INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Servidor de Multimédia Baseado em Conteúdo para Artistas

André Filipe da Silva Veríssimo

Dissertação para obtenção do Grau de Mestre em

Mestrado em Engenharia Informática e de Computadores

Júri

Presidente:	Prof. Dr. Joaquim Armando Pires Jorge
Orientador:	Prof. Dr. Andreas Miroslaus Wichert
Co-Orientador:	Prof.ª Dr.ª Helena Isabel de Jesus Galhardas
Vogais:	Prof. Dr. Manuel João da Fonseca

Outubro de 2009

Abstract

Art portals are repositories for large multimedia collections that, in most cases, rely exclusively on annotations written by the artist itself or in collaboration between several users for image search. We plan to demonstrate an alternative method for querying a multimedia collection for similar image by using the image's content, called content-based image retrieval. To accomplish this goal we use relevant information about the image from raw pixel data without human input and interpretation.

The system implements a fully functional web portal offering artists the opportunity to display their work online to a broader audience. The portal will feature an image hosting service, keyword and content-based search options, a personal page for each user and all the necessary tools to manage the profile and respective artwork.

In this thesis we focus on content-based image retrieval and high-dimensional indexing techniques such as the subspace tree indexing. We implemented a method that creates a pyramid of different dimensional spaces using a mapping function to generate them. This method speeds up the retrieval of similar objects by progressively discarding dissimilar objects as it visits the higher dimensional spaces.

The dissertation starts with a state of the art and contextual overview of content-based image retrieval methods and high-dimensional indexing techniques. We then documented the architecture and development of the resulting application and implemented indexing techniques. In the last chapters we describe our findings when testing the content-based search methods efficiency and viability and analyse the results.

Keywords

Content-Based Image Retrieval, High-Dimensional Indexing, Art Portal, Hierarchical Linear Subspace, Orthogonal Projection, Principal Components Analysis

Resumo

Portais Web com conteúdos para artistas são repositórios para grandes colecções de imagens que, utilizam para a pesquisa de imagens as anotações escritas pelos artistas ou em colaboração com diferentes utilizadores. Nesta dissertação pretende-se demonstrar um método alternativo para a pesquisa de imagens utilizando o seu conteúdo de forma a retornar imagens semelhantes - *content-based image retrieval*. Para isso são utilizadas as caraterísticas da imagem sem qualquer intervenção ou interpretação de pessoas.

O nosso sistema implementa um portal Web que proporciona a artistas a oportunidade de exibir o seu trabalho online para uma maior audiência. O portal oferece aos utilizadores um serviço de hosting de imagens, pesquisa por keywords e por conteúdo, uma página pessoal para cada artista registado e as ferramentas necessárias para gerir o seu perfil e portfolio de imagens.

Na dissertação é focada a pesquisa de imagens por conteúdo bem como técnicas de indexação de objectos com uma dimensionalidade muito alta, como o *subspace tree indexing*. É implementado um método que cria uma pirâmide de subespaços, cada um com diferentes dimensões. Cada subespaço é gerado utilizando uma função de mapeamento. Este método diminui o tempo necessário para cada pesquisa ao descartar objectos muito dissimilares à medida que vai percorrendo os vários espaços dimensionais mais elevados.

A dissertação começa por fazer o enquadramento teórico de forma a contextualizar o trabalho, seguindo-se a descrição da arquitectura e desenvolvimento da aplicação. São também documentado os métodos de indexação implementados, bem como os testes efectuados. Concluindo com uma analise aos testes e trabalho desenvolvido, e respectivas conclusões.

Palavras Chave

Content-Based Image Retrieval, Indexação de Objectos de grande Dimensionalidade, Portal de Arte, Hierarchical Linear Subspace, Orthogonal Projection, Principal Components Analysis

Part I

Dissertation

Contents

List of Figures

List of Tables

1

Introduction

Contents

There is an increasing number of portals on the web that host and present large collections of images, which range from stock photos and photography to digital art. These collections are very diverse and provide with search tools that mostly rely on information written by users. Image search engines have not kept up the pace with the collections they are searching and with user's necessities, focusing on the traditional keyword-based query that looks into text description when indexing, instead of the actual content of the image, objects, people, location, etc.

In this thesis we developed a framework that hosts an art portal and performs content-based queries on the collection. But the main focus is the high-dimensional indexing techniques it implements and compares to research which are the best methods when dealing with very large objects (up to the hundreds of thousands dimensions).

1.1 Motivations

Multimedia content is growing at an incredible pace over the internet, web portals like flickr and deviantART deal with millions of images and several hundreds of page views every second. Consequently managing the database and providing with effective search and indexing methods is a high priority. Content-based search methods emerge to help address the limitations of traditional search queries that rely on textual annotations and descriptions subject to human interpretation.

While users want to find images containing particular objects [1] content-based image queries are still not being implemented in mainstream web portals, instead relying on textual descriptions that can be incomplete and subjective to interpretation. The known portals that provide content-based search only offer this search method as a refinement for the results and not for the query itself. On the other hand there are museums and start-up projects that offer more advanced search options that query the image by content.

Our main motivation is to investigate and compare high-dimensional indexing techniques that can handle objects with dimensions up to the several hundreds dimensions. Most developed and researched techniques can only scale up the order of hundreds of few thousands dimensions, such as tree-indexing or space filling curves [2–6]. The retrieval access time of these techniques deteriorate their performance rapidly if the objects' dimension exceeds the limitation. Our goal is to test a hierarchical subspace method that scales beyond the limitations of the r-trees and kd-trees based methods.

We also present a fully functional art portal framework that presents the user with several methods of searching the collection, combining keyword and content-based search on image collection. In addition to its advance search capabilities the art portal will present other features to offer users, we pretend to implement user accounts for artists so they can exhibit their artwork on the internet to a broader audience. Each artist will have a personal page that will display their information and work allowing users to view the artist's collection, curriculum vita and exhibitions.

This framework will allow for the testing of the indexing methods and their performance, namely when retrieving the objects from the collection. It will perform high-dimensional indexing of images based on their content and keywords provided by the artist. By combining these two search methods we plan to extend the traditional type of query and offer the more accurate results.

The textual information that will be indexed is provided by the artist and includes the artwork's title, description, theme and descriptive keywords. On the other hand the data that is used to search by content is extracted automatically from the image without any user input making this type of query more objective. Although the goal is to read semantic content of the image, it is not possible using current technologies and existing research on a broad domain, therefore we have to rely on low-level information to search the image collection by content, mainly raw colour information. By using this information we can generate very large objects with dimensions beyond the order of thousands.

Content-based image retrieval (CBIR) will provide the user with additional search tools that will search the image itself instead of attached information, which is done by extracting data from the image, such as features or in more advanced implementations semantic objects, and compare against the given query. An image or desired features can be used as an example and the search engine will search for all images that conform to the search parameters.

1.2 Objectives

The main purpose of this master thesis is to compare different methods to perform content-based image retrieval, and answer the question „which the best method to implement CBIR queries „. For this purpose we will be testing several methods, (i) list matching (ii) histograms (iii) hierarchical subspace method using the orthogonal projection to reduce dimensions and also (iv) principal component analysis as a mapping function for the hierarchical linear subspace. The experiments should give a definite conclusion on the best method for this type of image artwork and also give some insight in general when comparing methods that use a hierarchical set of subspaces to produce the final results.

We also develop a scalable web portal framework for art content that can use the implemented CBIR techniques to search a large art collection. It should be fully implemented and tested using a very large collection of images in order to test both the efficiency and effectiveness of the system, and also research different methods for CBIR queries. The portal should also implement a search method that is able to search the database by keywords.

With this system the user will allow the user to find similar images, given an input that can be either a personal and private image or an existing image in the collection. This functionality will improve the user experience on browsing through collections, because when the user finds an image he can search for similar images to find what he likes best instead of relying on the categorisation written by users. The purpose of the CBIR component in this thesis is to research if the implemented methods can scale to large collections and determine how they perform, both in terms of efficiency and effectiveness.

The system must provide a simple and modular CBIR framework that is independent from the web application allowing it to be use used and ported in future works. The expected result is web portal that allows for anonymous users to browse and query the collection for similar images and for artists to register, submit images, create exhibitions and manage their portfolio and account. The portal must comply with web standards and have a good usability that translates to a simple

and straightforward browsing.

1.3 Solution

We implemented several high-dimensional indexing techniques that were tested and compared, they are:

- List Matching;
- Colour Histograms;
- Hierarchical Linear Subspace;
  - Orthogonal Projection;
  - Principal Components Analysis.

The list matching method queries every image on the original space, and while giving the exact distance between all images it is very slow and needs extensive resources to return results in real-time.

The colour histogram method is the fastest method as it indexes the image using a transformation that greatly reduces the dimensionality of the collection. Nevertheless, it is not an effective method as even small differences in the lighting may result on identical images, in content, be considered dissimilar with this method.

The hierarchical linear subspace method is, according to our experiments, the method that demonstrate the best results, both in performance and in the quality of the results. With this method we tested two types of mapping functions to create a pyramid of subspaces, (i) orthogonal projectional that uses the mean value of pixels to reduce the dimensionality of the images; (ii) principal components analysis that finds the correlated variables in a image collection and then is able to reduce the dimensionality by discarding them.

Our solution was developed based on a client-server architecture that will serve the artist community on exhibiting their work and making it searchable not only by text and keywords, but also by its content. The application stores the user's multimedia images and then creates a fingerprint that holds the image low-level features, namely colour distribution, and uses this information to index the multimedia images. The application offers any user that accesses the platform to query the collection using a content-based image retrieval query by example method.

The application implements all the necessary functionalities of an art portal that allows for the basic operations, such as image uploads and user registration, but also provides with more advanced features. Among them, we can highlight the exhibition management and the content-based image retrieval search methods. As with any art portal it is possible to browse the image and user collection, along with the most popular artwork.

The informational architecture evolved during the development of the application and keeps all the relevant metadata and information on the database while keeping the largest objects in the file system, the images and content-based image retrieval data. In order to keep a modular system and separate the artists and image information from the content-based image retrieval module we extended that separation to the information. Therefore in database, as in the file system, we keep

the image's data separate from the content-based image retrieval data, with the information being loosely coupled.

As mentioned above we required to keep some objects in a file system, namely the original, thumbnail and preview image, along with the generated data for the hierarchical linear subspace data. Due to its very small size the histogram data was kept in the database. In the filesystem we only kept the large binary data that was too big to be store in the database (the orthogonal projection data takes up more than 6 GB of disk space).

The application itself was developed as a client-server architecture with a tree layer system that allowed us to create a modular and very flexible system. This type of architecture also allows for reducing the maintenance costs over times as it has the functionality encapsulated and reutilises many functionalities (avoiding redundant code). The separation of layers comes naturally as the application was developed on top of the WebObjects framework, which design supports this type of architecture. The application is deployed on a central server that in case of increased load, the framework allows to spread the load over several servers. The user accesses the application through a web browser just as it would with any other web portal.

1.4 Structure of the document

The dissertation is structured in six chapters (not including indexes and appendixes):

Chapter 1 - Introduction (current chapter): In this chapter we describe the work and researched that was done for the master thesis and the motivation that precluded the work. We also detail the initial goals and the problem it proposes to solve;

Chapter 2 - State of the art: The chapter documents the researched work that has been done in the problem's area along with the current implementations of concurrent solutions and a analyse of their success and fails;

Chapter 3 - High-dimensional Indexing Techniques: This chapter documents the researched algorithms for high-dimensional indexing techniques that we implemented in our solution;

Chapter 4 - Software Architecture: In this chapter we present a description of the implemented architecture as well as an account of the development steps. We include the algorithms that we implemented in our solution, the information architecture used as well as the final application architecture;

Chapter 5 - Results: The experiments that were done in the scope of the thesis is documented in this chapter with all the relevant data for those experiments. We test the different algorithms implemented for the content-based image retrieval methods;

Chapter 6 - Conclusion: In the final chapter we present the final conclusions that resulted from the research and work that was done. Accordingly we analyse the work and what can be done from here and present the ideas and suggestion that came up during the master thesis but could not be materialised in the time frame.

2

State of the Art

Contents

A search query by definition is a method that will search for information by using a set of terms describing what the user wants. It is an operation that needs to be transparent to the user in a way that is fast and returns relevant results to that query. The search engine that powers the query is very complex, it needs to retrieve information from sources, a database or the internet, as they are updated. Then analyse the information calculating its relevancy, and finally provide with fast access to the data it has indexed.

Companies such as Google have become very successful by indexing websites, media and every kind of information that on the internet. The most common search query is a search by keywords, i.e. provide the search engine with a question or a combination of words and it returns information that considers relevant to that query. This is the traditional way of searching web content and it can be used to search for all types of information, textual or other media, as long as there is a method to extract a textual description that can be indexed, for instance, videos can be indexed by using the metadata information attached to it.

To optimise the retrieval and classification of data it is used additional information that accurate describes the content, this information include special fields, such as the title of a webpage or in the file itself, and it is called metadata [7]. An example of metadata present in a document would be the authors, title, subject, comments, language and tags. This kind of information will help systems to categorise the document, but adds an additional problem, as it would depend on humans.

Writing metadata is a long process, because it is done by people that need to read and analyse the data and then extract the relevant information. It is a process that heavily depends on the interpretation by human beings and when dealing with ambiguous content each person can have different opinions making it an unreliable method in some situations. It is also an expensive process due to the amount of information that is generated by applications and on the web, consequently having people interpret and write metadata is an extremely expensive and time-consuming process that should only be used in specific situations and automated [8], if possible.

The work being done in this field has the objective of creating standards to make metadata more objective and guarantee interoperability [9], it could then be used by search engines to index multimedia and other types of content [10]. Among the different metadata proposals that have been published there is MPEG-7 [11], EXIF and ID3, which provides standards in multimedia metadata.

The MPEG-7 standard, which is the most used one, will provide with a structured description on multimedia content (audio, video, picture, graph, 3D models, etc.) offering interoperability between systems and a flexible framework that can be used to provide fast and efficient annotation and searching methods [12]. The goal of this project is that once the content is created a description is attached to it and then distributed, making it accessible for anyone to read, query and edit it.

Search methods can be extended by using structured search terms that can give detailed information of what the user is looking for; it can be done in several ways:

- Adding specific information to the search terms by using a structured query, this will help filter the results, by explicitly saying what to include in the results (author, source, location, size, colour distribution, patterns, etc.).
- Allowing the user to combine several different types of keywords when searching, i.e.

using textual information with other data and features, such as images, sounds and videos.

By combining the keyword-based search method with these, search engines will have more information and return better results by cross-referencing the different terms and return more accurate results. Although these methods help increasing the accuracy of the search query there is a fundamental problem with any type of search method, and that is because the search engines do not understand the meaning of the queries and the indexed content.

This limitation is called the semantic gap [13–15] and while computer systems are able to read information they do not necessary understand the content. Research is being done in order to close the gap [16, 17] but there is still a long road ahead until it is possible for a search engine to interpret a query and understand what the user wants it.

2.1 Content-Based Image Retrieval

Content-based image retrieval tries to solve part of the problem and become a method for searching multimedia images by their content, in oppose to search by keywords and metadata. Current systems and search engines focus almost exclusively on textual information to search data, regardless of what type of data it is. Multimedia objects, such as images and videos, are indexed by the metadata written by a person or by the text that it is adjacent. This form of indexing, though cheap and fast, comes at a high price returning inaccurate results.

Search engines are already introducing content-based image retrieval in their searches and many other solutions are emerging. Content-based image retrieval by itself is not the solution, at least not at this moment. As the keyword-based search engines that can not understand the text they are indexing, content-based image retrieval systems still need time to perfect the feature and object extraction. In other words understand what the image represents.

As with text this is a very difficult task due to number of domains that exist and in the variety of objects that can be captured in images by photographs, paintings, etc ... We can start to consider just the human beings that are represented in multimedia images. In this domain we have to consider the anatomy of the human body, from birth to old age, man, woman, tall, short, thin, fat , injured (missing limbs) and every other characteristic. The system must also be able to recognise a person from every angle.

All this presents a large problem as it depends on too many variables, especially if we start considering all the different types of objects that can be represented in an image, people, animals, locations, objects, planets, stars, etc ... and every combination of them. The complexity is even greater if the art domain is added and abstract art, cubism, impressionism, portraits, etc ... are also considered.

The problem is too big to handle and for now the work that is being done is to create generic algorithms that start to handle some of the subdomains, for example facial detection. But most current systems still rely on colour, shape, texture and colour layout to query by content. Some search engines are introducing additional refinement of the results produced by a traditional query by adding options to search for:

- Similar images;
- A specific domain, that can range from high-level features such as images that include faces, head and shoulders, human figure, or low-level features as layout, size and colour;
- A specific style, i.e. illustrations or real photographs.

Other systems are using actual images or sketches of what the user is searching for as query terms and then search by content. An alternate solution for this problem is to implement a system that can extract patterns and given a training set for each object, it can learn and then apply that knowledge when extracting information from a multimedia image [18, 19].

There is a lot of potential in this area and as more work is done better results can be expected from search queries. And although it is difficult to remove human input from the process we can minimise their negative impact, i.e. the subjectivity, by implementing objective criteria in applications to categorise images and their objects instead of relying on metadata and textual descriptions.

2.1.1 Search by Content an Art Portals

An art portal will deal mainly with multimedia images, either digital art or photographs, and if successful it may have to manage millions of images increasing the complexity on the backend. but it may also become difficult to browse the collection as there are misspelled keywords or wrongly assigned A content-based search will help dealing with this problem and also improve the search capabilities of the portal.

Usually an art portal relies on the metadata information attached to each image when searching and organising content, that information can be retrieved in one of two ways, from the image itself or added at the upload process. It is very useful to the portal as it will be used to index the images, queries, classification and to display additional information on the portal complementing the image. The downside of using metadata is that it is not always accurate, complete and makes browsing ineffective when dealing with a large number of images. Content-based search can solve part of the problem, in this case a method that will extract information from an image and used it to query and group similar images in categories.

Content-based image retrieval (CBIR) will index the image collection and then query each image by its content. Making it a very effective search method for searching and offers tools to improve browsing. This type of query will look into each image's content and extract the features representing the image and aggregate them in one data structure, a signature. It can be reused and is create only once, usually when the image is uploaded. Then it is stored in the database allowing the system to compare the signatures instead of the whole image, which reduces the load and optimises the query process. Among the characteristics that can be extracted are:

- Colour information, it can be extracted by using histograms or colour-correlation.
- Shape, the system will look for known shapes on the image, usually after applying an edge detection algorithm such as sobel operator.
- Texture, which can be detected after applying a transformation function that can detect frequency.

- Size and related information such as orientation and layout.
- Objects, this is a high level feature that in order to be implemented needs an algorithm that understands what the content represents.

These properties can represent the global image or, in more advanced methods, regions [20]. The data structure that is used to group this information is called a feature vector, which consists on a vector with variable dimensions depending on the number of extracted features of the image. These feature vectors can reach a very high dimension and have to be reduced to a smaller one, because the smaller the dimension the more efficient is the query, index and storage process. To achieve this it is necessary to apply a method that reduces the spatial dimension of the vector, such as a transformation function or an indexing algorithm. It can be done by Discrete Wavelet Transform (DWT), Discrete-time Fourier Transform (DTFT), Principal Components Analysis (PCA) [21,22], R-Trees [23],Generic Multimedia Indexing [24] and Hierarchical Linear Subspace Method [25].

Essentially there are two different methods for searching content, either look for specific features or search for similar for image a given one by the user, both methods can use the feature vectors for the query or the image itself:

- Query by features, which will lookup features in images, such as colour, size, orientation, textures and shape, and include in the result images that have them. With this type of query it is possible to look for images that have predominant colours or a specific shape.
- Query by example, this type of query will use an existing image to perform a query on the collection, and then return a set of images ordered by similarity. This query has many possibilities because it can be used not only to search for similar images, but also to look for altered versions of an image.

There are other applications for CBIR besides queries as it can be used for recommending similar artwork or organising the images into categories by content. The portal can take advantage of clustering techniques from R-Trees, or other indexing methods, and use them to retrieve groups of images and recommendations for the users, for example to display thumbnails of similar images when browsing a page.

The current problem in CBIR is that there still exists a gap between the information that is extracted and the real semantic value of an image [13–15]. Methods still rely exclusively on low-level features and there still is no real interpretation of the content [26]. Simple methods for image enhancement, that for instance, changes the brightness of an image maintains the semantic value, but changes the colour information drastically. In the perspective of CBIR system the images can be considered very dissimilar, if the wrong feature is analysed. This occurs because the loss of information when extracting low-level features, namely semantic, will make the feature vectors of the images become dissimilar. The best solution for the problem is to actually extract the semantic information. If this is not feasible, then the focus should be on extracting several different features in order to avoid relying on just one type.

2.2 Web Portals and Content-Based Image Retrieval implementations

One of the first content-based image retrieval systems, IBM's Query by Image Content, was developed in the early 90's featuring colour, shape and texture extraction, but since then a lot of research has been done and this area evolved with some of the top web search engines already implementing this type of search.

While Google [1] and Bing [2] have started to introduce search refinement with content-based information, smaller search engines have emerged that implement content-based searches based on query images. Even social networks offer their users applications that search their friends photos for similar ones. Others create a business around content-based image retrieval giving artists and publishers the tools for finding unauthorised use of licensed material in magazines and on the web.

Regardless of the purpose of the system, many have systems have emerged over the years that use different approaches but with the same goal, the query of a multimedia image source collection for similar images that present common features. The next sections describe the existing systems and some real world implementations.

2.2.1 Content-Based Image Retrieval Systems

The existing CBIR systems available range from open source to proprietary solutions that extract many types of features from images in order to process queries, among those systems are;

- IBM's Query By Image Content (QBIC).
- GNU Image Finding Tool (GIFT).
- Semantics-sensitive Integrated Matching for Picture Libraries (SIMplicity).
- Visual Object Retrieval and Thesaurus Extension (VORTEX).
- Blobworld.

In the following paragraphs we present a description of algorithms behind the systems and their capabilities.

IBM's Query By Image Content

The best known content-based image retrieval system is IBM's query by image content (QBIC) search system [24, 27, 28]. This system uses the image's low-level information like colour, shape and texture to create a feature vector that holds the relevant information about the image will effectively reduce the amount of data needed for querying. QBIC presented the generic multimedia indexing approach (GEMINI) to reduce the dimensionality of the objects that in turn, will reduce the number of computation calculation needed to process a query. The idea behind GEMINI [24, 28] is to find a feature extraction function that maps the high-dimensional objects into a low dimension space. In this low dimensional space, a so called 'quick-and-dirty' test can discard the non-qualifying

[1] Google Similar Image from Google labs – `http://similar-images.googlelabs.com/` (last accessed July 17th, 2009)

[2] Microsoft Bing.com – `http://www.bing.com/` (last accessed July 17th, 2009)

objects. Objects that are very dissimilar in the feature space are expected to be very dissimilar in the original space. IBM's QBIC is currently deployed in the Hermitage [3] online museum and supports two different types of CBIR queries, by colour or layout.

GNU Image Finding Tool

The GNU image finding tool (GIFT) is an open source project and offers an extendable framework to search image collections by content [4]. It is based on the VIPER (Visual Information Processing for Enhanced Retrieval) project [5] relying solely on the content of images and is capable of weighting each feature relevance by calculating its frequency in the image and collection, it also gives the opportunity for the user to improve the query results by relevance feedback [29]. GIFT uses query by example to produce results and has been used mainly in medical systems, medGIFT [30, 31].

Semantics-sensitive Integrated Matching for Picture Libraries

This is an image retrieval system that uses semantic classification methods, feature extraction and segmentation of regions [18]. Each image is represented by regions, corresponding to objects, which are categorized by colour, texture, shape and location, and classifies it to a set of categories, such as textured / non-textured, graphs and photographs. The goal of this system is to narrow down each of the domains and be able to extract the most relevant features for each.

Visual Object Retrieval and Thesaurus Extension

The VORTEX approach [19] proposes a system which combines techniques from Computer Vision, such as feature extraction, with a thesaurus for objects and shapes. The system includes three components, an image collection, a thesaurus and a search engine, where each image stores a feature vector that is automatically created and has colour, shape, texture and spatial information. On the other hand, the thesaurus keeps a hierarchical list of shape templates for objects that needs to be maintained by humans. Each object is associated with different possible shapes the object can have, for example in different angles.

Blobworld

Similar to SIMplicity in the approach, it divides the objects in the image into regions allowing for queries at the object's level for better results [20]. The image representation is created by clustering pixels in a joint colour-texture-position feature space, where the segmentation algorithm is then applied to find all objects in the image. A particularity of this system is that it allows the user to review this representation in the results so that he can understand the results and better adjust the similarity metrics.

[3] The State Hermitage Museum, St. Petersburg, Russia – `http://www.hermitagemuseum.org/` (last accessed July 17th, 2009)

[4] GNU image finding tool (GIFT) home page – `http://www.gnu.org/software/gift/gift.html` (last accessed December 4th, 2008)

[5] Universit de Genve, Facult de Sciences – `http://viper.unige.ch/doku.php` (last accessed December 4th, 2008)

2.2.2 Art portals

The art world is very comprehensive as it contains a very wide range of artworks that used to be displayed in museums, galleries or on private collections. However, nowadays in the digital era, we can easily browse through the collections online and by the means of a user-friendly portal. These portals serve different purposes, some are created by museums to display their collections and attract a new audience, others are community driven that display and commercialise the user's work.

Each art portal has different features, but most are built with a common set of functionalities and focused on the the user:

- The display of artwork and related information in a user-friendly interface.
- The classification of the artwork in categories, collections, etc.
- Methods to search for content.

All these features have to be supported by a well implemented platform that can handle the heavy load associated with multimedia content. Having to deal with multimedia images is very different from text. First of all, because of the differences in size that represents a higher complexity in terms of storage, delivery and presentation of the content. Then there is also the indexing problem, whereas text can be indexed and searchable for keywords, multimedia content cannot. For traditional queries, i.e. search by keywords, a multimedia object must rely on metadata information attached to it and ultimately on the person that generated it. And unless there is a system that can analyse the content and return its semantic description we will have to continue to depend on the artist input for keyword searching. The alternative type of query is CBIR approach that extracts the images' low level features and creates a feature vector, which can contain colour spatial information, shapes or textures, and then use this information for indexing.

When developing an art portal there are some features that need to be well design in order to support its growth, these are:

- Security
- Storage
- Scalability
- Efficiency

Although the nature of the information and data is not necessary confidential, the need for security still persists, personal data from the users need to be protected as well as their intellectual property, the artwork. There is a need for a well designed and robust architecture for both the physical and application backend. The infrastructure has to be secure from attacks, the databases and storage should be protected by firewalls, their access and manipulation should be restricted. Two other security measures that should be implemented are accountability and traceability, it is as important to protect the data as is to trace how the data is accessed in case of an unauthorised access.

Storage is an essential part of the platform, because it will be dealing with large sized objects that need to be stored and retrieved in real time, either for queries or streaming [32]. To achieve

this it is necessary to make some decisions, a common solution to get a better performance is to use compression in order to reduce the size of the media objects, but some quality can be lost. There are pros and cons with every choice that is made but in the end what is really needed is a compromise that is both fast and delivers quality.

While developing a web application it is important to make it flexible and scalable, for instance, it must perform well if it only has hundreds of regular users or hundreds of thousands. Recent history shows that websites can grow exponentially and keeping up with this growth can became very difficult. If the infrastructure and applications are not ready for it [6].

These features amount to one fundamental characteristic, which is efficiency, because it is how the user evaluates the service. All of these problems must be transparent for the user as his only concern is that the portal does what it has set up to do and does it fast.

On top of these functionalities and features the portals need to find their niche and develop new characteristics based on their target audience and goals. For example, museums will provide with virtual tours of their collections and accurate information on current exhibits and collections on display in their facilities. On the other hand it is the interest of user-based websites to take advantage of social networks and web 2.0 [33] concepts to bring users together. It all comes down to what users want now and be able to predict what their future needs are.

In the following sections it will be described successful art portals and others that implemented content based image retrieval as a search method.

deviantArt

The deviantART portal [7] is one of the most successful art portals today, with over 8 million users and 62 millions submissions ranging from traditional art to literature and videos. It is a community-based portal that displays virtually all types of art (paintings, sculptures, digital art, logos, photography, animations, movies, typography fonts, literature, etc.) either by image, movie or compressed file and makes it easily accessible.

In addition to the main features described above, there are others that will stimulate the submission of quality artwork. The first are the public contests, in which the users are invited to submit their artwork for evaluation from the community or from an expert panel. The winners will receive a prize, usually in money, and gain reputation within the community making these contests very popular.

There is also an online shop where people can buy prints of submitted artwork and deviantART merchandising, along with advertising and s are the portal's source of revenue and although it is visible while browsing it is well integrated with the site's design.

Each user has its own subdomain URL that displays all of his artwork and relevant information. The homepage collects the user's personal information and relations, such as statistics on his activity, favourite artwork from other users and who is following his work. Users can also chat in real time and share ideas without having to use external applications for it delivering a rich user

[6] Seattle Conference on Scalability: YouTube Scalability, Cuong Do – http://www.youtube.com/watch?v=ZW5_eEKEC28 (last accessed December 8th, 2008)

[7] deviantART portal, where ART meets application – http://www.deviantart.com/ (last accessed December 3rd, 2008)

experience. This web portal understands today's need for connectivity and collaboration. It does so by exploring web 2.0 concepts allowing for the user to not only use the portal but participate and add value.

Due to its success deviantART has grown and so has its traffic, currently there are eight million registered users, of which at least thirty thousand are online at any given time, and fourteen million unique hits per month [8]. This numbers translate to 300 page views per second that need response from a backend that is prepared for that load and for future growth.

DeviantART's portal is one of the best examples on what features an art portal should have and how it should look like, its interface is well designed and gives the user a good experience and its features are what users expect on this web 2.0 age. One feature that is missing from this art portal is a CBIR method for users to use, in order to search for similar imager or just for a set of characteristics. Now it only implements two different types of search based on words:

- Keyword search, that looks for matches on the artwork's name, artist's comments and on a set of words that describe the artwork
- Category search, which allows for the user to either browse through the different categories or to filter the results of a keyword search.

Due to the size of the database, with more than 62 million artworks, it would be interesting to test the performance of a CBIR method, as it would put to the test how efficient and scalable the method really is.

flickr

Flickr is an image and video hosting site that operates in a larger scale than deviantART, it has recently reached the 3 billion images mark [9]. The video hosting service is something new, but their main focus, and what made flickr successful, is image hosting. Their approach is to keep it as simple as possible, but still offer a service that is as powerful as any application that manages and organises digital photos. They apply this idea to everything on the site, flickr's interface is minimalistic and functional that is focused on the content and users. Their processes are simple and fast, as an example uploading an image only takes two steps, where the user has to input relevant information, such as title, description and tags, decide on who is authorised to view the images and which set the images belong to.

The two main goals of flickr are:

1. Help people make their content available to the people that matter. It is done by implementing sharing tools and web 2.0 concepts that connect users. Users can share their photographs and images with just friends or, if they agree, with the world.
2. Provide with new ways of organising and managing media content.

CBIR could help flickr accomplish these goals by implementing a content-based search method searches for similar content but especially for people, i.e. using facial recognition on the pho-

[8] deviantart.com - Quantcast Audience Profile – `http://www.quantcast.com/profile/network-profile?wunit=wd%3Acom.deviantart` (last accessed December 14th, 2008)

[9] 3 Billion! " Flickr Blog – `http://blog.flickr.net/en/2008/11/03/3-billion/` (last accessed July 17th, 2009)

tographs, which represent a big percentage of the images they host, could help identify and group the photos where one person appears on. Such implementation would need to be careful deployed on the site as it raises privacy concerns.

2.2.3 Web portals implementing Content-Based Image Retrieval

There is still not a wide spread use of CBIR in mainstream web portals, however it is a feature that is beginning to surface in beta stages or as a way to filter the results of traditional queries. Several projects are developing these types of search and companies such as Hitachi, Microsoft and Ide are beginning to deploy CBIR systems on large scale portals and provide the users with the ability to search for similar images.

The following sections will describe implementations of CBIR search methods on museums, web portals, search engines and projects alongside with a description of their unique features.

Hermitage Museum

The hermitage museum located in Saint Petersburg, Russia, is one of the largest museums in the world and provides a web portal that displays their collection [10]. The site offers the user a virtual tour of museum that includes all the galleries and respective artwork, and the possibility to browse the entire collection, which can be done in three ways:

i. Browse through the different categories, titles, subject, authors, style, genre, theme or period.
ii. Search method using keywords.
iii. Query by image content.

The first two methods are implemented using a description of the artwork and the relevant information written by an expert that can cover the genre, theme, subject, etc. This information can be easily indexed and provide with an effective method of searching if the artwork's description is accurate and the right search terms are given.

The last search method is very different, because it only depends on the artwork visual information, colour and layout.

- Searching by colour. The user chooses a set of colours, using a visual tool, and the proportion in which they are present, then the system searches its image index for artwork that match these parameters. For example, if the user wishes to search for a beach painting, he would have to search for blue and light yellow, as he would be looking for the sea and sand.
- Searching by layout. The user provides with a visual layout of the artwork he is searching for and the system looks for images that have the same visual organisation as the sketch. This is done by combining geometrical shapes, such as squares and circles, with colour information, which will represent a very primitive sketch of what the user is looking for.

[10] The State Hermitage Museum, St. Petersburg, Russia – `http://www.hermitagemuseum.org/` (last accessed July 17th, 2009)

There are some problems with this method and with the options it offers, first of all each search takes over five seconds to return the results, and although it is understandable due to the size of the artwork collection it is still a long response time. Then there is also the large number of false positives returned in the result, in other words images that have the characteristics specified in the query, but are not what the user is searching for, only by using the layout search and spending time drawing a detailed sketch of what the user is looking for, can we get more accurate results. Therefore this method should only be used to search for a theme or general objects, never for a specific artwork, as it would require more time and effort that is contrary to what a search method should do, a quick and straightforward mean to get the right results.

eChase

eChase, which stands for electronic cultural heritage made accessible for sustainable exploitation, is a project co-funded by the European commission to find sustainable business models to explore Europe's vast cultural heritage online. The goal of this project is to create a central access point that brings together public and private entities and explores their cultural assets. The result would be a sustainable business model that could provide the public with a multilingual portal that would exhibit Europe's cultural heritage [34].

This project is still being developed, however there is an online demo [11] that features an implementation of the University of Southampton IT Innovation Centre that shows what has been done so far, it includes a database with 10.002 media objects and the following features:

- An interface that allows different multimedia collections, which are very different in their subject, content, structure and language, to be searched.
- A search engine that is able to search for text, filter the results by different classes and a content-based search that looks for similar images and colour distribution.
- Tools to collect interesting content and commenting.
- Social features, such as sharing favourites.
- Multimedia games based on Europe's history and cultural assets.

The search engine implements two different types of search, the first type is a keyword and category query, where the results can be restricted by collection, people, places, dates and format. The second is a CBIR that implements four different methods, which are:

- Image shape query that will look into the height and width of the image and offer three options based on the possible orientations, landscape, portrait and square.
- Colour or Black and White, which will return all the coloured images or the black and white ones.
- Query by colour. With this option the user can select a colour from a palette and the system will return the images in which that colour is most predominant. This method is not very effective, because the user can only choose one colour from 48 available in the palette.

[11] eChase: Project abstract – `http://echase.it-innovation.soton.ac.uk/echase/` (last accessed July 17th, 2009)

- Query by example, this search method allows the user to choose an image in his computer and look for similar images in the database.

From all of the methods described above the query by example is the most powerful one, while the others only look at general characteristics of the image such as shape and colour percentage, this method actually allows the user to actually find specific content in the image database, making it a viable alternative to the more traditional keyword search.

The most interesting aspect of the site is the work that is being done to implement a central access point to Europe's art and culture and the fact that it uses social tools such as sharing, and alternative search methods to browse the content.

2.2.4 Internet search engines

The more traditional web search engines only use keywords for indexing pages and media object, using metadata information and relevant text that references the object. However, there are some search engines that implement CBIR methods indexing online media content, namely images, Gazopa and Live.com are some of the sites that implement indexing techniques on image and video content.

GazoPa

This is a venture project by Hitachi that provides the user with a method to media content by keyword or content [12]. There are three ways of searching for media objects:

- Keyword search.
- Uploading an image and find similar, this method will use the user's image and search the image and video database for images with the same shape and colour features.
- Drawing the query. With this method the user draws a sketch using a flash application and submits the drawing as the query, it is a very simple application that only provides the user with a pen, a colour palette and some basic tools to draw the query.

The results, of both keyword and CBIR queries, will return a set of images and videos similar or related to the query. In order to index the videos the system breaks it down to individual frames and process each as an image.

This application can be accessed directly through their website or by a facebook application [13] that will find images similar to the ones on your profile. The project has only been released as a beta product, and it still is not production ready as their index needs to grow and functionalities tested. As of today it is only available by invite.

Idée

Ide is a private company that develops image recognition and visual search software, they offer several products that are able to find similar or modified version of the query image. Here are the

[12] GazoPa Similar Image Search – `http://www.gazopa.com/` (last accessed July 16th, 2009)

[13] Gazopa Book application for facebook – `http://www.facebook.com/apps/application.php?id=36689117046` (last accessed July 17th 2009)

two most relevant:

- Piximilar is a commercial service that searches through a collection of images and finds similar images, there are three methods of querying the collection, *i*) the user selects a set of colours and the application finds images where these colours are present; *ii*) the user uploads an image and it finds similar; and *iii*) the user combines the previous methods with a keyword search that filters the results [14].
- TinEye.com is a public website that offers a service that search for modified versions of the query image. This is a different approach from Piximilar that can be used by users to find if someone used their work without permission and keep track on how it is being used. Ide claims to have indexed just over one billion images and performs a query in less than a second.

TinEye is still in a beta stage, but provides with an interesting approach on CBIR, as it does not look for similar images, but instead for identical images that have been modified or used in another context. This type of approach must consider the image by regions instead of a whole and apply the indexing algorithms to them in order to find changes.

Bing & Google Similar Image

Both Bing [15] and Google [16] are popular search engines that are able to search the internet for web pages, images, videos, news articles and maps. Currently they are ranked first and third, respectively, as the most used search engines worldwide accounting for eighty percent of the web search market and having indexed billions of pages and multimedia content.

Both search engines operate similar as they offer the ability to search images by keywords and then refine the results by choosing an image and then search for similar ones. The user can also filter the results by size, aspect ratio, colour information (black and white or coloured image), style (illustration or photograph). In addition it is also able to look for people, namely human faces and figures. This is a high level implementation of CBIR as it not only looks at the raw image but also does facial and human recognition.

2.3 Technologies

The art portal will be developed on top of two java frameworks, WebObjects and Java SE, and will use MySQL as a database to store the data. These technologies were chosen because of their robustness, fast and iterative development, and portability, can be deployed on any hardware running any operation system (Windows Server, Linux, Mac OSX, FreeBSD).

[14] Piximilar - Ide Inc. - The Visual Search Company – `http://ideeinc.com/products/piximilar/` (last accessed July 16th, 2009)

[15] Microsoft Bing.com – `http://www.bing.com/` (last accessed July 17th, 2009)

[16] Google Similar Image from Google labs – `http://similar-images.googlelabs.com/` (last accessed July 17th, 2009)

Java & WebObjects

There are two frameworks that are being used in the development of the application, which are the WebObjects and Java framework. Java is a well known and robust programming language that was developed by Sun in 1995 and is currently on version 1.6, Java Standard Edition 6. One of the main characteristics of this framework is portability as an application develop on top of the framework can run on any hardware and operation system. This is achieved because the actual code is not compiled to machine code, but instead runs on a java virtual machine written specifically for the host hardware. Java simplifies most low level operations leaving the developer to focus on the application itself. The best example is the garbage collector that manages memory and it's in charge of the cleanup of unused objects.

Java SE framework provides with libraries that allow the user with generic methods of accessing the host specified features and several tools such as lists, maps, graphics, networking, threading, database access, etc. One of the libraries that will be used in this project in the Java Advanced Imaging (JAI) API that provides with the tool for manipulating and extracting information on digital images.

WebObjects provides the developer with a set of tools for the development and a java framework that is designed using a multilayer architecture consisting of presentation, application logic and data management. Each layer uses a different component of the framework:

- The presentation layer uses the WebObjects framework (WOF) and is responsible for the user's interaction with the application, i.e. interface design and state management. It allows the developer to create two different types of application, web and desktop. The main difference between the two is the location of the interface, the web applications are accessed through a web browser and the desktop are a standalone client that is run on the desktop that can share some of the computational load and have a more native look.
- The application logic layer uses the Java SE framework to develop the core of the application and auxiliary tools.
- The data layer uses the Enterprise Objects framework (EOF) and is the main feature in WebObjects as it is in charge of the communication between the application and database. For that it uses an abstraction layer that permits the developer to create the application independently from the relational database. It translates each row of the database to a Java object successfully separating the data from application. Even the tool that is used to manage the tables, EOModeler, does not require any SQL knowledge or code as the schema for each table is created as a Java object, which the application takes care of translating it to SQL-code automatically.

In addition to the framework that WebObjects provide it has also implemented mechanisms to keep up with the load requirements. It offers several load balancing algorithms and the ability to deploy multiple instances of the application in one or more servers that are managed automatically. These tools allow for the application to scale from few users to thousands with few modifications. However, the code needs to be optimised in order to support the growth.

Ultimately the reasons for choosing these technologies and not others, such as .NET or Java EE, is due to the explicit separation of the different layers of an application that is very important in order to maintain the code on the long term. The portability offered by being pure Java and specially because of the Enterprise Object Framework that relieves the developer with the burden of managing the database in addition to the application development.

Database

Among the existing solutions on the market we chose MySQL 5.1 as our relational database management system. MySQL is licensed as an open source project that carries no cost when used for academic purposes and we chose this specific database engine because of its feature set and high performance.

MySQL is an open source relational database management system that is cross-platform and has all the expected features in a relational database. It is ACID compliant through the default storage engine, supports indexes and the partitioning of tables, which are important features when dealing with large amounts of data and very fast queries. Although choosing the right database system is important, especially when managing huge amounts of data, it is not fundamental, because the application will be developed in a framework that separates data storage from the application itself.

In comparison with other database management systems MySQL may not be the industry standard, but has all the necessary features for our purposes. We compared many other solutions and their features, among the considered database management systems are:

1. Microsoft SQL Server;
2. Oracle;
3. DB2;
4. Ingres;
5. PostgreSQL;
6. MySQL.

The first feature we compared was the portability of the system, with special attention to the compatibility with Mac OSX 10.4.11 operating system that would be used for development. Because of this we had to discard Microsoft SQL Server and DB2 as they lack the support for this environment.

We then looked at the inherent costs of each solution, where the Open Source licensed systems had immediate advantages as with this type of license we can develop the application without any initial costs. It presents a big advantage especially considering that our development framework has a loose dependancy with the database and that any change in the database system can be done without affecting the application.

One of the features that some systems fail to support are the operators, where MySQL and Ingres do not have the "Intersect and "Except operators implemented.

Nevertheless, all the systems feature the required properties of a relational database, such as ACID (atomicity, consistency, isolation, durability), referential integrity, transactions, indexes,

partitioning and SQL's query language. They also support triggers, stored procedures, BLOB objects and large databases with millions of rows.

3

High-dimensional Indexing Techniques

Contents

In this chapter we will present different methods that reduce a high m-dimensional space while maintaining the distance between objects. It is important to use these methods when dealing with high-dimensional data (multimedia images) as comparing in the original space will require many resources and can become a very slow operation.

Comparing the image on the original space is called list matching and is an extremely expensive CBIR method as it compares each image pixel by pixel. This method uses the colour information on the individual pixels to compare two images, calculating the difference between each pixel pair using a distance function, for example the Euclidean distance. In order to compare between two images that have incompatible resolutions we need to scale all the images to a fixed resolution, the chosen resolution must have a small dimension as the number of comparisons needed grow exponentially the higher the dimension.

It has a very high computational cost that makes it impossible to query large collections in real-time, for example, in a collection with 20.000 images and using 256×256 as the resolution when comparing the images we need to compare 65.536 pixels per image, which adds up to 1.310.720.000 comparisons for each query. The method should only be used to assess its effectiveness or for queries on very small collections, as it is too expensive to apply.

Therefore, in order to be able to implement an efficient method for search a collection by content a high-dimensional indexing technique is needed, such as tree indexing. In particular the hierarchical linear subspace method [25] with the following mapping functions:

i. Orthogonal projection;
ii. Principal component analysis;

These methods can reduce the dimensionality of multimedia images while keeping the distance inferior or equal to the original space, lower bounding lemma [24]. In the following sections we will describe the methods along with the generic multimedia indexing and the hierarchical linear subspace [25] that use subspaces from the original image to reduce the computational costs without affecting the results.

3.1 Related Work

Tree indexing is a common technique used for fast retrieval of objects by similarity, but it has one downside, as the dimensionality of the data increases the performance deteriorates. Indexing trees can be divided in two classes that are composed by derivatives of the following trees, (i) from the kd-tree; (ii) from the R-tree.

In the first class the data space is divided along predefined hyper-planes regardless of the data distribution. This creates regions that are mutually disjoint, however many of these regions will not represent any objects. It becomes a real problem considering it requires exponentially more objects to populate the regions as the dimensionality increases.

Trees of the second class works around this problem by taking in consideration the data distribution when dividing the data space, which leads to the creation of overlapping regions. The M-tree [35] is an example and it performs exact retrieval with up to 10 dimensions, with its performance deteriorating in high dimensional spaces.

A possible solution for this problem involves approximating queries with a relative error during retrieval. Two examples of trees that use this approach to query data spaces with dimension of several hundreds are the M-tree [2] and A-tree [3]. Where the A-tree uses approximated minimum bounding rectangles (MBR) instead of the MBR of the R-tree. Approximate metric trees like NV-trees [4] work with an acceptable error up to 500 dimensions.

The most successful approximate indexing method used in these trees is based on hash tables. One implementation of this type of method is the locality sensitive hashing (LSH) [36] that is able to support fast queries with dimensions up to 100. The method uses a family of locality-sensitive hash functions where similar objects in the high-dimensional space are mapped into the same bucket with high probability. A similarity query can be performed by hashing a query object into a bucket and then rank all the objects in that bucket, which are called candidate objects, using a distance function that orders the results by similarity. The Multi-probe LSH [37] is one of many extensions of this method that reduces the space requirements for the hash tables.

The previous tree indexing techniques work in high-dimensional spaces using the original objects, whereas other approaches map these high-dimensional data (dimension 10 to 270) into a 1-dimensional space. In 1-dimensional space, efficient tree techniques such as B-trees can be applied for fast retrieval. Among the many examples are the space filling curves [5] and Pyramid Technique [6].

The Pyramid Technique divides the data space into 2-dimensional pyramids whose apexes share a common center point. The following step cuts each pyramid into several slices parallel to the foundation of the pyramid forming the data pages. Moreover, this technique maps each high-dimensional object onto a 1-dimensional space that represents the distance from the object to the top of the pyramid.

The NB-tree [38], on the other hand, maps the points in m-dimensional space into a 1-dimensional line, which can be stored in a simple B-tree for fast retrieval. It does so normalising the points in space to the unit hypercube $[0..1]^D$, where we can calculate the length of the corresponding points and the position in the line.

The major limitation of the methods described above is the inherit dimension of the data, although they performs well in a data space up to a dimension of several hundreds, it quickly starts to break when dealing with higher spaces. This limitation can be surpassed by using a subspace tree, which can handle these very high-dimensional spaces (order of several thousands) while yielding good performance and precise queries.

3.2 Generic multimedia indexing approach

The generic multimedia indexing approach (GEMINI) [24,28] is an indexing technique that was developed for IBM's QBIC system (see section 2.2.1) that uses a feature extraction function that maps high dimensional objects into a low dimensional space, the feature space. In this transformed space we can expect objects that are very dissimilar to also be dissimilar in the original space (see figure 3.1). Thus we can apply a quick and cheap query that will discard all the non-qualifying objects in the feature space.

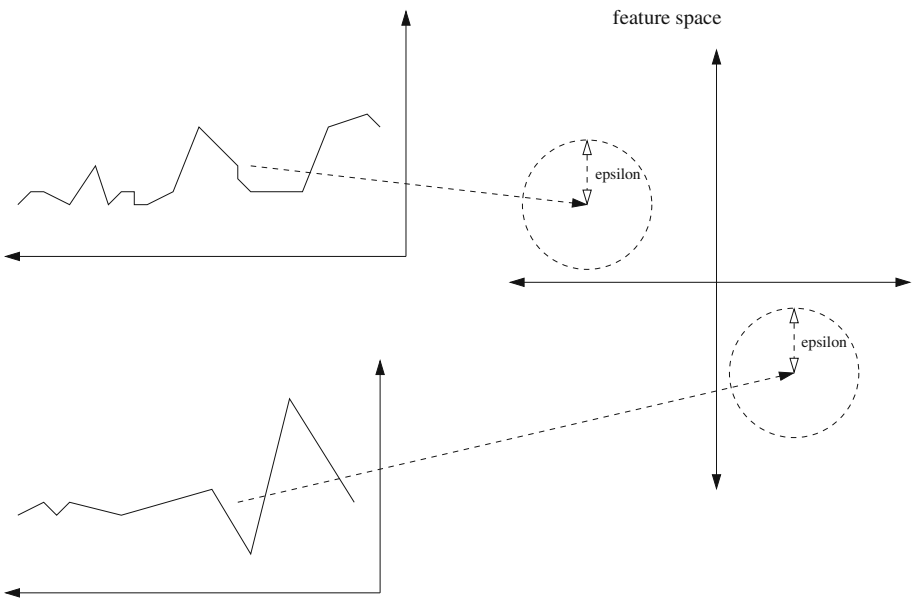


Figure 3.1: Feature extraction function which maps the high dimensional objects into a low dimensional space. The distance of similar objects should be smaller or equal to ϵ. This tolerance is represented by a sphere with radius ϵ in the feature space.

Ideally, the feature mapping should preserve the exact distances, but this is only possible if both spaces have the same dimension as some information is lost when we apply the mapping function. This limitation can be minimised depending on the function and on which object's characteristics it keeps. However, if distances in the feature space are always smaller or equal than distances in the original space a threshold ϵ can be determined so that the distance of similar objects that are smaller or equal to ϵ in the original space are also below that threshold in the feature space. No object in the feature space will be missed (false dismissals) in the feature space, though some objects that are very dissimilar in the original space may be under the ϵ threshold (false hints/alarms). That means that we are guarantee to have selected all the objects we wanted plus some additional false hits in the feature space. In a latter step, false hits will be discarded from the result set by performing a query in the original space.

The size of the collection in the feature space depends on ϵ and the proportion between both spaces may reach the size of the entire database if the feature space is not carefully chosen.

The lemma which guarantees that no objects will be missed in the feature space is called the "lower bounding lemma"" and is expressed mathematically as follows;

Lemma 3.1. *Let O_1 and O_2 be two objects; $F()$, the mapping of objects into f dimensional space should satisfy the following formula for all objects, where d is a distance function in the original space and $d_{feature}$, in the feature subspace:*

$$d_{feature}(F(O_1), F(O_2)) \leq d(O_1, O_2). \tag{3.1}$$

The mapping function that is used in the GEMINI approach has to satisfy the lower bounding lemma. When we apply the $F()$ function to an object it will transform it, preserving the relevant

object's characteristics or projecting it to a low dimension space. The goal of applying the mapping function is to return an object that has a dimensionality considerably smaller than the original space while still keep the important features, i.e. create a fingerprint of the object that will be used in queries.

Ideally the feature space should hold all the important characteristics of the image, in other words, the $F()$ mapping function should extract the important characteristics of the object and discard the irrelevant information.

Given Parseval's theorem, which states that the Discrete Fourier Transform (DFT) preserves Euclidian distances between signals, the DTF which keeps the first coefficients of the transform is an example of a feature function $F()$ [24,28]. Accordingly, any orthonormal transform can be used as a $F()$ mapping function, because they satisfy the lower bounding lemma as they preserve the distance between the original and transformed space. Other data transforms also hold to the lower bounding lemma, although with some particularities. Data dependant transforms can also be used as a mapping function, as they hold to the lower bounding lemma. The downside of using such methods is that the transform needs to be recalculated periodically or when new data is inserted. One data dependant transforms is the Karhunen-Loève transformation that is used in the principal components method.

3.3 Hierarchical Linear Subspace

The hierarchical linear subspace method is based on the generic multimedia indexing (GEMINI) approach [24,28] described before and makes use of the lower bounding lemma to create a pyramid of different dimensional spaces, or subspaces, with the original space in the bottom of the pyramid and the smallest on top. Each subspace is generated from the original space using a $F()$ mapping function that satisfies the lower bounding lemma and reduces the original m-dimensional space V, to a f-dimensional subspace U. This allows for the application to query the collection and progressively reduce the number of images and calculations as each subspace is "visited".

Lemma 3.2. *Let O_1 and O_2 be two objects; $F()$, the mapping of objects into f dimensional subspace U should satisfy the following formula for all objects, where d is a distance function in the space V and d_U in the subspace U:*

$$d_U(F(O_1), F(O_2)) \leq d(F(O_1), F(O_2)) \leq d(O_1, O_2). \tag{3.2}$$

Therefore a sequence of subspaces can be defined as $U_0, U_1, U_2, \ldots, U_n$ with $V = U_0$ in which each subspace is a subspace of another space, where $U_0 \supset U_1 \supset U_2 \supset \ldots \supset U_n$ and $dim(U_0) > dim(U_1) > dim(U_2) > \ldots > dim(U_n)$.

Example We have a DB with a collection of multimedia images at a $256x256$ resolution and with a 3-band RGB. We then use a $F()$ mapping function that satisfies the lower bounding lemma to create the subspaces U_0, U_1, U_2 and U_3 reducing the original to the respective resolutions 256×256, 64×64, 16×16, 8×8 (in this case the orthogonal projection can be used as a $F()$ function, as

any other orthonormal transform). The subspaces are then defined as,

$$U_0 \supset U_1 \supset U_2 \supset U_3$$

and

$$dim(U_0) > dim(U_1) > dim(U_2) > dim(U_3) \Leftrightarrow 196.608 > 12.288 > 768 > 192$$

In this case, as the same $F()$ mapping function is used to generate the subspaces, then the same distance function can be used in the query itself.

3.3.1 Orthogonal Projection

The orthogonal projection is a a linear mapping function $F()$ that calculate the mean value of the image's 3-band RGB to effectively reduce the original space V to the subspaces $U_1, U_2, U_3, \ldots, U_n$ and so on. This function is an orthonormal transformation and therefore satisfies the lower bounding lemma.

Theorem 3.3. *(Lower bounding) Let O_1 and O_2 be two objects; if $V = \mathbf{R^m}$ is a vector space and U is a f-dimensional subspace obtained by a projection and an Euclidian distance function $d = l_2$, then*

$$d_U(U((O_1), U(O_2)) \leq d(U(O_1), U(O_2)) \leq d(O_1, O_2). \tag{3.3}$$

Furthermore, we can map the computed metric distance d_U between objects in the f-dimensional orthogonal subspace U into the m-dimensional space V which contains the orthogonal subspace U by just multiplying the distance d_u by a constant $c = \sqrt{\frac{m}{f}}$,

$$d(U(O_1), U(O_2)) = \sqrt{\frac{m}{f}} \cdot d_U(U((O_1), U(O_2)). \tag{3.4}$$

This way not only can we reduce the original subspaces using the orthogonal projection but we can also approximate the distance between O_1 and O_2 to the original space.

Example 1 For example the orthogonal projection of points $\vec{x} = (x_1, x_2) \in \mathbf{R^2}$ on the bisecting line $U = \{(x_1, x_2) \in \mathbf{R^2} | x_1 = x_2\} = \{(x_1, x_1) = \mathbf{R^1}\}$ corresponds to the mean value of the projected points. The point $\vec{a} = (2, 4)$ is mapped into $P(\vec{a}) = 3$, and $\vec{b} = (7, 5)$ into $P(\vec{b}) = 6$. The euclidean distance in U is $d_u(P(\vec{a}), P(\vec{b})) = \sqrt{|6-3|^2}$, $c = \sqrt{2}$, so the distance in $\mathbf{R^2}$ is $d(P(\vec{a}), P(\vec{b})) = \mathbf{3} \cdot \sqrt{\mathbf{2}} \leq d(\vec{a}, \vec{b}) = \sqrt{\mathbf{26}}$ (see figure 3.2) [25].

Example 2 The average colour of an image $\vec{x} = (R_{avg}, G_{avg}, B_{avg})^T$ corresponds to an orthogonal projection. Because of that, the equation $d_{avg}(F_{avg}(\vec{x_1}), F_{avg}(\vec{x_2})) \leq d(\vec{x_1}, \vec{x_2})$ which is guaranteed by "Quadratic Distance Bounding" theorem is also guaranteed by the orthogonal projection, even the more efficient equation is valid [25].

$$d_{avg}(F_{avg}(\vec{x_1}), F_{avg}(\vec{x_2})) \leq d(F_{avg}(\vec{x_1}), F_{avg}(\vec{x_2})) \leq d(\vec{x_1}, \vec{x_2}) \tag{3.5}$$

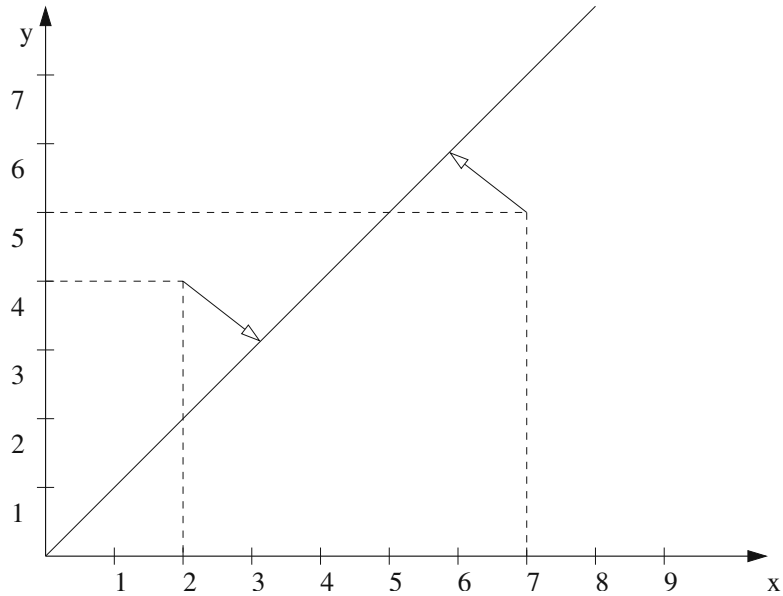


Figure 3.2: For example, the orthogonal projection of points $\vec{x} = (x_1, x_2) \in \mathbf{R^2}$ on the bisecting line $U = \{(x_1, x_2) \in \mathbf{R^2} | x_1 = x_2\} = \{(x_1, x_1) = \mathbf{R^1}\}$ corresponds to the mean value of the projected points. $\vec{a} = (2, 4)$ is mapped into $P(\vec{a}) = 3$, and $\vec{b} = (7, 5)$ into $P(\vec{b}) = 6$.

Example 3 A lower resolution of an image corresponds to an orthogonal projection in rectangular windows, which define sub-images of an image. The image is tiled with rectangular windows W of size $j \times k$ in which the mean value is computed (averaging filter). The arithmetic mean value computation in a window corresponds to an orthogonal projection of these values onto a bisecting line. Because of this, the different resolutions of an image correspond to a sequence of subspaces that satisfy the lower bounding lemma (see figure 3.3). The representation of images at several resolutions corresponds to a structure which is called "image pyramid" in digital image processing [39], [40]. The base of the pyramid contains an image with a high-resolution, its apex being the low-resolution approximation of the image (according to [25]). It should be noted, that a related approach was suggested for vector quantization using mean pyramids of codewords [41].

(a) 256×256 space. **(b)** 64×64 subspace. **(c)** 8×8 subspace. **(d)** 4×4 subspace.

Figure 3.3: Image of a landscape, where 3.3(a) is the original space U_0 and 3.3(b) (U_1), 3.3(c) (U_2) and 3.3(d) (U_3) are the subspace after applying the orthogonal projection.

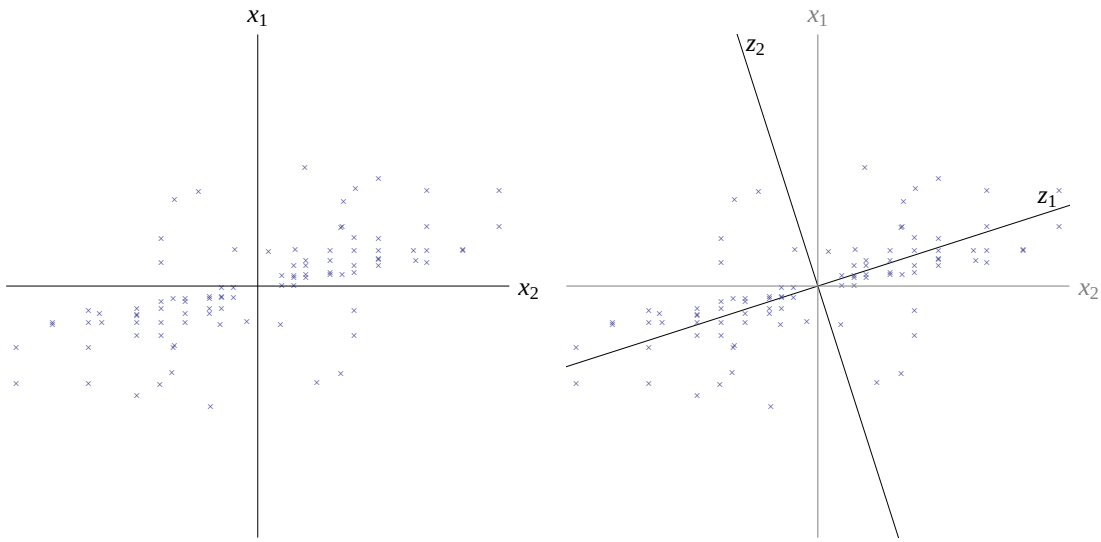
3.3.2 Principal Component Analysis

The principal component analysis method uses a data dependant transform to build its pyramid of low dimensional subspaces. It uses a $F()$ mapping function that is called the Karhunen-Loève transformation which satisfies the lower bounding lemma [24]. This method has to be implemented under a condition, it needs to be recalculated either periodically or after new data is inserted. It depends on the collection's data to accurately reduce the dimensionality of multimedia objects.

The PCA method was invented in 1901 by Karl Pearson [21] and is now used in many different fields, such as data mining, exploratory data analysis and data compression. It is also used as a common technique for finding patterns in high dimensionality data [22].

PCA was developed using simple statistical tools such as standard deviation, covariance, eigenvector and eigenvalues. It uses these mathematical techniques to determine the correlated variables in a data set that change together in space. By knowing which variables are common in the data we can discard some variables without affecting the distance between objects and only keep the variables that make each object different and unique.

Fundamentally, it analyses the data and extrapolates a new coordinate system, where the data is transformed. Then according to the variability of the data in each of the coordinates the PCA method will discard the coordinates that are not relevant and effectively reduce the dimension of the original data.



(a) Original data. **(b)** New coordinate system.

Figure 3.4: The new coordinate system after applying the PCA method to a sample data.

One of the most used implementation of the PCA method is the covariance matrix that will start by calculating the covariance matrix method, then it will calculate the eigenvectors (principal components) and respective eigenvalues by applying the Karhunen-Loève transformation. The result of the transformation will allow for the system or user to choose which principal components to keep. This method will derive a new data set projected into the new coordinate system.

3.3.2.a Covariance Matrix

Covariance is a statistical tool that is used to calculate the relationship between two variables and how they change together. If the same variable is used the result is the variable's variance. In a 3-dimensional space (x, y, z) one could measure the covariance between the x and y variables, the x and z variable, and the y and z variables. The formula for the covariance is very similar to the variance. While the variance can be defined as:

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{n-1} \tag{3.6}$$

We multiply the difference between the x value and the mean of x, by the difference between the y value and the mean of y. It can be defined as:

$$covar(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1} \tag{3.7}$$

If both X and Y are real-valued variables then the covariance calculation inherits the commutativity property from the multiplication as:

$$covar(X, Y) = covar(Y, X) \tag{3.8}$$

Example 1 Using a 2-dimensional data we have asked a sample of students how many hours in total they have spend studying for a class, and their final grade. We have two variables, the first is the hours of study H, and the grade G, table 3.1 holds the data and the calculation of $cov(H, G)$, i.e. the covariance between hours of study and the final grade. The results show that there is a relationship between the hours of study and the grade, as the more hours of study reflect better grades.

The covariance values can reflect one of three results, first, if the value is equal to 0 the two of variables are independent from each other. Second, if the value is greater than 0 the two variables are correlated and as one increases so does the other. Third, if the covariance is lower than 0 the variables are inversely correlated, i.e. as one variable increases the other decreases.

The results for this data set and in other low-dimensional spaces can be analysed manually or by plotting the results, but it is cannot be accomplished in a high-dimensional space. In fact, for a n-dimensional space the number of different covariance values that need to be calculated are:

$$covariance\,values = \frac{n!}{(n-2)} \tag{3.9}$$

The covariance matrix is use as an efficient method of saving and accessing the covariance values is by using a matrix. The definition of the covariance matrix for a data set with n-dimensions, where $C^{n \times m}$ is a square matrix with n rows and columns, $x_i^{(k)}$ and $x_j^{(k)}$ represent the $x$ith variables. The matrix is also symmetric and $c_{i,j} = c_{j,i}$.

	H	G	$(H_i - \bar{H})$	$(G_i - \bar{G})$	$(H_i - \bar{H})(G_i - \bar{G})/(n-1)$
	9	39	-4.92	-23.42	9.59
	15	56	1.08	-6.42	-0.58
	25	93	11.08	30.58	28.25
	14	61	0.08	-1.42	-0.01
	10	50	-3.92	-12.42	4.05
	18	75	4.08	12.58	4.28
	0	32	-13.92	-30.42	35.27
	16	85	2.08	22.58	3.92
	5	42	-8.92	-20.42	15.17
	19	70	5.08	7.58	3.21
	16	66	2.08	3.58	0.62
	20	80	6.08	17.58	8.91
total:	**167**	**749**			**112,70**
average:	**13,92**	**62,42**			**9,39**

Table 3.1: 2-dimensional data set and covariance calculation.

$$C^{n\times n} = C_{i,j} = covar(x_i^{(k)}, x_j^{(k)}) = (\frac{\sum_{k=1}^{n}(x_i^{(k)} - m_i)(x_j^{(k)} - m_j)}{n-1} \tag{3.10}$$

Example 2 A covariance matrix for a 3-dimensional space with the x,y and z variables will produce a 3×3 matrix that would be populated with the following values:

$$C^{3\times 3} = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(x,y) & cov(y,y) & cov(y,z) \\ cov(x,z) & cov(y,z) & cov(z,z) \end{pmatrix} \tag{3.11}$$

3.3.2.b Karhunen-Loève Transformation

The next step of the process is to perform the Karhunen-Loève transformation that will find the eigenvectors and respective eigenvalues of the covariance matrix C. This transformation rotates the feature space into alignment with uncorrelated features.

$$U^T CU = \Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)(\lambda I - C)u = 0 \tag{3.12}$$

There are n eigenvalues and eigenvectors and λ_i is the ith eigenvalue of C and u_i the ith column of U, the ith eigenvectors. The eigenvectors are always orthogonal and each is associated with the variance value that is represented by the eigenvalues.

$$C\vec{u}_i = \lambda_i \vec{u}_i \tag{3.13}$$

The resulting matrix U defines the Karhunen-Loève transformation and the transformed features are defined by:

$$\vec{y} = U^t \vec{x} \tag{3.14}$$

Uncorrelated features with higher variance, a high Λ_i, contain more information and therefore are the features that are kept when reducing the data. Correlated information should be discarded when reducing the dimension.

3.3.2.c Dimension Reduction

The last step of the method will take the Karhunen-Loève transformation matrix of eigenvectors and respective eigenvalues, and use this information to reduce the dimensionality of the original space. This is done by analysing the eigenvalue and determine which features are correlated in order to discard them. The process can be done using one of many objective criteria, among the existing are:

- Pre-determined dimensions criteria, in this method is mostly used for data compression as we set the resulting number of dimensions without taking into account their significance, keeping only the top N dimension.
- Kaiser criterion, we retain only the principal components with eigenvalues greater than 1, i.e. unless it extracts as much as the equivalent of one original variable, we drop that principal components, this criteria was proposed by Kaiser [42,43] and is the most widely used criteria.
- Joliffe criterion, similar to the Kaiser criterion, it retains the principal components that are greater than 0.7. This criterion will keep more features than the Kaiser.
- Mean eigenvalue, this criteria discards all principal components that have a eigenvalue lesser than the mean eigenvalue. As a result too few principal components are retained.
- Scree Test and elbow criteria, this criteria was proposed by Catell [44] and it suggest to sort by descending order and plot the eigenvalues in order to find the first point where the plot increases the most, the elbow, keeping the dimensions to the left of that point.

All the criteria have their merits, but the Kaiser criterion and the Scree Test are the most used criteria and have been studied in detail [44–47]. The general conclusion is that both of them behave well under normal conditions and can be used in choosing which principal components to keep.

A new matrix is then populated with the chosen eigenvectors, after applying a criterion to filter the eigenvectors. This new matrix U^* is defined by:

$$U^* = \begin{pmatrix} u^a \\ u^b \\ \dots \end{pmatrix} \tag{3.15}$$

and is then applied to the original m-dimensional space by:

$$\vec{y} = U^{*T} \vec{x} \tag{3.16}$$

The result of this process is a set of eigenvectors that when applied to a data set, it effectively reduces the dimension of the original space. However this is a lossy method that loses some information. In conclusion, we can use this $F()$ mapping function to create an "image pyramid" representation of the original space [39], where each image has its own resolution.

4

Software Architecture

Contents

Our goal is to implement a web-based application that supports the artist community in exhibiting their image portfolios and help them reach a broader audience. The application has all the basic features of popular art portals, such as flickr and deviantART (see sections 2.2.2 and 2.2.2) and also implement high-dimensional indexing techniques combined with content-based image retrieval queries.

Among the functional requirements that were documented during the development of the application (see appendix A) and subsequently implemented as features in the application we would like to point out the following;

- Any user may browse the portal and the image collection;
- Artists must be able to exhibit and manage their image portfolios;
- The application must implement content-based search methods to query the image collection;
- The application must also implement traditional keyword-based search methods;

The application had to be able to scale and handle increasing page views requests along with providing a fast and responsive interface to the user. The interface was designed to be minimalistic and easy to use. The content and styling were separated by using cascading style sheets that resulted on smaller pages and less network bandwidth, but mainly to avoid complex web pages that combine data and look and end up very complex and hard to maintain. To avoid this problem we complied with web standards when developing the interface to reduce the complexity and so that it could be viewable from any current browser.

We used a 3-tier client-server architecture that allowed us to separate the user interface from the application logic and database access layer. The CBIR methods and logic was developed as an weakly dependant module that can easily be ported to other research projects and application.

We implemented a series of content-based search methods that range from a "brute force" approach (list matching) to high-dimensional indexing techniques that greatly reduce the computational costs when querying the collection and provide real-time results. The public version of the application will feature one of these latter methods as an average user expect for quick and good results and is not willing to wait minutes or hours for the results of, in his view, a simple image query.

In the next sections we will describe the implemented search methods, the development methodology and the implemented architecture of the system with greater detail.

4.1 Implemented Search Methods

We implemented several methods that perform content-based queries on the collection in order to test their efficiency and the quality of the results. These methods have been described in the previous chapter (see chapter 3) and in this section we are going to give a description of the implementation details of the methods.

Firstly, we used a standard high-dimensional distancing function that can determine the distance between two images in space. One of such functions is the Euclidean distance. It is based

on the Pythagorean theorem and it calculates the ordinary distance from point a to b in any n-dimensional space. It is defined as;

$$d(O_1, O_2) = \sqrt{\sum_{i=1}^{n} (O_1^{(i)} - O_2^{(i)})^2} \tag{4.1}$$

Where O_1 and O_2 are the image that are being compared and $O_1^{(i)}$ is the ith dimension of the image. This function can be further defined as we are dealing with 3-band RGB images that can be divided in red, green and blue;

$$d(O_1, O_2) = \sqrt{\sum_{i=1}^{n} (R_1^{(i)} - R_2^{(i)})^2} + \sqrt{\sum_{i=1}^{n} (G_1^{(i)} - G_2^{(i)})^2} + \sqrt{\sum_{i=1}^{n} (B_1^{(i)} - B_2^{(i)})^2} \tag{4.2}$$

Other distance functions can be used with simple modification to the CBIR module and among the existing are the co-sine similarity and the Manhattan distance.

The Euclidean distance function was used in the implementation of all the search methods that are described on the following paragraphs.

List Matching The list matching method is a "brute force" approach that takes each of the RGB bands and compares each image on the original space. And while the results are satisfactory it is highly inefficient as the original space has a $256 \times 256 \times 3$ dimension (196 608), it should only be used as a point of comparison with the other methods and will not be a search option for the user in the public version of the application.

Colour Histograms This method extracts statistical information on the colour distribution of an image, which is done by counting the number of pixels of each colour. The colour histogram of an image is relatively invariant with translation and rotation about the viewing axis, and varies only slowly with the angle of view [48], being especially effective when looking for images that have the same objects or have been altered. In order to apply CBIR we can use colour histograms of two images and calculate the distance between them, which can be done by Euclidean Distance or Cosine similarity.

The major problem with colour histogram is that they are only depend on the colour of an object, discarding spatial, shape and texture information. It is enough for two images to be considered different if they have different lighting conditions, for instance, if a brightness filter is applied to image the content remains the same but the colour changes and so does the colour histogram making the two images very different when compared by this method.

When histograms are generated there is a loss of information as the original image has $pixel \times pixel \times 3$ dimension, whereas the colour histogram has $(n^o\, of\, colours) \times 3$, as a consequence images that have different content may have overlapping histograms. On the other hand, without spatial or shape information, both identical objects of different colours and different objects of the same colour are impossible to differentiate relying only on colour histograms.

Although there are drawbacks when using histograms for CBIR there are some advantages, the first is that colour information is very fast to compute and more efficient than other methods.

Secondly it presents with good results when searching for modified versions of an image, especially if the histograms are calculated by region instead of the image as a whole.

Hierarchical Linear Subspace The hierarchical linear subspace method was used in the application in order to create a subspace pyramid, i.e. different subspaces with increasing dimensions, for each image that will be used to the search queries. This pyramid will, in fact, allow to query images at a lower dimension discarding the images that are very dissimilar and therefore using less resources than doing this search exclusively on the original space.

We used two different mapping functions to generate the subspaces:

- Orthogonal Projection;
- Principal Components Analysis;

Both of these methods will give significant savings in computation time compared to list matching while maintaining the quality of the results, as the final subspace, the last subspace of the pyramid, is the original space.

4.2 Development Methodology

The software development process was set by a series of stages and intended to carry on an iterative and incremental development strategy. This methodology was based on the agile software development [49] with many of the initial requirements were redefined during the programming of the system and at the end of each iteration, or cycle, we had a working prototype with limited functionality.

A cycle would be schedule to last a week at maximum, for which we had to create a set of objective and realistic goals and a structured process that included *a*) requirement analysis, *b*) design, *c*) implementation and *d*) testing stage with the functionalities already deployed . This process was then repeated in order to implement a new feature of the system, each with increasing functionalities that allowed us to break the system's complexity in a series of smaller and objective goals. For example, the first iteration was the web application itself that allowed for browsing, user registration and image submitting. The next implemented image processing that extracts the features and store in cache the relevant CBIR information. The latter releases implemented a CBIR module that made available search methods allowing for content-based image retrieval. The following list details all the necessary cycles to build the system.

1. Basic web site functionality, such as user registration and image submitting;
2. Image processing to extract relevant features and data for the histogram, hierarchical linear subspace and list matching search methods;
3. Histogram search method;
4. List matching search method;
5. Hierarchical linear subspace search method;
6. Interface work that will separate content from style using cascading style sheets (CSS);
7. Development of a web crawler that would populate the collection with art images;

8. PCA experiments;
9. Implementation of the exhibition concept;

At the end of each of these cycles we had a working prototype with limited functionality. Each cycle would need, in average, a week with the exception of the PCA work that required more time than initially expected as many optimizations were needed during the testing stage and the generation of the test data was very extensive.

Requirement Analysis

As with any software development the first step in this process was the requirement analysis that aimed to determine all the expected functionalities and behaviour of the system. The requirements that resulted from this stage are detailed in the requirements section of the architecture chapter (see appendix A).

Design

On the next step of the development process we extrapolated the different use cases scenarios that the application required, the core features that needed to be implemented and the information architecture that would support the application. The architecture design evolved during the different cycles of development with the final version being very different from the first draft. The information architecture changed drastically since the beginning and now presents a flexible platform that effectively separates metadata information from the data itself. This allowed us to implement a normalised schema for the database. From all the models that were tested we believe that the final result is the best suited architecture to support this system.

Implementation and testing

The actual implementation, as mentioned, was planned to implement one feature at a time that allowed to build the system from the bottom up. It was followed by a testing phase that extensively test the implemented functionality and assess if the feature was well integrated with the system.

The code development was performed according to some guidelines that require good practises of programming in order to avoid a bloated application hard to maintain on the long term. The use of Java's best features and practises, such as encapsulation and polymorphism, was fundamental to implement a lean and flexible application. As an example,we can introduce new subspace for the hierarchical linear subspace queries by changing one line of code. We also use relative paths to determine where files are stored on disk, allowing for an easy migration of the system and collection between servers and partition.

4.3 Information Architecture

In this section we will describe the information architecture that supports the application and will be used to create the data layer that keeps and manages the data used by the system. The in-

formation architecture was designed based on the application requirements and then refined during the elaboration of the use cases and functionality of the system. During the implementation stage small modifications were done to address some problems and necessities that were not anticipated.

To keep with one of the major requirements of the applications, i.e. modular system, we decided to separate the image information in three entities, the metadata, the search data and the actual image. The figure below shows the model for the information architecture.

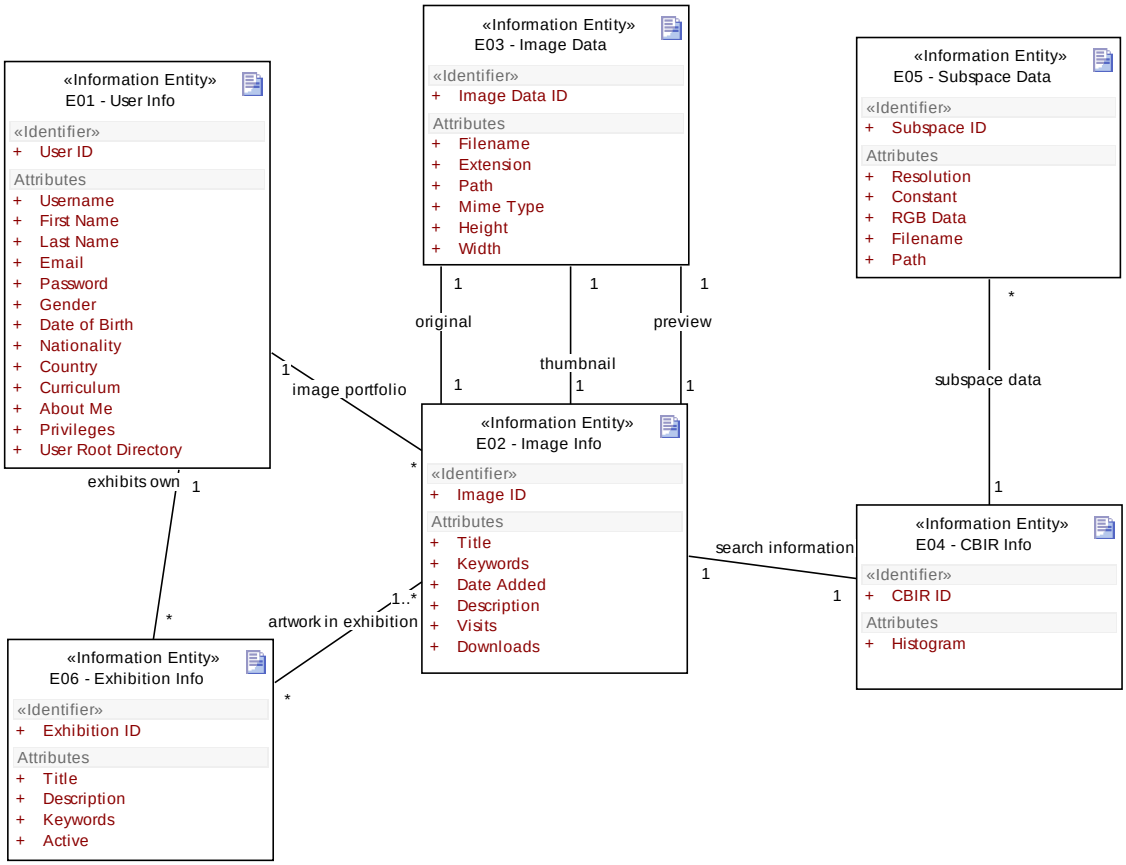


Figure 4.1: Diagram representing the existing information entities and their attributes.

On the following sections we describe the information entities of the system and what is their role, for a more detailed information on the information that each entity holds and the relationships between them see the tables in appendix B.

User Info represents every registered user in the system, both the artists and the administrators, and is the only entity that is independent of all others. This entity holds the user's personal and public information, including the contact in case there is a need to communicate with the user outside the context of the system.

Image Info represents every image in the system, containing relevant information and referencing the actual data that is stored in the Image Data (E03) entity. In additional to the original image, we also keep a preview and thumbnail image that saves network bandwidth and frees up resources.

Image Data represents the actual image data that is stored in disk. We felt the need to separate the metadata from the actual data in order to avoid repetitive attributes and to facilitate the use of different resolutions and the introduction of new features, such as an avatar for the user, which doesn't require any additional information.

CBIR Info keeps all the data necessary to perform content-based queries, in our case the histogram data and the different subspaces for the hierarchical linear subspace method. Because we separate the search data from the image entity we can easily maintain the search application and even introduce new search methods with minimal impact to the information architecture.

Subspace Data holds the different subspace needed by the hierarchical linear subspace method, i.e. the 3-band RGB information for the different subspaces and respective constant to approximate the subspace to the original space. The actual subspace data can either be kept on the database or on disk.

Exhibition Info represents an exhibition created by the user that is used to display part of his portfolio of images. Each exhibition can hold several images and has its own state, active or inactive. Only the active exhibitions are public and can be browsed by any user, consequently the inactive exhibitions only allow management operations until it returns to the active state.

4.3.1 Data Storage

The amount of data that the system needs to handle is colossal, with a minimum image of 20 000 images in the collection to produce a controlled real-world scenario we needed an efficient method for storing and accessing the data. It also had to be able to keep up with the needs of a real-time application.

The data that needs to be stored for each image object is significant, for regular browsing we need to store the metadata information along with an original, a preview and a thumbnail image. In addition, each object also needs to store the CBIR information that is used for the content-based queries in cache, as it is impractical to generate that data in runtime. It requires that the histogram and all the subspace data for the hierarchical linear subspace methods. Thus, dealing with many large files that should not be stored exclusively in the database and for it to be stored in disk it needs a directory structure that is prepared to scale fast.

We developed a hybrid method of storing the data between the database and the disk. In this approach we save all information that is under 768 bytes in the database , which includes the histogram and the 4×4 and 8×8 subspaces (hierarchical linear subspace) and along with metadata information. The data that is above this threshold is saved to disk and consists of the other subspaces, three JPEG images with different resolutions and using a compression rate of 90% (original, preview and thumbnail images). The following directory structure is detailed in figure 4.2.

The directory structure separates the CBIR data from the images as shown in figure 4.2 and is able to perform as it scaled to a 30.000 image collection. This structure evolved from a simplistic one that stored all the data from each image in their own directory. This initial structure caused

several bottlenecks and performance started to deteriorate once we started to test with large collection and negatively affected CBIR query-time.

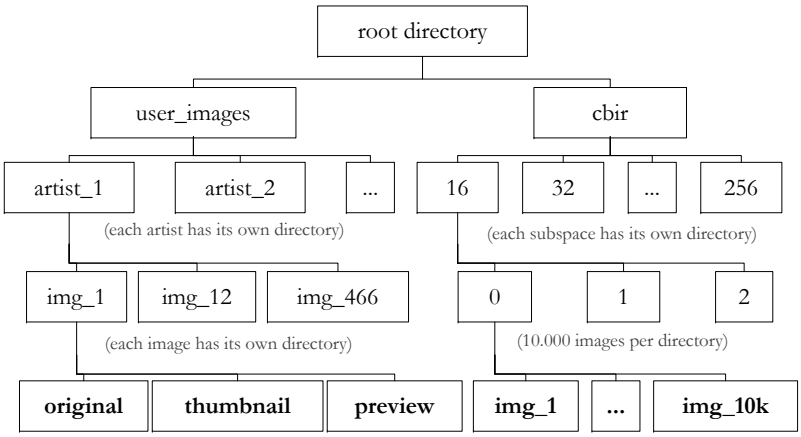


Figure 4.2: Directory structure for the files stored in disk. Due to filesystem restrictions we limit the number of files per directory to ten thousands (10.000).

One of the bottlenecks that we face was the number of files per directory that the filesystem could handle as performance quickly deteriorated after a certain point. That forced us to implement a restriction on the number of files per directory, 10 000. And not only resulted on good retrieval-time for random and sequential accesses but also allowed us to keep a sane file structure.

4.4 Business Process Architecture

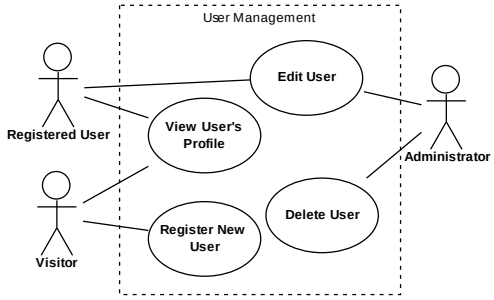
The business processes that the application must support must reflect the requirements that were set during the requirements analysis and refined during the implementation stage. These business process must reflect the most important activities of the application, either user triggered or not. But the most important outcome of these stage is to detect repetitive operations and redundant implementations of the same function that will be removed and unified in a sub-process that is shared. The actual result of these stage led to the creation of utilities classes that perform low-level operations, such as image scaling or disk accesses that are implemented only once and reused in the different processes.

First we need to determine the different actors in the system, which according to the requirements are the following:

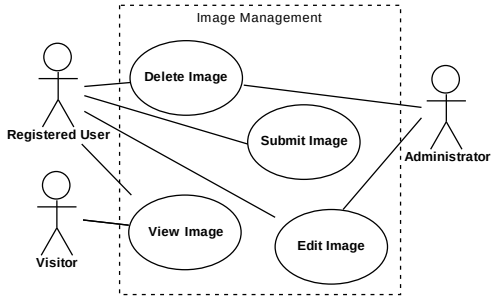
- Visitor, which is a user that has accessed the site but does not have an user account. This user can browse and search the site but does not have a profile and a image portfolio;
- Registered User, which is a user that has an artist account in the website. This user has more privileges than an visitor, as it has his own profile and is able to submit artwork and open exhibits.
- Administrator, which is a user that can manage all the accounts, image and exhibitions on the application. This user has full access to the non-confidential data, that is, he

cannot access the user password.

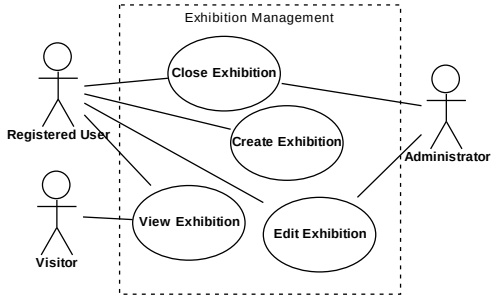
According to the requirements we decided to divide the use cases in four different categories, (a) User management (b) Image management (c) Exhibition management (d) search queries.



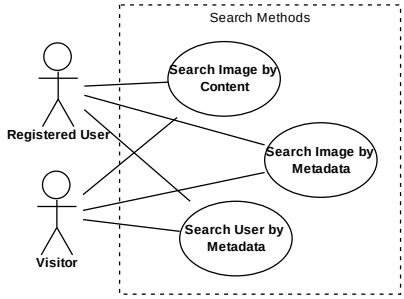
(a) User management use cases.



(b) Image management use cases.



(c) Exhibition management use cases.



(d) Search scenarios use queries.

Figure 4.3: The different use cases diagrams divided by categories.

The functionality that is implemented for these use cases is in general common among the different scenarios, as they mostly relate to management functions of the objects itself. The common actions for the user, image and exhibition management are the creation of new objects, editing, deleting and browsing of the objects. These actions are very similar differing only on which information they manipulate and how to display it, as each has different relevant information.

The exhibition has just one particularity as an artists can close and exhibition but this doesn't imply it is deleted from the application, as it may open it at a later date.

The search scenarios have its own unique use cases, with the different search queries that work similarly but return different types of information. There are two types of queries that the user can perform, the image and artist search. They query different data from the database, and return the results using a different presentation model, but are in essence similar, at least for the metadata search. The content-based search that can also be performed over the image collection uses a different search engine from the metadata search. In contrast these search will look at the image content instead of text strings in the metadata, but presents the results in a similar way.

4.5 Application Architecture

The application architecture that was developed results from a client-server architecture where the user interface, application logic and data access are logically separated as seen in figure 4.4. This type of architecture allowed us to implement a flexible and modular system, where each layer can be modified or even replaced without affection the other layers, as long as the methods' interface remains the same. For example, we can replace the database engine without any impact to the application's logic and interface.

- The user interface will define the art portal's web pages, allowing the user to browse the portal and access the information;
- The application logic layer is the core of the system where all the features in the portal are implemented here such as the user, image and exhibit management logic;
- The independent modules congregates the independent functionality that is used in the system, in our application that refers to the CBIR module and other frameworks that are used, for instance the java advanced imaging;
- The data access layer provides an interface for the application to access the data, i.e. the application reads and changes data in the database through an API making the operations transparent;

Our chosen development framework used to implement the system, WebObjects, had a big role in achieving this separation as it encourages it by design. The best example was the database access as it was unknown to the application which database was used or the query syntax. Using the interface we defined which database connector driver should be used (in our case jdbc) and some store procedures that involved more complex queries. The store procedure were defined using abstract objects that represented each object in the database (tables). Manual writing of queries should only be done in specific cases, which we didn't need. The application itself was implemented using a object-oriented programming language (Java) while the interface was designed in a framework-specific markup language that merged HTML and framework bindings.

4.5.1 Application

The application was developed using the Java foundation and WebObjects frameworks that allowed us to implement a modular system as described above. The actual application has three separate modules:

1. The application's logic that is responsible for all the underline logic and data, and provides the methods that the interface layer needs to serve as a front end to the application;
2. The utilities classes that provide the application with the tools it needs to manipulate images ("ImageUtils"), disk accesses("FileUtils"), validation of data fields ("ValidationUtils"), tracking of both memory usage and timing of operations ("MemoryUsage" and "Timer"), and other auxiliary classes that are needed ("BulkUtils", "SubspaceUtils", "UserUtils");

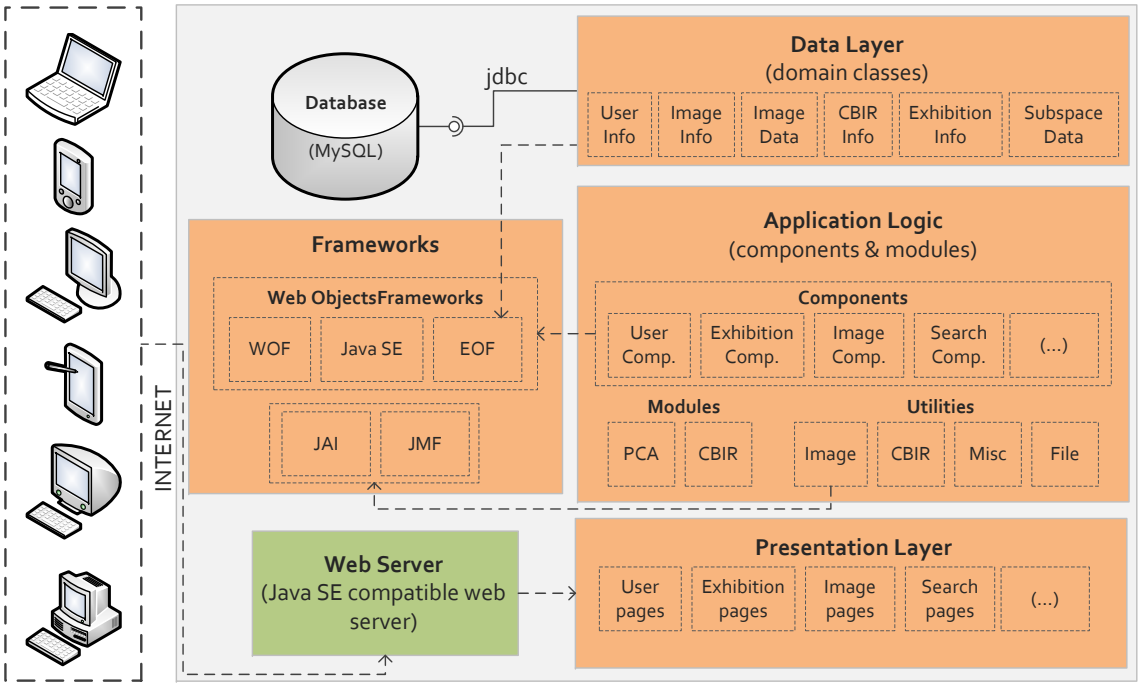


Figure 4.4: System's architecture.

3. The content-based image retrieval module that gives the application the tools it needs to create a high-dimensionality index on the image collection and the necessary methods to query it.

This latter module is the result of the research that was done in content-based image retrieval and high-dimensional indexing of multimedia images. It was planned to be an independent module that provides the application with the search algorithms and indexing capabilities of multimedia images. This module can be integrated with the web application or used as a standalone application for experiments. The value of developing the content-based image retrieval as an independent module is to allow it to be used in other contexts and research papers.

This required us to design the content-based image retrieval module as a "black box" loosely coupling with the rest of the application. The application only has access to some methods through an interface, which can be called with a given input (an image, the indexed collection and a ε-value) and then it returns a list of images ordered by similarity as an output.

The final version of the application has all the dependency identified and in case the content-based image retrieval functionality is removed, it should produce little impact as only few lines of code are affected. The opposite is also valid, for example, we can change the mapping function of the content-based image retrieval module to the principal components analysis without any impact to the rest of the application.

4.5.2 Interface

The interface was designed using a minimalistic design focusing on simplifying the task to ease the learning curve and offer a positive experience to users. Dealing with multimedia images and

artists we had to give special attention to the content having the homepage displaying the most popular and recent artwork and every other page on the site, either user, image or exhibition highlight the artwork.

The implemented interface of the application is very flexible due to the separation of content and styling that was enforced for many reasons, but especially to reduce the complexity and avoid repetitive information that bloated the web pages. As the styling of all web pages in the application were designed to follow the same look, we were able to create a general styling framework that is included in every page as a link. Ideally, as this styling information is static it should be stored in a separate server that only serves static data.

The actual implementation was done in hybrid extensible markup language that combines HTML tags with framework-specific bindings. This file dynamically generates HTML 4.01 transitional web pages that hold all the content, i.e. text, metadata, images and other objects. The file keeps all information logically separated and because we keep the content separated from styling the HTML file alone resembles an XML file that by itself produces a plain web page. The styling is not only common to every page that the application generates, but it is also kept external and referenced using a link tag.

5

Results

Contents

The experiments chapter presents the results of all tests done on the high-dimensional indexing techniques researched in the thesis scope and on the web application that serves as the interface for the implemented algorithms.

In line with the objectives that were defined in the beginning these experiments should test how the implemented algorithms perform, but also whether the application can be deployed for general use.

For this we setted up a sizeable collection of multimedia images retrieved from a popular art portal and tested different values for the ε-value threshold in order to gather information on which is the best value for this collection. By having this information we can balance the need of a populated result set and allow for the query to run in real-time.

5.1 Test framework

The multimedia image collection was built using a set of 30.000 images was downloaded from flickr.com so that it would simulate a real world scenario. All the downloaded images were published under a creative commons license that allows for non-commercial use that include academic and research purposes. In addition metadata information about the author and image was retrieved in order to comply with the terms of the license, when required.

A web crawler was used to download and retrieve the information from flickr as it performed different queries on the site with specific search parameters. About 3.000 images were downloaded with each query along with the necessary information where the following search terms were used: "abstract"; "art"; "arte"; "color"; "graffiti"; "landscape"; "paint"; "people"; "portrait"; "street". As a result the image collection included various types of artistic image and themes ranging from photography and portraits to oil paintings and digital art.

The original resolution of the images could not be used for these experiments as they are very different and therefore a standard resolution was specified with all images being scaled to that size. We chose the 256×256 pixels resolution as the standard for which all images are scaled. While this process loses a lot of information, especially on the quality of the image's content, it still maintains all of the important features necessary for the implemented algorithms to work, such as colour distribution and objects' shapes. This resolution will store 3-band RGB information for each pixel, that range from 0 to 255 projecting the original colour spectrum to this range and colour mode.

We had to perform additional pre-processing for each image in the collection by applying the orthogonal projection mapping function, used in the hierarchical linear subspace method. In contrary to the PCA implementation this information does not need to be recalculated and can be stored in cache, either on database or in disk.

Each image was scaled to the 128×128, 64×64, 32×32, 16×16, 8×8 and 4×4 resolutions using the mean value of the pixels' colour information.

For the principal components analysis tests we used a 32×32 scaled image in order to fit the memory requirements for PCA and available resources. And although there is some loss of information when reducing the images' size to this resolution, it is still representative enough of the original image as the layout and largest objects are still outlined. In these tests we also used the

32×32, 16×16, 8×8 and 4×4 subspaces generated for using the orthogonal projection mapping in order to compare the two mapping functions, their performance and results.

The collection itself needs 17 GB of disk space, and though this accounts for the most important information, the original images, the data that is generated when an image is inserted in the application needs additional disk space. As mentioned before when a new image is inserted it goes through pre-processing phase that generates all the necessary images and the data for the content-based image retrieval. It generates two scaled images that are used as thumbnail and preview images in order to optimise network bandwidth for the web application when deployed. These images need about 4 GB of disk space.

The data for the content-based image retrieval queries, in particular when using the orthogonal projection mapping function on the hierarchical linear subspace method, also requires large space in disk as table 5.1 shows, in total it needs about 7.32 GB. On the other hand, the principal components analysis mapping function does not require any pre-processing when inserting images. This function uses a data-dependant transformation and it has to calculate the subspaces in run-time and then discard it as the following transformation might apply a different transformation matrix.

		collection (MB)	
subspace	**each image** (kB)	**size**	**size in disk**
4x4	0.05	1.37	117.00
8x8	0.19	5.49	117.00
16x16	0.75	21.97	117.00
32x32	3.00	87.89	117.00
64x64	12.00	351.56	351.57
128x128	48.00	1 406.25	1 406.30
256x256	192.00	5 625.00	5 625.19

Table 5.1: Disk usage for the content-based image retrieval data.

Two different computers were used to generate the data for the experiments.

- A laptop running Linux operating system with a 2.2GHz Core2 Duo processor and 2GB of RAM memory;
- An Apple iMac running Mac OSX 10.4.11 operating system with 2.0GHz Core2 Duo processor and 1GB of RAM memory;

The laptop was used to generate the characteristics for the orthogonal projection and PCA functions implementation of the hierarchical linear subspace algorithm. It was also used to calculate the principal components for each sample input, since it was the only computer that could meet the memory requirements for such a large collection. The iMac was used to generate the collection, i.e. import and scale the images, and to run the web application.

The web application was built on top of the WebObjects framework and Java platform (Java SE), using many of the Java Advance Imaging (JAI) API for the image methods. We also used

a modified version of the Matrix Package (JAMA) was also used to optimise the eigenvectors decomposition computation. Then Mathematica was used to compile and plot the results of the tests.

5.2 Orthogonal Projection Results

The experiments that were done using this mapping function were performed with the main objective of documenting the hierarchical linear subspace method performance. But also how with just using a simple mapping function, that calculates the average pixel value, we can reduce the query time from minutes to just a few seconds proving that this function is mature and returns good results in a real world scenario that requires real-time results.

For this we chose to use the original space and six different subspaces with different dimensions in the experiments in order to test the mapping function and analyse the results. With the results we will be able to draw conclusions on whether it is worth to use this function but also if all the chosen subspaces are necessary to boost up queries. In the end we must balance the disk usage requirements with the accesses and comparisons that are done with the improvements in the results if there are any. The resulting application that is ready for deployment will have to review the results and determine which subspaces should be kept.

5.2.1 Metric indicators

For our experiments we chose to evaluate the performance of the method, the number of comparisons that are required as well as the number of images under the ε-value threshold. Both these indicators are objective and can clearly quantify the results, something that we cannot do for more subjective indicators such as the quality of the results.

Performance indicator holds two types of information:

1. The number of disk accesses that are required and how long it takes to load the data into memory;
2. The overall time it takes for a query to perform a full query of the collection;

These metrics, though objective, are ultimately dependant on the hardware were the application is deployed and on all the processes that are running, for example the disk specifications, block size, the operating system processes, Java's garbage collector, etc... All these influences the running time of a query, accordingly we will try account for these variations and minimise their impact on the final results.

Number of comparisons indicator will estimate the number of operations being done in each query and is the most impartial metric we can define, as it is independent from hardware and test conditions. And although it is a static number for the list matching method, it varies with the hierarchical linear subspace as this method depends on several variables,

i. the query image;

ii. the collection;
iii. the ε-value threshold.

The method will progressively reduce the size of the collection, as it visits each subspace, and consequently the number of comparisons needed. For example, for a list matching query using the test framework's collection we require 5 898 240 000 comparisons, whereas the hierarchical linear subspace method using the orthogonal projection function the number of comparisons ranges from 1% to 60% of the list matching.

Number of images under the ε-value threshold is directly related with the previous indicator as it shows how many images are discarded in each subspace but also, on testing the ideal size of the result set defined by the ε-value threshold.

Ratio indicator will calculate how many times does one algorithm outperforms the other, i.e. if the hierarchical linear subspace method using the orthogonal projection mapping function requires 5 times less operations than the list matching, we can conclude that it is 5 times faster than the list matching method. This indicator is extremely relevant in quantifying the gains of one method against another.

5.2.2 Characteristics

In order to estimate ε-value we defined a mean sequence (see equation 5.1), which describes the characteristics in the collection. It was impractical to calculate the characteristics using all images in the collection, as it would take approximately 70 days just for the original space, we then choose a 10% image sample that comprises of 3 000 images that present with a representative set of the collection. The images were chosen randomly from the collection making sure that group (defined by a keyword) had its share in the sample.

We chose the Euclidean distance as the distance function for orthogonal projection mapping function experiments, these function was used to calculate the distance between the query image and every image in the collection.

The calculation of the characteristics was done by querying each image of the sample against the entire collection using list matching algorithm in the original space and respective subspaces. We then ordered each query's results from the smallest to the highest value and then calculated the data's mean value which describes the characteristics of an image collection in our system:

$$[U_k(DB)]_n := \sum_{i=1}^{s} \frac{d[U_k(x^{(i)})]_n}{s}. \tag{5.1}$$

The characteristics allows us to estimate ε-value as shown in figure 5.1 and determine in average how many images are returned by using any given ε-value.

We set the ε-value at 53 014.290 and as a result we can see that in average the hierarchical linear subspace method will discard 15 854 images in the 4×4 subspace, 5 828, 3 978 , 2 228, 1 061, 540 and 311 in the others, respectively. By using this threshold the results will in average return 200 images and require 3% of the comparisons of the original space, i.e. if the query was performed exclusively

using this space. In fact, the application in average will only query the query image against 511 images in the original space. By using this method we can progressively discard irrelevant objects in the collection and minimise the comparisons that are made in the higher subspaces. The only drawback is that it will require additional storage space and disk accesses that adds complexity to the application and in the maintenance of the data integrity.

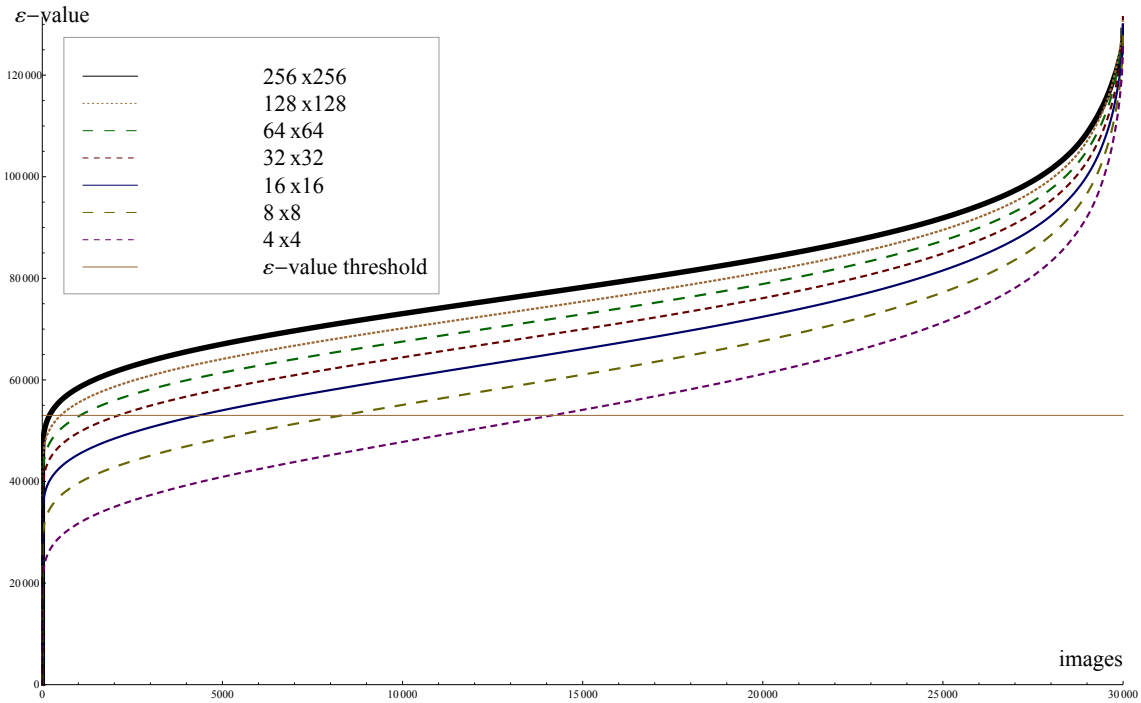


Figure 5.1: Characteristics plot with ε-value = 53 014.29.

This mapping function satisfies the lower bounding lemma [25] described in GEMINI [24, 28], therefore we can observe that the values of the any subspace are always lower than the other larger subspaces, including the values of the original space.

Table 5.2 shows how many images are discarded in each subspace and how it varies as the ε-value increases and more images fall into the threshold. Figure 5.2 plots the data in table which provides a visual help showing that there is not a linear growth in the number of images kept at each subspace as ε-value increases. At each point in the plot it shows the number of images that are below the ε-value threshold when querying the collection, i.e. the lower the line the higher the number of discarded images. We can observe that there are two points in the plot where the values start to grow faster (near the 40 000 and again about 10 000 after), this change are noticeable in the 8×8 subspace.

If we query the collection on the original space, list matching, the application needs to perform $256 \times 256 \times 3 \times 30\,000 = 5\,898\,240\,000$ pixel by pixel comparisons. Using the hierarchical linear subspace we can dramatically reduce this number by as much as 70.78 times, depending on the ε-value.

Analysing the results we can conclude that the hierarchical linear subspace is significantly more efficient than the list matching method (querying in the original subspace) on every scenario that

ε-value	256x256	128x128	64x64	32x32	16x16	8x8	4x4
49 715.15	50	169	424	1 026	2 544	5 818	11 529
51 242.28	100	293	664	1 464	3 306	6 933	12 744
52 241.43	150	404	865	1 810	3 868	7 704	13 539
53 014.29	200	511	1 051	2 112	4 340	8 318	14 146
55 840.67	500	1 084	1 957	3 488	6 302	10 639	16 318
57 303.67	750	1 513	2 583	4 372	7 434	11 871	17 390
58 434.44	1 000	1 915	3 150	5 131	8 358	12 827	18 187
60 198.83	1 500	2 678	4 176	6 429	9 854	14 311	19 376
63 781.03	3 000	4 767	6 761	9 409	12 994	17 187	21 548
67 056.12	5 000	7 266	9 557	12 359	15 826	19 574	23 229
73 047.76	10 000	12 740	15 104	17 634	20 419	23 137	25 599

Table 5.2: Number of images below the ε-value threshold on each subspace.

doesn't involve returning the entire collection of images. This exception is valid, because when we define the threshold high enough to return the entire collection, we are forcing the method to query all images in all subspaces, which in practise will need more comparisons than the list matching method. In this exception the list matching method would still require the 5 898 240 000 comparisons while the hierarchical linear subspace would need an additional 1 965 600 000 comparisons.

When testing the different ε-value threshold we can observe that when using the lowest threshold this method can outperform the list matching method 70.78 times which is impressive as it can reduce queries from 3 minutes and 10 seconds to just 2.688 seconds. These results are verified from our tests using the overall time it takes for a query to return the results in table 5.3. In this table we can observe that the predicted ratio, which is calculated based on the savings in computational requirements is very similar to the real value.

The results and speed of this method depend heavily on the chosen threshold. If a high threshold is chosen more images will be included in the results and more comparisons will be required and more images will be queried in the higher dimensional subspaces. This reflects negatively on the query time but increase the number of results. On the other hand, if we lower the threshold the query will become faster as more images are filtered in the lower dimensional subspaces. However, the result set will have fewer images. It is required to weight the consequences of the different thresholds available in order to chose one that meets the goals of the application where this method is implemented.

Table C.1 shows the prediction of the number of operations required for different combinations of subspaces that could be used. This data was calculated using the characteristics data. With this information we were able to predict in average the number of comparisons necessary to perform a query using the hierarchical linear subspace method and the mapping function we are evaluating in this chapter. Each column of the table shows the subspaces that were considered and the difference of the number of comparisons, in millions, when using all the subspaces (256×256, 128×128,

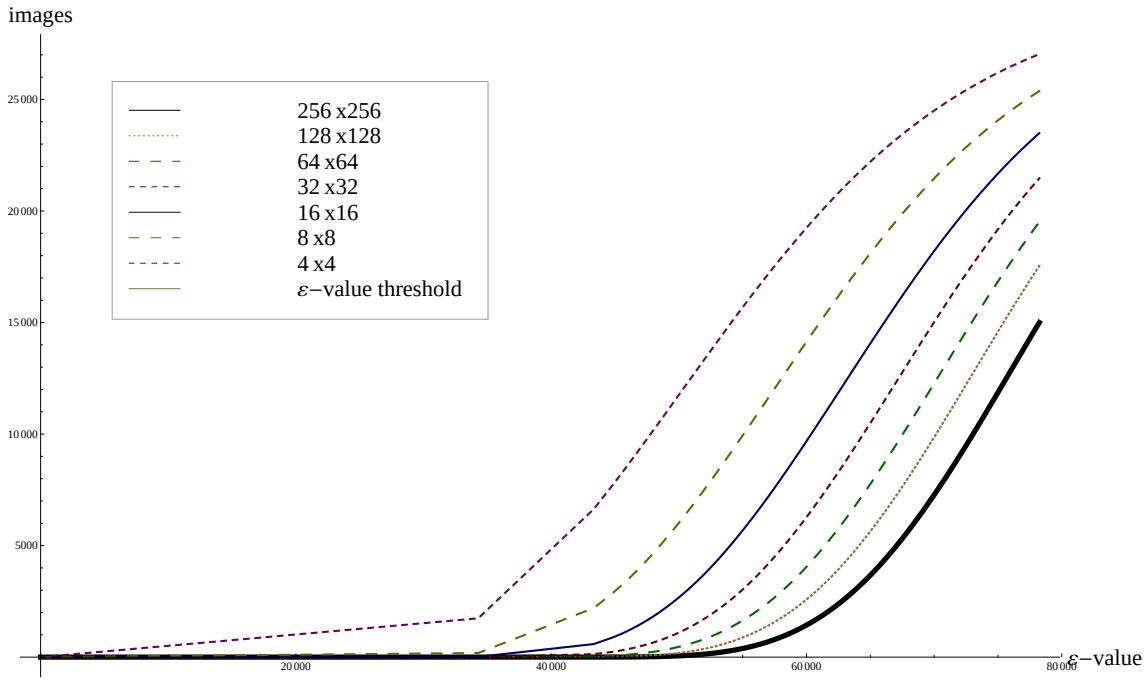


Figure 5.2: Graph that plots the number of images that are under the ε-value threshold as it changes.

64×64, 32×32, 16×16, 8×8 and 4×4) against using the marked subspaces.

After analysing the data we can see a need to keep all the available subspaces when using the hierarchical linear subspace method. These results show that when querying the collection using different subspaces we see an increase in comparisons with every combination for ε-values under 63 781.03.

The quality of the search results is an extremely difficult to quantify and the existing methods for doing so either, rely on human test subjects or in external systems. We performed a series of sample queries and extracted two that showed how the application performs and what problems affect these type of implementations (see appendix D. This high-dimensional indexing technique performs well but at the core still uses the pixel's RGB information as the main feature when comparing images. The distance function that is used, Euclidean Distance, compares pixel by pixel and band by band the colour information of two images to calculate their distance.

This has an impact on the quality of the results as images that are completely different may return as similar if they have a common colour distribution. If the first sample query (see figure D.1) the application returns similar results of landscapes with mountains in the background. These results are satisfactory as the content is similar. On the other hand, the second sample query (see figure D.2) returns very dissimilar images that look very different from the query image. There are two explanations for this result that are not mutually exclusive, either the (i) collection has a very limited set of images that show houses, or (ii) the algorithm is flawed or incomplete. After reviewing the rest of the result pages, we detected some images that should have ranked higher, nevertheless none was very similar to the query image.

In general our implementation returns good results and in an acceptable time. These anomalies can be minimised by introducing new features for the algorithm to consider, such as shape, texture,

ε-value	Orthogonal Projection (ms)	Standard Deviation (ms)	Comparisons	Ratio	Predicted Ratio
49 715.15	2 688	210	13 984 704	70.78	71.40
51 242.28	4 142	277	40 698 240	45.93	46.22
52 241.43	5 383	339	74 304 192	35.34	35.53
53 014.29	6 544	397	128 105 472	29.07	29.21
55 840.67	12 431	532	279 273 600	15.30	15.35
57 303.67	16 647	655	382 818 816	11.43	11.46
58 434.44	20 519	883	492 031 296	9.27	9.29
60 198.83	27 695	1 294	700 515 648	6.87	6.88
63 781.03	46 625	1 485	1 260 938 688	4.08	4.09
67 056.12	68 431	1 669	1 886 741 952	2.78	2.78
73 047.76	114 595	1 844	3 292 073 088	1.66	1.66

Table 5.3: Query time of both the hierarchical linear subspace method and list matching and respective ratio to determine how many times it outperforms the latter method.

etc... In terms of performance the hierarchical linear subspace method presents encouraging results when combined with the orthogonal projection mapping function. However, an optimal threshold that balances the number of hits in the results with the query time is necessary. For situations where not all results need to be calculated this high-dimensional indexing technique and function can be deployed guaranteeing a good performance, especially with high ε-values.

5.3 PCA Results

The purpose of this experiment is to test whether PCA can become a viable mapping function for the hierarchical linear subspace method to reduce the high-dimensionality of multimedia image in CBIR systems. It will also be compared against the orthogonal projection function in order to determine which is the best mapping function for the hierarchical linear subspace. We will apply the PCA on a multimedia collection and then perform queries in order to test it.

The PCA behaves well with high dimensionality up a certain order, where the memory and computational requirements become massive. When calculating the principal components' matrix, the dimension size is crucial, as:

$$Dim = resolution\, of\, the\, image^2 \tag{5.2}$$

If we have an image with MxN resolution with a 3-band RGB representation and 64 bits precision then the eigenvectors matrix's size is:

$$Dim = (M \times N \times 3)^2 \times 64 \tag{5.3}$$

Table 5.4 shows the memory usage for different image resolutions.

		Memory usage (MB)	
Resolution	**Eigenvector Matrix cells**	64 bits precision	32 bits precision
256x256	38 654 705 664	294 912.00	147 456.00
128x128	2 415 919 104	18 432.00	9 216.00
64x64	150 994 944	1 152.00	576.00
32x32	9 437 184	72.00	36.00
16x16	589 824	4.05	2.25
8x8	36 864	0.28	0.14

Table 5.4: Eigenvector's matrix memory requirements for the different image resolutions.

The PCA is most effective when dealing with similar and correlated data, as the more information the data has in common the more dimensions it can positively discard. Keeping only the variables that show the most variability. When dealing with sparse data the PCA method does not perform as well, since most variables are not correlated. Multimedia images are very different and unless a specific type of image and theme are targeted, such as portraits or landscapes, we may find that most of the variables cannot be discarded. Therefore when using very sparse data we may not be able to reduce the dimension of the collection to a desirable size that makes using the PCA useful and worth applying.

In order to avoid this problem with sparse collections we can try to find a smaller sample of images that is representative of the collection and apply the PCA based on that sample. We will be using the Euclidean distance to calculate the distance between the multimedia images.

Preparing the PCA data

We choose to use the co-variance method to calculate the principal components and then apply the Kaiser criteria to determine which to keep. This approach does not allow us to choose the resulting number of principal components as it is dependent on the variability of the original data. Because of this we needed to use different samples of the collection as input for the PCA and then choose which ones to keep for the tests, table 5.5 shows the results and the samples that were chosen for the next step of the tests.

Results of the experiments

In order to compare the PCA against the hierarchical linear subspace method we use the characteristics data to analyse the results. This allows us to predict the average number of retrieved images at each dimension that is determined by the ε-value, i.e. only the images with a Euclidean distance inferior to the ε-value are retrieved.

To estimate ε-value we define a mean sequence which describes the characteristics of an image in our system:

Sample size		Resulting Principal Components
100	(*)	99
200		199
300		299
400		399
500	(*)	498
600		598
700		698
800		797
900		895
1 000	(*)	995
1 500	(*)	1 487
2 000	(*)	1 969
3 000	(*)	2 527
6 000		2 997
6 500		3 022
7 000		3 040
7 500		3 054
9 000	(*)	3 072

Table 5.5: Principal Components generated for each of the sample sizes, the (*) marks the chosen dimensions for the CBIR tests.

$$d[U_k(DB)]_n = \sum_{i=1}^{n} \frac{d[U_k(x^{(i)})]_n}{n} \tag{5.4}$$

Table 5.6 shows the time it took for to calculate the characteristics data, using a 10% sample from the collection, 3.000 of the 30.000 images in the collection.

In figure 5.4 we see that the ε-value threshold can be applied to the subspace with 995 dimensions and higher, whereas the characteristics for the lower subspaces are below the threshold. Consequently we can use the PCA method to reduce this collection to less than one third of the original dimension, and while the number of operations are significantly reduced we still need to consider whether it is worth to use the smaller subspaces as few images are discarded.

The number of images retrieved on each subspace must be below a certain threshold otherwise it is not worth to use that subspace in the query, as it will not give an advantage. The following formula calculates the number of images that must be retrieved in order for the dimension to be worth using.

PCA Sample	Principal Components	Transformation to the New Coordinate System (time)	Applying the PCA (time)
100	**99**	4 m 39.90 s	05 m 20.24 s
500	**498**	7 m 39.60 s	10 m 32.82 s
1000	**995**	9 m 30.97 s	15 m 43.70 s
1500	**1 487**	10 m 14.45 s	20 m 50.78 s
2000	**1 969**	11 m 23.84 s	26 m 4.36 s
3000	**2 527**	13 m 23.44 s	31 m 8.36 s
9000	**3 072**	14 m 56.26 s	36 m 24.3 s

Table 5.6: Performance of the PCA method under different samples.

$$number\,of\,images < size\,of\,collection \times \frac{(dim_i - dim_j)}{dim_i}, dim_i > dim_j \tag{5.5}$$

For example, the combination of using the 498 and 995 subspaces does not discard enough images to reduce the number of calculations. When the formula above is applied we see that the minimum number of images that must be discarded is 14.985, which in turn needs a very small ε-value that only returns two images in the original space and this is just the value where using this dimension is computational cheaper.

This behaviour is present on many of the possible combinations and unless we use dimensions that are far apart we don't significantly reduce the number of operations necessary to process the query. Table 5.7 shows some of the possible ε-value that can be used and the number of images retrieved at each dimension.

	Images retrieved for each dimension in PCA						
ε-value	3072	2527	1969	1487	995	498	99
3 225.95	200	833	3 250	8 879	20 990	29 350	30 000
3 427.48	500	1 686	5 422	12 351	23 817	29 698	30 000
3 616.26	1 000	2 894	7 937	15 621	25 772	29 893	30 000
3 745.73	1 500	3 976	9 850	17 740	26 771	29 967	30 000
3 848.21	2 000	4 979	11 427	19 297	27 402	29 995	30 000
3 934.79	2 500	5 914	12 777	20 511	27 841	30 000	30 000

Table 5.7: Number of retrieved images with each dimension.

For the same ε-values the table below shows the average number of operations needed for the query, starting by comparing in that dimension and then using the result as the input for the next dimension. Among the different existing combinations the best result is achieved when we start with the 1969 dimension and that only is valid until a certain ε-value. And although these results are slightly better than when searching in the original space, the number of required operations is

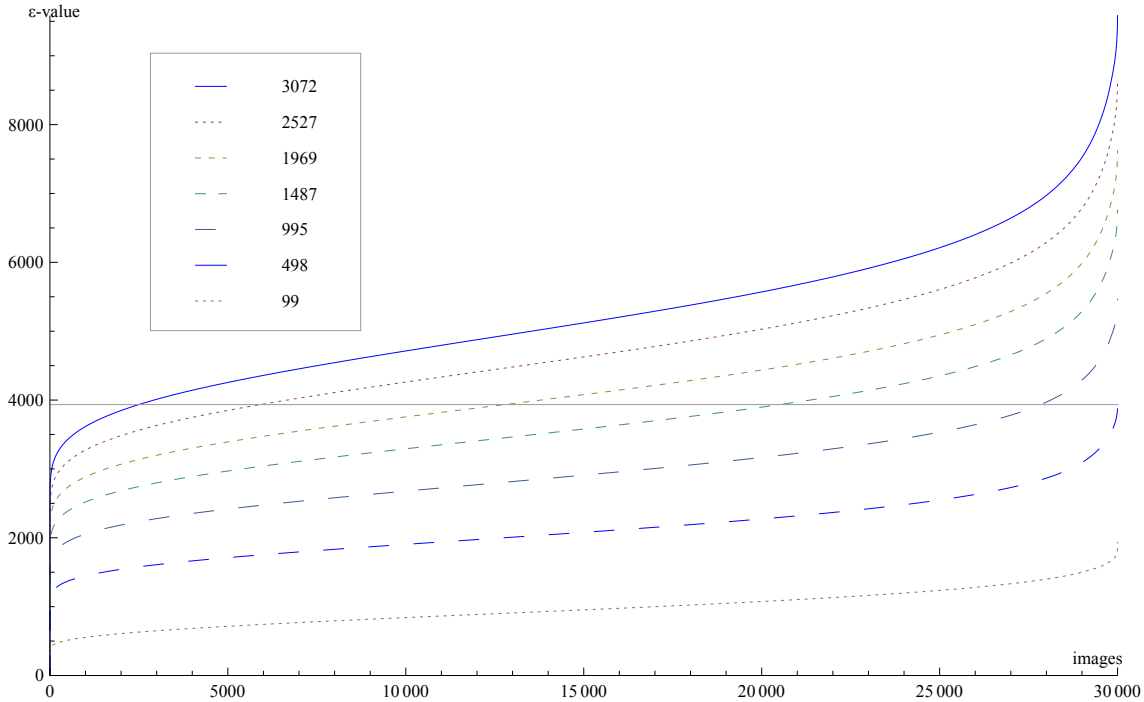


Figure 5.3: Characteristics plot using the PCA function and $\varepsilon = 3\,934.79$

still considerable higher when compared with the hierarchical linear subspace method.

	Operations necessary for the query (in millions)						
ε-value	**PCA method**						**Orthogonal**
	3 072	2 527	1 969	1 487	995	498	**Projection**
3 225.95	92.16	78.37	69.84	72.86	89.32	103.61	6.92
3 427.48	92.16	80.99	77.95	87.81	108.47	123.11	10.51
3 616.26	92.16	84.70	88.02	104.31	127.88	142.71	15.13
3 745.73	92.16	88.02	96.18	116.65	141.69	156.60	18.99
3 848.21	92.16	91.11	103.24	126.78	152.76	167.70	22.45
3 934.79	92.16	93.98	109.53	135.45	162.09	177.03	25.64

Table 5.8: Number of comparisons required in average for the PCA and the orthogonal projection.

The hierarchical linear subspace method is able to achieve such a better performance because it applies for each subspace a constant that estimates the results in the original space, allowing the characteristics to be close together and converging to the same value, as shown in figure 5.4. This allows the use of very small subspaces that greatly reduces the calculations, for example we can use the 4x4 subspace that has 48 dimensions.

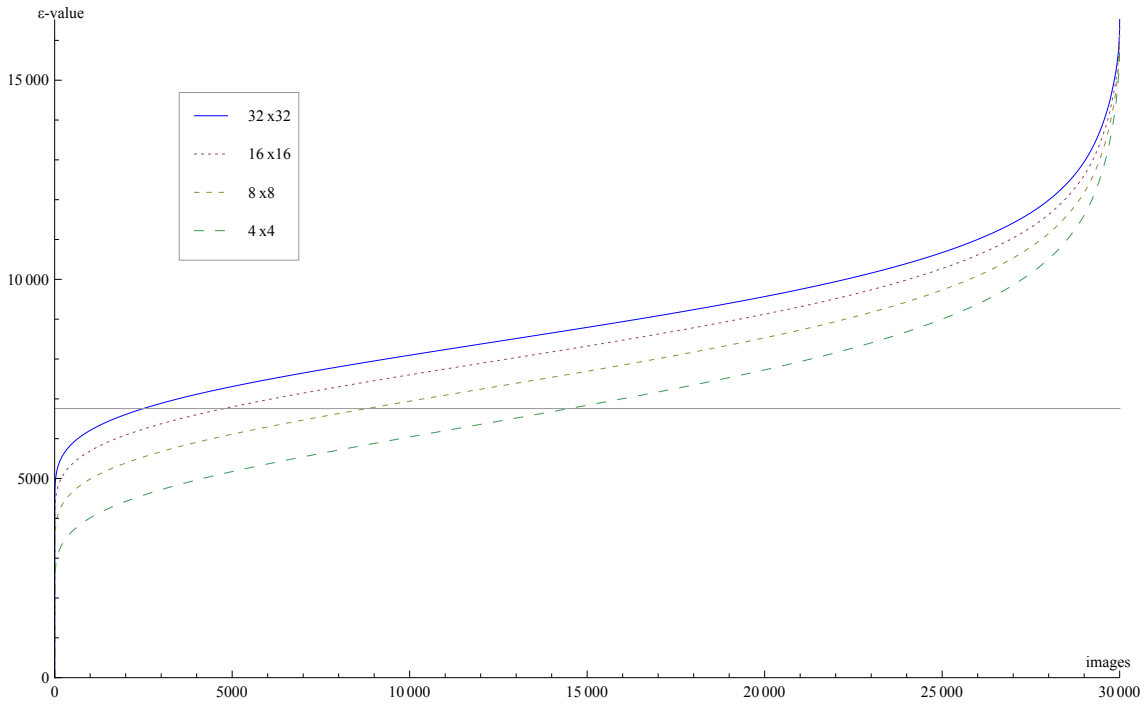


Figure 5.4: Characteristics plot using the orthogonal projection function and $\varepsilon = 3\,934.79$

5.4 Orthogonal Projection vs. Principal Components Analysis

The principal components method performs poorly with a large sparse collection of images as it does not significantly reduces the number of calculations required to perform a query on the collection, i.e. although it can reduce the dimension of the images, the result cannot be used to greatly improve the performance of a query. When compared with the orthogonal projection function it needs up to seven times more operations to execute the query, which translates on more computation power required to run an application that uses this method.

The main problem with using the PCA as a mapping function for the hierarchical linear subspace method is due the distance between results of different subspaces, as the loss of information when reducing the dimension cannot be compensated or approximated.

6

Conclusion

Contents

The development and research done in the scope of the thesis resulted on an web application that fulfills the initial requirements of being able to serve an artist community and perform content-based image retrieval queries on its collection. The series of experiments lead to a real case study on the viability of using principal components analysis to create a subspace pyramid on a multimedia image collection. Our experiments also demonstrate encouraging results when using the the orthogonal projection combined with the hierarchical linear subspace method. They show that this combination can be used for CBIR queries in a live web portal with large collections. We can then conclude that the master thesis achieved all the goals that were initially proposed.

The developed art portal is an web application that is capable of hosting images of digital artwork created by artists and it is "content-addressable. Any user, whether they are artists or not, can enter the art portal and search for art by executing a query by example using their image or another artwork on the portal. This application was developed as a modular system that is extremely flexible and open. It is ready to be extended or retrofitted for other purposes, such as other master thesis. This platform can be used as a starting point to test other algorithms in content-based image retrieval and compare against the implemented methods.

One of the best features of our implementation is the creating of an independent CBIR module that is used in this application, but can easily be adapted for other uses that require CBIR functionality. For instance, medical systems, image hosting portals, copyright infringement detection, etc...

The algorithm researched on this thesis show great potential, but still require more study as they still use too much irrelevant information when querying the database. From the experiments and existing research by other authors we concluded that the hierarchical linear subspace method is the most efficient and presented the best results from all algorithms. This was proven especially when using the orthogonal projection as a mapping function. The principal components analysis revealed too many limitations despite being able to reduce the dimension of the objects. While the list matching was too expensive and time consuming to be considered anything other than a good method to test the distance functions and to compare against.

The orthogonal projection mapping function delivered very promising results, but it still lacks the speed it is needed to be implemented on a collection that stores hundreds of thousands of images. Because it uses low-level features such as colour distribution and then the mean value to reduce the dimensionality, it just does not allow for the scalability required for the current reality. Web portals that can grow at an exponential rate, and this type of application can quickly be required to host millions of images. We have some suggestions that can adapt the implementation of a hierarchical linear subspace method and allow for faster results that we will present in the following section.

The principal components analysis failed completely as a mapping function as it is very limited, especially when it involves the higher dimensional spaces. We showed that the requirements of using covariance matrix method to calculate the principal components requires too much memory resources as the original dimensional space gets higher. In the experiments we could only test using a 3072 dimensional space and calculate the principal components, this is the equivalent of a image with a 32×32 resolution. These dimension does not have the necessary granularity to expect good results from queries. Notwithstanding these limitations, the real problem with this method is that

the distance between objects in the transformed dimension is too different in comparison with the original space. And there is no method of approximate the results, as the orthogonal projection does by introducing the multiplication of a constant. As a result we can not eliminate sufficient objects in the smaller spaces to produce a saving on the number of comparisons that makes an impact on the query time.

Overall we can say with confidence that performing the hierarchical linear subspace method with the orthogonal projection mapping function is a viable method of querying a collection of high-dimensional objects. However as we said, the performance is directly linked to the ε-value threshold, which is extremely important as it need to be carefully chosen in order to limit the number of images in the result set.

6.1 Main Contributions

The main contributions from this thesis are:

- It has shown that the hierarchical linear subspace method can perform an effective indexing of very large objects with dimensions in the order of the hundreds of thousands;
- The PCA method, when used as a mapping function does not achieve the performance and computational savings as the orthogonal projection for a sparse collection. In fact, it performs very badly as we can not approximate the subspaces to the original space;
- A modular and flexible framework that can be extended and used for future projects.

6.2 Future work

The developed application is by design open to further development and allows the introduction of new functionalities for the web portal and other content-based image retrieval indexing techniques. A social component can be added with blogging capabilities for the artists to talk about their work, commenting of the submitted work would also be an interesting functionality. Nevertheless the real value of the web portal is its content-based image retrieval methods and indexing techniques that can be further developed.

In this area several modifications can be done to improve the efficiency of the results. First instead of choosing a standard ε-value for all queries to obey, it would be interesting to implement a dynamic method that adapts the threshold to the requirements of the query. When a user performs a search he is first presented with the initial page results that contains n-images. Because he will not have immediate access to the other results we believe that the following pages can be queried as the he navigates from page to page.

Use case scenario Test_User enters the system and sees an image that he likes, he then performs a search for similar images using it as the search term. The system returns the initial page with the 6 most similar images and links to navigate to other results page. In the backend the system performed a CBIR query that used a low ε-value that returned 30 images. As Test_User navigates

from page to page, once he reaches page 15 (the 30 images), the system recalculates the query using a higher value that will return more results.

In the ambit of content-based image retrieval, it would be interesting to introduce different features to the search method. At the moment only colour distribution is used when searching for similar images, other characteristics, such as shape, textures or even the detection of regions should prove very valuable to improve the quality of the results. Furthermore, if we could detect the style of the image, it could allow the application to automatically categorise images in the collection. So that portraits, landscapes, photos, paintings, etc... could be logically grouped and made available for users to browse through.

Another very important feature that should be researched are other high-dimensional indexing techniques that exist and compare against the hierarchical linear subspace. Despite the dimensionality of the images prevent many algorithms from being efficiently implemented, other methods should be investigated. Other mapping functions should also be tested and compared against the orthogonal projection and principal components analysis. As well as the use of these function using different feature vectors, presently our vector only includes colour information.

During the development of the application we performed a series of undocumented tests that showed good results in compressing the indexed data. As we scale our images down from the original size, we also change the colour model to 3-band RGB with values that range from 0 to 255. Our test show that many colours are repeated sequentially. Thus allowing to save space in disk and possibly some comparisons can be predicted instead of calculated. In other words, if we know the pixel's colour information for the following 5 pixels of both images, we can calculate the distance in the first, and propagate that information without having to repeat the operations. The next step would be to implement the detection of similar regions in the image (see 2.2.1, which would allow for a better compression of the data while keeping the precision in the results.

6.3 Personal Experience

This master thesis was the most challenging, demanding and important work I have ever undertaken. It marks and important step of my academic journey and life. The time that was spent investigating the related work, the application development and finally the writing of the thesis demanded as much discipline as motivation. I do not believe that I could follow through if the problem was not challenging and interesting. Though I do not have an artistic talent for painting and designing, this area always attracted me and added to the initial motivation that set up the pace for work that was done. Furthermore, I wanted to perform new research in a area to developed something that I could call my own, which with the PCA experiments is something that I can proudly say I have accomplished.

During the year that was spent researching and developing for the dissertation I have grown to be more responsible and, because it covers so many areas, it allowed me to improve my qualifications and consolidate the knowledge I have acquired during the course. Additionally, it introduced new technologies that offered new insight into development methodologies and technical skills that can be used from hereon after.

The most important aspect of the work, was that it required for me to develop an idea from conception. For the first time, the majority of the burden relied on me, which also became an interesting aspect as the work that was done (initial research, conceptual design, development, tests and writing) had to be open and flexible, as it evolved to a final result that was to be discovered and not predetermined. This allowed for a test of my skills, technical, conceptual and human, that resulted on this research document.

The master thesis is the culmination of my academic life and serves as the bridge between now and a professional career, one that I am better prepared because this past year.

Part II

Bibliography

Bibliography

[1] D. Forsyth, J. Malik, and R. Wilensky, "Searching for digital pictures," Scientific American Magazine, no. 276 (6), pp. 72–77, June 1997.

[2] P. Ciaccia and M. Patella, "Searching in metric spaces with user-defined and approximate distances," ACM Transactions on Database Systems, vol. 27, no. 4, 2002.

[3] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima, "Spatial indexing of high-dimensional data based on relative approximation," VLDB Journal, vol. 11, no. 2, pp. 93–108, 2002.

[4] A. Olafsson, B. Jonsson, and L. Amsaleg, "Dynamic behavior of balanced nv-trees," in International Workshop on Content-Based Multimedia Indexing Conference Proceedings, IEEE, 2008, pp. 174–183.

[5] C. Zaniolo, S. Ceri, R. T. Snodgrass, R. Zicari, and C. Faloutsos, Advanced Database Systems. Morgan Kaufmann, 1997.

[6] C. Böhm, S. Berchtold, and A. K. Kei, D., "Searching in high-dimensional spaces—index structures for improving the performance of multimedia databases," ACM Computing Surveys, vol. 33, no. 3, pp. 322–373, 2001.

[7] R. Henshaw and E. J. Valauskas, "Metadata as a catalyst: Experiments with metadata and search engines in the internet journal," Libri, vol. 51, no. 2, pp. 86–101, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.102.6248

[8] G. Stamou, J. van Ossenbruggen, J. Pan, and G. Schreiber, "Multimedia annotations on the semantic web," Multimedia, IEEE, vol. 13, no. 1, pp. 90, 86, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1580438

[9] J. Smith and P. Schirling, "Metadata standards roundup," Multimedia, IEEE, vol. 13, no. 2, pp. 84–88, 2006.

[10] A. Dawson, "Creating metadata that work for digital libraries and google," Library Review, vol. 53, no. 7, pp. 347 – 350, 2004. [Online]. Available: http://www.emeraldinsight.com/10.1108/00242530410552278

[11] S. Chang, T. Sikora, and A. Purl, "Overview of the MPEG-7 standard," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 11, no. 6, pp. 688–695, 2001.

[12] B. S. Manjunath, P. Salembier, and T. Sikora, Introduction to MPEG-7: Multimedia Content Description Interface, Apr. 2002. [Online]. Available: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471486787.html

[13] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, no. 12, pp. 1349–1380, 2000.

[14] N. Rasiwasia, P. Moreno, and N. Vasconcelos, "Bridging the gap: Query by semantic example," Multimedia, IEEE Transactions on, vol. 9, no. 5, pp. 923–938, 2007.

[15] A. Hanjalic, R. Lienhart, W. Ma, and J. R. Smith, "The holy grail of multimedia information retrieval: So close or yet so far away?" Proceedings of the IEEE, vol. 96, no. 4, pp. 541–547, 2008.

[16] C. Dorai and S. Venkatesh, "Bridging the semantic gap in content management systems: Computational media aesthetics," 2001. [Online]. Available: http://citeseer.ist.psu.edu/dorai01bridging.html

[17] R. Baeza-Yates, M. Ciaramita, P. Mika, and H. Zaragoza, "Towards semantic search," in Natural Language and Information Systems, 2008, pp. 4–11. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-69858-6_2

[18] J. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: Semantics-Sensitive integrated matching for picture LIbraries," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 9, pp. 947–963, 2001. [Online]. Available: http://citeseer.ist.psu.edu/449651.html

[19] L.-J. Hove, "Improving content based image retrieval systems with a thesaurus for shapes," 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.4856

[20] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: image segmentation using expectation-maximization and its application to image querying," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 8, pp. 1026–1038, 2002.

[21] K. Pearson, "On lines and planes of closest fit to systems of points in space," Philosophical Magazine, vol. 2, no. 6, pp. 559–572, 1901.

[22] I. Jolliffe, Principal Component Analysis, 2nd ed., ser. Springer Series in Statistics. Springer, Oct. 2002. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0387954422

[23] A. Guttman, "R-trees: a dynamic index structure for spatial searching," SIGMOD Rec., vol. 14, no. 2, pp. 47–57, 1984. [Online]. Available: http://dx.doi.org/10.1145/971697.602266

[24] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, "Efficient and effective querying by image content," Journal of Intelligent Information Systems, vol. 3, no. 3/4, pp. 231–262, 1994. [Online]. Available: http://dx.doi.org/10.1007/BF00962238

[25] A. Wichert, "Content-based image retrieval by hierarchical linear subspace method," Journal of Intelligent Information Systems, vol. 31, no. 1, pp. 85–107, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10844-007-0041-4

[26] K. Yanai, N. V. Shirahatti, P. Gabbur, and K. Barnard, "Evaluation strategies for image understanding and retrieval," in Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval. Hilton, Singapore: ACM, 2005, pp. 217–226. [Online]. Available: http://portal.acm.org/citation.cfm?id=1101862&dl=

[27] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: the QBIC system," Computer, vol. 28, no. 9, pp. 23–32, 1995. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=410146

[28] C. Faloutsos, "Modern information retrieval," in Modern Information Retrieval, R. Baeza-Yates and B. Ribeiro-Neto, Eds. Addison-Wesley, 1999, pp. 345–365.

[29] D. M. Squire, W. Muller, H. Muller, and J. Raki, "Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback," Pattern Recognition Letters, vol. 21, pp. 143–149, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.3029

[30] H. Müller and A. Geissbühler, "medGIFT - using the GNU image finding tool for medical image retrieval," 2003.

[31] H. Müller, C. Lovis, and A. Geissbühler, "The medGIFT project on medical image retrieval," University and Hospitals of Geneva, Service of medical informatics, vol. Medical Imaging and Telemedicine, 2005.

[32] L. Dunckley, Multimedia Databases: An Object-Relational Approach. Pearson Education, 2003. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0201788993

[33] G. Vickery and S. Wunsch-Vincent, Participative Web And User-Created Content: Web 2.0 Wikis and Social Networking. OECD Observer, 2007. [Online]. Available: http://www.oecd.org/document/40/0,3343,en_2649_34223_39428648_1_1_1_1,00.html#HTO

[34] M. J. Addis, S. Hafeez, D. Prideaux, R. Lowe, P. H. Lewis, K. Martinez, and P. A. S. Sinclair, "The eCHASE system for cross-border use of european multimedia cultural heritage content in education and publishing," Leeds, Dec. 2006. [Online]. Available: http://eprints.ecs.soton.ac.uk/14265/1/eCHASE_axmedis2006.pdf

[35] P. Z. Paolo Ciaccia, Marco Patella, "M-tree: An efficient access method for similarity search in metric spaces," in VLDB, 1997, pp. 426–435.

[36] A. Andoni, M. Dater, P. Indyk, N. Immorlica, and V. Mirrokni, "Locality-sensitive hashing using stable distributions," in Nearest Neighbor Methods in Learning and Vision: Theory and Practice, MIT-Press, Ed. T. Darrell and P. Indyk and G. Shakhnarovich, 2006, ch. 4.

[37] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: Efficient indexing for high-dimensional similarity search," in Proceedings of the 33rd international conference on Very large data bases, 2007, pp. 950–961.

[38] M. J. Fonseca and J. A. Jorge, "Indexing high-dimensional data for content-based retrieval in large databases," in Proceedings of the 8th International Conference on Database Systems for Advanced Applications, 2003, pp. 267–274.

[39] P. J. Burt and E. H. Adelson, "The laplacian pyramidas a compact image code," IEEE Trans. Commin, vol. COM-31, no. 4, pp. 532–540, 1983.

[40] R. C. Gonzales and E. W. Woods, Digital Image Processing, 2nd ed. Prentice Hall, 2001.

[41] C. Lee and L. Chen, "A fast search algorithm for vector quantization using mean pyramids of codewords," IEEE Transactions on Communications, vol. 43, no. 3-5, pp. 1697–1702, 1995.

[42] H. F. Kaiser, "Comments on communalities and the number of factors," in Conference on the Communality Problem in Factor Analysis, Washington University, St. Louis, 1960.

[43] ——, "A note on guttman's lower bound for the number of common factors," British Journal of Statistical Psychology, vol. 14, no. 1, p. 1, 1961.

[44] R. Cattell, A general plasmode No. 30-10-50-2 for factor analytic exercises and research, 1967.

[45] A. Hakstian, "The behavior of Number-Of-Factors rules with simulated data," Multivariate Behavioral Research, vol. 17, no. 2, pp. 193–219, Apr. 1982.

[46] R. L. Linn, "A monte carlo approach to the number of factors problem." Psychometrika, vol. 33, no. 1, pp. 37–71, Mar. 1968.

[47] L. Tucker, R. Koopman, and R. Linn, "Evaluation of factor analytic research procedures by means of simulated correlation matrices," Psychometrika, vol. 34, no. 4, pp. 421–459, Dec. 1969. [Online]. Available: http://dx.doi.org/10.1007/BF02290601

[48] G. Stockman and L. G. Shapiro, Computer vision. Hemel Hempstead: Prentice Hall, 2000.

[49] E. A. Edmonds, "A process for the development of software for non-technical users as an adaptive system," General Systems, vol. 19, p. 215–217, 1974.

Part III

Appendixes

A

Requirements

Contents

A.1 System's Requirements

R1. The system must separate the data, business logic and presentation.

R2. The application must be modular with its functionality separated and encapsulated.

R3. The system must be able to distinguish different types of users, administrators, registered users and visitors.

R4. The system must be accessible from a web browser.

R5. The system must be able to handle large image collection.

R6. The system must be able to scale in handle an increase in page hits.

R7. The system should be able to offer different licenses to the user when submitting an image.

R8. The application must offer CBIR methods to query the image collection.

R9. The application should offer metadata-based search methods to query the image collection.

R10. The application should offer metadata-based search methods to query the artist information.

R11. The system must allow unregistered users to create an account.

R12. The system must allow any user to query the collection using any of the available search methods (content-based and metadata-based).

R13. The system must allow any user to browse all the images in the image collection.

R14. The system must allow any user to browse all the active exhibitions.

R15. The system must only allow registered users to submit images.

R16. The system must allow registered users to edit their own account.

R17. The system must allow registered users to edit their own images and respective information.

R18. The system must prevent users from editing information that does not belong to them.

R19. The system must allow registered users to create exhibitions with their artwork.

R20. The system must allow registered users to edit their own exhibitions and respective information.

R21. The system must allow registered users to close their own exhibitions.

R22. The system must allow registered users to reopen their closed exhibitions

R23. The system must allow registered users to delete their own exhibitions.

R24. The system must allow administrators to manage all images and respective information.

R25. The system must allow administrators to manage all exhibitions and respective information.

R26. The system must allow administrators to manage all user accounts and respective information.

R27. The system must prevent administrators from accessing confidential and sensitive information (passwords, information deemed private, etc).

B

Information Architecture

User Info (E01) represents every registered user in the system, both the artists and the administrators, and is the only entity that is independent of all others. This entity holds the user's personal and public information, including the contact in case there is a need to communicate with the user outside the context of the system.

Attribute	Description
Username	Public name for the user.
First Name	First name of the user.
Last Name	Last name of the user.
Email	Email contact of the user.
Password	String of characters that allows the system to authenticate the user.
Gender	Gender of the user (male/female)
Date of Birth	Date of birth of the user.
Nationality	Original country where the user was born.
Country	Country where the user is currently residing.
Curriculum	Curriculum of the user.
About Me	Small description of the user.
Privileges	Privileges of the user in the system (regular / administrator).
User Root Directory	Directory where the user's data is stored.

Table B.1: User Info's attributes and their descriptions.

Name of relation	Source	Target	Multiplicity
Image Portfolio	User Info (E01)	Image Info (E02)	1 - *
Exhibits own	User Info (E01)	Exhibition Info (E06)	1 - *

Table B.2: User Info's relation with other information entities.

Image Info (E02) represents every image in the system, containing relevant information and referencing the actual data that is stored in the Image Data (E03) entity. In additional to the original image, we also keep a preview and thumbnail image that saves network bandwidth and frees up resources.

Image Data (E03) represents the actual image data that is stored in disk. We felt the need to separate the metadata from the actual data in order to avoid repetitive attributes and to facilitate the use of different resolutions and the introduction of new features, such as an avatar for the user,

Attribute	Description
Title	The image title.
Keywords	List of words that are associated with the image.
Date Added	Date when image was uploaded.
Description	Textual description of the image content.
Visits	Number of times the image's page was visited.
Downloads	Number of downloads of the image
License URL	Web address of the license under which the image has been published.

Table B.3: Image Info's attributes and their descriptions.

Name of relation	Source	Target	Multiplicity
Image Portfolio	Image Info (E02)	User Info (E01)	* - 1
Preview Image	Image Info (E02)	Image Data (E03)	1 - 1
Thumbnail Image	Image Info (E02)	Image Data (E03)	1 - 1
Original Image	Image Info (E02)	Image Data (E03)	1 - *
Search Information	Image Info (E02)	CBIR Info (E04)	1 - 1
Artwork In Exhibition	Image Info (E02)	Exhibition Info (E06)	1..* - *

Table B.4: Image Info's relation with other information entities.

which doesn't require any additional information.

Attribute	Description
Filename	Image's filename.
Extension	Image's extension.
Path	Path to the image stored in disk.
Mime Type	The image's type (JPEG, PNG, BMP, ...)
Height	The height of the image.
Width	The width of the image.

Table B.5: Image Data's attributes and their descriptions.

CBIR Info (E04) keeps all the data necessary to perform content-based queries, in our case the histogram data and the different subspaces for the hierarchical linear subspace method. Because we separate the search data from the image entity we can easily maintain the search application and even introduce new search methods with minimal impact to the information architecture.

Name of relation	Source	Target	Multiplicity
Preview Image	Image Data (E03)	Image Info (E02)	1 - 1
Thumbnail Image	Image Data (E03)	Image Info (E02)	1 - 1
Original Image	Image Data (E03)	Image Info (E02)	1 - 1

Table B.6: Image Data's relation with other information entities.

Attribute	Description
Histogram	Histogram binary data that is stored in database.

Table B.7: CBIR Info's attributes and their descriptions.

Name of relation	Source	Target	Multiplicity
Search Information	CBIR Info (E04)	Image Info (E02)	1 - 1
Subspace Data	CBIR Info (E04)	Subspace Data (E05)	1 - *

Table B.8: CBIR Info's relation with other information entities.

Subspace Data (E05) holds the different subspace needed by the hierarchical linear subspace method, i.e. the 3-band RGB information for the different subspaces and respective constant to approximate the subspace to the original space. The actual subspace data can either be kept on the database or on disk.

Attribute	Notes
Resolution	Resolution of the subspace.
Constant	Constant used to approximate to the original space.
RGB Data	RGB data when kept in database, for most cases the data is stored in disk.
Filename	Name of the file with the subspace data stored in disk.
Path	Path to the file where the subspace data is stored.

Table B.9: Subspace Data's attributes and their descriptions.

Name of relation	Source	Target	Multiplicity
Subspace Data	Subspace Data (E05)	CBIR Info (E04)	1..* - 1

Table B.10: Subspace Data's relation with other information entities.

Exhibition Info (E06) represents an exhibition created by the user that is used to display part of his portfolio of images. Each exhibition can hold several images and has its own state, active or inactive. Only the active exhibitions are public and can be browsed by any user, consequently the inactive exhibitions only allow management operations until it returns to the active state.

Attribute	Description
Title	Title of the exhibition.
Description	Description of the exhibition.
Keywords	Set of keywords that accurate describe the exhibition.
Active	Whether the exhibition is still active and open for everyone to visit.

Table B.11: Exhibition Info's attributes and their descriptions.

Name of relation	Source	Target	Multiplicity
Exhibits Own	Exhibition Info (E06)	User Info (E01)	* - 1
Artwork In Exhibition	Exhibition Info (E06)	Image Info (E02)	* - 1..*

Table B.12: Exhibition Info's relation with other information entities.

C

Subspaces Test

	256x256	x	x	x	x	x	x	x	x	x	x	x	x	x
	128x128	x	x	x	x	x	x			x	x	x	x	x
	64x64	x	x	x	x	x		x	x	x	x	x		x
	32x32	x	x	x	x		x	x		x	x	x	x	
	16x16	x	x	x		x	x	x	x		x			x
	8x8	x	x		x	x	x	x	x			x	x	x
	4x4	x		x	x	x	x	x	x	x				
49 715.15	50	(82.61)	2.11	2.17	5.59	10.84	16.98	29.29	40.13	20.92	14.92	7.70	24.68	13.01
51 242.28	100	(127.6)	1.87	2.02	5.82	12.48	21.33	40.30	52.78	21.22	13.83	7.69	29.02	14.49
52 241.43	150	(166.03)	1.72	1.88	5.87	13.41	24.21	48.12	61.53	21.19	13.08	7.59	31.80	15.29
53 014.29	200	(201.95)	1.60	1.76	5.83	14.05	26.20	54.51	68.55	21.02	12.50	7.44	33.63	15.81
55 840.67	500	(384.28)	1.19	1.23	5.15	15.22	32.39	75.45	90.67	19.47	10.30	6.34	38.73	16.45
57 303.67	750	(514.88)	0.98	0.90	4.51	14.79	34.21	83.41	98.20	18.13	9.14	5.49	39.70	15.69
58 434.44	1 000	(634.84)	0.83	0.62	3.88	13.98	34.32	87.98	101.96	16.85	8.26	4.71	39.03	14.60
60 198.83	1 500	(857.20)	0.60	0.17	2.70	11.81	31.74	89.26	101.07	14.54	6.89	3.30	35.04	11.98
63 781.03	3 000	(1443.86)	0.18	**-0.79**	**-0.32**	4.13	14.54	59.72	63.85	8.94	4.26	**-0.14**	14.40	3.35
67 056.12	5 000	(2119.72)	**-0.14**	**-1.65**	**-3.52**	**-6.01**	**-14.14**	**-19.32**	**-25.33**	3.25	2.11	**-3.66**	**-17.80**	**-7.67**
73 047.76	10 000	(3550.72)	**-0.60**	**-3.02**	**-9.42**	**-28.51**	**-92.33**	**-277.61**	**-306.12**	**-6.77**	**-1.08**	**-10.01**	**-102.35**	**-31.53**
epsilon	**result size**													

Table C.1: Each column represents, in millions, the difference in the number of comparisons needed when using all the available subspace against using the subspaces marked with an "x" for the respective ε-value (a negative number represents a saving while a positve shows an excess in number of comparisons).

D

Sample Queries

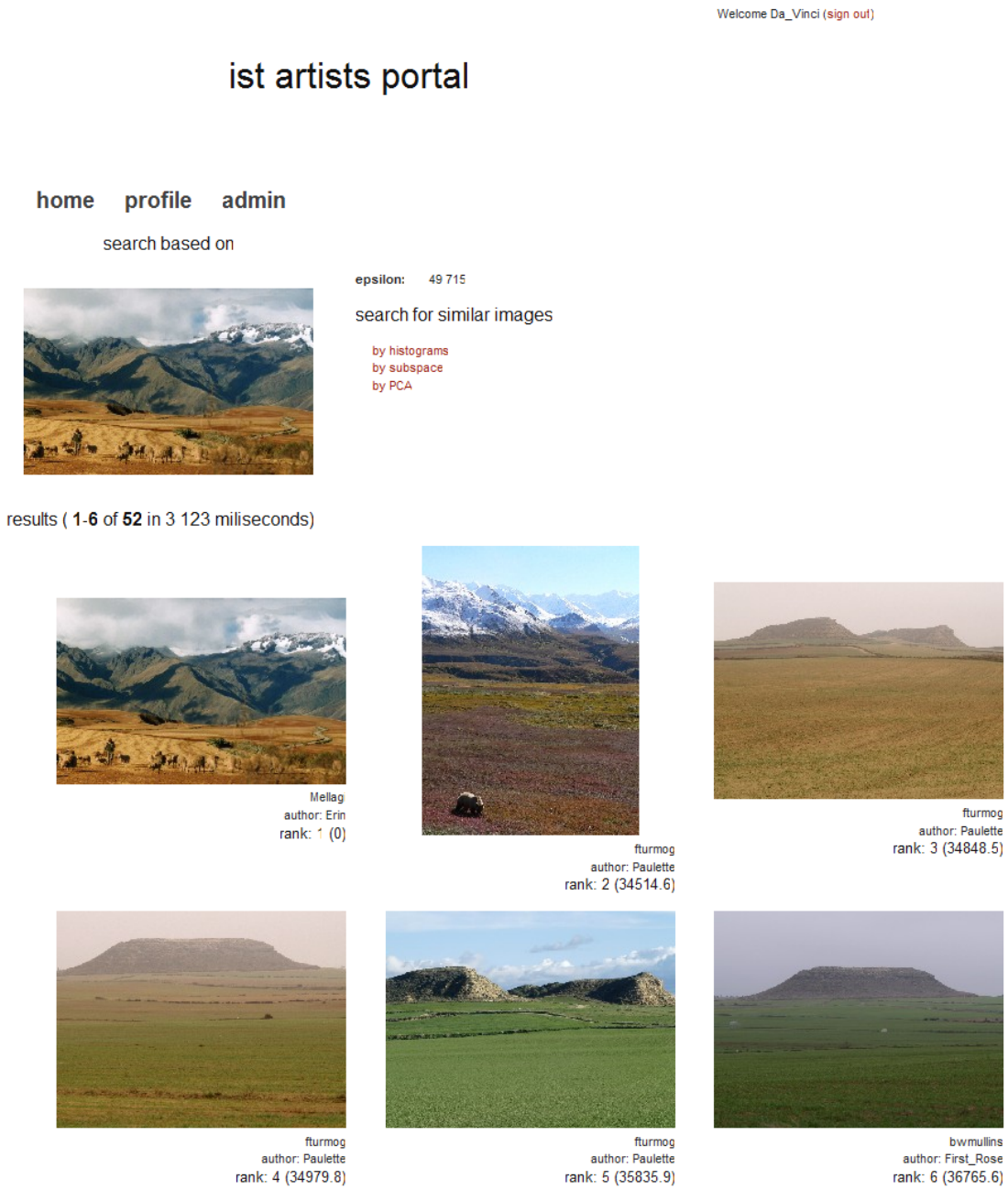


Figure D.1: Sample query 1.

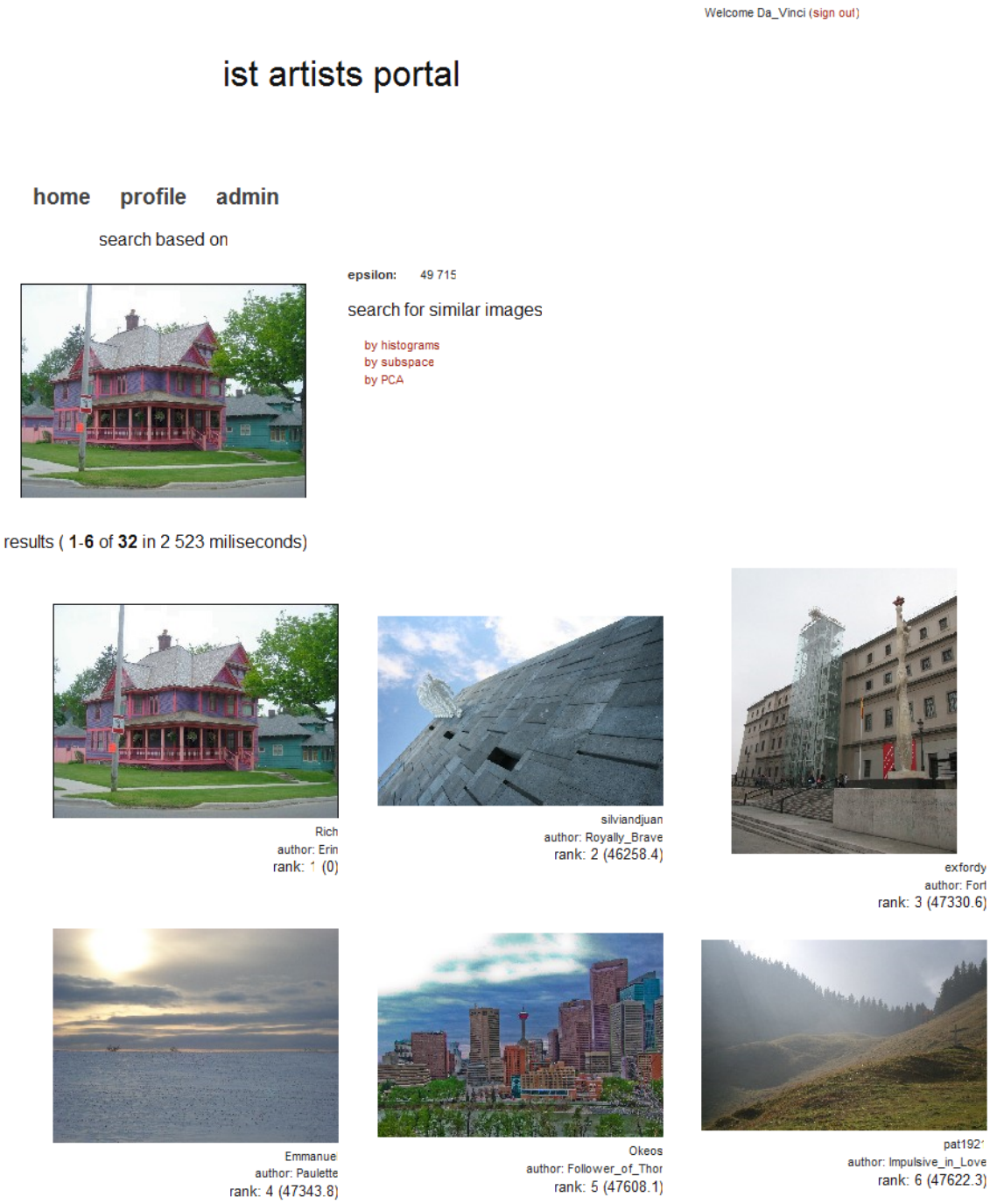


Figure D.2: Sample query 2.