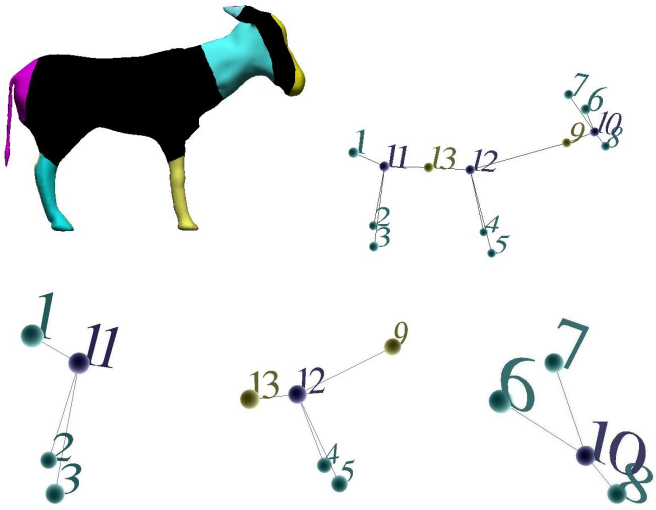


Figure 4.6: Subgraph decomposition of a model of a goat.

A subgraph distance contributes more to the total similarity if the size of the subgraphs is larger. The total distance between graphs g_1 and g_2 based on their subgraphs is defined as

$$d_{\text{sg}}(g_1, g_2) = \frac{1}{\text{MaxS} \times (|g_1| + |g_2|)} \sum_{s=1}^{MaxS} s \times d_{\text{sg\_d}} \tag{4.11}$$

where $|g_1|$ and $|g_2|$ are the number of nodes in g_1 and g_2 respectively, where s is the size of the subgraphs and where MaxS is the smallest maximal subgraph that could be decomposed from either g_1 or g_2. See equation 4.10 for a definition of $d_{\text{sg\_d}}$.

Using this distance function, a ranked list of all the database objects can be produced for a given query. Together with the ranked list based on the comparison of complete graphs (see section 4.4.1), two complementary ranked lists are produced for one query. To provide the user one set of results for a given query, these ranked lists need to be combined into one.

4.4.3 Rank aggregation

The distance functions (4.9) and (4.11) each produce a ranked list of the database for a given query. These ranked lists are usually highly complementary, i.e. correctly retrieved models that occur at the top of one ranked list are often positioned at lower ranks in the other and vice versa. Therefore, in order to combine these ranked lists and benefit maximally from both, it is necessary to emphasise models that occur at the top of the rankings, and damp their influence when they are ranked at lower positions. Furthermore, the range of distances of the two distance functions can be very different. During combination of the two ranked lists, we therefore don't consider distances anymore, but ranks.

Specifically, a score value for each database object is calculated based on its position in the two rankings. This score is then used to re-order the database into a new ranking that reflects both original rankings. The score of a database object i is calculated as $\mathrm{score}(i) = \log(\mathrm{r_{sg}}(i)) + \log(\mathrm{r_{full}}(i))$, where $\mathrm{r_{sg}}(i)$ and $\mathrm{r_{full}}(i)$ are the ranks of object i in the ranked lists according to subgraph matching and complete matching respectively. By taking the logarithm of the ranks, the influence of lower ranks is damped and the top of the rankings is emphasised.

4.5 Experimental results

We implemented our framework in C++, using an implementation of the ALGLIB project (www.alglib.net) to solve the Hermitian eigenproblem. It has been tested on the dataset that was used in the Watertight 3D Models track of the SHape REtrieval Contest 2007 (SHREC '07) [130]. This dataset consists of 400 models, divided over 20 classes of 20 models each. All models were preprocessed using the described segmentation and shape analysis software. After preprocessing, the implemented framework ranks the complete database of 400 models with respect to one query in around 0.2 seconds on a 3.00 Ghz Intel Xeon processor. Each model was used as a query once; the goal is to to see the complete class of the query at the top of the ranking.

However, before running the experiments on the whole dataset, we investigated the ability of the approach to discriminate between topologically similar, but geometrically and semantically different models. To this end, we have chosen the class of human and the class of Teddy bear models. Figure 4.7 shows retrieval results for some sample queries when only this subset of 40 models is considered. As can be seen from these results, the Teddy bears can be separated from the topologically sim-

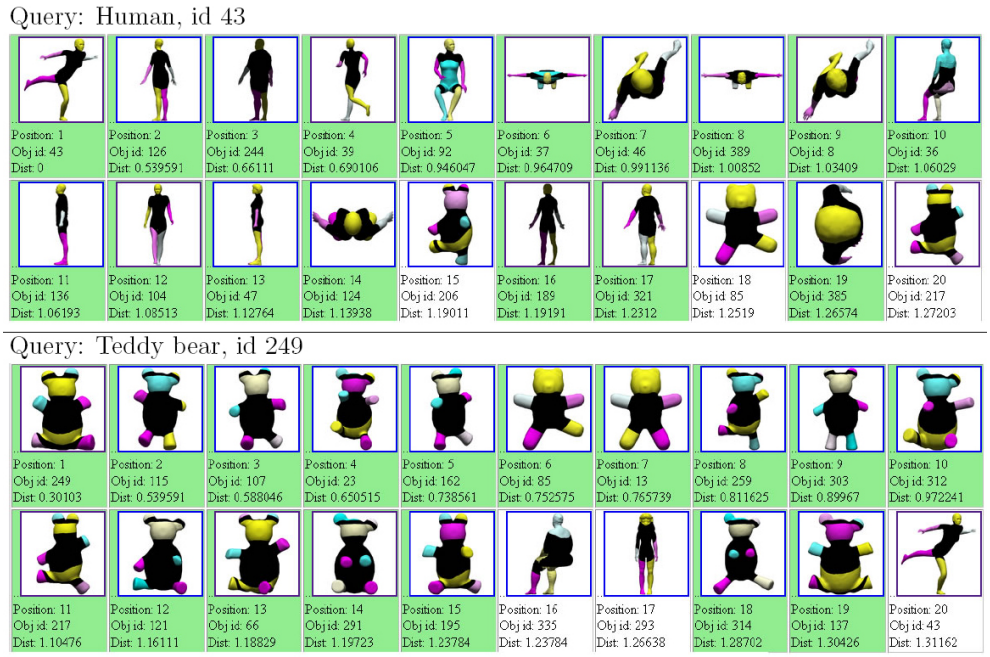


Figure 4.7: Retrieval results when only the subset of human and Teddy bear models is considered: top 20 retrieved items are displayed. Colored border means successful hit.

ilar humans, by means of the encoded shape analysis.

To be able to compare the results to other participants of this contest, the same performance measures are used. These are:

- Precision and recall rates when the scope is set to 20, 40, 60 or 80 items, i.e. first until fourth tiers,
- Average Dynamic Recall $ADR = \frac{1}{20}\sum_{i=1}^{20}\frac{RI(i)}{i}$, where $RI(i)$ is the number of relevant retrieved items in the top i of the ranking. $ADR \in [0,1]$, where 1 is the best retrieval result,
- success rates for the first item (PS1) and second item (PS2),
- Average Ranking AVG of all class members and Last Place Ranking $LPR = 1 - \frac{\text{Rank}-20}{380}$, where Rank is the rank of the last relevant item. $LPR \in [0,1]$, where 1 is the ideal retrieval result.

Table 4.2: Performance comparison of the proposed approach (CFV), using Laplacian matrices (Lapl), using adjacency matrices (Adj) and the approach by Tung et al. Measures P1 to P4 represent precision in the first to fourth tiers, R1 to R4 represent recall in the first to fourth tiers, PS1 represents the success rate for the first item, PS2 represents the success rate for the second item, ADR represents Average Dynamic Recall, AVG represents the Average Rank of all class members and LPR represents the Last Place Ranking.

Measure	CFV	Lapl	Adj	Tung	Measure	CFV	Lapl.	Adj.	Tung
P1	0.42	0.22	0.20	0.71	R1	0.42	0.22	0.20	0.71
P2	0.29	0.16	0.16	0.41	R2	0.57	0.33	0.31	0.83
P3	0.22	0.14	0.14	0.29	R3	0.66	0.43	0.41	0.87
P4	0.18	0.12	0.12	0.23	R4	0.72	0.50	0.48	0.90
PS1	1	1	1	1	ADR	0.58	0.36	0.31	0.86
PS2	0.65	0.33	0.17	0.98	AVG	77	125	132	31
					LPR	0.48	0.25	0.17	0.74

Table 4.2 displays the above performance values for our method (abbreviated as *CFV*, for Complex Fiedler Vectors) and the method that performed best in SHREC 2007, a multi resolution graph matching method by Tung et al. [117]. At this point we would like to emphasise that our method is an indexing method rather than a matching method; all participating methods in SHREC are matching methods encoding a large amount of shape information. This means that with our method, parts of the database can be discarded upon querying time, whereas the matching methods need an exhaustive search through the database to obtain the ranking. Our method does improve with respect to other spectral based indexing methods; we compared it to using the Fiedler vectors of adjacency and Laplacian matrices.

The table shows that precision and recall values for small scopes (first and second tiers) are almost twice as high for the proposed approach compared to using adjacency matrices and Laplacian matrices. In the third and fourth tier, the difference is slightly smaller but still significant. All methods score a success rate of 1 for the first item; this means that each method finds an object from the correct class on top of the ranking. This is usually the query itself. It is therefore interesting to look at the performance for the second item on the ranking: in 65% of the cases this is a true positive when using the proposed approach. For adjacency matrices, Laplacian matrices and Tung's approach the values are 33%, 17% and 98% respectively. The relative performance of the four methods stays more or less consistent according to the other methods, Average Dynamic Recall, Average Rank and Last Place Ranking.

4.6 Concluding remarks

We have presented a new shape retrieval method using complex Fiedler vectors of Hermitian property matrices. In these matrices, the topology of a graph that represents the shape is stored, together with two additional properties per edge. Several constraints are imposed on these properties, such that this Hermitian matrix mimics the well known Laplacian matrix from a spectral point of view. Therefore, known theorems about the important second smallest eigenvalue, and its associated eigenvector, the Fiedler vector, extend to the Hermitian case.

Using a recently proposed shape segmentation and analysis schema, we adopted the framework to the retrieval of watertight 3D models. Extended Reeb Graphs were attributed with shape classification labels and weights related to segment size, obeying the constraints necessary to mimic the Laplacian matrix. The approach was tested on a benchmark of 400 models, that was subject to a recent contest: the watertight models track in the 2007 SHape Retrieval Contest (SHREC). By combining partial and complete retrieval, the framework shows strong performance on this benchmark of 400 watertight 3D models.

Chapter 5

Visual diversification of image search results

Relevant publications:

R.H. van Leuken, L. Garcia, X. Olivares, R. van Zwol
Visual diversification of image search results
In Proc. of the World Wide Web conference, 2009

5.1 Introduction

In the previous chapters, we have investigated techniques for content-based retrieval of multimedia, either by graph based methods or by vector space embeddings of a dissimilarity space. The main goal so far has been to retrieve similar objects for a given query. We obtained strong retrieval performance by maintaining a highly discriminant representation of the objects throughout the different retrieval pipelines that have been described. In this chapter, the goal will be slightly different. We will now extend our interest toward obtaining *diverse* results. Not only should the results be relevant, the top of the ranking should in fact reflect the diversity of the relevant objects that are present in the collection. We will study techniques within the context of image search on the world wide web.

The common modality used for image search on the web is text, used both in indices of large search engines or in a more restricted environment such as social media sites like Flickr. Although not without its flaws, the assumption that a relevant image resides on a web page surrounded by text that matches the query is reasonable. Along the same lines, tags and textual descriptions of photos prove to be powerful ways to describe and retrieve images that are uploaded daily in massive quantities to dedicated sharing sites. The retrieval models deployed on the Web and by these photo sharing sites rely heavily on search paradigms developed within the field Information Retrieval. This way, image retrieval can benefit from years of research experience, and the better this textual metadata captures the content of the image, the better the retrieval performance will be.

On the other hand however, it is commonly acknowledged that a picture has to be seen to fully understand its meaning, significance, beauty, or context, simply because it conveys information that words cannot capture, or at least not in any practical setting. This explains the large number of papers on content-based image retrieval (CBIR) that have been published since 1990, the breathtaking publication rates since 1997 [107], and the continuing interest in the field [34]. Moving on from simple low-level features to more discriminative descriptions, the field has come a long way in narrowing down the semantic gap by using high-level semantics [71]. Unfortunately, CBIR-methods using higher level semantics usually require extensive training, intricate object ontologies or expensive construction of a visual dictionary, and their performance remains unfit for use in large scale online applications such as the aforementioned search engines or websites. Consequently, retrieval models operating in the textual metadata domain are deployed here.

In these applications, image search results are usually displayed in a ranked list. This

ranking reflects the similarity of the image's metadata to the textual query, according to the textual retrieval model of choice. There may exist two problems with this ranking.

First, it may be lacking visual diversity. For instance, when a specific type or brand of car is issued as query, it may very well be that the top of this ranking displays many times the same picture that was released by the marketing division of the company. Similarly, pictures of a popular holiday destination tend to show the same tourist hot spot, often taken from the same angle and distance. This absence of visual diversity is due to the nature of the image annotation, which does not allow or motivate people to adequately describe the visual content of an image.

Second, the query may have several aspects to it that are not sufficiently covered by the ranking. Perhaps the user is interested in a particular aspect of the query, but doesn't know how to express this explicitly and issues a broader, more general query. It could also be that a query yields so many different results, that it's hard to get an overview of the collection of relevant images in the database.

We propose to create a visually diverse ranking of the image search results, through clustering of the images based on their visual characteristics. To organise the display of the image search results, a cluster representative is shown to the user. Depending on the interest of the user in one of the representatives, he can then explore the other images in that cluster. This approach guarantees that the user will be presented a visually diverse set of images.

An example clustering of one of our algorithms is given in Figure 5.1. The example uses the ambiguous query "*jaguar*". The image search result is not only ambiguous from a topical point of view (car, mammal), but also from a visual point of view. The algorithm separates mammals with a tiger print from black mammals and mammals behind bars. It also groups pictures from a new car model at an expo from cars in the street, and groups the accidentally found pictures of a lady at a car expo. The cluster representatives together form a diverse set of image search results.

5.1.1 Contributions

In this chapter we introduce new methods to diversify image search results [122]. Given a user query, we first determine dynamically appropriate weights of visual features, to best capture the discriminative aspects of the resulting set of images that is retrieved. These weights are used in a dynamic ranking function that is deployed in a lightweight clustering technique (i.e. easy to implement, efficient to run, for

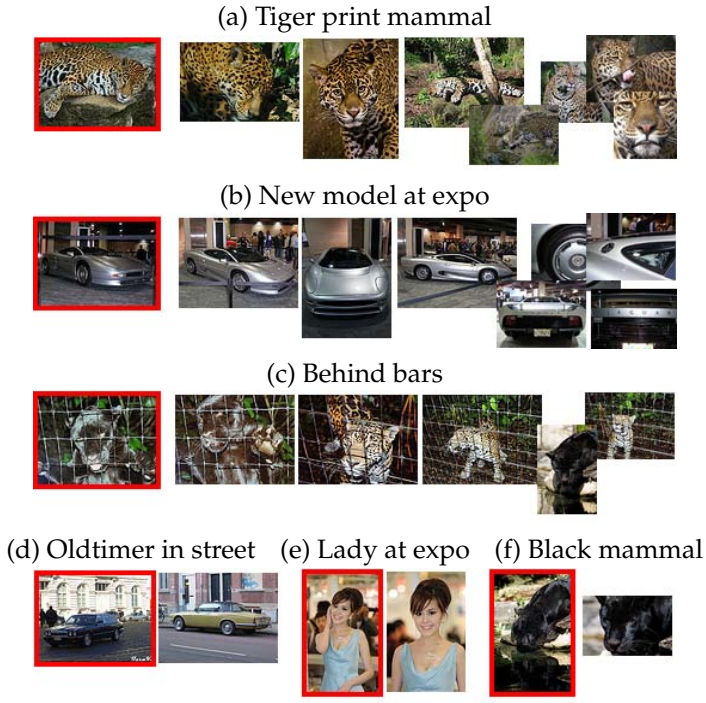


Figure 5.1: Example clustering: output of the reciprocal election algorithm for query *jaguar*. Cluster representatives are indicated by a red border.

example with no training required) to obtain a diverse ranking based on cluster representatives. We propose three clustering algorithms that are both effective and efficient, called *folding*, *maxmin* and *reciprocal election*. In the case of folding, the original ranking is respected by preferring higher ranked items as representatives over lower ranked items. Maxmin on the other hand discards this original ranking and aims for maximal visual diversity of the representatives. The key idea behind reciprocal election is to let each image cast votes for other images that it is best represented by: a strategy close to the intuition behind a clustering. We have implemented the methods and performed a performance evaluation in a user-study using 75 topics of both an ambiguous and non-ambiguous nature.

5.1.2 Related work

In the context of the general task of this chapter, we discuss the related work by first discussing the state of the art in image clustering, and then by focusing on related work in diversifying search results.

5.1.2.1 Image clustering

Most image clustering techniques are not dynamic, and therefore not suitable for clustering image search results. To begin with, we are only interested in unsupervised clustering techniques, which makes techniques such as presented in [64] unsuitable for our task. Furthermore, clustering techniques often partition the entire database in static clusters to facilitate faster browsing and retrieval [51].

In [77] a method for extracting meaningful and representative clusters is presented that is based on a shared nearest neighbours (SNN) approach that treats both content-based features and textual descriptions (tags). They describe, discuss and evaluate the SNN method for image clustering and present some experimental results using the Flickr collections showing that their approach extracts representative information of an image set. Such techniques are often effective, but require a lot of processing power to produce a final clustering. When clustering image search results, the input varies depending on the user's query and it is essential that the clustering technique is not only effective, but the results can be efficiently computed.

In our case, we want the approach to be dynamic such that it best captures the particular context of the user's query. We'll therefore rely on a dynamic ranking strategy, which allows us to dynamically weight the importance of the visual dimensions such as colour, shape an texture, in combination with lightweight clustering strate-

gies. Although not incorporated in the current implementation, we can easily extend the dynamic ranking strategy to include a textual modality as is used in the aforementioned related work.

In Cai et al. [26] the problem of clustering Web image search results is studied, by organising the results into different semantic clusters that facilitates users' browsing. They propose a hierarchical clustering method using visual, textual and link analysis that is mainly targeted at clustering the search results of ambiguous targets. However they do not report the effectiveness of their approach, and illustrate the performance through a single example. In a related work by Wang et al. [134], a different approach named IGroup is evaluated, for semantic clustering of image search results, based on a textual analysis of the search results. Through a user study they report a significant improvement in terms of efficiency, coverage, and satisfaction.

5.1.2.2 Diversity in search results

In some prior work, diversification of image search results was studied in two different contexts. In [136], a method is presented for detecting and resolving the ambiguity of a query based on the textual features of the image collection. If a query has an ambiguous nature, this ambiguity should be reflected in the diversity of the result set. Furthermore, in [127] it is investigated how the topical (textual) diversity of image search results can be achieved through the choice of the right retrieval model. The focus in the current chapter is on *visual* diversity of the search results. Our solution for the visual diversity builds upon the results of these two works, as it takes as input the ranked list of images produced by the retrieval models for topical diversity.

In Zhang et al. [145] diversity of search results is examined in the context of Web search. They propose a novel ranking scheme named Affinity Ranking to re-rank search results by optimising two metrics: diversity and information richness. More recently, Song et al. [109] also acknowledge the need for diversity in search results for image retrieval. They propose a re-ranking method based on topic richness analysis to enrich topic coverage in retrieval results, while maintaining acceptable retrieval performance.

Zeigler [147] studied topic diversification to balance and diversify personalised recommendation lists in order to reflect the user's complete spectrum of interests. Although their system is detrimental to average accuracy, they show that the method improves user satisfaction with recommendation lists, in particular for lists generated using the common item-based collaborative filtering algorithm. They intro-

duced an intra-list similarity metric to assess the topical diversity of recommendation lists and the topic diversification approach for decreasing the intra-list similarity.

In a different setting, Yahia et al. [128] propose a method to return a set of answers that represent diverse results proportional to their frequency in the collection. Their algorithm operates on structured data, with explicitly defined relations, which differs from our setting, as we aim to diversify through visual content based on a dynamic ranking strategy, rather than using predetermined fractions.

5.2 Image similarity

One of the key elements to any clustering algorithm or retrieval system, is a similarity measure between the objects. In content-based image retrieval or clustering, it is common to use several features simultaneously while calculating the similarity between images. These features represent different aspects of the image, such as colour features, edge features, texture features, or alternatively concept detectors [93]. Each feature has its own representation (e.g. a scalar, a vector, a histogram) and a corresponding matching method (e.g. Euclidean distance, hamming metric, earth mover's distance).

The fusion of different modalities into a single ranking is not trivial. Various techniques have been proposed to effectively fuse multiple-modalities into a single ranking, using a simple linear weighting, principle component analysis [126], or by using a weighted schema for aggregating features based on document scores [137].

In this chapter we introduce a dynamic ranking strategy that weights the importance of the different features based on the (normalised) variance of the similarities of all images in the results set. In our case, the similarity measure defined on the images has to reflect visual similarity, but the clustering algorithms presented in this chapter work with any distance measure between two images.

In this section we first describe the dynamic feature weighting function that implements the ranking strategy, followed by a short description of the 6 well-known visual image features that we have adopted for our experiments.

5.2.1 Dynamic feature weighting

Based on the features described below, the similarity between two images can be expressed in 6 similarity values. These values, that may be of entire different range

and distribution, need to be aggregated into one value for use in the clustering algorithm. Moreover, it is a priori unclear what the relative importance is of these features within the context of a specific set of image search results.

One assumption we make, is that the images retrieved by the textual retrieval model are topically relevant to the query. For each feature, we then calculate the variance over all image similarities within the set of image results. This variance is used as a weighting and normalising factor at the same time. The image similarity according to a certain feature is divided by the variance of that feature in the result set. This brings image similarities according to different features in a similar range, and assigns a larger weight to features that are a good discriminator for the results that are presented to the user. The rationale is that when the variance of a certain feature is small, the images in the result set resemble each other in terms of that feature closely and thus it is a striking feature for this specific set.

More formally, the similarity between two images a and b is calculated as follows:

$$d(a,b) = \frac{1}{f}\sum_{i=0}^{f}\frac{1}{\sigma_i^2}d_i(a,b),$$

where f is the total number of features, $d_i(a,b)$ is the similarity between a and b in terms of the i-th feature and σ_i^2 is the variance of all image similarities according to the i-th feature within this set of image search results.

5.2.2 Features

For our experiment we have extracted 6 visual features from each image to capture the different characteristics such as the colour, shape and texture of an image. Below follows a short description.

Colour histogram. A colour histogram describes the global colour distribution in an image. To compute the colour histogram, we define a discretization of the RGB colour space into 64 colour bins. Each bin contains the number of pixels in the image that belong to that colour range. Two colour histograms are matched using the Bhattacharrya Distance [15], defined on histograms H_1 and H_2, both with n bins, as

$$\text{bhattacharrya}(H_1,H_2) = \sum_{i=1}^{n}\sqrt{H_{1_i}\cdot H_{2_i}}$$

where H_{1_i} and H_{2_i} are the sizes of the i-th bins in H_1 and H_2 respectively.

Colour layout. Colour layout is a resolution invariant compact descriptor of colours used for high-speed image retrieval [93]. Colour layout captures the spatial distribution of the representative colours in an image. The image is divided into 64 blocks. For each block a representative colour is obtained using the average of the pixel colours. Every colour component ($YCbCr$) is transformed by a 8x8 DCT (discrete cosine transformation) obtaining a set of 64 coefficients, which are zigzag-scanned and the first coefficients are non linearly quantized. The distance between two colour layout descriptors is calculated using the L_2-norm.

Scalable colour. Scalable colour can be interpreted as a Haar-transform applied to a colour histogram in the HSV colour space [93]. First, the histogram values (256 bins) are extracted, normalised and non linearly mapped to a 4-bit integer representation. Afterwards, the Haar transform is applied across the histogram bins to obtain a smaller descriptor allowing a more scalable representation. Two feature vectors are matched using a standard L_1-norm.

CEDD. The colour and edge directivity descriptor (CEDD) incorporates both colour and texture features in a histogram [28]. It is limited to 54 bytes per image making this descriptor suitable for large image databases. First, the image is split in a preset number of blocks; a colour histogram is computed over the HSV colour space. Several rules are applied to obtain for every block a 24-bins histogram (representing different colours). Then 5 filters are used to extract the texture information related to the edges presented in the image and classified in vertical, horizontal, 45-degree diagonal, 135-degree diagonal and non-directional edges. The whole process results in a descriptor in vector format. Two descriptors A and B are matched using the Tanimoto coefficient [114], which is an extension of the cosine similarity metric, defined as

$$T(A,B) = \frac{A \cdot B}{||A||^2 + ||B||^2 - A \cdot B}.$$

Edge histogram. The edge histogram represents a local edge distribution of the image [93]. First, the image is divided in a 4×4 grid. Edge detection is performed to each block and the edges are grouped into 5 types: vertical, horizontal, 45 degrees diagonal, 135 degrees diagonal and non-directional edges. The feature therefore consists of $16 \times 5 = 80$ coefficients. For matching two feature vectors the standard L_1- norm is used.

Tamura features Tamura et al. [112] identified properties of the images that play an important role in describing textures based on human visual perception.

They defined six textural features (coarseness, contrast, directionality, line-likeness, regularity and roughness). We used 3 Tamura features to build a texture histogram: coarseness, contrast and directionality. The Tamura features are matched using the standard L_2-norm.

5.3 Clustering algorithms

In this section we present the clustering algorithms, called *folding*, *maxmin* and *reciprocal election*. First, we introduce some notation. A set of image search results I contains n images. I can be stored either in a ranked list $L = L_1, L_2, \ldots, L_n$, sorted in decreasing degree of relevance to the query, where $L_1, \ldots, L_n$ are the images. Alternatively, I can be stored in a set $S = S_1, S_2, \ldots, S_n$, where there is no particular ordering of the images $S_1, \ldots, S_n$. The input to a clustering algorithm can be either L or S, and its output is a clustering C: a partitioning of I. In C, all the images are divided over K clusters $C_1, C_2, \ldots, C_k$ such that $C_k \bigcap C_l = \emptyset$ for all $l, k \in K$ and $\bigcup_{k=1}^{K} C_k = I$. The number of images in cluster C_k is n_k, so $\sum_{k=1}^{K} n_k = n$, and in each cluster C_k one image is declared representative, called R_k. All the representatives together form the set R. Note that K is not fixed and may be different for each clustering.

The three clustering algorithms differ from each other in viewpoint. The folding algorithm considers the original ranking of the search results as returned by the textual retrieval model. Images higher in the ranking have a larger probability as being selected as a cluster representative. In one linear pass the representatives are selected, the clusters are then formed around them. The maxmin approach also performs representative selection prior to cluster formation, but discards the original ranking and finds representatives that are visually different from each other. Reciprocal election lets all the images cast votes for other images that they are best represented by. Strong voters are then assigned to their corresponding representatives, and taken off the list of candidates. This process is repeated as long as there exist unclustered images.

The number of clusters is never fixed, because it is impossible to predict a priori what a good value is. This should always be dynamically set for a specific clustering. We now present the algorithms in more detail.

5.3.1 Folding

In some cases, it is important to take the original ranking of the image search results into account while performing the clustering. For example, it might be that the query is very specific and only the top of the ranking is sufficiently topically relevant. It might also be that retrieval speed is valued over accuracy, and the uncertainty about topical relevance of the retrieved items decreases quickly while going through the ranking. Folding (see algorithm 5.1) is an approach that appreciates the original ranking, by assigning a larger probability of being a representative to higher ranked images.

The first step of the approach is to select the representative images, while traversing through the ranking L from top to bottom. The first image in the ranking, L_1 is always selected as representative. While going down the ranked list, each image is compared to the set of already selected representatives. When an image is sufficiently dissimilar to all the selected representatives in R, it is added to R. After this pass through the ranking, clusters are formed around each representative using a nearest neighbour rule: each image in L is assigned to the closest representative.

Key to this approach is the definition of *sufficiently dissimilar* while selecting the candidates. This parameter is set automatically and dynamically as well. It is defined as the mean distance all images in I have to the *average features*. The average features do not form a real image, but are synthetic features that only exists in feature space and are constructed by aggregating for each feature all the images into one canonical feature. Since all features are histogram-like features, this aggregation step follows from their definition. It would be also possible to select a canonical image from I using a heuristic, e.g. the image with the smallest mean distance to all the other images.

5.3.2 Maxmin

The maxmin approach (see algorithm 5.2) doesn't take the original ranking into account like folding does. It rather tries to get as visually diverse representatives as possible. To achieve this, it uses a maxmin heuristic on the distances between cluster representatives. The algorithm takes as input I stored as unordered set S, and the first representative R_1 is selected at random. The second representative R_2 is the image of S with the largest distance to R_1. For each following representative, the image is selected that has the largest minimum distance to all the other selected representatives. This process is continued until this maximum minimal distance is smaller than ϵ, which is again defined as the mean distance all images have to the

Algorithm 5.1 Folding
Input: Ranked list L of I
Output: Clustering C

1: Let the image L_1 be the first representative R_1
2: **for** Each image L_i **do**
3: **if** $d(L_i, R_j) > \epsilon$ for all representatives $R_j \in R$, where ϵ is the mean distance all images have to the *average feature* **then**
4: add L_i to the set of representatives R
5: **for** Each image $L_i \notin R$ **do**
6: Find representative $R_j \in R$ that is closest to L_i
7: Assign L_i to the cluster of R_j

average image.

When the representatives are selected using this heuristic, cluster formation is again carried out using a nearest neighbour rule. Each image is assigned to the closest representative.

Algorithm 5.2 Maxmin
Input: Set S containing I
Output: Clustering C

1: Select the first representative R_1 randomly
2: **while** All pairwise distances in $R > \epsilon$ **do**
3: **for** Each image $L_i \notin R$ **do**
4: Let d_i be $\underset{d(L_i,R_j),R_j \in R}{\arg\min}$
5: Add to R the image with $\underset{d_i}{\arg\max}$
6: **for** Each image $S_i \notin R$ **do**
7: Find representative $R_j \in R$ that is closest to S_i
8: Assign S_i to the cluster of R_j

5.3.3 Reciprocal election

In contrast to folding and maxmin, the reciprocal election approach interleaves the processes of representative selection and cluster formation. The key idea behind this approach (see algorithm 5.3 and figure 5.2) is that every image in I decides by which image (besides itself) it is best represented. They all cast votes for the other

images, and all the votes an image receives determine its chances of being elected as representative. This process of voting is based on calculating reciprocal ranks in rankings of I. For each image S_i, the whole set of image search results I is ranked into L_i based on visual similarity to S_i. The image S_i then casts its highest vote for the image that appears at the top of that ranking (excluding itself), its second highest vote to the number two of that list etcetera. More specifically, each image in L_i receives as a vote from S_i its reciprocal rank, i.e. $1/r$ where r is its rank in L_i.

When all the images have cast their votes, the image with the highest number of votes is selected as first representative R_1. Immediately, the cluster around R_1 is formed, by inserting those images that have R_1 in the top-m of their ranking, where m is a user defined parameter [7]. The rationale is that because R_1 appears so high in their ranking, they are sufficiently well represented by R_1. After cluster C_1 has been formed, its members and its representative are excluded from the list of candidate representatives, and the process is repeated until every image has been either selected as representative or assigned to a cluster.

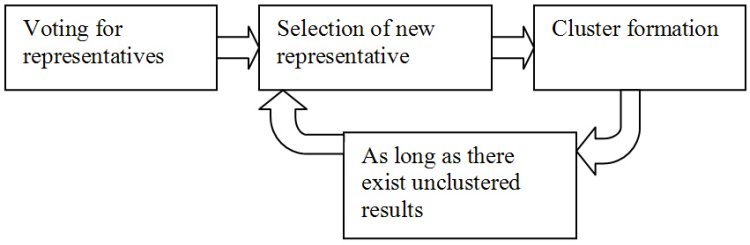


Figure 5.2: Overview of the reciprocal election algorithm

5.4 Experimental setup and results

Our test corpus consists of a pool of 75 topics that were randomly selected from the Flickr search logs. Based on the method for resolving query ambiguity as presented in [136], we have divided our pool in 25 textually ambiguous queries, and 50 textually non-ambiguous queries. This method measures how well a query can appear in two different contexts. Specifically, within a proposed probabilistic framework, it looks at the KL-divergence of different keywords that are commonly associated with the query. Take for instance the query jaguar: the tags "mammal" and "jaguar" come

[7] Alternatively, the threshold determining the scope in which to search for R_1 can be defined by a maximum distance.

Algorithm 5.3 Reciprocal election
Input: Set S containing I, parameter m
Output: Clustering C

```
Initialise Votes map V[0, ..., n] = 0, ..., 0
for Each image S_i in S do
    Rank S into L_i based on visual similarity to S_i
    for Each image S_j in L_i do
        V[j] += 1/r, where r is the rank of S_j in L_i
while V is not empty do
    Let R_i be the item with the highest score in V
    Remove R_i from V
    Initialise new cluster C with representative R_i
    for All items s in V do
        if R_i is in top-m of L_s then
            add s to cluster C
            remove s from V
```

often with the query jaguar, but these rarely appear in the same context. This suggest a high level of ambiguity. See figure 5.3 for an illustration of this example. There are more kinds of ambiguity than just this word sense ambiguity. Some examples are spatial ambiguity (e.g. there is a "Cambridge" both in Massachusetts and in the UK), temporal ambiguity (e.g. "World War" may refer to the first or second) and language ambiguity (e.g. "handy" means convenient for use or skillful with the hands in English, whereas the Germans use it to refer to a mobile phone). Using this method for measuring tag ambiguity allows us to investigate the difference in performance of the visual clustering methods on both types of queries. For each query we have retrieved the top 50 results from a slice of 8.5 Million photos on Flickr.

To retrieve a list of 50 results for the non-ambiguous queries we have used a dual index relevance model that produces a focused result set. To obtain the top 50 results of the ambiguous queries, we have used a tags-only index relevance model that produces a balanced list of diverse results. The details of both retrieval models are described in [127]. The intuition behind these choices is simple. If the terms in a query are textually diverse, then we want to produce a diverse set of images that embodies many possible interpretations of the user's query. Consider again the example query "jaguar", which carries at least three different word-senses that are present in the Flickr collection: the mammal, the car, and the operating system. On the other hand, if a query is textually non-ambiguous, e.g it has a clear dominant sense, the precision can be improved by returning more focused results. The query

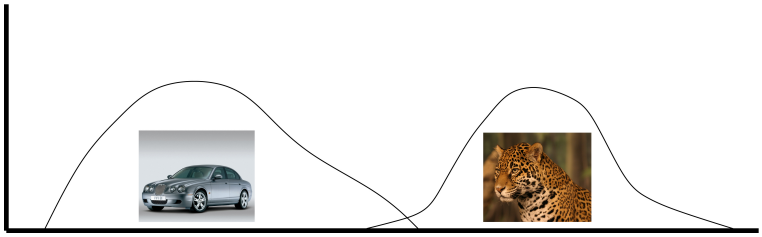


Figure 5.3: Measuring query ambiguity: the two keywords that are often associated with the query "jaguar" have a high KL-divergence

"jaguar x-type" serves as an example for a non-ambiguous query. In both cases, the result sets produced contain visually diverse images on which we will test our methods.

To evaluate the performance of the proposed algorithms, we compare their output to clusterings that were created by human assessors. The following sections present details on the establishment of the ground truth, the evaluation criteria and the experimental results.

5.4.1 Human assessments

To establish a ground truth, we divided the 75 topics over 8 independent, unbiased assessors and we asked them to cluster the images based on their visual characteristics. We implemented the following procedure.

1. Select a topic, and inspect the top 50 results during at least one minute. This allows the assessors to get an overall impression of the images in the result set, to get a rough idea of how many clusters will be needed, and of their level of inter cluster dissimilarity. At any point in the assessment, the assessor could switch to this overview.

2. Form image clusters by assigning each image to a cluster pool by entering the cluster id. In total the assessor could create 20 clusters, and he/she could undo the last assignment if needed to correct for errors. See Figure 5.4 for an example of this interface.

3. Once all images in the results were assigned to a cluster, the assessor was asked

to label each cluster and to identify one image in each cluster that could serve as a cluster representative.

In total we obtained 200 topic clusterings created by the assessors, because each topic was assigned to multiple assessors. We are therefore able to calculate inter assessor variability, that provides us with a base line during the performance evaluation of the algorithms.

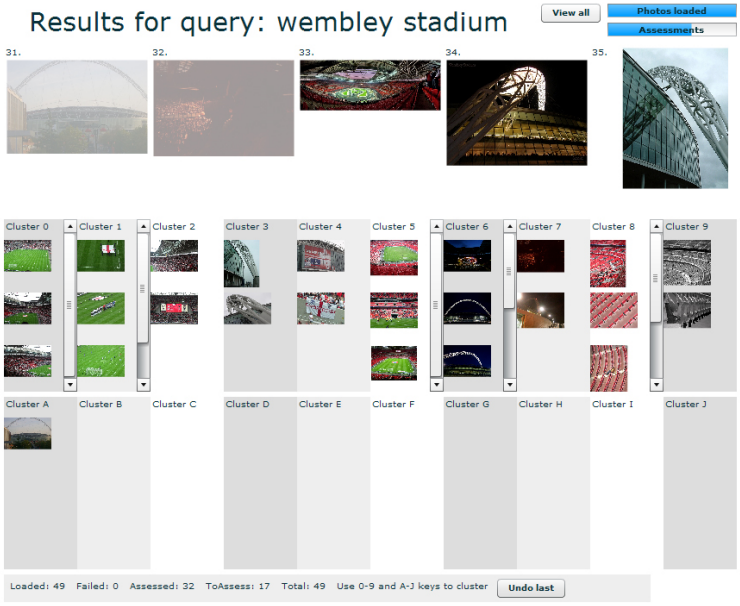


Figure 5.4: Example of the clustering interface used by the assessors.

5.4.2 Evaluation criteria

Comparing two clusterings of the same data set is an interesting problem itself, for which many different measures have been proposed. We adopt two clustering comparison measures that appreciate different properties. In this subsection we describe them briefly.

One popular category of comparison measures is based on counting pairs. Given a result set I and two clusterings C and C', all possible image pairs based on I are

divided over the following four classes:

- N_{11} : image pairs in the same cluster both under C and C'
- N_{00} : image pairs in a different cluster both under C and C'
- N_{10} : image pairs in the same cluster under C but not under C'
- N_{01} : image pairs in the same cluster under C' but not under C

From now on, we will refer to the *cardinality* of these classes simply by its class name. These cardinalities are input to the comparison measures. The first clustering comparison measure we use is the Folwkes-Mallows index [46] that can be seen as the clustering equivalent of precision and recall. A high score of the Folwkes-Mallows index indicates that the two clusterings are similar. It is based on two asymmetric criteria proposed by Wallace [133]:

$$W_I(C, C') = \frac{N_{11}}{N_{11} + N_{01}}$$
$$W_{II}(C, C') = \frac{N_{11}}{N_{11} + N_{10}}$$

The Folwkes-Mallows index is the geometric mean of these two, making it a symmetric criterion:

$$FM(C, C') = \sqrt{W_I(C, C')W_{II}(C, C')}$$

Another class of comparison measures is based on a structure called the *contingency table* or *confusion matrix*. The contingency table of two clusterings is a $K \times K'$ matrix, where the kk'-th element is the number of points in the intersection of clusters C_k of C and $C'_{k'}$ of C'. Our second clustering comparison measure is the *variation of information* criterion, $VI(C, C')$, as introduced by Meilă [74]. Variation of information uses the contingency table, and is based on the concept of conditional entropy.

Given a clustering C, the probability that a randomly picked image belongs to cluster k with cardinality n_k, is

$$P(k) = \frac{n_k}{n}$$

.

This defines a random variable taking K values. The uncertainty about which cluster an image is belonging to is therefore equal to the entropy of this random variable:

$$H(C) = -\sum_{k=1}^{K} P(k) \log P(k).$$

The mutual information $I(C, C')$, the information one clustering has about the other, can be defined similarly. First, the probability that a randomly picked image belongs to cluster k in C and to cluster k' in C', is

$$P(k, k') = \frac{|C_k \bigcap C'_{k'}|}{n}.$$

Then, the mutual information $I(C, C')$ is defined as the sum of the corresponding entropies taken over all possible pairs of clusters:

$$I(C, C') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} P(k, k') log \frac{P(k, k')}{P(k)P'(k')}.$$

The mutual information coefficient can be seen intuitively as a reduction of uncertainty from one clustering to the other. For a random image, the uncertainty about its cluster in C' is measured by $H(C')$. Suppose now that it is given which cluster in C the image belongs to; how much does this reduce the uncertainty about C'? This reduction, averaged over all images, is equal to $I(C, C')$.

Finally, the variation of information is defined as the sum of the two conditional entropies $H(C, C')$ and $H(C', C)$. The first measures the amount of information about C that we loose, while the second measures the amount of information about C' that we gain, when going from clustering C to C'. It can be written as

$$VI(C, C') = [H(C) - I(C, C')] + [H(C') - I(C, C')].$$

The variation of information coefficient focuses on the relationship between a point and its cluster. It measures the difference in this relationship, averaged over all

points, between the two clusterings, hence a low variation of information score indicates that two clusterings are similar.

5.4.3 Results

All 200 clusterings of the 75 topics that were obtained as a result of the human assessments are compared to the clusterings generated by the different techniques. Using the described comparison measures, variation of information and the Folwkes-Mallows index, performance is evaluated. In this section, results are presented for ambiguous topics separately, non-ambiguous topics separately and all topics together.

Interassessor variability and random clustering

As a base line for the performance we use the inter assessor variability. A technique cannot be expected to produce clusterings that resemble on average the human created clusterings better than the assessors agree among themselves. To put a bound on expected performance on the other end as well, we compare the human created clusterings with randomly generated clusterings. For this purpose, for each topic a random number (between 2 and 20) of clusters was generated. Every image was clustered randomly into one of the clusters, all random distributions were uniform. We expect that the performance of each of the three methods to lay within these two performance bounds.

Results on Fowlkes-Mallows Index

The best performing method according to the Fowlkes-Mallows index is folding, followed by reciprocal election and maxmin. Mean values and first and third quartiles are given in Figure 5.5 for both ambiguous and non-ambiguous topics. The boxes show the average and the first and third quartiles for all comparisons, i.e. 50% of the 200 clustering comparisons fall within the box. The figure is showing the performance of reciprocal election, folding and maxmin; it is also showing the comparison results of a randomly generated clustering and the inter-assessor agreements according to the same comparison measure. Please note that a higher FM-index corresponds to better performance, as it indicates more agreement between the method and the assessors on point pairs that fall in the same cluster. Table 5.1 presents performance of the methods averaged over all topics, and with an FM-index of 0.282 folding outperforms again reciprocal election (FM = 0.250) and maxmin (FM = 0.214).

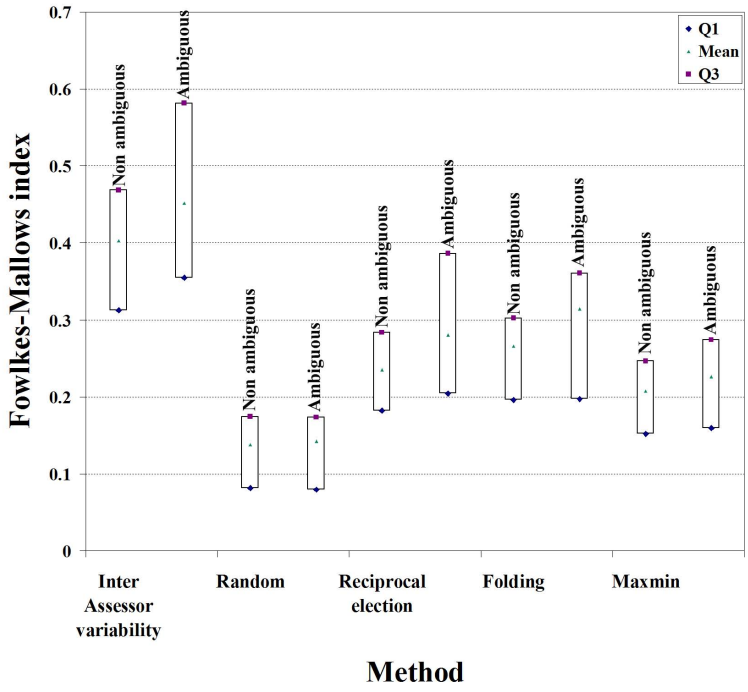


Figure 5.5: Performance of the three methods on the Fowlkes-Mallows index, compared to human assessments and the random baseline.

To test these claims for significance, we calculated p-values of two tailed t-Tests. The null-hypothesis that all methods perform equally well is rejected both times, with $p = 0.006$ for reciprocal election and $p = 2.3 \times 10^{-9}$ for maxmin. Moreover, Figure 5.6 shows per topic the FM-index for folding against the FM-index for reciprocal election and maxmin. For every topic under the equality line $y = x$, folding outperforms the other method. With respect to reciprocal election, folding outperforms 58% of the topics, and for maxmin this value is even 73%.

The Fowlkes-Mallows index measures the degree of agreement on point pairs that fall in the same cluster under both clusterings. This measure is therefore rather sensitive to the number of clusters. The folding approach benefits from its strong mechanism to automatically and dynamically select a proper number of clusters.

	Inter-assessor variability	Random	Reciprocal election	Folding	Max-min
FM	0.419	0.139	0.250	0.282	0.214
VI	1.463	2.513	1.975	2.081	2.129

Table 5.1: Average performance over all topics and assessors.

Results on Variation of Information Metric

A different relative performance is given by the variation of information criterion. According to this measure, reciprocal election outperforms folding and maxmin. Mean, first and third quartile performance is given in Figure 5.7, while Table 5.1 presents the performance averaged over all topics. In this case, a lower variation of information indicates a better performance. It denotes that there is less change in cluster membership while going from one clustering to the other.

Significance tests (two tailed t-Test) support the superiority of reciprocal election. The null hypothesis of all methods performing equally well is rejected with $p = 0.002$ for folding and with $p = 7.3 \times 10^{-7}$ for maxmin. Figure 5.8 presents relative performance comparisons per topic. It shows that the majority of folding and maxmin clusterings have a larger variation of information coefficient than reciprocal election, respectively 63% and 70%. In this figure, for every topic under the line reciprocal election achieves a better performance.

Rather than counting image pairs that fall in the same cluster under both clusterings, variation of information focuses on the relationship between an image and its cluster. It measures the difference in this relationship, averaged over all images, between the two clusterings. As this is a more general and less sensitive measure than counting successfully clustered image pairs, reciprocal election has a better overall performance. We conjecture that this is due to how the approach follows the intuition behind a cluster. Images in a cluster should all be well represented by that cluster, a notion that translates directly to how the reciprocal ranks are used as votes.

Ambiguous Topics vs. Non-ambiguous Topics

One more interesting result can be observed. Both figure 5.5 and 5.7 clearly show that the assessors agree more on ambiguous topics than on non-ambiguous topics. This is probably due to the fact that a more generally accepted clustering exists for topics that produce semantically different clusters. On non-ambiguous topics the assessors

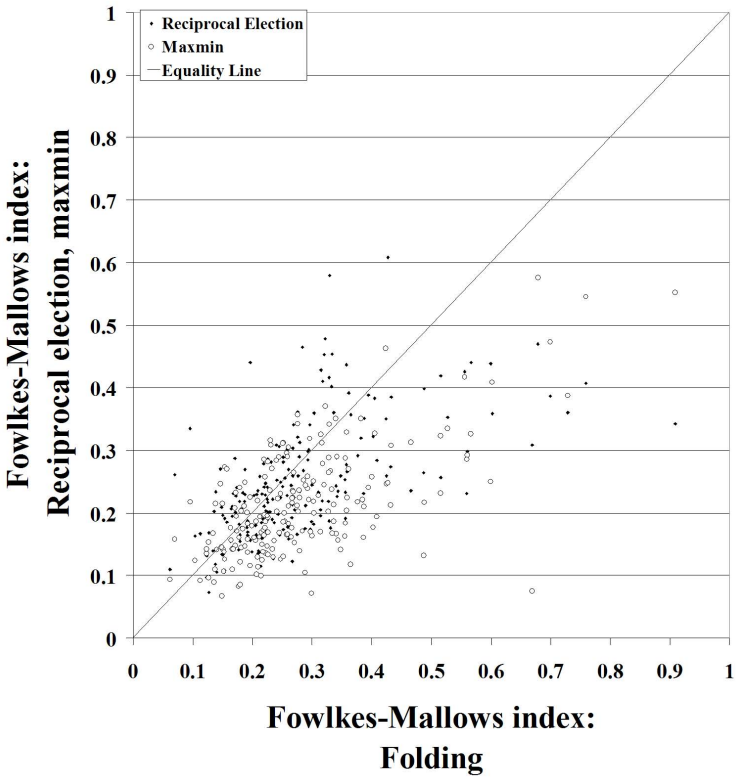


Figure 5.6: Performance evaluation per topic comparison. The plots show the performance of folding on the x-axis for each clustering comparison, with respect to the performance of the reciprocal election and maxmin on the y-axis. For every topic under the line, folding outperforms the corresponding other method.

may choose more different criteria to base their clustering on. This behaviour is also visible in the performance of the methods; the performance on ambiguous topics is significantly better than on non-ambiguous topics. This indicates the existence of clear visual dissimilarity between semantically different images.

Parameter sensitivity of the algorithms

Both folding and maxmin are parameter-free approaches. The number of clusters is determined automatically based on threshold ϵ, for which the appropriate value is calculated given a set of image search results. The image similarity measure or

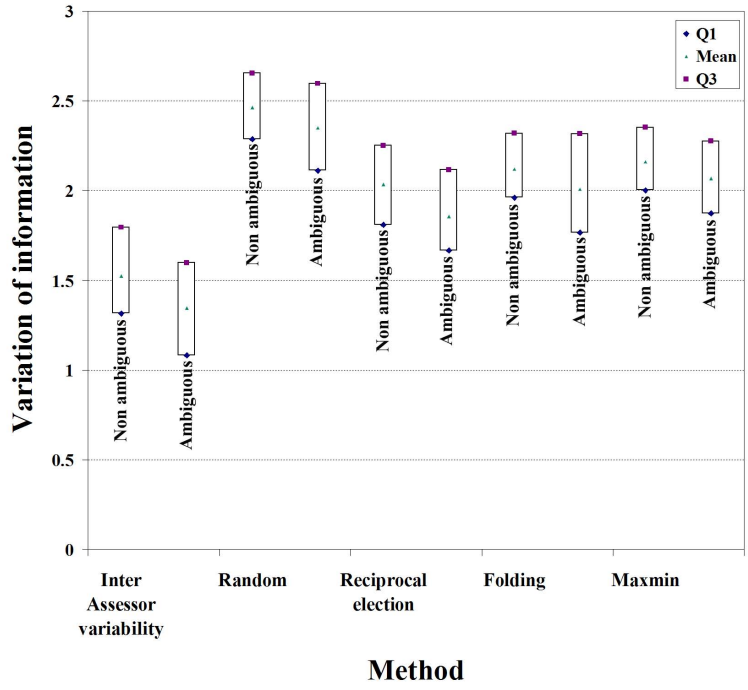


Figure 5.7: Performance of the three methods on the variation of information metric, compared to human assessments and the random baseline.

ranking function is also dynamic: weights for the visual features are established automatically for each set of image results.

Reciprocal election requires only parameter m, that determines the window size with which the ranked lists are inspected to decide upon cluster membership after a new representative has been found. We experimented with several values for this parameter m, and found more or less consistent performance for values between 3 and 8. The best performance was obtained using $m = 4$. With $5 \leq m \leq 8$, the variation of information increases slightly, but the method still outperforms the other approaches. Relative performance according to the Fowlkes-Mallows index did not change significantly either. With $m \geq 9$, performance degrades quickly, because then the images are assigned to a representative too easily and clusters become too large.

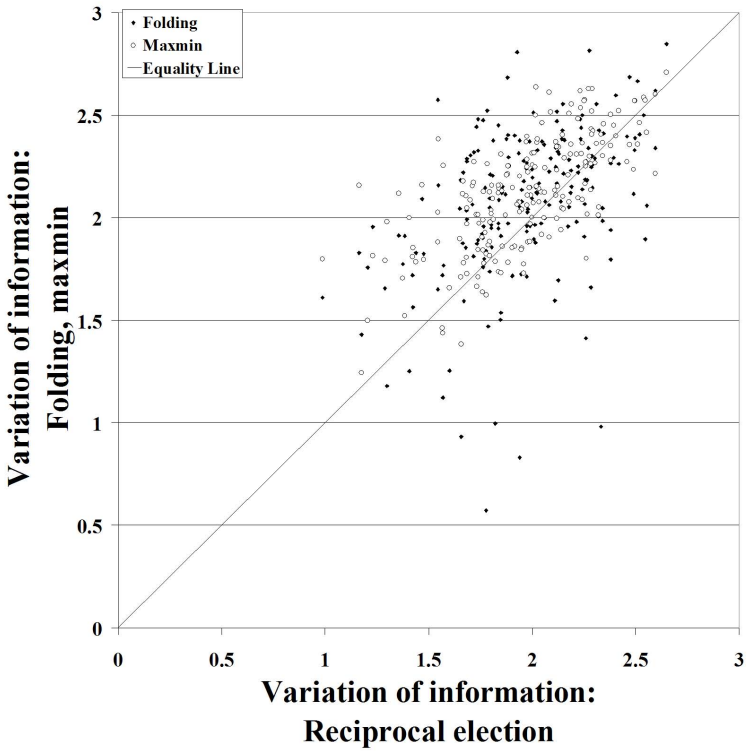


Figure 5.8: Performance evaluation per topic comparison. The plots show the performance of reciprocal election on the x-axis for each clustering comparison, with respect to the performance of the folding and maxmin on the y-axis. For every topic above the line, reciprocal election outperforms the corresponding other method.

5.5 Concluding remarks

Image search engines on the Web still rely heavily on textual metadata, causing a lack of visual diversity in image search results. Still, diversity is a highly desired feature of search results, no less in image search than in other search applications. In this chapter, we present new methods to visually diversify image search results that deploy lightweight clustering techniques. These methods are effective, efficient and require no training nor parameter tuning. Given a user query, they adapt automatically to the set of image search results. The weights for visual features in a dynamic ranking function are computed on the fly to emphasise highly discriminant features for this set of results, and the number of clusters is adaptive as well.

The folding strategy respects the ranking order and picks the cluster representatives accordingly, while Reciprocal election aims to optimise the clustering and the (s)election of cluster representatives by a voting strategy where each image determines a list of candidate images that it would best represented by. After performing a large user-study to establish a ground truth and a baseline, we measure performance of all methods.

Folding shows a better performance according to the Folwkes-Mallows index, a performance measure that focuses on image pairs that can be formed with images from the same cluster. This indicates that the folding approach benefits from its strong mechanism to automatically and dynamically select a proper number of clusters. On the other hand, reciprocal election significantly outperforms the other methods in terms of variation of information, a more general performance measure. The selection of candidates and the decision on cluster membership both follow an intuitive notion behind a clustering. We conjecture that this is rewarded by means of a low variation of information, and therefore conclude that reciprocal election achieves the strongest overall performance.

Chapter 6

Conclusion and future research

...this is my dilemma. I'm a guy who makes things up as I go along so nothing is ever finished-there are so many layers. So when you solo, yeah, you might get into one thing, but then, hey, everything has implications! You can hear the next level. That's how I feel-there's always another level.

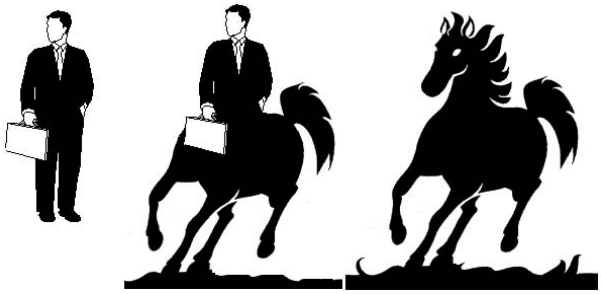
– Sonny Rollins (1930-) –

We started in chapter 2 with a study of vantage indexing. A vantage index is basically a vector space embedding of a general metric space. The performance of this embedding can be improved by a proper choice of vantage objects; they correspond to the axes of the vector space. As false negatives are (under metric restrictions) not possible when using vantage indexing, the goal is to avoid as many false positives as possible. We introduced two quality criteria for vantage objects that are directly concerned with false positives: a criterion for the relevance of a single vantage object (spacing) and a criterion for the redundancy of a vantage object with respect to other vantage objects (correlation). To select vantage objects according to these criteria and to build the vantage index simultaneously, we proposed a heuristic algorithm. Vantage objects that are selected using this algorithm outperform vantage objects that are selected using other, existing techniques, on a large variety of application domains and dataset sizes.

One other aspect influencing the performance of vantage indexing is the number of vantage objects. We investigated the difference in performance using different numbers, but did not propose an automatic way of setting this parameter. A concept that is closely related to an appropriate number of vantage objects is the *intrinsic dimensionality* ρ of the dataset. Given an object A_i from a dataset A consisting of n objects and a metric defined on these objects, one can say that n features are known for A_i: its distances to all other objects. This specific dataset therefore spans an n-dimensional space. However, without loss of information, i.e. while preserving the pair-wise distances, this database can probably be embedded in a space of a lower dimensionality d. This value d is closely related to the intrinsic dimensionality ρ of the dataset, which can be estimated with $\rho = \mu^2/2\sigma^2$ [42], where μ denotes the mean and σ^2 the variance of the distances. It is a promising idea to translate a definition of intrinsic dimensionality to a function with which an appropriate number of vantage objects can be estimated. A possible drawback of the definition given above would be its dependency on a normal distribution: the definition is based on the behaviour of uniformly distributed vector spaces under Minkowski metrics. In many practical cases however, the object space is not uniformly distributed, and the underlying distance function is not a Minkowski metric. It is interesting to see how much the estimation would suffer from this discrepancy, or how the approach can be relieved from the strong assumption of normality.

A possible limitation of the vantage indexing scheme is that it requires the underlying distance measure to be metric. The triangle inequality is a particularly strong constraint. In many application domains the human perception does not satisfy the triangle inequality, and therefore a distance measure should ideally not be limited by this constraint either. One reason for violating the triangle inequality is the sup-

Figure 6.1: Examples of how through partial matching the triangle inequality is violated: the distance between man and horse is larger than the distance between man and man-horse plus the distance between man-horse and horse.



port for partial matching. Figure 6.1 illustrates how through partial matching the triangle inequality no longer holds: the distance between the man and the horse is larger than the distance between man and man-horse plus the distance between man-horse and horse. Future work could include relieving vantage indexing from the metric constraint, since it would make the method significantly wider applicable. An interesting idea to bridge the gap between non-metric distance measures and metric spaces is presented by Skopal [105]. Based on samples of the database, an order-preserving function is constructed that transforms the distances such that the triangle inequality is satisfied. Order preserving means that the order of the database objects, when sorted based on distance with respect to a certain object, is not altered by the function. Their distances do change, and for the triplets of objects in the sample, the triangle inequality will hold. As a consequence, the intrinsic dimensionality may increase and this makes it harder to embed the space in a vector space. Future research could include a investigation of how this idea combines with vantage indexing, and how it influences the performance of the vantage object selection approach.

In chapter 3, we studied a more restricted class of indexing strategies: those that handle objects that are represented by graphs. We introduced indexing through Laplacian spectra, which essentially embeds the graphs in a vector space that can be queried efficiently. We presented a framework that supports both partial and complete similarity. Many subgraphs are considered to account for partial similarity, and we proposed to use spectral integral variation to speed up the computation of indexing signatures of these subgraphs. There is some art involved in finding a good graph representation for objects. We performed experiments on 2D silhouette images and 3D models for which we both used existing graph representations. In

the domain of trademark images and melody scores we introduced two new graph representations. The framework displays strong performance on all these graph representations.

A possible limitation of indexing through Laplacian spectra is overcome in chapter 4. Two graphs that have the same topology end up having the same indexing signature, while the objects they represent may be different. The graph representation is therefore enriched with additional object properties that are stored in a Hermitian matrix, that is constructed such that it mimics the Laplacian matrix from a spectral point of view. The additional properties are reflected in the Fiedler vector, that is used as indexing signature. Again we use subgraphs to account for partial similarity. In this case we reuse the Fiedler vector to partition the graph in meaningful subgraphs and thereby limit the number of subgraphs to consider, thus increasing the efficiency of the framework.

Finding good graph representations poses a challenge yet another time, particularly because of the additional constraints the Hermitian matrix imposes. We show how the representation of 3D models can be extended to the new situation; we enrich it with shape properties. Experiments show indeed the increased performance of the new representation. Finding more graph representations, possibly also in other domains, remains an interesting future research challenge.

The in chapters 3 and 4 proposed frameworks compute similarity between graph pairs for indexing purposes, based on topology and in the case of chapter 4 also based on additional object properties. Future research could be to design a matching approach based on a similar idea. Such an algorithm would not merely compute a similarity value between graphs, but would try to establish node correspondences using both topology and object characterisations, as encoded in the spectral or vectorial representation. This in turn could guide the similarity computation and could make the frameworks more accurate.

Accuracy, in terms of both precision and recall, is together with efficiency the main evaluation criterion for an indexing framework. In chapter 5 we add diversity to the wish list: search results should not only be relevant, but diverse as well. Ideally, they should reflect the diversity of objects that is present in the collection for a given query. To this end, we proposed a method that dynamically determines the discriminating features for a given set of results, and uses these features in unsupervised clustering algorithms to present diverse results to the user.

The proposed approach provides more perspectives. For instance, the dynamic ranking function based on the feature analysis could lead to a better understanding of the

usefulness of the individual features. But of particular interest would be the evaluation of the quality of the cluster representatives and their suitability to serve as visually disambiguated query expansion. In this case the cluster representative is used as a query itself, and its results can be fused with the original result set using rank aggregation strategies such as proposed in [39]. This would diversify the image search results beyond the scope of an initially returned set of images. Finally, as we have done throughout this thesis, other domains should be investigated as well to further study the intricate equilibrium between accuracy and diversity within the context of both the proposed methods and new techniques.

Bibliography

[1] H. Alt and M. Godau. Measuring the resemblance of polygonal curves. In *Proc. Symposium on Computational geometry*, pages 102–109, 1992.

[2] S. Alwis. *Content-Based Retrieval of Trademark Images*. PhD thesis, Dept. of Computer Science, University of York, UK, 1999.

[3] S. Alwis and J. Austin. An integrated framework for trademark image retrieval using gestalt features and cmm neural network. In *Proc. Image Processing and its applications*, pages 290–295, 1999.

[4] S. Alwis and J. Austin. Trademark image retrieval using multiple features. In *Proc. Challenge of Image Retrieval*, 1999.

[5] W. N. Anderson and T. D. Morley. Eigenvalues of the laplacian of a graph. *Linear and Multilinear Algebra*, 18:141–145, 1985.

[6] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, and J.S.B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(3):209–216, 1991.

[7] Vassilis Athitsos, Jonathan Alon, Stan Sclaroff, and George Kollios. Boostmap: An embedding method for efficient nearest neighbor retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(1):89–104, 2008.

[8] M. Attene, S. Biasotti, and M. Spagnuolo. Shape understanding by contour-driven retiling. *The Visual Computer*, 19(2-3):127–138, 2003.

[9] S. Ayra, D.M. Mount, N.S. Netanyahu, R.Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[10] K. Balińska, D. Cvetković, Z. Radosavljević, S. Simić, and D. Stevanović. A survey on integral graphs. *Univ. Beograd. Publ. Elektrotehn. Fak. Ser. Mat.*, 13:42–65, 2002.

[11] S. Barik and S. Pati. On algebraic connectivity and spectral integral variations of graphs. *Linear Algebra and its Applications*, 397:209–222, 2005.

[12] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust acces method for points and rectangles. In *Proc. ACM-SIGMOD*, pages 322–331, 1990.

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

[14] S. Beucher. Watersheds of functions and picture segmentation, acoustics, speech, and signal processing. In *Proc. International Conference Acoustics, Speech and Signal Processing*, pages 1928–1931, 1982.

[15] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by probability distribution. *Bulletin of the Calcutta Mathematical Society*, 35:99–109, 1493.

[16] S. Biasotti. Reeb graph representation of surfaces with boundary. In *Proc. Shape Modeling International*, pages 371–374, 2004.

[17] S. Biasotti, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, S. Marini, G. Patane, and M. Spagnuolo. 3D shape description and matching based on properties of real functions. In *Eurographics - Tutorials*, 2007.

[18] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Extended reeb graphs for surface understanding and description. In *Proc. Discrete Geometry for Computer Imagery*, pages 185–197, 2000.

[19] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):177–196, 2008.

[20] S. Biasotti, S. Marini, M. Spagnuoloa, and B. Falcidieno. Sub-part correspondence by structural descriptors of 3d shapes. *Computer-Aided Design*, 38(9):1002–1019, 2006.

[21] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *Proc. ACM-SIGMOD*, pages 357–368, 1997.

[22] T. Bozkaya and M. Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Systems*, 24(3):361–404, 1999.

[23] N.R. Brisaboa, A. Farina, O. Pedreira, and N. Reyes. Similarity search using sparse pivots for efficient multimedia information retrieval. In *Proc. International Symposium on Multimedia*, pages 881–888, 2006.

[24] C. Buckley and E.M. Voorhees. Evaluating evaluation measure stability. In *Proc. ACM-SIGIR*, pages 33–40, 2000.

[25] B. Bustos, G. Navarro, and E. Chavez. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 24(14):2357–2366, 2003.

[26] D. Cai, X. He, Z. Li, W.-Y. Ma, and J.-R. Wen. Hierarchical clustering of www image search results using visual, textual and link information. In *Proc. International Conference on Multimedia*, pages 952–959, 2004.

[27] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical models and image processing: GMIP*, 59(5):321–332, 1997.

[28] S.A. Chatzichristofis and Y.S. Boutalis. Cedd: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval. In *Proc. International Conference on Computer Vision Systems*, pages 312–322, 2008.

[29] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. Very Large Databases*, pages 426–435, 1997.

[30] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. *Discrete and Computational Geometry*, 32(2):231–244, 2004.

[31] M.S. Costa and L.G. Shapiro. Relational indexing. In *Proc. Structural and Syntactical Pattern Recognition*, pages 130–139, 1996.

[32] D.M. Cvetković, M. Doob, and H. Sachs. *Spectra of Graphs: Theory and Application*. VEB Deutscher Verlag der Wissenschaften, Berlin, 2nd edition, 1982.

[33] E. R. V. Dam and W. H. Haemers. Spectral characterizations of some distance-regular graphs. *Journal of Algebraic Combinatorics*, 15(2):189–202, 2002.

[34] R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computer Surveys*, 40(2):1–60, 2008.

[35] F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, and S. Dickinson. Object recognition as many-to-many feature matching. *International Journal of Computer Vision*, 69(2):203–222, 2006.

[36] M.F. Demirci, R.H. van Leuken, and R.C. Veltkamp. Shape indexing through laplacian spectra. In *Proc. Visual and Multimedia Digital Libraries*, pages 21–26, 2007.

[37] M.F. Demirci, R.H. van Leuken, and R.C. Veltkamp. Indexing through laplacian spectra. *Computer Vision and Image Understanding*, 110(3):312–325, 2008.

[38] Ate Doornbosch. *Onder de groene linde*. Uitgeverij Uniepers, 1987-1991.

[39] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proc. Conference on World Wide Web*, pages 613–622, 2001.

[40] E. E. Saund. Finding perceptually closed paths in sketches and drawings. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(4):475–491, 2003.

[41] J.P. Eakins, K. Shields, and J.M. Boardman. Artisan: A shape retrieval system based on boundary family indexing. In *Proc. Storage and Retrieval for Image and Video Databases (SPIE)*, pages 17–28, 1996.

[42] E.Chavez and G. Navarro. Searching in metric spaces. *ACM Computer Surveys*, 33(3):273–321, September 2001.

[43] C. Faloutsos and K.I. Lin. FastMap: A fast algorithm for indexing, datamining and visualization of traditional and multimedia datasets. In *Proc. ACM-SIGMOD*, pages 163–174, 1995.

[44] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematics*, 23:298–305, 1973.

[45] M. Fingerhut. Music information retrieval, or how to search for (and maybe find) music and do away with incipits. In *Proc. IAML-IASA Congress*, 2004.

[46] E.B. Fowlkes and C.L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[47] V. Gaede and O. Gunther. Multidimensional access methods. *ACM Computer Surveys*, 30(2):170–231, 1998.

[48] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[49] P. Giannopoulos and R.C. Veltkamp. A Pseudo-Metric for Weighted Point Sets. In *Proc. European Conference on Computer Vision*, pages 715–730, 2002.

[50] C.D. Godsil and B.D. McKay. Constructing cospectral graphs. *Aequationes Mathematicae*, 25:257– 268, 1982.

[51] J. Goldberger, S. Gordon, and H. Greenspan. Unsupervised image-set clustering using an information theoretic framework. *IEEE Trans. on Image Processing*, 15(2):449–458, 2006.

[52] D. B. Goldgof, H. Lee, and T. S. Huang. Matching and motion estimation of three-dimensional point and line sets using eigenstructure without correspondences. *Pattern Recognition*, 25(3):271–286, 1992.

[53] R. Grone, R. Merris, and V. S. Sunder. The laplacian spectrum of a graph. *SIAM Journal on Matrix Analysis and Applications*, 11:218–238, 1990.

[54] A. Gutman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM-SIGMOD*, pages 47–54, 1984.

[55] W. H. Haemers and E. Spence. Enumeration of cospectral graphs. *European Journal on Combinatorics*, 25(2):199–211, 2004.

[56] C. Henning and L.J. Latecki. The choice of vantage objects for image retrieval. *Pattern Recognition*, 36(9):2187–2196, 2003.

[57] M. Hilaga, Y. Shinagawa, T. Kohmura, and T.L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proc. SIGGRAPH*, pages 203–212, 2001.

[58] G.R. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(5):530–549, 2003.

[59] V.J. Hodge, J. Eakins, and J. Austin. Inducing a perceptual relevance shape classifier. In *Proc. International Conference on Image and Video Retrieval*, pages 138–145, 2007.

[60] C. M. Hoffmann. *Group-theoretic algorithms and graph isomorphism*. Springer-Verlag, Berlin, 1982.

[61] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs. In *Proc. Symposium on Theory of Computing*, pages 172–184, 1974.

[62] R. Horaud and T. Skordas. Structural matching for stereo vision. In *Proc. International Conference on Pattern Recognition*, pages 439–445, 1988.

[63] G. Hristescu and M. Farach-Colton. Cluster-preserving embedding of proteins. Technical Report 99-50, DIMACS, 8, 1999.

[64] J. Huang, S.R. Kumar, and R. Zabih. An automatic hierarchical image classification scheme. In *Proc. International Conference on Multimedia*, pages 219–228, 1998.

[65] R. Jain. Nsf workshop on visual information management systems. *SIGMOD Record*, 22(3):57–75, 1993.

[66] J.P.Eakins, J.M. Boardman, and M.E. Graham. Similarity retrieval of trademark images. *MultiMedia*, 5(2), 1998.

[67] S. Kirkland. A characterization of spectral integral variation in two places for laplacian matrices. *Linear and Multilinear Algebra*, 52(2):79–98, 2004.

[68] J.B. Kruskal and M.Wish. Multidimensional scaling. Sage university series, Beverly Hills, California, 1978.

[69] S. W. Lee and J. H. Kim. Attributed stroke graph matching for seal imprint verification. *Pattern Recognition Letters*, 9:137–145, 1989.

[70] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[71] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[72] S. W. Lu, Y. Ren, and C.Y. Suen. Hierarchical attributed graph representation and recognition of handwritten chinese characters. *Pattern Recognition*, 24(7):617–632, 1991.

[73] S. Marini, M. Spagnuolo, and B. Falcidieno. From exact to approximate maximum common subgraph. In *Proc. Graph-based Representations in Pattern Recognition*, pages 263–272, 2005.

[74] M. Meilă. Comparing clusterings : an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007.

[75] R. Merris. Laplacian matrices of graphs: a survey. *Linear Algebra Applications*, 197/198:143–176, 1994.

[76] B.T. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12):1979–1998, 1999.

[77] P.-A. Moëllic, J.-E. Haugeard, and G. Pitel. Image clustering based on a shared nearest neighbors approach for tagged collections. In *Proc. Content-based image and video retrieval*, pages 269–278, 2008.

[78] B. Mohar. The laplacian spectrum of graphs. *Graph Theory, Combinatorics and Applications*, 2:871–898, 1991.

[79] B. Mohar. Laplace eigenvalues of graphs: a survey. *Discrete Mathematics*, 109(1-3):171–183, 1992.

[80] B. Mohar. Some applications of laplace eigenvalues of graphs, 1997.

[81] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Proc. Image Databases and Multimedia Search*, pages 35–42, 1996.

[82] M. Mortara, G. Patane, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38(1):227–248, 2004.

[83] D. Müllensiefen and K. Frieler. Optimizing measures of melodic similarity for the exploration of a large folk song database. In *Proc. International Conference on Music Information Retrieval*, pages 274–280, 2004.

[84] J. Norden. *England, an Intended Guyde for English Travailers*. 1625.

[85] S. Pati. The third smallest eigenvalue of the laplacian matrix. *Electronic Journal of Linear Algebra*, 8:128–139, August 2001.

[86] E. Pękalska, R.P.W. Duin, and P. Paclik. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2005.

[87] L. Peusner. A Graph Topological Representation of Melody Scores. *MIT Press Journals; Leonardo Music Journal*, 12(1):33–40, 2002.

[88] A. Pinto, R.H. van Leuken, M.F. Demirci, F. Wiering, and R.C. Veltkamp. Indexing music collections through graph spectra. In *Proc. International Conference on Music Information Retrieval*, pages 153–156, 2007.

[89] Alberto Pinto and Goffredo Haus. A novel xml music information retrieval method using graph invariants. *ACM Trans. Information Systems*, 25(4):19:1–19:44, 2007.

[90] H. Qiu and Edwin R. Hancock. Graph partition for matching. In *Proc. Graph-based Representations in Pattern Recognition*, pages 265–270, 2003.

[91] D.H. Rouvray and A.T. Balaban. Chemical applications of graph theory. *Applications of Graph Theory*, pages 177–221, 1979.

[92] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. In *Proc. International Conference on Computer Vision*, pages 59–66, 1998.

[93] P. Salembier and T. Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[94] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[95] S. Sarkar and K.L. Boyer. On optimal infinite impulse response edge detection filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(11):1154–1171, 1991.

[96] J. Schietse, J. Eakins, and R.C. Veltkamp. Practice and challenges in trademark image retrieval. In *Proc. Conference on Image and Video Retrieval*, pages 518–524, 2007.

[97] S. Sclaroff and A. Pentland. Modal matching for correspondence and recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(6):545–561, 1995.

[98] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The r -tree: A dynamic index for multi-dimensional objects. In *Proc. Very Large Databases*, pages 507–518, 1987.

[99] K. Sengupta and K. L. Boyer. Modelbase partitioning using property matrix spectra. *Computer Vision and Image Understing*, 70(2):177–196, 1998.

[100] L. G. Shapiro and R. M. Haralick. Organization of relational models for scene analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 4(11):595–602, 1982.

[101] L. S. Shapiro and J. M. Brady. Feature-based correspondence: an eigenvector approach. *Image and Vision Computing*, 10(5):283–288, 1992.

[102] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S.W. Zucker. Indexing hierarchical structures using graph spectra. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(7):1125–1140, 2005.

[103] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker. The hamilton-jacobi skeletons. *International Journal of Commputer Vision*, 48(3):215–231, 2002.

[104] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.

[105] T. Skopal. Unified framework for fast exact and approximate search in dissimilarity spaces. *Transactions on Database Systyms*, 32(4):29, 2007.

[106] A.W.M. Smeulders, M.L. Kersten, and T. Gevers. Crossing the divide between computer vision and databases in search of image databases. In *Proc. Visual Database Systems*, pages 223–239, 1998.

[107] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[108] W. So. Rank one perturbation and its application to the laplacian spectrum of a graph. *Linear and Multilinear Algebra*, 46:193–198, 1999.

[109] K. Song, Y. Tian, W. Gao, and T. Huang. Diversifying the image retrieval results. In *Proc. International conference on Multimedia*, pages 707–710, 2006.

[110] H. Sossa and R. Horaud. Model indexing: The graph-hashing approach. In *Proc. Computer Vision and Pattern Recognition*, pages 811–814, 1992.

[111] O. Symonova and R. De Amicis. Shape analysis for augmented topological shape descriptor. In *Proc. Eurographics*, 2007.

[112] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. *Systems, Man and Cybernetics*, 8(6):460–473, 1978.

[113] J.W.H. Tangelder and R.C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools and Applications*, 39(1):441–471, 2008.

[114] T.T. Tanimoto. Interal report 17th november. Technical report, IBM, 1957.

[115] A. H. Tewfik and M. Deriche. An eigenstructure approach to edge detection. *IEEE Trans. Image Processing*, 2(3):353–368, 1993.

[116] J. Tierny, J.-P. Vandeborre, and M. Daoudi. Reeb chart unfolding based 3D shape signatures. In *Eurographics*, pages 13–16, 2007.

[117] T. Tung and F. Schmitt. Augmented reeb graphs for content-based retrieval of 3d mesh models. In *Proc. Shape Modeling International*, pages 157–166, 2004.

[118] M. Turk and A.P. Pentland. Eigenfaces for recognition. *Cognitive Neuroscience*, 3(1):71–86, 1991.

[119] R. Typke, P. Giannopoulos, R.C. Veltkamp, F. Wiering, and R. van Oostrum. Using transportation distances for measuring melodic similarity. In *Proc. International Symposium on Music Information Retrieval*, pages 107–114, 2003.

[120] R. Typke, R.C. Veltkamp, and F. Wiering. A measure for evaluating retrieval techniques based on partially ordered ground truth lists. In *Proc. International Conference on Multimedia and Expo*, pages 128–135, 2006.

[121] R.H. van Leuken, M.F. Demirci, V.J. Hodge, J. Austin, and R.C. Veltkamp. Layout indexing of trademark images. In *Proc. Conference on Video and Image Retrieval*, pages 21–26, 2007.

[122] R.H. van Leuken, L. Garcia, X. Olivares, and R. van Zwol. Visual diversification of image search results. In *Proc. of the World Wide Web conference*, 2009.

[123] R.H. van Leuken, O. Symonova, R.C. Veltkamp, and R. de Amicis. Complex fiedler vectors for shape retrieval. In *Proc. of Structural and Syntactic Pattern Recognition*, pages 167–176, 2008.

[124] R.H. van Leuken and R.C. Veltkamp. Selecting vantage objects for similarity indexing. Technical Report 002, UU-CS Technical Report, 2008.

[125] R.H. van Leuken, R.C. Veltkamp, and R. Typke. Selecting vantage objects for similarity indexing. In *Proc. International Conference on Pattern Recognition*, pages 453–456, 2006.

[126] R. van Zwol. Multimedia strategies for b3-sdr, based on principle component analysis. In *Proc. Advances in XML Information Retrieval*, pages 540–553, 2006.

[127] R. van Zwol, V. Murdock, L. Garcia, and G. Ramirez. Diversifying image search with user generated content. In *Proc. Multimedia Information Retrieval*, pages 67–74, 2008.

[128] E. Vee, U. Srivastava, J.Shanmugasundaram, P. Bhat, and S.A. Yahia. Efficient online computation of diverse query results. In *Proc. International Conference on Data Engineering*, pages 228–236, 2007.

[129] R.C. Veltkamp and M. Tanase. A survey of content-based image retrieval systems. *O. Marques, B. Furht; Content-Based Image and Video Retrieval*, pages 47–101, 2002.

[130] R.C. Veltkamp and F.B. ter Haar. SHREC2007: 3D Shape Retrieval Contest. Technical Report UU-CS-2007-015, Utrecht University, 2007.

[131] J. Venkateswaran, D. Lachwani, T. Kahveci, and C. Jermaine. Reference-based indexing of sequence databases. In *Proc. Very Large Databases*, pages 906–917, 2006.

[132] J. Vleugels and R.C. Veltkamp. Efficient image retrieval through vantage objects. *Pattern Recognition*, 35(1):69–80, 2002.

[133] D.L. Wallace. Comment. *Journal of the American Statistical Association*, 78(383):569–576, 1983.

[134] S. Wang, F. Jing, J. He, Q. Du, and L. Zhang. Igroup: presenting web image search results in semantic clusters. In *Proc. Computer/Human Interaction*, pages 587–596, 2007.

[135] X. Wang, J. Tsong-Li Wang, K.I. Lin, D. Shasha, B.A. Shapiro, and K. Zhang. An index structure for data mining and clustering. *Knowledge and Information Systems*, 2(2):161–184, 2000.

[136] K. Weinberger, M. Slaney, and R. van Zwol. Resolving tag ambiguity. In *Proc. International Conference on Multimedia*, pages 111–120, Vancouver, Canada, 2008.

[137] P. Wilkins, P. Ferguson, and A.F. Smeaton. Using score distributions for query-time fusion in multimediaretrieval. In *Proc. Multimedia information retrieval*, pages 51–60, 2006.

[138] R.C. Wilson, E.R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *IEEE. Trans. Pattern Analysis and Machine Intelligence*, 27:1112–1124, 2005.

[139] R.C. Wilson and P. Zhu. A study of graph spectra for comparing graphs and trees. *Pattern Recognition*, 41(9):2833–2841, 2008.

[140] E. K. Wong. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25(3):287–303, 1992.

[141] D.M. Wuescher and K.L. Boyer. Robust contour decomposition using a constant curvature criterion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(1):41–51, 1991.

[142] S.A. Yahia, M. Benedikt, L.V.S. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. In *Proc. Very Large Databases*, pages 710–721, 2008.

[143] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. Symposium on Discrete Algorithms*, pages 311–321, 1993.

[144] F. Yizheng. On spectral integral variations of graphs. *Linear and Multilinear Algebra*, 50(2):133–142, 2002.

[145] B. Zhang, H. Li, Y. Liu, L. Ji, W. Xi, W. Fan, Z. Chen, and W.-Y. Ma. Improving web search results using affinity graph. In *Proc. ACM SIGIR*, pages 504–511, 2005.

[146] J. Zhang, K. Siddiqi, D. Macrini, A. Shokoufandeh, and S. J. Dickinson. Retrieving articulated 3-d models using medial surfaces and their graph spectra. In *Proc. Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 285–300, 2005.

[147] C.-N. Ziegler, S.M. McNee, J.A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proc. World Wide Web*, pages 22–32, 2005.

[148] S.W. Zucker. Region growing: Childhood and adolescence. *Computer Graphics & Image Processing*, 5:382–399, 1976.

Summary

The demand for efficient systems that facilitate searching in multimedia databases and collections is vastly increasing. This demand is raised within a large number of application domains, including criminology (face and fingerprint recognition), musicology (music information retrieval), trademark registration (automatic trademark retrieval), medicine (DNA fingerprinting) and image or video retrieval on the web. This thesis discusses content-based retrieval techniques that can be applied on these databases. The most important operation to support is query-by-example: retrieve items that are to some extent similar to a given query.

A common, high-level paradigm for this operation consists of the following steps. First, a digital representation is built for all the objects in the collections, that characterises the object's most important features. This representation can take different forms, for example vectors, graphs, point sets. To find a similarity or dissimilarity value between two objects, their representations need to be compared by a matching algorithm. The query can then be compared with every object in the collection to find similar objects. To speed up this computationally expensive process, this thesis presents a number of indexing strategies that can be deployed to prune the database.

In chapter 2 we study a specific indexing strategy called vantage indexing. This is a generic approach, not limited to a specific domain, because it only requires a (preferably metric) way of calculating a distance between the objects. With vantage indexing, objects are no longer compared directly, but it is investigated how similar their resemblance is to a set of reference objects: the vantage objects. In a sense, vantage indexing provides an embedding of a general metric space into a vector space. We focus particularly on the selection of vantage objects, since a good choice of vantage objects increases retrieval performance. We develop two criteria that are directly concerned with the retrieval performance, and present a heuristic algorithm that selects vantage objects according to these criteria and builds the vantage index simultaneously. Experiments with datasets of different size and modality demonstrate the scalability and show that the proposed strategy improves on existing methods.

In many applications, it is possible to represent the objects by graphs. For example,

a 3D model can be represented by a graph that resembles its skeleton, and a melody can be transformed into a graph whose nodes represent pitch classes that are connected when they occur consecutively in the melody. In chapter 3 we propose a specific indexing strategy for these scenarios. The assumption is that the graph's topology can be used to describe the underlying object. This topology is stored in a square matrix called the Laplacian matrix, and the sorted set of eigenvalues (spectrum) of this matrix is used as an indexing signature. To account for partial similarity, the difference in spectra of many subgraphs is considered as well. Both theory and experimental work support the claim that similarity in Laplacian spectrum predicts similarity between the original objects. The proposed representation outperforms existing methods in various application domains.

A possible limitation of the approach presented in chapter 3 is that two graphs with the same topology share the same representation, while the underlying objects may be different. In chapter 4, we enrich the graph representation therefore by storing additional object properties, that have to be reflected in the spectral representation as well. We develop a complex-valued analogue of the Laplacian matrix, a Hermitian matrix, and use its eigenvector associated to the second smallest eigenvalue as indexing signature. This eigenvector is known to be informative about the graph, and reflects the information stored in the matrix. Moreover, it can be reused to partition the graph into meaningful subgraphs, resulting in less subgraphs to compare for partial similarity. We provide an instance of this framework within the context of 3D object retrieval. We find that the enriched representation generally outperforms the previous one, and show in a specific example the added benefit of the enrichment.

In chapter 5 we extend the objective of a content-based retrieval system slightly. With content-based image retrieval on the web as example application, we claim that good retrieval results are not only relevant to the query, they should in fact reflect the diversity of the relevant objects that are present in the collection as well. First we retrieve images from a large collection based on their metadata: textual annotations and descriptions. The resulting images are topically relevant, but not necessarily visually diverse. We propose to cluster the retrieved images based on their visual characteristics and to show the most important image(s) from each cluster. This clustering should be query-dependent and unsupervised. Given a user query, we first dynamically determine appropriate weights of visual features, to capture the discriminative aspects of the resulting set of images that is retrieved. These weights are used in a dynamic ranking function that is deployed in a clustering technique to obtain a diverse ranking based on cluster representatives. We provide three lightweight and efficient clustering algorithms, and show that the algorithmic output coincides consistently with the results of a user study.

Samenvatting

De vraag naar efficiente systemen die het mogelijk maken te zoeken in multimediale databanken neemt toe. Er zijn vele toepassingsgebieden waar deze systemen gewenst of noodzakelijk zijn, waaronder criminologie (gezichts- en vingerafdrukherkenning), musicologie (music information retrieval), handelsmerk registratie (automatische logoherkenning) en beeld- en videoherkenning op het Web. Dit proefschrift bespreekt inhoudsgebaseerde herkenningstechnieken die kunnen worden toegepast op dergelijke databanken. De belangrijkste operatie die moet worden ondersteund is zoeken op basis van een voorbeeld: vind objecten die tot op zekere hoogte gelijkenis vertonen met een gegeven object als zoekopdracht.

Een veelgebruikt, algemeen paradigma voor deze operatie bestaat uit de volgende stappen. eerst dient er een digitale representatie gevormd te worden voor elk object in de databank, die de belangrijkste kenmerken van dat object weergeeft. Deze representatie kan vele vormen aanneemen, zoals vectoren, grafen, punten sets enzovoorts. Om een gelijkheidsscore of afstand te berekenen tussen twee objecten, dienen hun representaties vergeleken te worden door een vergelijkingsalgoritme. Voor een voorbeeldobject als zoekopdracht kan op deze manier ook een representatie gebouwd worden, en deze kan worden vergeleken met alle objecten in de collectie om gelijkende objecten te vinden. In dit proefschrift bestuderen we indexeringstechnieken, die dit rekenintensieve proces kunnen versnellen. Hierdoor hoeft de zoekopdracht niet met elk object in de collectie te worden vergeleken, maar wordt er automatisch al een snelle voorselectie gemaakt en kan een groot deel van de collectie buiten beschouwing worden gelaten.

In hoofdstuk 2 bestuderen we een specifieke indexeringstechniek, genaamd vantage indexing. Dit is een generieke aanpak, niet gebonden aan een specifiek domein, aangezien het alleen een (bij voorkeur metrische) afstandsmaat nodig heeft die gelijkheid tussen objecten uitdrukt. Bij gebruikmaking van vantage indexing worden objecten niet meer rechtstreeks met elkaar vergeleken, maar er wordt gekeken naar het verschil in gelijkheid ten opzichte van een set referentie objecten: de vantage objecten. In zekere zin genereert vantage indexing zo een inbedding van een generieke metrische ruimte in een vector ruimte. We concentreren ons op de selectie van van-

tage objecten, aangezien een goede keuze van vantage objecten de kwaliteit van het zoekresultaat verbetert. We definiëren twee criteria waaraan goede vantage objecten voldoen, die rechtstreeks betrekking hebben op de kwaliteit van het zoekresultaat. Tevens presenteren we een heuristisch algoritme dat vantage objecten uitkiest op basis van deze criteria, en tegelijkertijd de vantage index bouwt. Experimenten met dataset van verschillende grootte en aard tonen de schaalbaarheid van de methode aan, en laten zien dat de gepresenteerde selectiemethode tot beter resultaat leidt dan bestaande selectiemethoden.

Het is in veel toepassingsgebieden mogelijk om de objecten te representeren door grafen. Een 3D model kan bijvoorbeeld worden gerepresenteerd door een graafmodel van het skelet, en een melodie kan worden getransformeerd tot een graaf waarin de knopen de 12 tonen uit het octaaf voorstellen die verbonden worden door kanten indien ze opeenvolgend voorkomen in de melodie. In hoofdstuk 3 presenteren we een specifieke indexeringstechniek voor deze scenario's. De aanname is dat de topologie van de graaf gebruikt kan worden om het onderliggende object te beschrijven. Deze topologie is opgeslagen in een vierkante matrix genaamde de Laplaciaan matrix, en de gesorteerde eigenwaarden (het spectrum) van deze matrix kunnen worden gebruikt als indexeringssignatuur. De gelijkheid in spectra is dan een voorspeller geworden voor de gelijkheid van de objecten. Om ook gelijkheid in delen van het object mee te laten wegen, wordt het verschil in spectra van vele subgrafen ook berekend. Zowel theorie als experimenten tonen aan dat spectrale ge-lijkheid inderdaad een goede gelijkheidsvoorspeller is. De techniek geeft een beter zoekresultaat dan bestaande andere technieken in verschillende toepassingsgebieden.

Een mogelijke beperking van de techniek gepresenteerd in hoofdstuk 3, is dat twee grafen met dezelfde topologie dezelfde indexeringssignatuur krijgen, terwijl de onderliggende objecten zeer verschillend kunnen zijn. In hoofdstuk 4 verrijken we daarom de graafrepresentatie met additionele objecteigenschappen. Deze eigenschappen dienen ook terug te komen in de spectrale representatie. We definiëren een analoog van de Laplaciaan matrix in het complexe domein, een Hermitsche matrix, en gebruiken de eigenvector die hoort bij de op één na kleinste eigenwaarde als indexeringssignatuur. Deze eigenvector staat erom bekend verschillende eigenschappen en invarianten van de graaf te reflecteren en bevat de aan de matrix toegevoegde objecteigenschappen. Bovendien kan deze eigenvector efficiënt worden hergebruikt om de graaf te partitioneren in betekenisvolle subgrafen, waardoor er in totaal minder subgrafen vergeleken hoeven te worden. We geven invulling aan de Hermitse matrix in de context van 3D object herkenning. We concluderen op basis van resultaten dat in zijn algemeenheid de nieuwe representatie beter presteert, en illustreren

met een specifiek voorbeeld de verrijking van de representatie.

In hoofdstuk 5 verbreden we de doelstelling van een inhoudsgebaseerd zoeksysteem. Met inhoudsgebaseerde fotoherkenning op het Web als voorbeeldapplicatie, stellen we dat goede zoekresultaten niet alleen relevant zijn ten opzichte van de zoekopdracht, ze dienen ook de diversiteit van de in de collectie aanwezige relevante objecten te reflecteren. We presenteren een techniek die de zoekresultaten groepeert in clusters, en van elk cluster de belangrijkste foto('s) weergeeft. Deze clustering moet daarom zoekopdracht-onafhankelijk zijn en volledig automatisch. We bepalen gegeven een zoekopdracht eerst juiste gewichten voor verschillende visuele eigenschappen, om de onderscheidende karakteristieken van de set zoekresultaten te vinden. Deze gewichten worden dan gebruikt in een dynamische rankschikkingsfunctie die wordt toegepast in een clusteringtechniek om een gediversificeerde rankschikking te verkrijgen. We presenteren drie lichtgewicht en efficiënte clusteringstechnieken, en laten op basis van een gebruikersexperiment zien dat de resultaten van deze algoritmen overeenkomen met de menselijke perceptie.

Curriculum vitae

Reinier Henricus van Leuken was born in Leidschendam, The Netherlands, on September 18th 1981. He received his preparatory education at the Allfrink College in Zoetermeer from 1993 to 1999. In 1999 he studied Computer Science at Utrecht University, passing the propaedeutic exam in 2000 *cum laude* and receiving the MSc degree in 2005. His Master's thesis, entitled "Segmenting the human bronchial tree in multislice CT images of the lungs" was completed at the Image Sciences Institute located in the University Medical Centre Utrecht. The research for this thesis was carried out from 2005 to 2009 under the supervision of prof. dr. Remco Veltkamp, and was further supported by a visit to Yahoo! Research Barcelona during the summer of 2008.

Acknowledgements

First and foremost I would like to thank my advisor Remco Veltkamp. Remco, I greatly admire your clear mind that never loses track of the aspects that really matter. Whenever I got stuck, you managed to get me going again in no-time after a short, keen analysis of the problem at hand simply by asking me a few key questions. Never have I heard from you that you were too busy to discuss whatever issue. Thank you for the confidence you have always expressed in me, and for giving me so much freedom in my research while keeping a reassuring close eye on me at the same.

During the course of my PhD I have worked together with many people, a fact for which I consider myself to be lucky. I want to thank all my coauthors, and in particular Fatih Demirci and Olga Symonova who have both been important in the development of this thesis. Fatih, it was you who introduced me to the field of spectral methods when we started to work on trademark layout. Thank you for providing me with this solid base for future research, that has given me many interesting results and insights beyond trademarks. Olga, the effortlessness of our collaboration still amazes me. We've come a long way shaping the research that now spans the fourth chapter of this thesis, carrying out some very difficult and involved experiments. You stayed motivated and never lost hope, even when everything seemed to be lost. Thank you for being so persevering during our absorbing work together.

I'm also greatly indebted to Roelof van Zwol for inviting me to Yahoo! Research during the summer of 2008. I've had a terrific time in Barcelona, both during and after work, where I found a motivating and inspiring research environment. I thank you and all the others who received me with such great hospitality, for giving me the opportunity to share the Yahoo!/Barcelona experience during those months.

A word of thanks goes to all the members of the reading committee, prof. dr. H. Alt, prof. dr. ir. R.L. Lagendijk, prof. dr. Th. Gevers, prof. dr. A.P.J.M. Siebes and prof. dr. J. Austin, for helpful suggestions and comments.

I'm thankful to all my colleagues from the *Multimedia and Geometry* and *Games and Virtual Worlds* groups at the ICS department. Frank ter Haar, I have enjoyed the years during which we shared A203 a lot. Thanks for all the good advice, fun and strange-noise-tolerance. You were the perfect office mate in all respects. Thank you: Geert-Jan for all the good laughs and the programming advice; Arno for taking beautiful pictures (both of my bands and for my thesis cover) and for always beating me at Foosball; Dennis for always loosing with me at Foosball from Arno and for all the good times we had (and continue to have!); both Esther Moet and Frank van der Stappen for all the discussions we had that usually moved swiftly from very fun to very serious things and back; Martijn for still making me smile when I think of your wisecracks on the smaller and bigger things in life, I have always regretted (but understood) your decision to move on / abroad; Bas for being the other jazz enthusiast in the group and thank you to all the other people from the groups: in the end, you make a working environment into a pleasant place to be.

I'm proud to have Rodrigo Silveira and Reinier Meerwaldt as paranymphs during the defence of this thesis. Rodrigo, you've been a terrific colleague. When I was really stuck or really tired or really disappointed or really excited about results, you were always there for a talk, that more often than not ended up taking all afternoon. More importantly, you have become a dear friend outside work and I hold great memories to the trips we made. Reinier, I think one has to search hard and long to find two people who studied more intensively together than we did in the years 1999-2004. It makes great sense to me that you are a paranymph during the defence. You're one of the most intelligent, honest and positively insane guys I have ever met.

I want to express my gratitude to my parents and my sister Renske for their unconditional love, wine supply, dvd supply and all other kinds of support that kept me going the last four years. Besides, I would not have finished this thesis without having some of my father's drive in me, combined with my mother's calmness and sense of reality.

Every thesis needs a muse, every acknowledgement a finale. Marjolein, you have inspired me during these years beyond words, while working on this thesis and in every aspect of life. I look forward to all the years to come.