FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

# Search and navigation for photo collections

**Pedro Miguel Correia Teixeira**

Master in Informatics and Computing Engineering

Supervisor: Jorge Alves da Silva (PhD)

Co-Supervisor: Luís Filipe Pinto de Almeida Teixeira (PhD)

July, 2010

# Search and navigation for photo collections

**Pedro Miguel Correia Teixeira**

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: João António Correia Lopes (PhD)
External Examiner: António José Ribeiro Neves (PhD)
Supervisor: Jorge Alves da Silva (PhD)

---

$31^{st}$ July, 2010

# Abstract

Images play a more and more important role in sharing, expressing and exchanging information in our daily lives. Accompanying this revolution, mobile hand-held devices with embedded digital camera are also undergoing considerable progress. A typical user can take hundreds of photos, save and stack them in a hard disk, remaining unlabelled and improperly organized.

There is a need to develop new ways of photo searching in a personal collection and give users a better experience when browsing them. In this document, the key components of an image retrieval system for personal image collections are presented and various critical aspects of their design are studied.

A content-based method of retrieving images using a computational model based on Wavelets has been implemented and evaluated. An image retrieval interface has been implemented, incorporating the ability to execute query-by-example in collaboration with keyword information.

In order to simplify image navigation, an unsupervised image clustering framework has been implemented through an adaptation of the k-means algorithm. The clustering method is based on hierarchical image grouping using content-based and metadata features. Experimental results demonstrate the performance of the proposed clustering methods on a real image database. Also, a new visualization and interaction method of image browsing is proposed for mobile devices. This technique is based on the previously calculated image hierarchical structure and is aimed for an optimal ratio between displayed content and screen area usage.

# Resumo

As imagens desempenham um papel cada vez mais importante na partilha, expressão e troca de informação no nosso dia-a-dia. Seguindo essa revolução, os dispositivos móveis com câmara integrada também são alvo de progressos consideráveis. Qualquer utilizador pode tirar centenas de fotografias, salvaguardá-las e acumulá-las num disco rígido, deixando-as sem legenda e desorganizadas.

Existe actualmente a necessidade de desenvolver novas formas de pesquisa de fotografias numa colecção pessoal para facilitar a navegação dos utilizadores. Este documento apresenta os principais componentes de um sistema de recuperação de imagens para as colecções de imagens pessoais, bem como vários aspectos críticos da sua concepção. Implementou-se e avaliou-se um método baseado no conteúdo de recuperação de imagens utilizando um modelo computacional baseado em Wavelets. Foi também implementada uma interface de recuperação de imagens, com capacidade de executar *query-by-example* associada a uma palavra-chave.

Para simplificar a navegação de imagens, desenvolveu-se uma *unsupervised image clustering framework* através da adaptação do algoritmo *K-means*. O método de *clustering* baseia-se no agrupamento de imagens hierárquicas usando características baseadas no conteúdo e em metadados. Os resultados experimentais demonstram o desempenho dos métodos de *clustering* propostos numa base de dados de imagens reais. Além disso, propõe-se um novo método de visualização e interacção de pesquisa de imagens para os dispositivos móveis. Esta técnica baseia-se na estrutura hierárquica de imagens previamente calculada e visa o mais eficiente rácio entre o conteúdo apresentado e a utilização da área do ecrã.

# Acknowledgements

I would like to show my gratitude to Dr. Luís Filipe Pinto de Almeida Teixeira whose advice and comprehensive support were essential to the successful completion of this research. The feedback provided by Dr. Jorge Alves da Silva was critical to the success of this work. I am greatly appreciative of the insights he provided, all of which enhanced my understanding and generated countless new ideas. Without the company and collaborative efforts of my colleagues at Fraunhofer Portugal Research Institute the creation of this dissertation would have been a far less enjoyable experience. In particular, I would like to thank Dr. Filipe Sousa and Dr. Paula Silva for the brainstorming talks and ideas regarding this research. A special thanks to my dear friend João Mendes who provided endless support. My parents and brother unconditional love was an unwavering source of strength throughout the writing of this dissertation. Last but not least, my girlfriend Christine Costa made me a happy person and gave me the extra strength, motivation and love necessary to get things done.

Pedro Miguel Correia Teixeira

# Contents

**Abstract** **i**

**1 Introduction** **1**
1.1 Context . . . . . 1
1.2 Common usage of real world image retrieval systems . . . . . 2
1.3 The Problem and motivation . . . . . 4
1.4 Document Structure . . . . . 5

**2 State of the art** **7**
2.1 Image retrieval in the real world . . . . . 7
2.1.1 System Design . . . . . 8
2.1.2 User Intent . . . . . 9
2.1.3 Data Scope . . . . . 10
2.1.4 Query Modalities . . . . . 10
2.1.5 Visualization . . . . . 12
2.2 Core Problem Techniques . . . . . 12
2.2.1 Extraction of Visual Signature . . . . . 13
2.2.2 Image Segmentation . . . . . 13
2.2.3 Colour Histograms and Textures . . . . . 13
2.2.4 Shapes . . . . . 14
2.2.5 Image Similarity . . . . . 15
2.3 Browsing models for CBIR . . . . . 16
2.3.1 In defense of browsing . . . . . 16
2.3.2 Challenges of browsing . . . . . 17
2.4 Summary . . . . . 18

**3 Image representation and distance metrics** **19**
3.1 Introduction and overview . . . . . 19
3.2 Wavelets and image decomposition . . . . . 20
3.2.1 One-dimensional Haar wavelet decomposition and compression . 20
3.2.2 Image (two-dimensional) Haar wavelet compression and edge extraction . . . . . 25
3.2.3 The image compression algorithm . . . . . 27
3.2.4 Image distance metric and scoring algorithm . . . . . 29
3.3 Test application . . . . . 31
3.4 Summary . . . . . 35

**4 Unsupervised Learning** **37**
4.1 Introduction and overview . . . . . 37
4.2 Image clustering using an adaptation of k-means algorithm . . . . . 38
4.3 Experimental results of k-means adaptation algorithm . . . . . 41
4.4 Experimental results of hierarchical k-means adaptation algorithm . . . . 46
4.5 Summary . . . . . 47

**5 Visualizing image collections on Mobile Devices** **49**
5.1 Introduction . . . . . 49
5.2 Mobile Visualization Design and User Interaction . . . . . 50
5.3 Hierarchical Image Structure Visualization . . . . . 51
5.3.1 Non-Distortion based Visualizations . . . . . 51
5.3.2 Generalized Distortion-based Visualizations . . . . . 52
5.3.3 Hierarchical Tree Visualization . . . . . 53
5.4 Methodology . . . . . 53
5.5 Interface Design . . . . . 53
5.5.1 Proposed "Cube Tree Visualization" layout for Mobile Devices . . 53
5.5.2 Application prototype using proposed "Cube Tree Visualization layout" . . . . . 55
5.6 Summary . . . . . 57

**6 Conclusions and future work** **59**
6.1 Image Distance Metric Improvements . . . . . 60
6.1.1 Possible content-based scoring improvement . . . . . 60
6.1.2 Metadata related improvements . . . . . 60
6.2 Clustering Improvements . . . . . 61
6.3 Future Analysis and Evaluation of Visualization Process . . . . . 61
6.3.1 Information Visualization Process . . . . . 61
6.3.2 Criteria . . . . . 61
6.3.3 Techniques . . . . . 62
6.3.4 Metrics . . . . . 62
6.3.5 Process . . . . . 63

**References** **65**

# List of Figures


1.1 Google Image search — Find similar images with car example. . . . . 2
1.2 Gazopa — Find by example, uploaded car image. . . . . 2
1.3 Google Goggles usage on finding information on web for a given image . 3
1.4 Editing image metadata using Adobe Photoshop Lightroom. . . . . 3
1.5 Proposed Architecture: Image retrieval system running on a personal image collection and serving mobile device clients via WiFi. . . . . 4
1.6 Proposed System Layer Architecture . . . . . 5

2.1 Visualizing image retrieval from a user perspective [RD08]. . . . . 8
2.2 Visualizing image retrieval from a system perspective [RD08]. . . . . 8
2.3 Example of an image colour histogram with the representation of the color distribution in an image. . . . . 14
2.4 Example of an image texture segmentation . . . . . 14
2.5 Example of an image shape segmentation . . . . . 15
2.6 Different types of image similarity measures, their mathematical formulations and techniques for computing them. [RD08] . . . . . 15
2.7 Google Swirl Demo — clusters centroids with "Eiffel" search. . . . . 17
2.8 Google Swirl Demo — Step navigation after selecting first Eiffel image in Figure 2.7. . . . . 18

3.1 Haar wavelet scaling function $\phi(x)$ on left and Haar wavelet function $\psi(x)$ on right. . . . . 23
3.2 The box basis for $V^2$ in the interval [0,1). . . . . 24
3.3 A sequence of decreasing-resolution approximations to a function (left), along with the detail coefficients required to recapture the finest approximation (right). . . . . 25
3.4 Standard decomposition (left), non-standard decomposition (right) [EJS95]. 25
3.5 Original image (a) represented using 21% of its coefficients with 5% error (b), 4% of its coefficients with 10% error (c) and (d) 1% of coefficients with 15% error. [EJS95] . . . . . 26
3.6 Original image (left), wavelet coefficients display using wavelet toolbox in MATLAB. [kn:b] . . . . . 26
3.7 Original image (left), Un-thresholded wavelet coefficients representation (center), thresholded wavelet coefficients representation (right). . . . . 27
3.8 Test application architecture overview. . . . . 31
3.9 Test application home page. . . . . 32
3.10 Find images by text. . . . . 32

3.11 Find similar images (cow original image query at top left corner). . . . . 33
3.12 Find similar images (mountain original image query at the top left corner). 33
3.13 Butterfly image and variances sorted by content similarity. . . . . 34

4.1 Randomly placed 5 centroids on a data space [kn:a]. . . . . 39
4.2 Points mapped to centroids through the use of Euclidian distance objective function [kn:a]. . . . . 39
4.3 New centroid calculation for each cluster [kn:a]. . . . . 39
4.4 Final result of clustering process after some repeating iterations [kn:a]. . 39
4.5 Image database categories 1-8. . . . . 41
4.6 Image database categories 9-17. . . . . 42
4.7 Overview of database clustering result with $k = 10$. . . . . 43
4.8 Focus cluster containing category 10. . . . . 44
4.9 Focus cluster containing categories 7, 9 and 17. . . . . 44
4.10 Focus cluster containing category 11. . . . . 44
4.11 Focus nested clusters in the tree structure containing categories 2, 3 and 16. 46
4.12 Focus nested clusters in the tree structure containing categories 11, 14 and others. . . . . 46
4.13 Focus nested clusters in the tree structure containing categories 6 and others. 46

5.1 RadialMobile screenshot. . . . . 52
5.2 Perspective Wall, ConeTree and TreeMap. . . . . 52
5.3 Images represented as colors in a hierarchical structure scheme. . . . . 54
5.4 Image top-down navigation in the hierarchical structure scheme. . . . . 54
5.5 Image navigation using cube to perform top-down, bottom-up and side navigations. . . . . 55
5.6 Screenshots of the application prototype: simple layout and cube vertical rotation . . . . . 56
5.7 Screenshots of the application prototype: simple layout and cube horizontal rotation . . . . . 56

# List of Tables

3.1 Image decomposition - First step. . . . . 21
3.2 Image decomposition - Final step. . . . . 21
3.3 Different weights for “painted like” approximation images (left) and for “scanned like” approximations (right). . . . . 29
3.4 Original image and variances similarity values and luminance difference value. . . . . 34

4.1 Number of images in each category. . . . . 42
4.2 Image category distribution through clusters. . . . . 45

LIST OF TABLES

# Chapter 1

# Introduction

The growth of the Internet, the falling price of storage devices, the availability of image capturing devices such as digital cameras, image scanners, and the increasing pool of available computing power, made it necessary and possible to manipulate very large repositories of digital information. Due to its convenience and affordability, a typical user can take hundreds of photos, save and stack them in a hard drive remaining unlabelled and improperly organized. Efficient image searching and browsing tools are required by users from diverse domains, such as, fashion, crime prevention, medicine, publishing, etc. For this purpose, many image retrieval systems have been developed, and current state-of-the-art systems hold enough maturity to be useful for real-world applications.

## 1.1 Context

There are two main real-world image retrieval system frameworks: text-based and content-based image search and browsing. In the text-based approach, images are manually labelled with high-level features (concepts), such as, keywords, text descriptors, to interpret images and measure their similarities which are then used to perform image retrieval by a database. Since a considerable amount of human work is needed, there is the disadvantage of possible inaccuracies due to the subjectivity of human perception. To overcome the disadvantages of text-based approach, content-based image retrieval (CBIR) approach was introduced. As we see it today, CBIR is any technology that helps organizing image archives by their visual low-level features (colour, texture, shape, spatial layout, etc), ranging from image similarity functions to a robust image annotation engine. The fundamental difference between the two approaches is that human interaction is an indispensable part of the text-based one, and in general there is no direct link between high-level concepts and low-level features. Due to these limitations, in some sense, it may be easier to find such a picture by looking through an organized structure of image collections

and making unconscious matches with the one drawn by imagination, than to use other descriptions that fail to capture the true meaning of the given search.

## 1.2 Common usage of real world image retrieval systems

Not many image retrieval systems are deployed for public usage, Google Images, Bing or Yahoo! Images for example primarily use the surrounding meta-data associated with images but they have also capabilities to find similar images for a given selected image as illustrated in Figure 1.1.

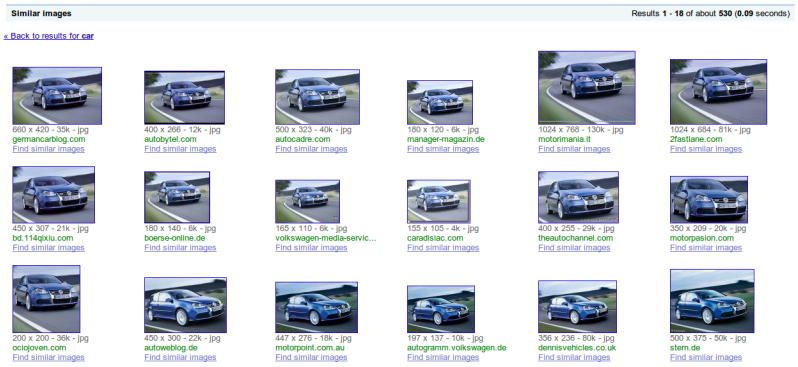



Figure 1.1: Google Image search — Find similar images with car example.

Recently, GazoPa search engine was released. In addition to images found using keyword searches, GazoPa enables the use of users'own photos and drawings, and images found on the web, as search keys to locate similar images. An example is demonstrated in Figure. 1.2

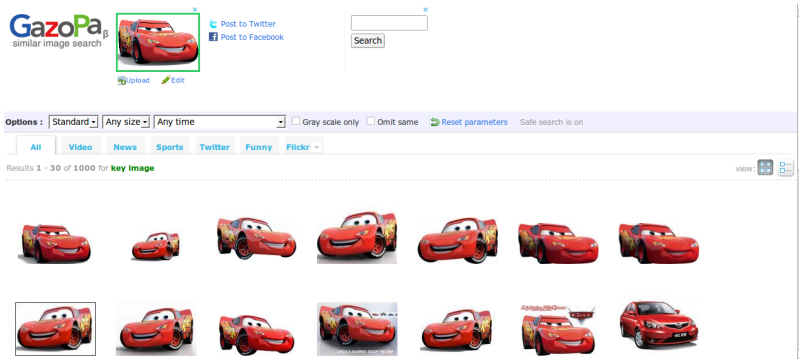



Figure 1.2: Gazopa — Find by example, uploaded car image.

On mobile devices environments there are also image search systems, for example, Google Goggles, an Android downloadable image recognition application created by Google Inc. It is currently available in Google Labs as a beta version and it is used for searches based on pictures taken by hand-held devices. For example taking a picture of a

famous landmark would search for information about it, or taking a picture of a products bar code would search for information on the product, Figure 1.3.

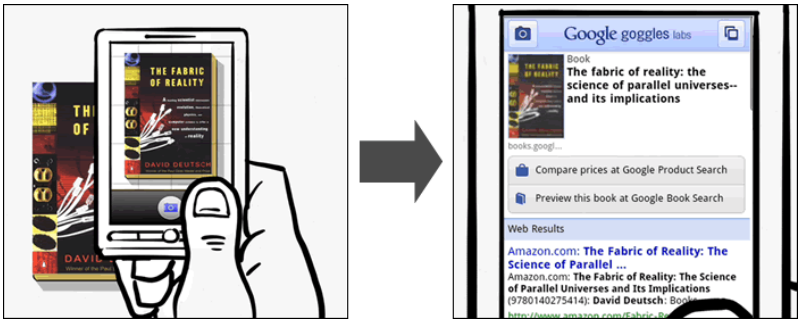



Figure 1.3: Google Goggles usage on finding information on web for a given image

One thing in common on the examples above is the use of the world wide web as image domain, where images generally possess some metadata associated. The web has a semi-structured, non-homogeneous and massive volume of images usually stored in large disk arrays that has a key crawler component which regularly updates its local database. On personal image collections, images are saved on their owner local storage media, are organized in folders, and their related metadata is typically inserted by hand. An image metadata usually contains information about the contents, copyright status, origin and history of the image, as shown on Figure 1.4. Some information such as creation date, and camera settings (Exif data on the case of JPEG image format) is automatically assigned by digital cameras.

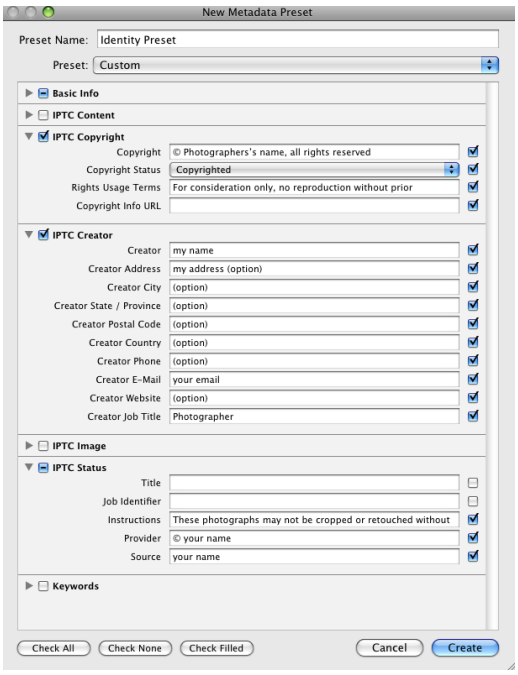



Figure 1.4: Editing image metadata using Adobe Photoshop Lightroom.

## 1.3 The Problem and motivation

Now people can easily capture and share personal photos on mobile devices anywhere and at any time. Personal image collections contain large volumes of unstructured homogeneous photos pertaining to specific topics stored on multiple disks. The usual way images are sorted is by their metadata (more commonly the creation-date and name) which can be used as filters in image research. Besides labeling practices, the most common used method of organizing image collections is to manually associate them with specific albums. However, it can be a tedious task and not always appropriate for further image search.

The aim of this project is to develop solutions that can obviate the manual organization of image collections through an automatic organization process, providing to the user a better experience when browsing a large image collection. An extension of image metadata filtering is provided by image contents. The researching plan is as follows:

1. To build an image computational model for content-based comparisons.
2. To merge image metadata and content-based features to produce image distances.
3. To build image hierarchical groups through unsupervised clustering methods.
4. To develop a visualization and interaction method of image browsing on mobile devices.

The physical architecture model which agregates the wanted solutions is shown on figure 1.5. It consists of a current domestic image share network, using the computer power to perform heavy tasks such as image clustering and extend image browsing and search features to mobile devices. Taking into account the advantages and restrictions of mobile devices environments, it will be studied how state of the art mobile image visualization techniques can be used regarding the developed image organization principle.

Figure 1.5: Proposed Architecture: Image retrieval system running on a personal image collection and serving mobile device clients via WiFi.

The system layer architecture model is shown on Figure 1.6. It consists on two major components, a *daemon* running the image similarity and clustering engines and an *intranetservice* providing a bridge between the user and system using HTTP protocol for communication.

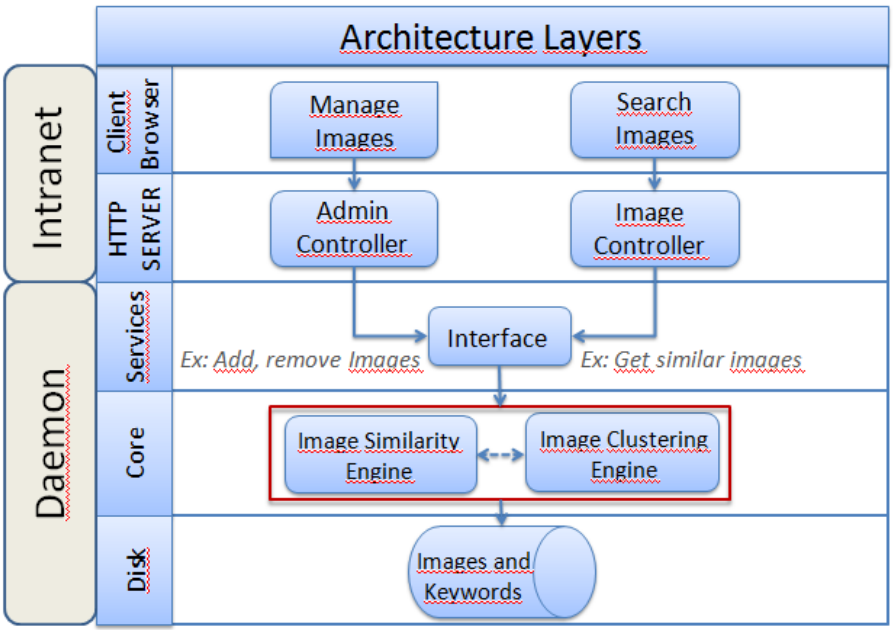



Figure 1.6: Proposed System Layer Architecture

## 1.4 Document Structure

For building a useful system in the real-world, a number of issues need to be taken into account as well as various critical aspects of its design. Chapter 2 is devoted to the analysis of key techniques of core problems and to an overview of the different browsing models for CBIR, that have been explored over the last decade and can be used to assist photo's automatic cataloguing. Chapter 3 focuses on building internal representations of images in order to produce distance metrics regarding image content-based features, used to improve comparisons with query by example facilities. Chapter 4 focuses on building the internal image database structure used to help the user task in the image browsing process. Chapter 5 presents display methods on mobile devices regarding the proposed image database structure organization and chapter 6 concludes with possible future research paths and strategic points that can be improved.

# Chapter 2

# State of the art

## 2.1 Image retrieval in the real world

Through extensive experiments on CBIR systems, image content often fails to describe high-level concepts, and human has traditionally outperformed machines in this kind of tasks. One reason that causes this distinction is that images are a mere replica of what a human has seen since birth, and descriptions are relatively elusive. As described in [AWMSJ00] the gaps that motivate most of the related problems are:

- *The sensory gap* — the gap between the object in the world and the information in a (computational) description derived from a recording of that scene.
- *The semantic gap* — the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data has for a user in a given situation.

The former makes recognition from image content challenging due to recording limitations, the latter represents the difficulty of capturing user's interpretations. Another challenge on CBIR systems is the domain of images. As described in [AWMSJ00], the image domain is classified as *narrow* and *broad*, and to date this remains an important distinction as it can be used for system design purposes. Narrow domains usually have limited variability and better-defined visual characteristics (e.g., cars related images), which makes content-based image search easier to formulate. On the other hand, broad domains have high variability and unpredictability for the same underlying semantic concepts (e.g., Web images), which makes the problem of systems design more challenging.

### 2.1.1 System Design

Designing an omnipotent real-world image search engine capable of serving all categories of users requires understanding and characterizing user-system interaction and image search from both the user and system points of view. The dual characterization and attempt of representing all known possibilities of interaction and search are shown in Figures 2.1 and 2.2.

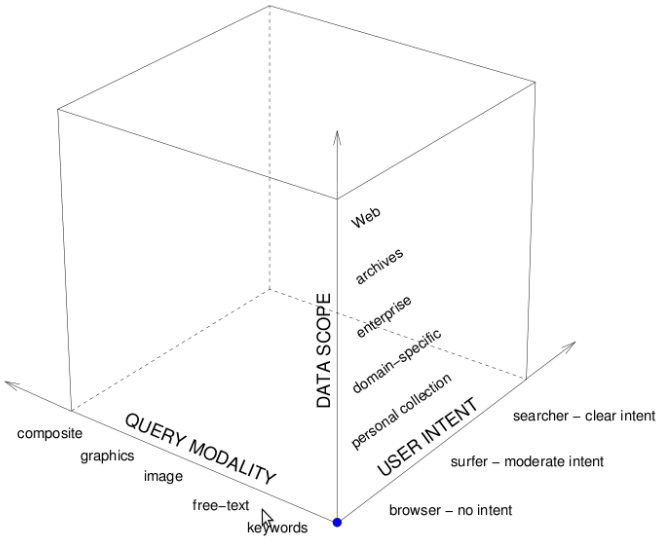



Figure 2.1: Visualizing image retrieval from a user perspective [RD08].

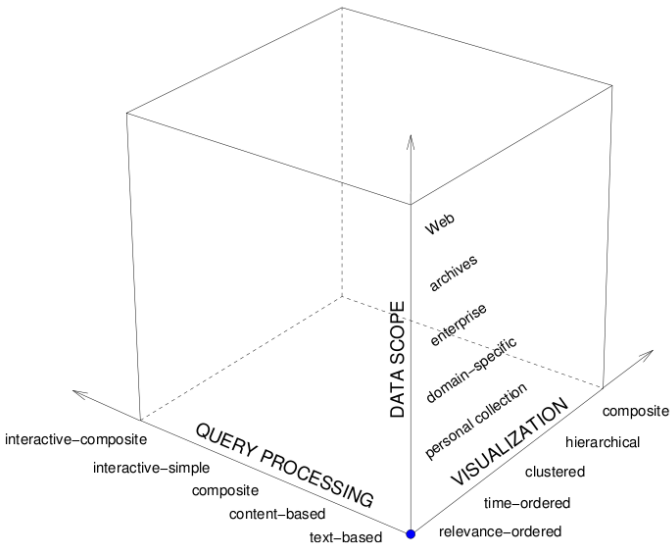



Figure 2.2: Visualizing image retrieval from a system perspective [RD08].

From a *user perspective* (Figure 2.1), image search involves considering and taking decisions on the following fonts:

1. Clarity of the user about what he/she wants.
2. Where does the user wants to search.
3. In what form does the user has his query.

An alternative view, from an image retrieval *system perspective* (Figure 2.2), a search translates to making arrangements as per the following factors:

1. How does the user wish the results to be presented.
2. Where does the user desire to search.
3. What is the nature of user input/interaction.

These factors, with their respective possibilities form the axis of the Figures 2.1 and 2.2. Image instances can be considered as isolated points if they have only one query modality, data scope and user intent, or point clouds if they have multiple query, modality and user intent features. Therefore, an image search can be perceived as a trajectory, starting on a sample image and ending up on the target image.

### 2.1.2 User Intent

While searching for images, user intent on what they desire may vary. The clarity of intent plays a key role in users expectations from a given search system, and it can also act as a guideline for system design. User intent can be characterized as follows [RD08] :

- *Browser* — A user browsing for pictures with no clear end-goal. A browser's session would consist of a series of unrelated searches. A typical browser would jump across multiple topics during the course of a search session.
- *Surfer* — A user surfing with a moderate clarity of an end-goal. A surfer's actions may be somewhat exploratory in the beginning with a difference that subsequent searches are expected to increase the surfers clarity of what she wants from the system.
- *Searcher* — A user who is very clear about what is searching for in the system. A searcher's session would typically be short with coherent searches leading to an end-result.

The importance of building human-centred systems has been expressed lately, and in order to gain wide acceptance, image retrieval systems need to acquire a human-centred perspective as well.

### 2.1.3 Data Scope

The nature of data scope also plays an important role in the complexity of image search systems design. The diversity of user-base and expected user traffic for a search system also influence the design. Search data is classified into the following categories [RD08]:

- *Personal collection* — A largely homogeneous collection generally small in size, accessible primarily to its owner, and usually stored on a local storage media.
- *Domain-specific collection* — A homogeneous collection providing access to controlled users with very specific objectives. The collection may be large and be hosted on distributed storage, depending upon the domain. Examples of such a collection are biomedical and satellite image databases.
- *Enterprise collection* — A heterogeneous collection of pictures accessible to users within an organization's Intranet. Pictures may be stored in many different locations. Access may be uniform or non-uniform depending upon the Intranet design.
- *Archives* — These are usually of historical interest and contain large volumes of structured or semi-structured homogeneous data pertaining to specific topics. May be accessible to most people on the internet, with some control on usage. Data is usually stored in multiple disks or large disk arrays. This is our case study image domain.
- *Web* — World Wide Web pictures are accessible to practically everyone with an Internet connection. Current WWW image search engines such as Google images and Yahoo! images have a key crawler component which regularly updates their local database to reflect on the dynamic nature of the Web. The image collection is semi-structured, non-homogeneous, and massive in volume and is usually stored in large disk arrays.

### 2.1.4 Query Modalities

Another important parameter is the level of complexity of the queries supported by the system. Below, the various querying modalities, characteristics, and the required system support are described [RD08]:

- *Keywords* — User poses a simple query in the form of N words. This is currently the most popular way to search images, e.g., the Google and Yahoo! image search engines.
- *Free-text* — User frames a complex phrase, a sentence, a question, or a story about what she desires from the system.

- *Image* — The user wishes to search for an image similar to a query image. Using an example image is perhaps the most representative way of querying a CBIR system in the absence of reliable meta-data.

- *Graphics* — A hand-drawn or computer-generated picture or graphics could be presented as query.

- *Composite* — These are methods that involve using one or more of the above modalities for querying a system. This also covers interactive querying such as in relevance feedback systems.

The query modalities require different processing methods, and become more complex when visual queries interactions are involved and are characterized from a system perspective:

- *Text-based* — Text based query processing usually boils down to performing one or more simple keyword based searches and retrieving matching pictures. Processing a free-text could involve parsing, processing, and understanding the query as a whole. Some form of natural language processing may also be involved.

- *Content-based* — Content based query processing lies at the heart of all CBIR systems. Processing of a query (image or graphics) involves extraction of visual features and/or segmentation and search in the visual feature space for similar images. An appropriate feature representation and a similarity measure to rank pictures, given a query, are essential here. These will be discussed in detail in Chapter 3.

- *Composite* — Composite processing may involve both content and text-based processing in varying proportions. An example of a system which supports such processing is the story picturing engine [DJ06].

- *Interactive-simple* — User interaction using a single modality needs to be supported by a system. An example is a relevance feedback based image retrieval system.

- *Interactive-composite* — The user may interact using more than one modality (e.g., text and images). This is perhaps the most advanced form of query processing that is required to be performed by an image retrieval system.

Nevertheless, there are issues relate to the presence of reliable meta-data with pictures as a prerequisite for supporting text-based queries, and images rarely come with reliable human tags.

### 2.1.5 Visualization

As mobile devices became widespread, mobile users have limited querying capabilities due to scrolling and typing constraints. Hence it becomes necessary to design intelligent feedback methods to cater to users with small displays. Presentation of search results is perhaps one of the most important factors in the acceptance and popularity of an image retrieval system. The visualization schemes for image search are characterized as follows [RD08]:

- *Relevance-ordered* — The most popular way to present search results, as adopted by Google and Yahoo! for their image search engine. Results are ordered by some numeric measure of relevance to the query.
- *Time-ordered* — Pictures are shown in a chronological ordering rather than by relevance.
- *Clustered* — Clustering of images by their meta-data or visual content has been an active research topic for several years. Clustering of search results, besides being an intuitive and desirable form of presentation, has also been used to improve retrieval performance.
- *Hierarchical* — If meta-data or content-based features can be associated with images in a tree order, it can be a very useful aid in visualization. Hierarchical visualization of search results is desirable for archives especially for educational purposes.
- *Composite* — Combining one or more of the above forms of visualization schemes especially for personalized systems. Hierarchical clustering and visualization of concept graphs are examples of composite visualizations. This will be discussed in detail in Chapter 4.

## 2.2 Core Problem Techniques

To implement image search features in the system it is necessary to understand and interpret visual content for indexing and retrieval. In spite of the effort of the last decade in image retrieval research, there is not yet a universally acceptable way of replacing human vision in interpreting images. Hence, it is not surprising to see continued effort in this direction, building on prior work and exploring novel directions. As described in [AWMSJ00], image search technology boils down to two intrinsic problems:

1. How to mathematically describe an image.
2. How to assess the similarity between a pair of images based on their abstracted descriptions.

These issues arise because of the mathematical representation of an image (for retrieval purposes), referred as its *signature*, corresponds poorly to human visual response, let alone semantic aspects.

### 2.2.1 Extraction of Visual Signature

Most of the image search systems perform feature extraction as a pre-processing step. Once obtained, visual features act as inputs to subsequent image analysis tasks such as similarity estimation, concept detection, or annotation. Below visual signature extraction methods are presented. Since it has been seen great interest in region-based signatures in the past decade, image segmentation will be presented first, because it is the essential first step.

### 2.2.2 Image Segmentation

Image segmentation refers to the process of partitioning a digital image into multiple segments (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse [SS01]. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.

The problem of image segmentation is mapped to a weighted graph partitioning problem where the vertex set of the graph is composed of image pixels and edge weights represent some perceptual similarity between pixel pairs. Achieving good segmentation is a major step toward image understanding. Some issues plaguing current techniques are computational complex.

### 2.2.3 Colour Histograms and Textures

In early years of CBIR, the exploration of colour features was active, with emphasis on exploiting colour spaces that seem to coincide better with human vision than the basic RGB colour space. Nowadays, research on colour features has focused more on the summarization of colours in an image, that is, the construction of signatures out of colours, such as *color histograms* (representation of the distribution of colours, Figure 2.3), spatial colour descriptors and *textures*, Figure 2.4.

Textures features are intended to capture the granularity and repetitive patterns of surfaces within pictures. For instance, grass land, brick walls. These features, which define a spatial arrangement of texture constituents, help to single out the desired texture types, e.g. fine or coarse, close or loose, plain or twilled or ribbed textile fabrics. It is difficult to use human classifications as a basis for formal definitions of image textures, because there is no obvious ways of associating these features, easily perceived by human vision,

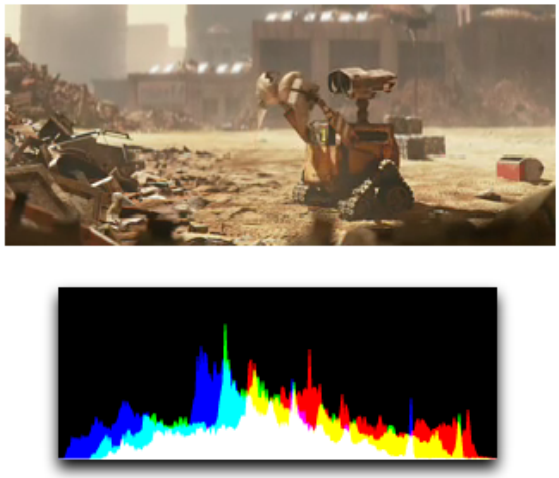


Figure 2.3: Example of an image colour histogram with the representation of the color distribution in an image.

with computational models that have the goal to describe the textures. Nonetheless, after several decades of research and development of texture analysis and synthesis, a variety of computational characteristics and properties for indexing and retrieving textures have been found.

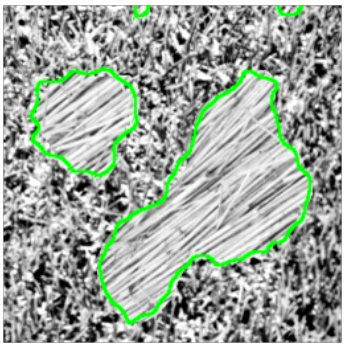


Figure 2.4: Example of an image texture segmentation

### 2.2.4 Shapes

Shape is a key attribute of segmented image regions, Figure 2.5, and its efficient and robust representation plays an important role in image retrieval. Shape does not refer to the shape of an image but to the shape of a particular region that is being sought out by first applying segmentation. A shape descriptor for similarity matching, referred to as *shape context* is proposed by [SBP02].

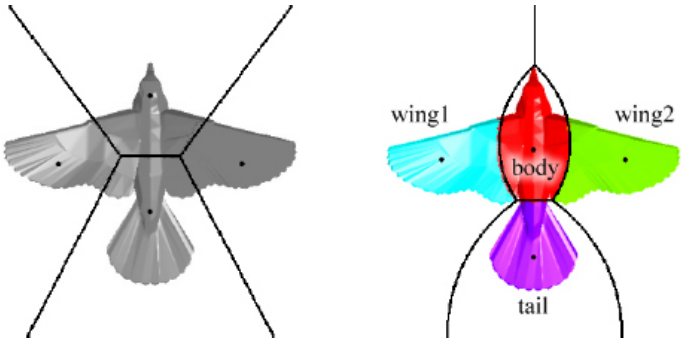



Figure 2.5: Example of an image shape segmentation

### 2.2.5 Image Similarity

Once a decision on the choice of image signatures is made, how to use them for accurate image retrieval is the next concern. There has been a large number of different importances given to the methods proposed in the recent years, summarized as follows [RD08]:

- *Agreement with semantics*
- *Robustness to noise (invariant to perturbations)*
- *Computational efficiency (ability to work real-time and in large-scale)*
- *Invariance to background*

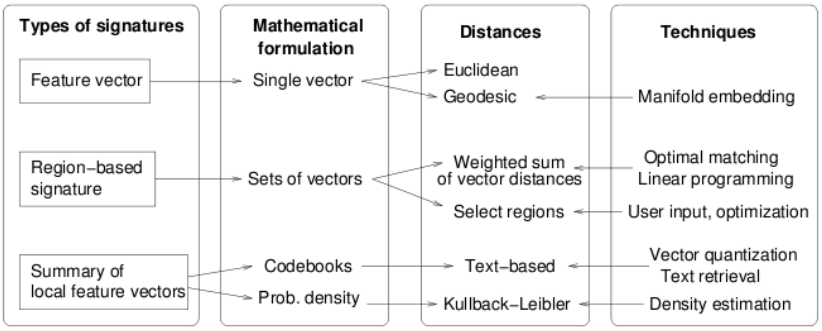



Figure 2.6: Different types of image similarity measures, their mathematical formulations and techniques for computing them. [RD08]

The different distance measures illustrated in Figure 2.6 have their own advantages and disadvantages. While simple methods, like single vector representation with Euclidian distance, lead to very efficient computation, there are often not effective enough, producing poor quality in the results to be useful. The use of quality distance measures,

like region-based signatures with weighted sum of vector distances lead on the other hand to a poor computation efficiency [RD08].

## 2.3 Browsing models for CBIR

Browsing is defined here as the exploration of spaces through a sequence of local decisions or navigational choices and provides an interesting alternative to systems requiring explicit query formulation, involving the measure of similarities of image features, but it has, by comparison, received only scant attention [For01].

The problem of estimating the relative significance of different features pertains to information retrieval in general and all these methods have in common that at some point users issue an explicit query, be it textual or pictorial. This division of roles between the human and the computer system as exemplified by many early CBIR systems seems warranted on the grounds that search is not only computationally expensive for large collections but also amenable to automation.

Unsupervised clustering techniques are a natural fit when handling large, unstructured image repositories such as the Web, but when one considers that humans are still far better at judging relevance, and can do so rapidly, the role of the user seems more important for restricted repositories. The introduction of relevance feedback into image retrieval has been an attempt to involve the user more actively and has turned the problem of learning feature weights into a supervised learning problem [Hee08]. Although the incorporation of relevance feedback techniques can result in substantial performance gains, such methods fail to address a number of important issues. Users may, for example, not have a well-defined information need in the first place and may simply wish to explore the image collection.

Most of the browsing models cast the collection into a structure that can be navigated interactively. Arguably one of the greatest difficulties of a browsing approach is to identify structures that are conducive to effective search in the sense that they support fast navigation, provide a meaningful neighbourhood for choosing a browsing path and allow users to position themselves in an area of interest.

### 2.3.1 In defense of browsing

The query in CBIR often takes the form of an example image. This query mode is inadequate when query images are not readily at hand — a *Mental query*. Indeed, users would perhaps need to access a collection first by some other means to identify suitable query images. However, images, whether in our mind or not, can be inordinately more expressive than words and to find what lies beyond verbal description, a visually guided search is likely to remain the more effective strategy.

*User's cognitive abilities* can be exploited as the human visual system is able to recognise patterns reliably and quickly. Given our present limitations in understanding and emulating cognitive vision, the most promising way to leverage the potential of computers is to combine their strengths with those of users and to achieve a synergy through interaction. Such synergy can be achieved through browsing as users are continuously required to make decisions based on the relevance of items in relation to their current information need [Hee08].

### 2.3.2 Challenges of browsing

The greatest challenge is to identify good organisation principles for structuring a collection. Intuitively, we should wish objects to be near each other and easily accessible from one another if they are similar. Images admit to a number of different representations in terms of visual features and not all features are equally useful to find, for a particular image, those that are similar. The question of how to weigh different features when constructing structures for browsing is far from trivial. Some classes of organizations are distinguished in [Hee08]:

- *Static hierarchical structures* — This structure is used in daily lives, such as the arrangement of books in a physical library, postal addresses and many more. They have been studied for many years as a possible remedy against the linear time complexity of exhaustive nearest neighbour searches [RK97]. The general idea here is to find nearest neighbours by descending a tree of hierarchically organised cluster centroids. A useful distinction is between normal clustering and fuzzy clustering models [RZ05]. In the former, an item is unambiguously assigned to only one cluster, while in the latter an item may belong to several clusters. Users, who at every step compare their internal query image with the cluster centroids at a particular level of the hierarchy, decide along which path to continue. Google Image Swirl is presented here as an example through Figures 2.7 and 2.8.

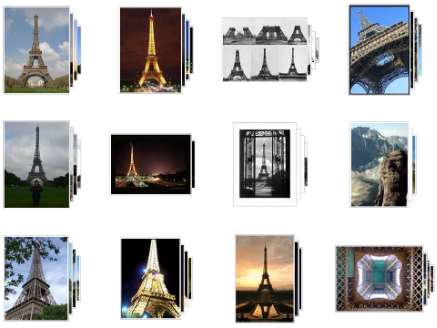


Figure 2.7: Google Swirl Demo — clusters centroids with "Eiffel" search.

Figure 2.8: Google Swirl Demo — Step navigation after selecting first Eiffel image in Figure 2.7.

- *Static networks* — In this type of structure the physical realisation of nodes is in the form of neurons or clusters of interconnected neurons. In these distributed structures the hierarchical nature of much data is implicit in the weights associated with pairs of connected nodes. Typically, networks are built on the basis of similarity data between images. There are different approaches that differ in the way edges are established between vertices [Cox95] [RK97].

## 2.4 Summary

In our case study, the user photo collection contains large volumes of unstructured semi-homogeneous photos pertaining to specific topics. They are archives possibly labeled and stored on multiple disks. Since the user may be very clear about what he/she wants, a very important step in the design of the system is the choice of image visual signatures and distance measures to develop a search by image-query feature. A constraint in this matter is the need of computational efficiency, having the ability to work real-time and in large-scale because mobile handsets are the systems primary client targets and they require quick results. Chapter 3 discusses with more detail the choice of the technique and developing process of this feature. Another user intent is to be a browser and surfer. With this in mind surges the need of structuring the image collection database in a manner thatf provides a guide for image exploration. The conceptual main idea is to automatically organize images into albuns or to provide a way of giving a few images as database representants and using user interactions to refine or filter the collection and therefore search for intended image targets. These conceptual ideas match the image clustering process and the use of hierarchical structure of image collection. More details on this structuring process can be found on Chapter 4. Chapter 5 explains how search features and image navigation process can be used in mobile handsets, possessing small screen displays.

# Chapter 3

# Image representation and distance metrics

## 3.1 Introduction and overview

This chapter focuses on building a database with internal representations of images and image distance metrics used for further content-based comparisons. The usual raw pixel representation of an image is shifted to a feature-level one, based on overall image average color for each color channel and on multi-resolution wavelet decomposition for edges information extraction. The color information and the coefficients of the decomposition are distilled into small "signatures" for each image. These signatures are then used in image querying metrics, analyzing average luminance color channel discrepancies and how many significant wavelet coefficients of one image has in common with potential targets.

A query image may be different from a target image, so the retrieval method must allow for some distortions and should be applied in conjunction with keyword-based querying, producing possibly better results, depending on keywords quality since they are manually entered by humans and error prone. In spite of not achieving perfect results, the use of this content-based image analysis technique was chosen due to the low processing power and storage of mobile devices, as end-targets. Thus, it was required a simple, little signature storage overhead, and a fast algorithm to be performed on a database of 3500 plus multi-resolution images. Successful experiments with this algorithm with hundreds of queries and databases with 1000 and 20,000 images using conventional color histogram norms can be found in [CEJ95].

## 3.2 Wavelets and image decomposition

Wavelets are a mathematical tool for hierarchically decomposing functions. They allow a function to be described in terms of a coarse overall shape, plus details that range from broad to narrow. Regardless of whether the function of interest is an image, a curve, or a surface, wavelets offer an elegant technique for representing the various levels of detail. They have advantages over traditional Fourier methods in analyzing physical situations where the signal contains discontinuities and sharp spikes. Wavelets were developed independently in the fields of mathematics, quantum physics, electrical engineering, and seismic geology. Interchanges between these fields during the last years led to many new wavelet applications such as image compression, turbulence analysis, human vision, radar, and earthquake prediction. For more detailed information on wavelets applications see [kn:c]. As described in [CEJ95], Haar wavelets are the fastest to compute due to their simplicity, and in this work, it will be described how "lossy" image compression with this kind of wavelet decompositions produces a basis to create a metric used for content-based image similarities. In section 3.2.1 it is described how one-dimensional functions can be decomposed and compressed using Haar wavelets. In Section 3.2.2 it is described how this compression technique is extended to two dimensions (images) and in Section 3.2.4 it is described how image edge information is extracted from compressed images.

### 3.2.1 One-dimensional Haar wavelet decomposition and compression

As explained in [EJS95], to get a sense of how wavelets work, suppose a one-dimensional "image" with a resolution of 4 pixels is given, having the following pixel values:

$$[8 \quad 4 \quad 1 \quad 3]. \tag{3.1}$$

A representation of this image through Haar wavelet basis can be done as follows. First, average the pixels together, pair wise, to get a lower resolution image with pixel values:

$$[6 \quad 2]. \tag{3.2}$$

Some information has been lost in this averaging and downsampling process. In order to be able to recover the original four pixel values from the two averaged pixels, we need to store detail coefficients, capturing the missing information.

In this example, it is chosen 2 for the first detail coefficient, since the average we computed is 2 less than 8 and 2 more than 4. This single number allows the recovering of the first two pixels of the original 4-pixel image. Similarly, the second detail coefficient is -1, since 2 + (-1) = 1 and 2 – (-1) = 3. The summary of this decomposition into lower-resolution version with detail coefficients is shown in table 3.1.

Table 3.1: Image decomposition - First step.

| Resolution | Averages | Coefficients |
|---|---|---|
| 4 | [8 4 1 3] | |
| 2 | [6 2] | [2 -1] |

Repeating this process recursively on the averages gives the full decomposition shown in table 3.2.

Table 3.2: Image decomposition - Final step.

| Resolution | Averages | Coefficients |
|---|---|---|
| 4 | [8 4 1 3] | |
| 2 | [6 2] | [2 -1] |
| 1 | [4] | [2] |

With the final 4-pixel image wavelet transformation we get a single coefficient representing the image overall average color, followed by the detail coefficients in order of increasing resolution. Thus, the one-dimensional wavelet transformation of the example image is

$$[4 \quad 2 \quad 2 \quad -1]. \tag{3.3}$$

No information has been lost by this process. The original image had 4 coefficients, and so does the transform. It is possible to reconstruct the image to any resolution by recursively adding and subtracting the detail coefficients from the lower-resolution versions. Storing the wavelet transform of the image, rather than the image itself, has some advantages. One of them is that often a large number of the detail coefficients turn out to be very small in magnitude, thus, thresholding or removing these small coefficients introduces only small errors in the reconstructed image, giving a form of “lossy” image compression. After this introductory example on how wavelets work using images as sequences of coefficients, we can now alternatively think of images as piecewise-constant functions. To do so, it is necessary to understand the concept of vector spaces from linear algebra, because wavelets make use of them. Basically, a vector space is a set of vectors

such as

$$V = [(0,1),(1,0),(\frac{1}{2},1/2),(\frac{-1}{2},0)]. \tag{3.4}$$

In addition to defining a vector space by explicitly stating all the vectors that make up the space, it is also possible to define a vector space using a function. When reading material on wavelets, the vector space $R^2$ is often mentioned meaning the 2 dimensional plane of real numbers. This vector space can be thought of as the plane made up by the $x$ and $y$ axis. There are three important concepts needed for understanding vector spaces:

- *Linear combination* — This is the easiest term to understand. It is simply the sum of a set of vectors or equations where each vector, value or function is multiplied by some real constant. It can be represented mathematically as follows:

$$\sum_{n=1}^{k} c_n v_n \tag{3.5}$$

- *Span* — A span is a set of vectors whose linear combination can create all of the other vectors in the vector space. For example, using $i = (1,0)$ and $j = (0,1)$ is possible to recreate any of the vectors previously mentioned through a linear combination. So the vectors

$$[(1,0),(0,1)] \tag{3.6}$$

  span the vector space $V$ mentioned above.

- *Basis* — The smallest set of vectors that can span a vector space. If it is created a sub set of vector space $V$ called $S$ consisting of the set $[i,j,k]$, it is true that it spans $V$, but is not a basis for $V$, because $k$ can be eliminated from $S$ since $k$ can be created using $i$ and $j$. So, $S$ is not a basis, but a set of the vectors $[i,j]$ is a basis for the vector space $V$.

For practical purposes one can think of an image as a vector space such as $V^j$, $V^{j+1}$ would be an higher resolution version and $V^{j-1}$ would be that image at a lower resolution until it is reached $V^0$ where we have one pixel in the entire image. Now, one needs to define a basis for each vector space $V^j$ called scaling functions. The basis function for the spaces $V^j$ are usually denoted by the symbol $\phi$. This basis is obtained through the use of Haar wavelets. Their piecewise functions and graphic are as follows:

$$\phi(x) = \begin{cases} 1 & \text{if } 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

$$\psi(x) = \begin{cases} 1 & \text{if } 0 \leq x < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2}\, x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

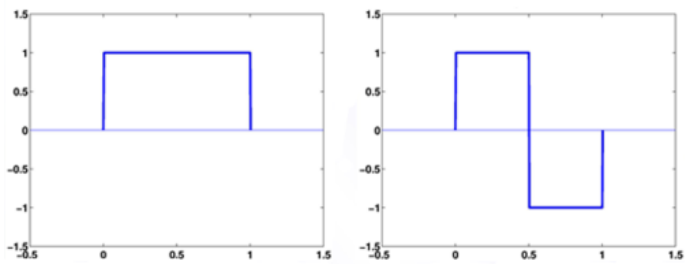


Figure 3.1: Haar wavelet scaling function $\phi(x)$ on left and Haar wavelet function $\psi(x)$ on right.

These are the root functions for the Haar wavelet. The function $\phi(x)$ as previously mentioned is called the scale function of the Haar wavelet, or the "box" function, and the function $\psi(x)$ is the actual wavelet, their graphics are shown on Figure 3.1. With these two functions alone it is not possible to do much, so it must be created a function who translates and scales both of these functions which are the following:

$$\phi_{i,j}(x) = \phi(2^j - i) \qquad \text{for } i = 0, ..., 2^j\text{-}1 \tag{3.9}$$

$$\psi_{i,j}(x) = \phi(2^j - i) \qquad \text{for } i = 0, ..., 2^j\text{-}1 \tag{3.10}$$

The $j$ is responsible for the scaling of the function; it basically shrinks and expands the graph. The $i$ is the translation of the graph across the time axis. As an example, Figure 3.2 shows the four box functions forming a basis of $V^2$ for the interval $[0,1)$.

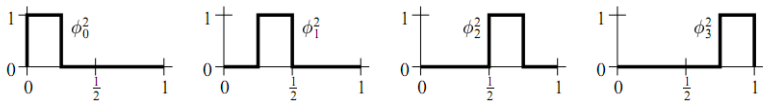


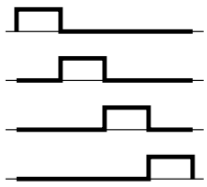

Figure 3.2: The box basis for $V^2$ in the interval [0,1).

As described in [EJS95], the next step is to create a new vector space $W^j$ orthogonal complement to $V^j$. This new vector can be thought as a means of using wavelets to represent the parts of $V^{j+1}$ that cannot be represented in $V^j$. The basis functions of $W^j$ have two important properties:

1. The basis functions $\psi_{i,j}$ of $W^j$, together with the basis functions $\phi_{i,j}$ of $V^j$, form a basis for $V^{j+1}$.
2. Every basis function $\psi_{i,j}$ of $W^j$ is orthogonal to every basis function $\phi_{i,j}$ of $V^j$ under a chosen inner product.

So, beginning by expressing an example "image" [9 7 3 5] as linear combination of the box basis functions

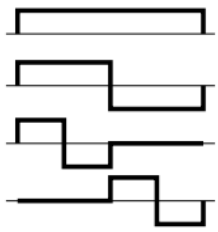
$$\begin{aligned} \mathcal{I}(x) &= 9 \times \\ &+ 7 \times \\ &+ 3 \times \\ &+ 5 \times \end{aligned}$$



, (3.11)

it is possible to rewrite the expression in terms of basis functions in $V^0$, $W^0$ and $W^1$:

$$\begin{aligned} \mathcal{I}(x) &= c_0^0\,\phi_0^0(x) + d_0^0\,\psi_0^0(x) + d_0^1\,\psi_0^1(x) + d_1^1\,\psi_1^1(x) \\ &= 6 \times \\ &+ 2 \times \\ &+ 1 \times \\ &+ -1 \times \end{aligned}$$



. (3.12)

In conclusion, these four final coefficients in the linear combination are the Haar wavelet transform “detail coefficients” of the original image. In Figure 3.3 different images reconstructions are demonstrated using different amounts of wavelet coefficients.

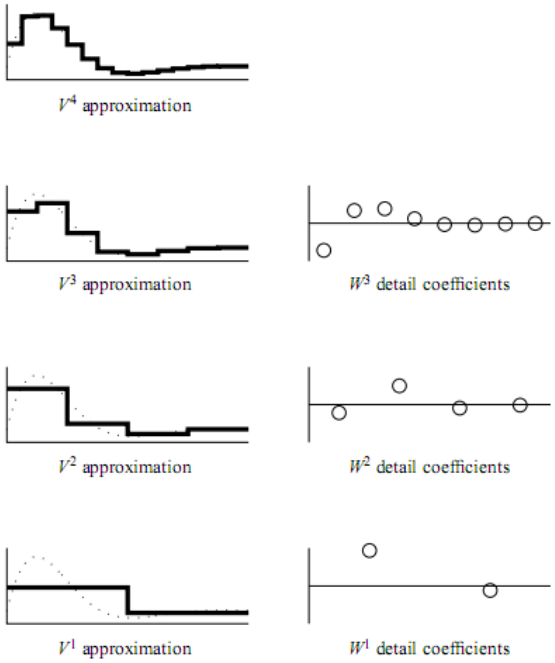



Figure 3.3: A sequence of decreasing-resolution approximations to a function (left), along with the detail coefficients required to recapture the finest approximation (right).

### 3.2.2 Image (two-dimensional) Haar wavelet compression and edge extraction

For image compression, the one-dimensional wavelet transform described in previews section is generalized into two dimensions. There are two ways to do the decomposition, in the standard manner, it is applied the one-dimensional wavelet transform to each row of pixel values, next, to theses transformed rows as if they were themselves an image, is applied the one-dimensional transform to each column. The non-standard decomposition of an image alternates between operations on rows and columns; a visual representation of these decompositions is shown on Figure 3.4.

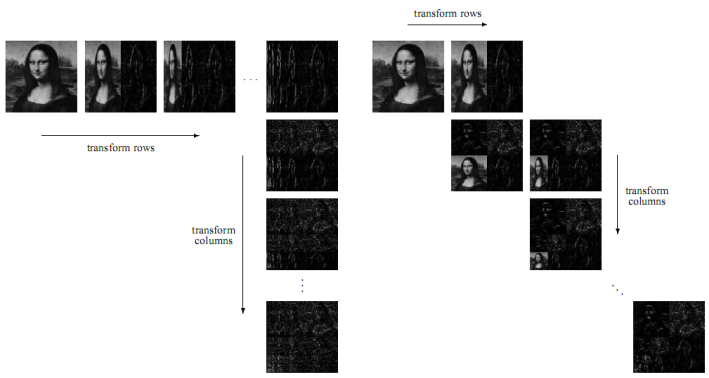



Figure 3.4: Standard decomposition (left), non-standard decomposition (right) [EJS95].

For a 128 x 128 pixel image, there are $128^2 = 16,384$ different wavelet coefficients for each color channel. Rather than using all of these coefficients in the metric to preserve the entire image detail, it is preferable to threshold the coefficients, keeping only the ones with largest magnitude. This threshold reduces storage for the database and provides acceleration in searching for a query. An example of images reconstruction using different amounts of wavelet coefficients is shown on Figure 3.5.

(a) (b) (c) (d)

Figure 3.5: Original image (a) represented using 21% of its coefficients with 5% error (b), 4% of its coefficients with 10% error (c) and (d) 1% of coefficients with 15% error. [EJS95]

The objective of the previous two types of image decomposition (standard and non-standard) is to extract edge information, and they provide different quality in results. In [CEJ95] it was shown that the non-standard manner is better at identifying image features that are about as wide as they are high, and the standard basis works best for images containing lines and other rectangular features giving better general results, thus, it is the method chosen once the objective is to work with general type images. Edges in conjunction with color information are likely to be among the key features to an image query. With image wavelet decomposition, edge information is extracted from the calculated coefficients, using them as if they were pixels themselves. In Figure 3.6 a display of image wavelet coefficients is represented using a wavelet toolbox within the MATLAB framework.

Figure 3.6: Original image (left), wavelet coefficients display using wavelet toolbox in MATLAB. [kn:b]

Although the use of the wavelet transform is a good approximation of the edges in an image, even in the presence of image noise, there are often many false edges as it can be

seen in the example of shown central image in the Figure 3.7. Therefore, thresholding the largest-magnitude of the coefficients also appears to improve the discrimination of image as shown on the right image in Figure 3.7.

Figure 3.7: Original image (left), Un-thresholded wavelet coefficients representation (center), thresholded wavelet coefficients representation (right).

### 3.2.3 The image compression algorithm

At a high level, a two-dimensional standard Haar wavelet decomposition is done for every image in the database as described in the previews sections, and just stores the overall average color and the indices and signs of the $m$ largest-magnitude wavelet coefficients. The indices for all of the database images are then organized into a single data structure that optimizes searching in the program. Then, for each new query image, we perform the same wavelet decomposition, and again throw away all but the average color and the largest $m$ coefficients are stored. The score for each target image T is then computed as explained in the Section 3.2.4. As demonstrated in Section 3.2.2, the standard two-dimensional Haar wavelet decomposition of an image involves a one-dimensional decomposition on each row, followed by a one-dimensional decomposition on each column of the result. The following pseudo code performs this one-dimensional decomposition on an array $A$ of $h$ elements, where $h$ is a power of two [CEJ95]:

**proc** DecomposeArray(A: array [0..h-1])
$A \leftarrow A/\sqrt{h}$
**while** h > 1 **do**
  $h \leftarrow h/2$
  **for**
  $i \leftarrow$ 0 to h - 1 **do**
    $A'[i] \leftarrow (A[2i]+A[2i+1])/\sqrt{2}$
    $A'[h+i] \leftarrow (A[2i]+A[2i+1])/\sqrt{2}$
  **end for**
  $A \leftarrow A'$
**end while**
**end proc**

The entries of “A“ are assumed to be 3-dimensional color components. The entire image T can thus be decomposed as follows [CEJ95]:

```
proc DecomposeImage(T: array [0..r-1,0..r-1] of color)
for row ← 1  to  r  do
  DecomposeArray(T[row,0..r-1])
end for
for col ← 1  to  r  do
  DecomposeArray(T[0..r-1,col])
end for
end proc
```

After the decomposition process, the entry $T[0,0]$ retains the average color of the overall image, while the other entries of $T$ contain the wavelet coefficients. Thresholding the magnitudes of the coefficients appears to have more discriminatory power for image querying than the features precise magnitudes. Thus, quantizing each significant coefficient to just two levels: 1 and $-1$, representing large positive coefficients and large negative coefficients works remarkably well, allowing for a fast comparison algorithm [CEJ95].

Finally, $T[0,0]$ is stored with the indices and signs of the thresholded and quantized $m$ wavelet coefficients of $T$. To optimize the search process, the $m$ wavelet coefficients of all images in the databases are organized into a set of six arrays, called the search arrays, with one array for every combination of sign ($+$ or $-$) and color channel (such as $R$,$G$, and $B$). For example, let $R+$ denote the “positive” search array for the color channel $R$. Each element $R_+[i,j]$ of this array contains a list of all images $T$ having a large positive wavelet coefficient $T[i,j]$ in color channel $R$. Similarly, each element $R_-[i,j]$ of the “negative” search array points to a list of images with large negative coefficients in $R$.

In the implementation used, the search arrays are created as a pre-process for a given database and stored on disk. The new added images are decomposed and therefore augment the database search arrays accordingly.

### 3.2.4 Image distance metric and scoring algorithm

In order to calculate the content-based image distance metric, consider $Q$ and $T$ as representing just a single color channel of query and target images wavelet decompositions. Let $Q[0,0]$ and $T[0,0]$ be the coefficients corresponding to the overall average intensity of that color channel. Further, let $Q'[i,j]$ and $T'[i,j]$ represent the $[i,j]$-th thresholded and quantized wavelet coefficients of $Q$ and $T$; these values are either $-1$ or $+1$. The distance metric between $Q$ and $T$ is

$$W_0|Q[0,0]-T[0,0]|-\sum_{i,j}W_{bin(i,j)}(\tilde{Q}[i,j]=\tilde{T}[i,j]) \tag{3.13}$$

The expression $\tilde{Q}[i,j] = \tilde{T}[i,j]$ is represented by 1 when $\tilde{Q}[i,j]$ and $\tilde{T}[i,j]$ are equal and 0 otherwise. The weights $W_0$ and $W_{bin(i,j)}$ in the equation provide a convenient mechanism for tuning the metric to different styles of image querying, allowing for some image distortions in the following manner: the coefficients are distributed into a small number of groups. Then each group is scaled to a given importance by some weights w[b], where b is the group "id" as presented in Table 3.3, which has a good set of weights that was found experimentally and used in [CEJ95] using YIQ color space and standard decomposition:

Table 3.3: Different weights for "painted like" approximation images (left) and for "scanned like" approximations (right).

| b | $w^Y[b]$ | $w^I[b]$ | $w^Q[b]$ | $w^Y[b]$ | $w^I[b]$ | $w^Q[b]$ |
|---|---|---|---|---|---|---|
| 0 | 4.04 | 15.14 | 22.62 | 5.00 | 19.21 | 34.37 |
| 1 | 0.78 | 0.92 | 0.40 | 0.83 | 1.26 | 0.36 |
| 2 | 0.46 | 0.53 | 0.63 | 1.01 | 0.44 | 0.45 |
| 3 | 0.42 | 0.26 | 0.25 | 0.52 | 0.53 | 0.14 |
| 4 | 0.41 | 0.14 | 0.15 | 0.47 | 0.28 | 0.18 |
| 5 | 0.32 | 0.07 | 0.38 | 0.30 | 0.14 | 0.27 |

The resultant expression is a weighted sum of the difference in the average color between $Q$ and $T$, and the number of stored wavelet coefficients of $T$ whose indices and signs match those of $Q$. To compute the score, each color channel is looped. First, the differences between the query's average intensity in that channel $Q[0,0]$ and those of the database images are computed.

Next, for each of the $m$ non-zero thresholded wavelet coefficients $Q'[i,j]$, it is searched through the image database the images containing the same magnitude coefficient and sign. Those image's scores are updated accordingly. The algorithm embodiment of the equation is [CEJ95]:

```
func ScoreQuery(Q: array [0..r-1,0..r-1] of color; m: int)
DecomposeImage(Q)
Initialize scores[i] ← 0 for all i
for each color channel c do
  for each database image T do
    scores[index(T)] += w^c[0]*|Q^c[0,0]-T^c[0,0]|
  end for
  Q̃ ← TruncateCoefficients(Q,m)
  for each non-zero coefficient Q̃^c[i,j] do
    if Q̃^c[i,j] > 0 then
      list ← D̃^c_+[i,j]
    else
      list ← D̃^c_-[i,j]
    end if
    for each element l of list do
      scores[index(l)] −= w^c[bin(i,j)]
    end for
  end for
end for
return scores
end func
```

Finally, the smallest (typically, the most negative) scores are considered to be the closest matches. Since the score itself is dependent on the nature of the images, for example, making comparisons between different amounts of edges in each one, only returns a score relative to the image query itself. Thus, given an image A and B, if they return a similarity score of -19, does not mean they are more similar to each other than images C and D with a score of -15. But if images A and C produce a score of -15, means that B is more similar to A than C. This property arises because an image compared with itself can produce a "maximum similarity score" different of another image compared with itself. Thus, to normalize scoring values, each image is compared to itself, getting the relative value of 100% similarity. Then for each image that is compared to the original, the similarity percentage is calculated using the previous relative value as reference and augmented by 10% for each keyword in common (only used in the next chapter clustering process). After the score query is terminated, a "Heap-Select" algorithm is used to find the $m$ requested closest matches in linear time.

## 3.3 Test application

A simple interactive test web application, that incorporates the previous image querying algorithm and search arrays, was written in C++, using Ruby on Rails for the interface. The application architecture overview can be seen in Figure 3.8 and the homepage in Figure 3.9. For the test image collection domain, 5000 random images with associated metadata, were collected from Flickr, taking into account that for each image there are at least a few similar ones, to avoid a production of an heterogenic local image collection. The test application has functionalities such as find images by text (Figure 3.10), find images by similar content (Figures 3.11, 3.12), and a mix of text and content search by checking the "Use metadata helper to find similar images" option in the application homepage. The disk usage for the image database grows linearly with the number of images. As an example, a database with 5000 images takes 5.2 Mbytes of disk space and RAM memory. Adding images to a database has a constant complexity, i.e. it always takes the same time regardless of how big the image collection is. When it comes to querying for similar images, the complexity is also linear with the size of the image collection. If the collection with 10.000 images takes 5 seconds to query, and are increased to 1.000.000 images, then it should take 50 seconds to query.

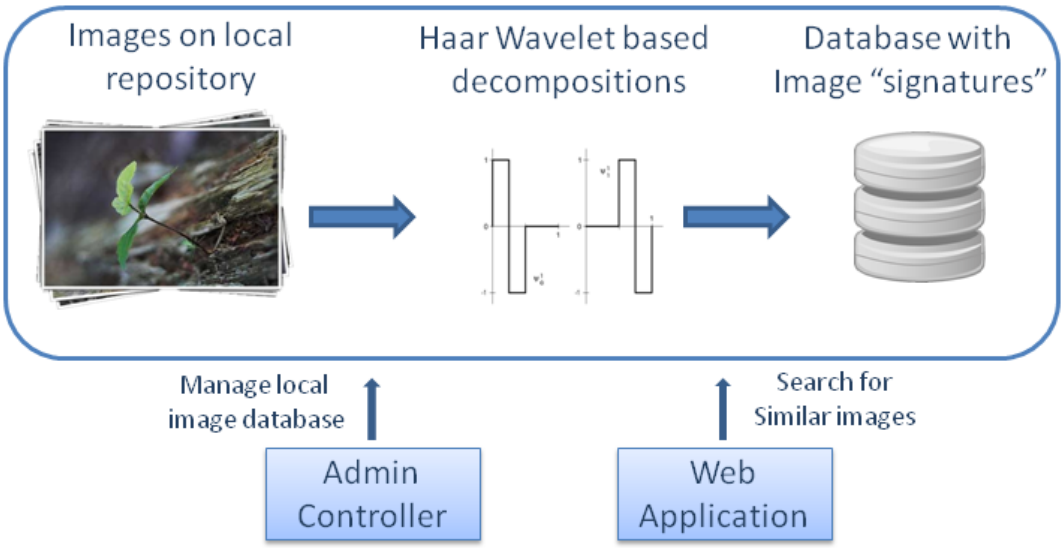



Figure 3.8: Test application architecture overview.

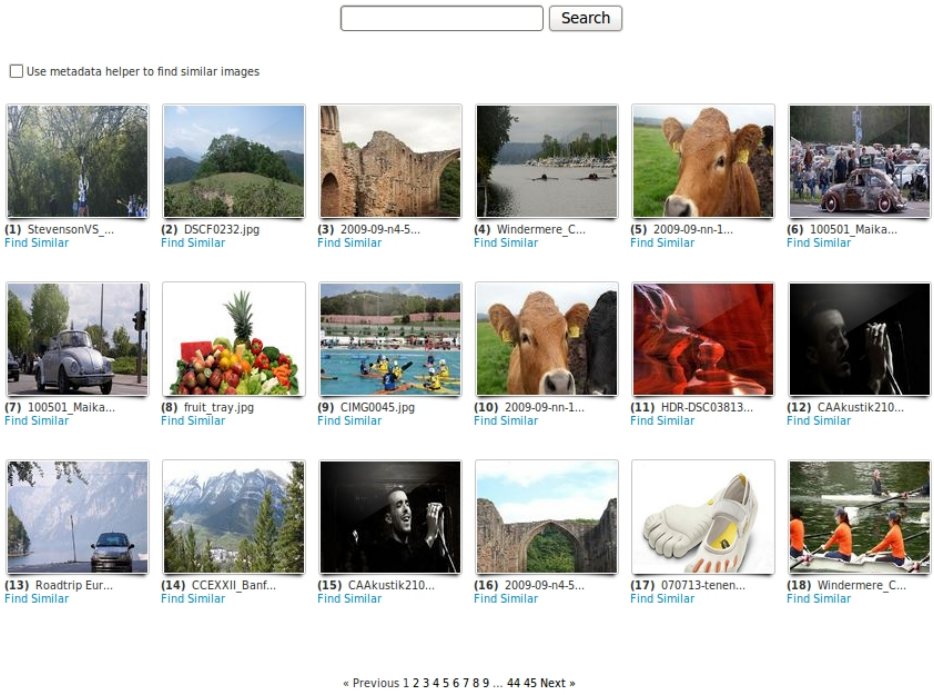



Figure 3.9: Test application home page.

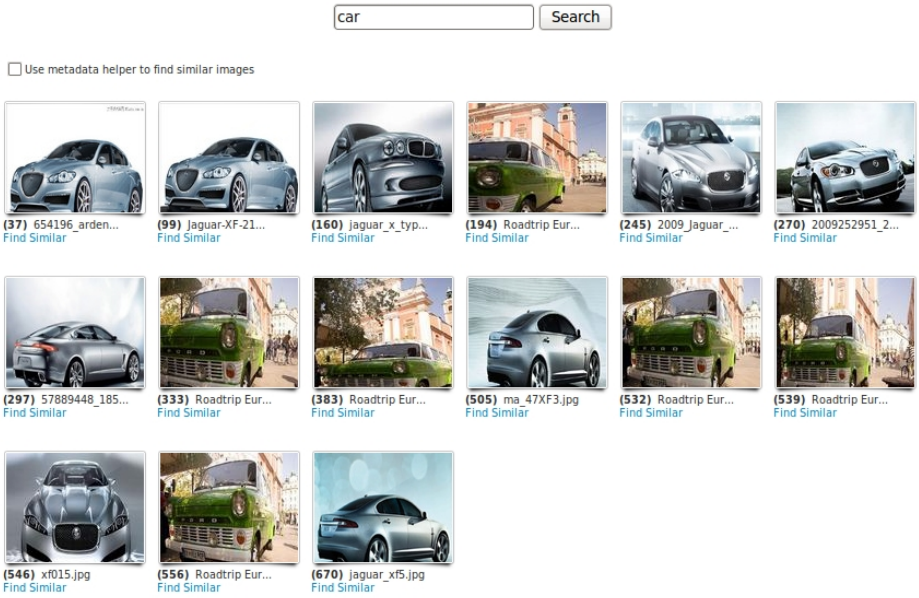



Figure 3.10: Find images by text.

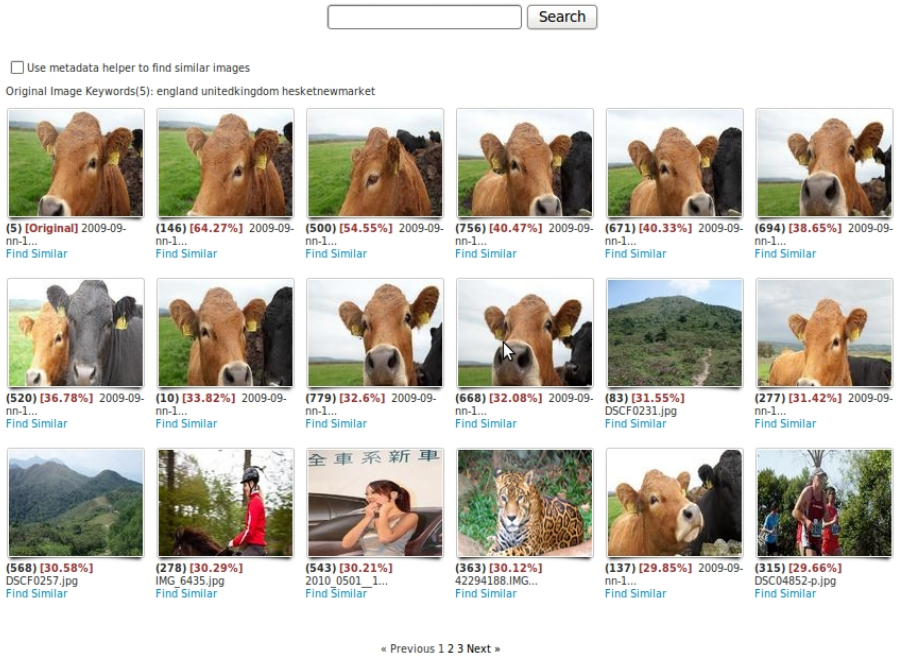



Figure 3.11: Find similar images (cow original image query at top left corner).

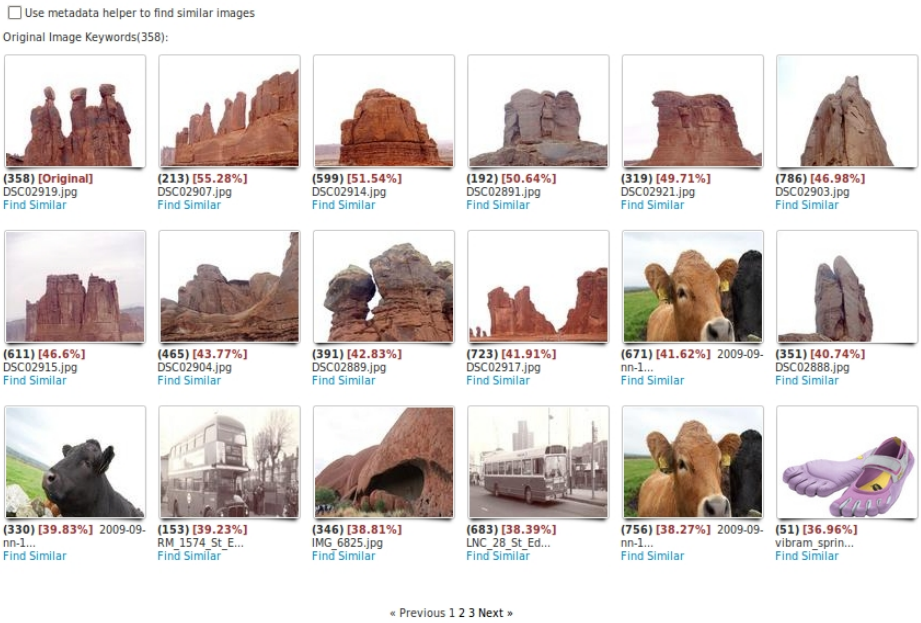



Figure 3.12: Find similar images (mountain original image query at the top left corner).

Using the metadata helper option, when the user presses “Find Similar”, the images set is first reduced by using the original image metadata (only images that share at least one similar tag are presented) and then a sort is applied by analyzing the content of the previous results. In Figures 3.11 and 3.12 some results are presented regarding only content-based features. The original example image is located at the top left corner and then the sorted similar images are displayed ordered according to the relative similar percentage, shown in red. In some cases, not intended images appear with better scores than expected ones. For instance in Figure 3.11 the green mountain image appears first than other cows images, and in Figure 3.12, the cow image appears first than the other mountain image. This kind of results are due to the nature of the algorithm itself, since the cow and mountain images can share similar luminance averages in the three color channels, and similar edges. To analyze this effect, an algorithm sensitivity test was performed using a butterfly image and variations regarding color shifts and shape rotations. Then, they were compared to each other producing the sorted similar image results in Figure 3.13, and more detailed values in table 3.4.

Figure 3.13: Butterfly image and variances sorted by content similarity.

Table 3.4: Original image and variances similarity values and luminance difference value.

| | Image similarity value | Average color difference | Similarity Score |
|---|---|---|---|
| Original copy | -40.700 | 0.000 | 100% |
| Black and white | -25.772 | 0.087 | 62.69% |
| Color shift | -19.580 | 0.229 | 38.04% |
| Different shape | -14.998 | 0.061 | 36.11% |
| Rotated 90º | -12.979 | 0.000 | 33.52% |
| Rotated 180º | -20.149 | 0.000 | 52.04% |

The black and white version of the original image has the better similarity results in the test, followed by 180º rotation, color shift, different shape and 90º rotation. Thus, it appears that the algorithm is more sensitive to edge variations than color luminance variations. This phenomenon is due to the weights values given at edge information in the scoring function, causing a greater sensitivity to edge variations than color variations.

## 3.4 Summary

The use of image wavelet compression provides a metric that is fast to compute, requiring little storage for each image database due to good image approximation with just a few largest magnitude coefficients. Wavelet decompositions can be used to extract and encode edge information. Edges allied to color are likely to be among the key features for matching content-based similarities with little processing time. The coefficients of wavelet decomposition provide information that is independent of the original image resolution. Thus, a wavelet-based scheme allows the resolutions of the query and the target to be effectively decoupled. Wavelet decompositions are fast, requiring linear time in the size of the image and very little code. This technique is however sensitive to edge variances of the query and target images, producing sometimes unexpected results. In conclusion, the use of wavelet decompositions for content-based image search does not always produce robust results in order to be fast. Thus, its use is only recommended in synergy with text-based techniques.

# Chapter 4

# Unsupervised Learning

## 4.1 Introduction and overview

Through the use of metadata and image wavelet decomposition, an image distance metric was created. This metric was used as a foundation of a previously constructed test application, where results appeared sorted by content-based similarities and high-level features such as a query by text. This searching method requires that, at some point, users issue an explicit query, be it textual or pictorial. In some cases, users may not have a well-defined information and simply wish to browse the image collection to identify suitable images to use them as queries. This problem increases when browsing unstructured large databases, where the access through a simple image collection list becomes increasingly difficult, requiring a visually guided search. With this in mind, a problem arises, that of finding a good organization principle for structuring the image collection database.

This chapter focuses on building the internal structure organization. Starting by using an adaptation of the k-means algorithm, images are arranged into clusters with content-based and metadata similarities. Next, a static hierarchical structure is created, by dividing recursively the image database, using the same algorithm. The image database is therefore represented as a tree, using representative images (general images) as top nodes and more concrete images as descending nodes. The reason to use this kind of structure is its extensive use and good results in other fields of science, having a simple implementation, quickness of calculation, capability of identifying nested clusters and flexibility on producing different forms of image trees.

## 4.2 Image clustering using an adaptation of k-means algorithm

Clustering is one of the most important unsupervised learning problems. It deals with finding a structure in a collection of data points. This kind of analysis is based on partitioning a collection of data points into a number of subgroups, where the objects inside a subgroup (a cluster) show a certain degree of closeness or similarity. Each cluster must contain at least one data element, and each data element must belong to exactly one group. The notion of what constitutes a good cluster depends on the application and there are many methods for finding clusters subject to various criteria, both ad-hoc and systematic. These include approaches based on splitting and merging such as ISODATA [BH64, JD88], randomized approaches such as CLARA [KR90], CLARANS [NH94], methods based on neural nets [Koh89], and methods designed for scaling large databases, including DBSCAN [ME95], BIRCH [TZ97], ScaleKM [PB98]. Among clustering formulations that are based on minimizing a formal objective function (such as the distance between images in a cluster), perhaps the most widely used and studied is k-means algorithm [KAS98], and for that reason, it was the method chosen in this case study.

There are no efficient solutions known to any of these problems and some formulations are NP-hard [GJ79]. One of the most popular heuristics for solving the problem using k-means algorithm is based on a simple iterative scheme for finding a locally minimal solution. The main idea is to define $k$ centroids, one for each cluster. These centroids should be placed in a smart way because different locations cause different results. So, a better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate $k$ new centroids as centers of the clusters resulting from the previous step. After we have these $k$ new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the $k$ centroids change their location step by step until no more changes are done. In other words, centroids do not move any more. As an illustrative example of this clustering process, suppose the Euclidean k-means problem where the objective function is to minimize the sum of positional distances to the nearest centroid. First, we have some points in a cartesian spaces. We ask the user how many clusters he/she wants, for example $k = 5$, then we randomly guess the cluster center locations as shown in Figure 4.1. Each data point finds out which center it is closest to, dividing the collection in 5 groups. Each center owns a set of data points, Figure 4.2. Then each center finds the new central point of its own cluster, Figure 4.3. Repeating the process of binding points to its closer centroid and re-calculation of $k$ centroids until there are no modification of centroids or until a pre-fixed number of iterations is reached, the process terminates, Figure 4.4.

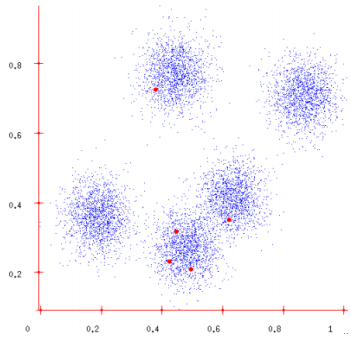



Figure 4.1: Randomly placed 5 centroids on a data space [kn:a].

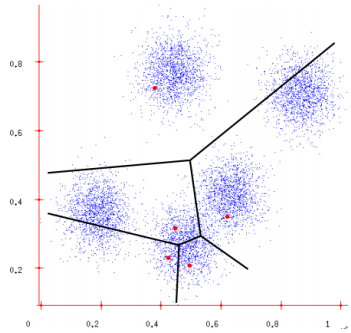



Figure 4.2: Points mapped to centroids through the use of Euclidian distance objective function [kn:a].

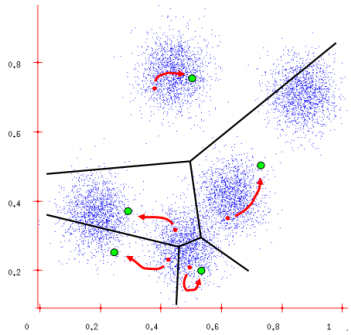



Figure 4.3: New centroid calculation for each cluster [kn:a].

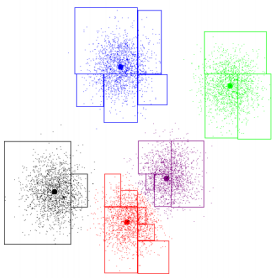


Figure 4.4: Final result of clustering process after some repeating iterations [kn:a].

Since in our case study the images decompositions structures are not mapped as points on the Cartesian space an adaptation of k-means algorithm is necessary. This relevant property means that centroids always need to be an image and not a random point in space as in the original algorithm. Another issue arises on the re-calculation of $k$ new centroids, because it is not possible to do arithmetic calculations regarding the central position of the cluster. But, as the image collection conceptual structure can be perceived as a strongly connected graph, each node being an image and each edge representing the similarity distance between a pair of images, the problem of finding the new centroid becomes a graph theory problem, causing a calculation overhead when compared to the original algorithm. In our case study, a simplistic approach was used. Having a cluster graph, the new image centroid is found as the image with the least sum value of distances to other images in the same cluster, meaning that, it is the image closer to all other images.

The "k-means adaptation" pseudo-code is presented as follows:

**Input**
$E = e_1, e_2, ..., e_n$ (set of entities to be clustered)
$k$ (number of clusters);
$MaxIters$ (limit of iterations)
**Output**
$C = c_1, c_2, ..., c_k$ (set of cluster centroids);
$L = l(e)|e = 1, 2, ..., n$ (set of cluster labels of $E$)
**for** $c_i$ in $C$ **do**
  $c_i \leftarrow e_j \in E$ (random selection)
**end for**
**for** $e_i$ in $E$ **do**
  $l(e_i) \leftarrow argminDistance(e_i, c_j) j \in 1..k$
**end for**
$changed \leftarrow false$; $iter \leftarrow 0$;
**repeat**
  **for** $c_i$ in $C$ **do**
    updateCluster($c_i$) (find central node in graph)
  **end for**
  $changed \leftarrow false$
  **for** $e_i$ in $E$ **do**
    $minDist \leftarrow argminDistance(e_i, c_j) j$ in $1..k$
    **if** $minDist \neq l(e_i)$ **then**
      $l(e_i) \leftarrow minDist$; $changed \leftarrow true$;
    **end if**
  **end for**

$iter++$
**until** $changed = false$ or $iter \geq MaxIters$

## 4.3 Experimental results of k-means adaptation algorithm

When performing unsupervised clustering on a data set which we have no prior knowledge, an interesting question arises: What is the actual number of clusters present in the database? Since the aim of the project is to navigate through images centroids on a small screen display, we do not have much space to watch many centroids images at a time. Thus, for this experiment, k=10 was chosen. The database used throughout the experiments consists of 794 images, selectively hand-picked from a database of 5000 images, randomly downloaded from Flickr. The selection of images was due to the creation of 17 image categories, in order to avoid a heterogeneous collection. The images within each category have similar colors and color spatial layout, and can be labeled with a high-level semantic description. Some sample images from each of the categories are presented in figures 4.5 and 4.6. Table 4.1 summarizes the number of images extracted for each of the categories. Figure 4.7 presents the overview of the clusters created by the algorithm, using Graphviz visualization tool. Figure 4.8, 4.9 and 4.10 focus some resulting clusters with more detail, each one presenting a centroid connected to other images in the cluster through an undirected edge.

Figure 4.5: Image database categories 1-8.

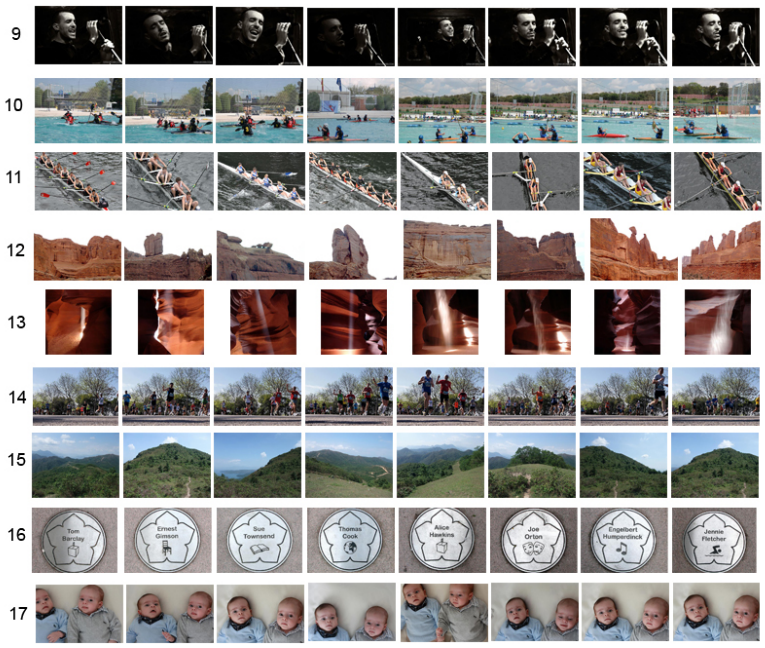



Figure 4.6: Image database categories 9-17.

Table 4.1: Number of images in each category.

| Category | Number of images |
|---|---|
| 1 | 32 |
| 2 | 30 |
| 3 | 13 |
| 4 | 21 |
| 5 | 149 |
| 6 | 12 |
| 7 | 37 |
| 8 | 22 |
| 9 | 15 |
| 10 | 36 |
| 11 | 53 |
| 12 | 30 |
| 13 | 132 |
| 14 | 59 |
| 15 | 134 |
| 16 | 9 |
| 17 | 10 |

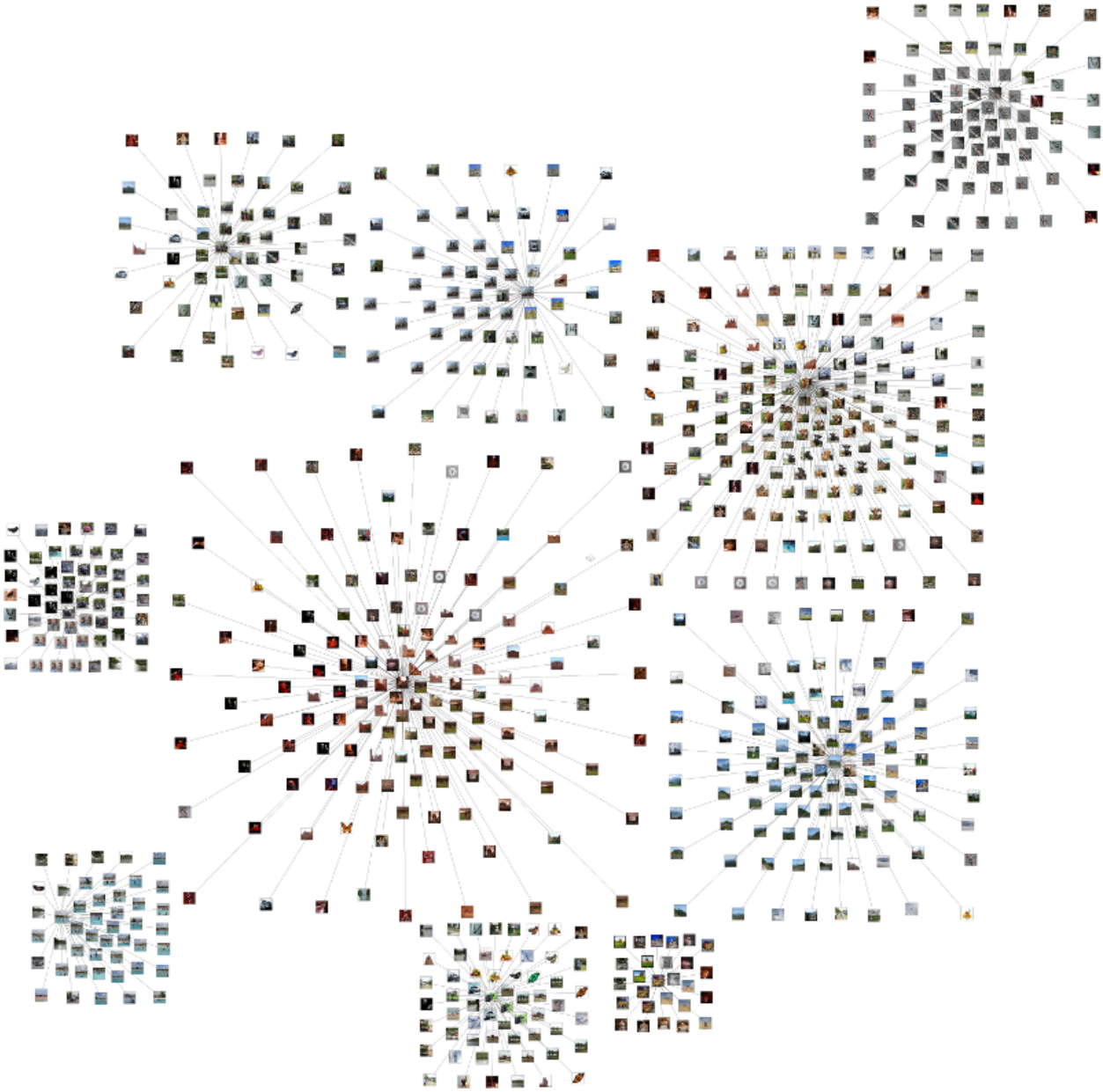


Figure 4.7: Overview of database clustering result with $k = 10$.

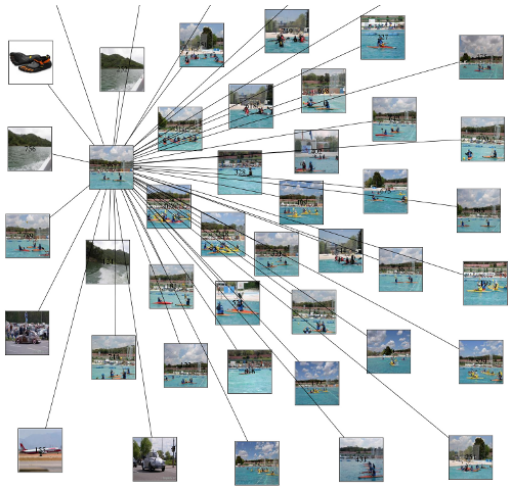


Figure 4.8: Focus cluster containing category 10.

Figure 4.9: Focus cluster containing categories 7, 9 and 17.

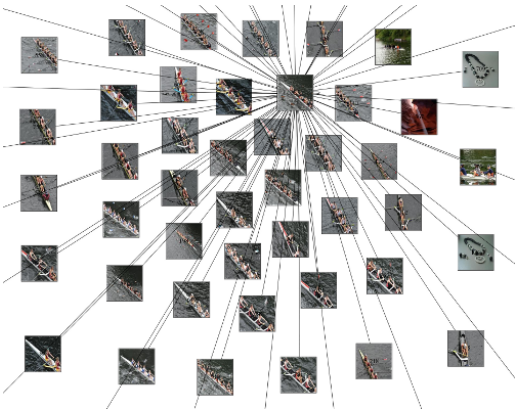


Figure 4.10: Focus cluster containing category 11.

The key features of k-means that make it efficient are often regarded as its risky drawbacks; an inappropriate choice of the number of clusters as well the starting position of its centroids and an inappropriate distance metric between images can produce unexpected results. In our experiment though, it was clear the division of the database in groups of similar images in spite of some minor image categories being splited through different clusters. This phenomenon is due to the lack of robustness of image distance metric and possible errors of image labels. As the number of clusters used in this experiment is smaller than the actual categories in the database, categories with big amount of images tend to occupy almost one entire cluster as shown in Figures 4.8 and 4.10. In counterpart, small categories tend to be merged into one cluster, as shown in Figure 4.9, being represented by an image of the bigger category (the centroid). The image category distribution through clusters can be seen in Table 4.2.

Table 4.2: Image category distribution through clusters.

| | Clusters | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Category | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 8 | 0 | 1 | 12 | 5 | 5 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 8 | 22 | 0 | 8 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 13 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 21 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 132 | 0 | 0 | 0 | 0 | 17 | 0 |
| 6 | 0 | 0 | 0 | 4 | 0 | 4 | 0 | 0 | 0 | 4 |
| 7 | 0 | 5 | 0 | 0 | 18 | 0 | 6 | 8 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 22 | 0 | 0 | 0 |
| 9 | 0 | 1 | 0 | 0 | 0 | 5 | 9 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 36 | 0 | 0 |
| 11 | 45 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 9 | 0 | 21 | 0 | 0 | 0 | 0 |
| 13 | 5 | 0 | 0 | 25 | 0 | 102 | 0 | 0 | 0 | 10 |
| 14 | 0 | 19 | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 10 | 16 | 0 | 0 | 0 | 88 | 4 | 10 | 6 |
| 16 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 1 | 9 | 0 | 0 | 0 | 0 | 0 | 0 |

## 4.4 Experimental results of hierarchical k-means adaptation algorithm

In order to isolate image categories within multi-category clusters, the same algorithm is applied recursively to each cluster, until a cluster size less or equal than $k$ is reached. In this experiment we have used again $k = 10$ due to mobile screen size constraints.

A tree structure is built by keeping a connection of previously calculated centroids to their "inner-centroids", representing a sequence of nested clusters. On the top are the first centroids, as calculated in Section 4.3 which includes all other clusters. At the bottom are the images representing single elements of the clusters. The top calculated centroids represent a generic topic of images (representants of their "inner-clusters") and descending centroids represent the concrete subcategories. Using this kind of structure with a top-down navigation, the system can offer a fast way of finding images the user wants through a process of decisions, leading to the chosen filtered images. Figures 4.11, 4.12, 4.13 focus some nested clusters of the resulting tree.

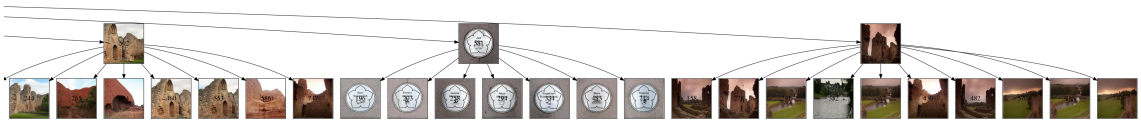


Figure 4.11: Focus nested clusters in the tree structure containing categories 2, 3 and 16.

Figure 4.12: Focus nested clusters in the tree structure containing categories 11, 14 and others.

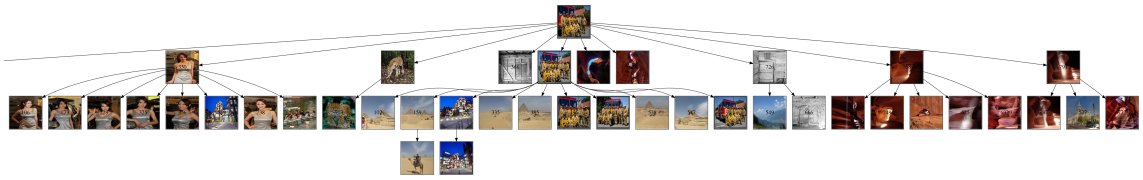


Figure 4.13: Focus nested clusters in the tree structure containing categories 6 and others.

## 4.5 Summary

Through the synergy of the obtained hierarchical image tree structure with the "find similar" feature presented in Chapter 3, we have reached a building foundation of an interactive image navigation system. The user has a first contact with the system through 10 representative images (the first calculated centroids). Each one represents a path from a general topic to a more concrete subtopic(s). Using connections between centroids and sub-centroids, one can "take a look" and have an idea of what they bring and then choose to continue or not the path. Through this kind of top-down tree navigation, the image navigation may be somewhat exploratory in the beginning with a difference that subsequent refined paths are expected to increase the clarity of what the user wants from the system. If some similar images belong to different clusters and the user chooses to watch all separated images, instead of moving up and down in the tree structure, he/she can always use one image as query to bring all similar images as a filtered list through "find similar" feature. In Chapter 5 we will discuss the strategy of displaying the tree structure and how users can navigate it using a small screen display.

# Chapter 5

# Visualizing image collections on Mobile Devices

## 5.1 Introduction

Computer users have always relied on visual tools and techniques such as maps, diagrams and charts to better understand problems and solve them in less time. The continuous improvement of computers performance and graphics made it possible to implement a wide range of advanced visualization techniques for several different domains. However, adapting these techniques to mobile devices such as PDAs (Personnal Digital Assistants) and Smartphones poses several technical and usage limitations problems. In order to harness the power of visualization on mobile devices an additional effort is needed to understand how to get the most from these tiny devices we carry everyday in our pockets. Compared to mobile desktop computers, mobile devices have a set of restrictions that limit the development of applications [Chi06]:

1. Displays are smaller, have lower resolution and smaller color range.
2. The width/height aspect ratio differs from desktop displays.
3. Onboard hardware such as CPU, GPU and memory are limited.
4. Input devices such as tiny keyboards, microjoysticks and scroll balls are inadequate for complex user tasks.
5. Connectivity is slower, affecting interactivity when a significant quantity of data is retrieved/stored remotely.
6. Form-factor, performance and input devices vary greatly.
7. Development tools and graphic libraries tend to be very limited.

Due to mobile devices characteristics, applications developed for desktop computers do not scale to mobile devices. Despite the current development on mobile devices, some of these limitations are unlikely to disappear as the devices must remain compact in size to be practical [HYY06].

## 5.2 Mobile Visualization Design and User Interaction

Mobile visualization design should address a set of steps in order to avoid situations such as misleading the user, the user getting lost when navigating using a map or presenting great looking and impressive graphics that are difficult to read or interpret. The main steps are [Chi04]:

- *Mapping* — the way information is visually encoded is crucial. Visualization turns numbers, objects, relations and strings into graphics with several features– colors, lines, lengths, curvatures, animations. An effective mapping should be consistently applied through the application and easy to interpret.
- *Selection* — Due to display size limitation and mobile context, already mentioned, it is important to show only the essential information. Insufficient information will lead to suboptimal or wrong decisions while excess of information may confuse the user or make it difficult to reason.
- *Presentation* — Even with an efficient mapping it is important to think about how information is laid out on the display. Special attention must be given to the useful screen area and if it is enough to display all the required information.
- *Interactivity* — The way users interact with information is crucial in order to engage their attention. Current devices with touch capabilities make it easier to develop new techniques that are very intuitive.
- *Human Factors* — Users must be able to quickly recognize and easily interpret a visualization. The application interface should take into account human perceptual and cognitive capabilities.
- *Evaluation* — Usability testing is the most crucial. The best way of doing user evaluation is to test the application in quasi-real situations. Mobile application evaluation should be done with real devices (phone emulators can lead to unreliable results) and distraction should be measured.

The existing mobile devices provide specific sensors and ways of interacting with the interface. The following list provides a briefing over the existing technologies [BBB06]:

- *Pen* — An accessory similar to a pen is used as a picking device on touch screen displays.
- *Multi-touch* — The display is capable of detecting more than one simultaneous touch on the display surface.
- *Accelerometer* — It provides the acceleration of the device, usually on three axes XYZ, and can be used to determine the device orientation.

## 5.3 Hierarchical Image Structure Visualization

Many hierarchies, such as organization charts or directory structures, are too large to display in their entirety on a computer screen. The conventional display approach maps all the hierarchy into a region that is larger than the display and then uses scrolling to move around the region. This approach has the problem that the user can not see the relationship of the visible portion of a tree to the entire structure (without auxiliary views). It would be useful to be able to see the entire hierarchy while focusing on any particular part so that the relationship of parts to the whole can be seen and so that focus can be moved to other parts in a smooth and continuous way. Over the years there were several attempts to represent hierarchies using other than the typical 2D graph with nodes and edges. The main goal of researchers is to display large amounts of data that are in someway inter-related more effectively [YXC05]. In order to achieve this, there are several techniques developed over the years that focus on data per area ratio improvement [YXC05, BBB02, HYY06, Fur86], node and edge cost visualization [JR05], that employ distortion techniques to distinguish between nodes [JL95, Fur86, WR02] and navigation/searching focused visualizations [Cou07, HZ07].

### 5.3.1 Non-Distortion based Visualizations

The simplest technique to visualize a tree on a two dimensional surface optimizing the amount of space available is by using a radial layout. RadialMobile [HYY06] applies this technique to mobile devices. It uses circles centered in the origin of the screen to distribute nodes according to their level in the hierarchy. The root node is in the center while its descendants are drawn at an $n$ distance from the center. The distance $n$ corresponds to the nodes level in the hierarchy (Figure 5.1).

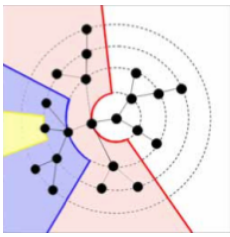


Figure 5.1: RadialMobile screenshot.

### 5.3.2 Generalized Distortion-based Visualizations

Many of distortion-based visualizations such as the document lens [Fur86] and the perspective wall [YXC05] can be applied to browsing trees using conventional two dimensional layout techniques. The problem is that there is no satisfactory conventional two dimensional layout of a large tree, because of its exponential growth. If leaf nodes are to be given adequate spacing, then nodes near the root must be placed very far apart, obscuring the high level tree structure and leaving no good way to display the context of the entire tree. The Cone Tree [WR02] modifies the above approach by embedding the tree in a three dimensional space (Figure 5.2).

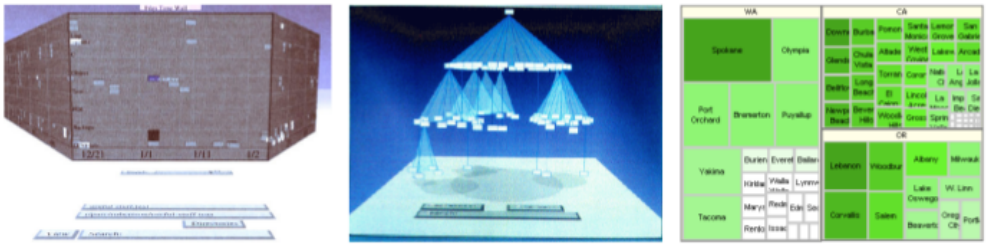


Figure 5.2: Perspective Wall, ConeTree and TreeMap.

The embedding of the tree has joints that can be rotated to bring different parts of the tree into focus. This requires expensive 3D animation support. Furthermore, trees with more than approximately 1000 nodes are difficult to manipulate. Another novel tree browsing technique is TreeMaps [BBB02] (Figure 5.2) which allocates the entire space of a display area to the nodes of the tree, by dividing the space of a node among itself and its descendants, according to properties of the node. The space allocated to each node is then filled according to the same or other properties of the node. This technique utilizes space eficiently and can be used to look for values and patterns amongst a large collection of values which agglomerate hierarchically, however it tends to obscure the hierarchical structure of the values and provides no way of focusing on one part of a hierarchy without losing the context.

### 5.3.3 Hierarchical Tree Visualization

In our case study, a hierarchical tree visualization was used. The hierarchy browser prunes or filters the tree to allow selective display of portions of the tree that the user has indicated. The degree of interest can then be used to selectively display the nodes of interest and their local context. Though this technique is quite powerful, it still does not provide a solution to the problem of displaying the entire tree.

## 5.4 Methodology

The goal of this methodology section is to summarize the processes and steps that led from image structure organization (chapter 4) to the visualization software prototype. The visualization process has several steps involved, to transform the hierarchical image structure into an interactive visualization. Each of the visualization steps has specific technical challenges that need to be taken into account before the solution design. These steps are:

- *Analysis* — The process of acquiring and parsing the data structures received. It needs to convert the data into an internal representation of the software module.
- *Organization* — Reduces the dimensionality of the resulting data set by filtering and mining according to the views selected.
- *Visualization* — A graphical user interface representing a visualization of the selected content. It allows interaction over the hierarchical image tree, to select portions of the tree or filter images, using query by image example features. The visualization and organization steps work in parallel. The user actions regarding data manipulation (e.g. removing images from database) are received by the organization step that filters and selects images being visualized.

## 5.5 Interface Design

The interface design is an exercise that tries to organize the requirements of Section 1.3 into a set of features.

### 5.5.1 Proposed "Cube Tree Visualization" layout for Mobile Devices

The essence of the layout of the proposed image collection produced in Section 4.4 is a simple graph structure, but presented in a way that simplifies navigation. First, in order to understand more deeply the proposed layout, consider the image collection scheme (Figure 5.3) representing the image structure produced in Section 4.4.

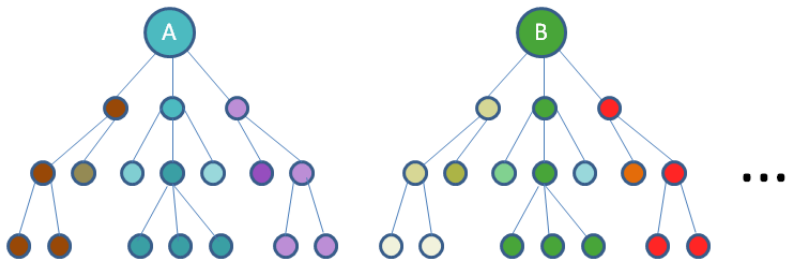



Figure 5.3: Images represented as colors in a hierarchical structure scheme.

Each tree represents one of calculated $k$ groups (clusters) with one root image as the group representant (the centroid). Each tree has its own subtrees, meaning that it possesses inner subgroups. Thus, making a top-down tree search means that we are jumping from general images to more concrete ones. As mobile devices possess small displays, we are only allowed to see small ammounts of images at a time, requiring a focus of the structure. First we need to display all the image roots at once as a simple image wall. To navigate through the structure, supose we select the image $A$ to focus the first cluster. The following layout can be as follows: the centroid image (the root) is positioned at the center and the next level images of the tree are positioned around it as shown in Figure 5.4. Selecting one surrounding image, we make a simple top-down navigation where the selected image becomes the new layout centroid and its inner-centroids surround it (Figure 5.4).

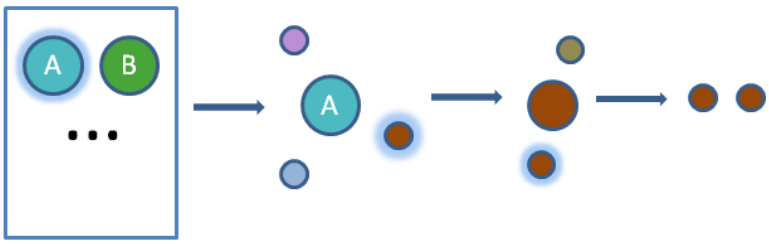



Figure 5.4: Image top-down navigation in the hierarchical structure scheme.

Top-down navigation is solved through this kind of layout display, but to make bottom-up navigation and *side* navigation (explained briefly) we need to do a small adjustment. Imagine a cube as the main navigation controller. The cube replaces the centroid image position and displays it as its top face. In order to perform bottom-up navigation, the cube simply rotates vertically. With this kind of mechanism we can perform *side* navigations by rotating the cube horizontally. This navigation is better explained by examining the Figure 5.5.

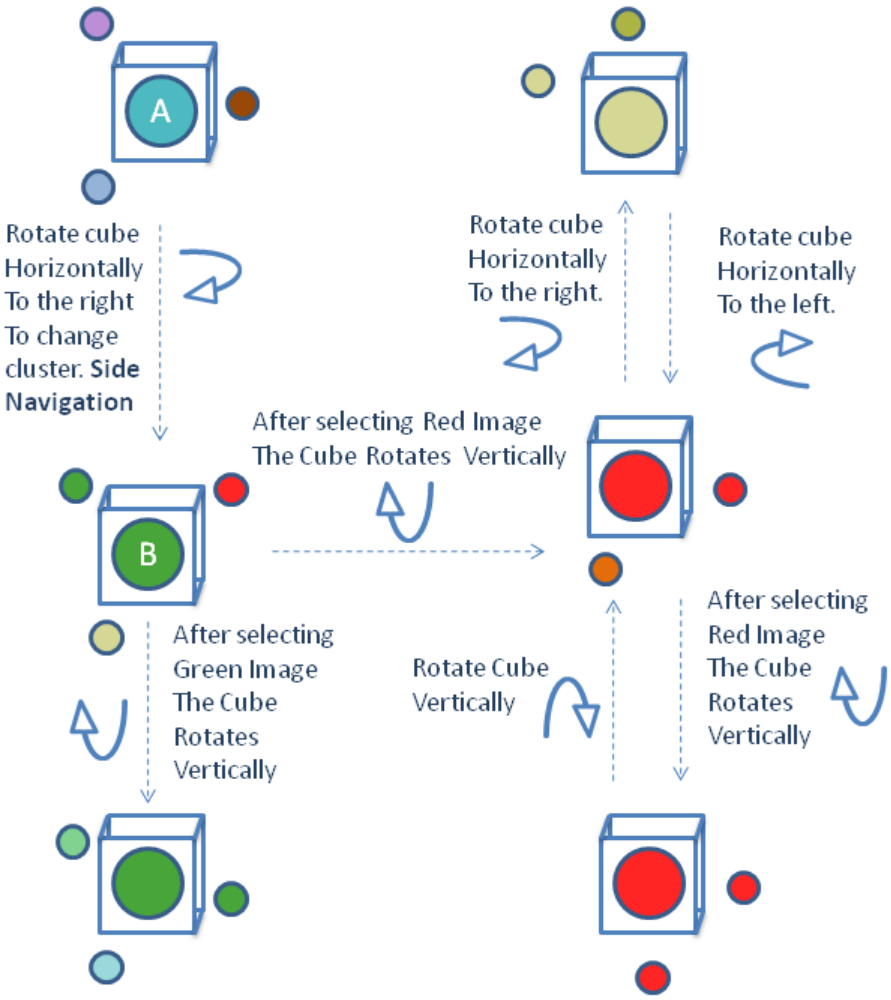



Figure 5.5: Image navigation using cube to perform top-down, bottom-up and side navigations.

### 5.5.2 Application prototype using proposed "Cube Tree Visualization layout"

The application prototype was thought and developed exclusively as a showcase of the visualization module features. They were built using the experimental results of Section 4.4, Figures 5.6 and 5.7.

Figure 5.6: Screenshots of the application prototype: simple layout and cube vertical rotation

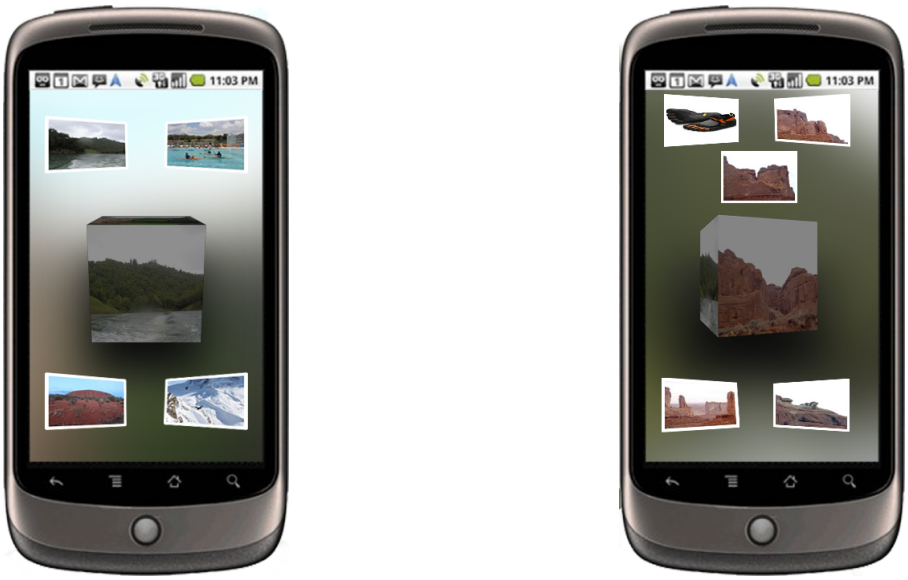



Figure 5.7: Screenshots of the application prototype: simple layout and cube horizontal rotation

## 5.6 Summary

Adapting computer graphics techniques to mobile devices poses several technical and usage limitations problems. Due to its characteristics, application developed for desktop computers do not scale to mobile devices. As mobile devices must remain compact in size to be practical, some of these limitations are unlikely to disappear. Using the results of Section 4.4, user image collection possesses a hierarchical tree structure. Thus, to browse images, one must perform tree navigations. The proposed browser filters the tree to allow selective display of user indicated tree portions. The mobile application prototye uses the proposed Cube Tree Visualization layout, supposed to be used a lightweight application for browsing the user's photos in a intuitive way.

# Chapter 6

# Conclusions and future work

Although user personal images are not as diverse as web images, their domain was also classified as *broad*, with high variability and unpredictability for the same underlying semantic concepts, being an important distinction used for the system design. The system conception took into account the different types of user clarity about what the user wants, browsing for pictures with no clear end-goal as well as being very clear about what he/she is searching for in the system. The image scope consists on a personal collection generally small in size, accessible primarily to its owner, and usually stored on a local storage media. In searching for images, the system support various level of queries modalities, such as the use of simple keywords, a more complex query-by-content feature when in the absence of reliable meta-data and a composite way involving one or more modalities of querying the system.

For image content-based system queries, the usual raw pixel representation of an image was shifted to a feature-level one, based on overall image average color for each color channel and on multi-resolution Wavelet decomposition for edges information extraction. The color information and the coefficients of the decomposition were distilled into small "signatures" for each image. These signatures were then used in image querying metrics by analyzing average luminance color channel discrepancies and how many significant wavelet coefficients of one image has in common with potential targets. The use of this technique required a simple, little signature storage overhead, and a fast algorithm to be performed on a normal (3500 plus images) user image collection. Sensitive to edge variances of the query and target images, producing sometimes unexpected results, the use of wavelet decompositions for content-based image search does not always produce robust results in order to be fast. Thus, its use used in synergy with text-based techniques.

The alternative of requiring to systems explicit query formulations, involving the measure of similarities of image features, was done through browsing, defined here as the exploration of spaces through a sequence of local decisions or navigational choices. Unsupervised clustering techniques were a natural fit when handling this kind of unstructured

personal image repositories. One of the greatest difficulties of a browsing approach was to identify a structure that was conducive to effective search in the sense that they support fast navigation, provide a meaningful neighbourhood for choosing a browsing path and allow users to position themselves in an area of interest. Starting by using an adaptation of the k-means algorithm, images were arranged into clusters with content-based and metadata similarities. Next, a static hierarchical structure was created, by dividing recursively the image database, using the same algorithm. The image database were therefore represented as a tree, using representative images (general images) as top nodes and more concrete images as descending nodes. The user has a first contact with the system through 10 representative images (the first calculated centroids). Each one represents a path from a general topic to a more concrete subtopic(s). Using connections between centroids and sub-centroids, one can “take a look” and have an idea of what they bring and then continue searching. Through this kind of top-down tree navigation, the image navigation may be somewhat exploratory in the beginning with a difference that subsequent refined paths are expected to increase the clarity of what the user wants from the system.

A visualizatoin software prototype was created in order to navigate through the built image structure organization through mobile devices. A new visualization layout called "Cube Tree Visualization" was proposed. It consists of navigating the built hierarchical structure through cube rotations, performing tree top-down, bottom-up and side navigations aiming for an optimal ratio between displayed content and screen area usage.

## 6.1 Image Distance Metric Improvements

### 6.1.1 Possible content-based scoring improvement

Although the wavelet-based image distance metric provides high performance with regards to speed, it lacks perfect results. One point that can be improved is a better use of color information. Currently the overall luminance intensity is analysed for each color channel. This information belongs to the scoring expression (3.13). Replacing the image luminance intensity difference with a more robust color histogram analysis would theoretically improve the results. This improvement will likely represent a calculation overhead and therefore this trade-off should be considered.

### 6.1.2 Metadata related improvements

Currently, we use image metadata information to boost image distance scores for a given number of common keywords. This method assumes that every keyword has the same importance value, possibly leading to some unharmonious results. Take for example the keywords "2010" and "car", since the former is more general and perhaps more used than the latter, it should have less impact in image discrimination. Thus, in order to

improve results, an image collection tag-cloud based exclusively on the use frequency should be created. Further analysis on high semantic density (high frequency) and on more discriminatory keywords (low frequency) should redefine keyword discrimination power.

## 6.2 Clustering Improvements

K-Means clustering uses an iterative procedure that converges to local minima. This local minimum is highly sensitive to the selected initial partition for the K-Means clustering. Currently this initial partition is done randomly. Initial starting points generated randomly often make the clustering results reach local optima. The better clustering results can be achieved after computing more iterations. However, it is difficult to decide the computation limit, which can give the better result. To overcome this difficulty, new approaches to initial centroid positioning should be studied. For example, a recursive method for initializing the means by running $K$ clustering problems is discussed in [DH73]. A variant of this method consists of taking the entire data and then randomly perturbing it $K$ times [KA04]. Bradley and Fayyad (1998) proposed an algorithm that refines initial points by analyzing distribution of the data and probability of data density [BF98]. In [JP99] an empirical comparison was presented for four initialization methods of the K-means algorithm and the authors concluded that the random and Kaufman initialization method outperformed the other two methods with respect to the effectiveness and the robustness of the K-means algorithm. Once again, further analysis should be taken regarding the result improvement tradeoff with the calculation overhead.

## 6.3 Future Analysis and Evaluation of Visualization Process

The evaluation of a prototype is an important step of any software project. In this project it is particularly important to evaluate the $UserInterface$ with a group of potential users. This is the way to obtain results about the system's usability and overall usefulness. The formal process is described in this section as a guide for future work.

### 6.3.1 Information Visualization Process

The techniques and criteria described in this section are focused on the specific characteristics of the information visualization process.

### 6.3.2 Criteria

The criteria [Pla04] that can be addressed in an evaluation process are:

- *Functionality* — It consists on comparing the user requirements with the features provided by the system.
- *Effectiveness* — It is focused on the visualization purposes and determining if it provides new insights compared to other previous solutions.
- *Efficiency* — Analyses if the visualization helps the users achieve a better performance with their tasks using it.
- *Usability* — Tests how the users interact with the system and of the information provided is clear and intelligible.
- *Usefulness* — Analyses the visualization within its usage context. It determines if the vi- sualization is useful and how users may benefit from it.

### 6.3.3 Techniques

The evaluation process has several possible techniques that can be grouped into two main categories, analytics and empiric evaluations [TZ06]:

- *Analytic evaluation* — Based on formal analysis models and conducted by experts.
  - *Heuristic* — Experts act as less experienced users and describe the potential problems they foresee for such users.
  - *Cognitive walkthroughs* — Experts walk through a specific task using a prototype.
- *Empirical evaluation* — Realized through experiments with user test subjects.
  - *Quantitative studies* — Consist of an analysis to determinate hypotheses tested through direct measurements. Usually done within controlled experiments, where an evaluator can manipulate a number of factors associated with interface design and study their effects on various aspects of user performance.
  - *Qualitative studies* — Involves the analysis of qualitative data, which may be ob- tained through questionnaires, interviews and observing users using the sys- tem, to understand and explain social phenomena. Usually done with a focus group of selected individuals.

### 6.3.4 Metrics

The metrics used for HCI (Human Computer Interaction) evaluation are not appropriate for assessing information visual- ization systems. Some of the HCI metrics used in this field are:

- Time required to learn the system.
- Time required to achieve a goal.
- Error rates.
- Retention of the use the interface over the time.

### 6.3.5 Process

The developed visualization module aims to demonstrate that:

- The Cube Tree Visualization method to display content on a mobile device is useful.
- Manipulating and interacting with media content is effective by helping users to discover forgotten or unfamiliar images.
- The interface and the visualizations provided are well integrated and make the content easy to navigate.

In order to test each of these items, a small user study should be designed to touches upon each of them. It starts with a focus group of 5 users unfamiliar with the visualization and application goals. It takes about 30 minutes. During this period the visualization is explained briefly, the user has the chance to play with the visualization and execute specific tasks. The results are evaluated through a qualitative questionnaire about the user experience. Each 30-minutes evaluation session follows this structure:

1. (5 min) Introduction to the visualization module. The explanation is complemented with a brief demo where the main interaction techniques are shown.
2. (10 min) The subject tries to use the visualization freely in order to understand the main action-effect behaviors of the application.
3. (10 min) The user is asked to execute specific tasks:
    - View the photos of a specific category.
    - Filter the photo collection through a selection of a photo and its use with query by image example featutre.
    - View the photos that are related to the tag “car”.
4. (5 min) Fill the questionnaire.

# References


[AWMSJ00] Simone Santini Amarnath Gupta Arnold W. M. Smeulders, Marcel Worring and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[BBB02] Martin Wattenberg Benjamin B. Bederson, Ben Shneiderman. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Transactions on Graphics*, 21(4):833–854, 2002.

[BBB06] Amy K. Karlson Benjamin B. Bederson. Understanding single-handed mobile device interaction. *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*, 2006.

[BF98] Paul S. Bradley and Usama M. Fayyad. Refining initial points for K-Means clustering. In *Proceedings of the International Conference on Machine Learning*, pages 91–99, 1998.

[BH64] G.H. Ball and D.J. Hall. Some fundamental concepts and synthesis procedures for pattern recognition preprocessors. In *Proceedings of the International Conference on Microwaves, Circuit Theory, and Information Theory*, 1964.

[CEJ95] David H. Salesin Charles E. Jacobs, Adam Finkelstein. Fast multiresolution image querying. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, pages 277–286, 1995.

[Chi04] Luca Chittaro. HCI aspects of mobile devices and services. *Personal Ubiquitous Computing*, 8(2):69–70, 2004.

[Chi06] Luca Chittaro. Visualizing information on mobile devices. *Computer*, 39(3):40–45, 2006.

[Cou07] Rémi Coulom. Treemaps for search-tree visualization. In *Proceedings of the Seventh Computer Olympiad Computer-Games Workshop*, 2007.

[Cox95] K. Cox. *Searching through browsing.* PhD thesis, 1995.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classication and Scene Analysis*. 1973.

[DJ06] Z. Zhuang W. Weiss M. Friedenberg J. Wang J. Li D. Joshi, R. Datta. PARAgrab: A comprehensive architecture for web image management and multimodal querying. In *Proceedings of the International Conference on Very Large Data Bases*, pages 1163–1166, 2006.

[EJS95] David H. Salesin Eric J. Stollnitz, Tony D. DeRose. Wavelets for computer grahics: A primer, part i. *IEEE Computer Graphics and Aplications*, 15(3):76–84, May 1995.

[For01] D. Forsyth. Benchmarks for storage and retrieval in multimedia databases. *Storage and Retrieval for Media Databases*, 2001.

[Fur86] G. W. Furnas. Generalized fisheye views. *ACM SIGCHI Bulletin*, 17(4):16–23, 1986.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[Hee08] Daniel Heesch. A survey of browsing models for content based image retrieval. *Multimedia Tools and Applications*, 40(2):261–284, November 2008.

[HYY06] Suh Hyun Cheon. Hee Yong Yoo. Visualization by information type on mobile device. In *In Proceedings of the Asia-Pacific Symposium on Information Visualisation*, pages 143–146, 2006.

[HZ07] Jie Hao and Kang Zhang. A mobile interface for hierarchical information visualization and navigation. In *IEEE International Symposium on Consumer Electronics*, pages 1–7, 2007.

[JD88] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[JL95] Peter Pirolli John Lamping, Ramana Rao. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 401–408, 1995.

[JP99] P. Larrañaga J.M Peña, J.A Lozano. An empirical comparison of four initialization methods for the K-Means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, October 1999.

[JR05] S.P. Mudur Juergen Rilling. 3D visualization techniques to support slicing-based program comprehension. *Computers and Graphics*, 29(3):311–329, 2005.

[KA04] Shehroz S. Khan and Amir Ahmadb. Cluster center initialization algorithm for K-Means clustering. *Pattern Recognition Letters*, 25(11):1293–1302, August 2004.

[KAS98] Sanjay Ranka Khaled Alsabti and Vineet Singh. An efficient K-Means clustering algorithm. In *Proceedings of the First Workshop High Performance Data Mining*, 1998.

[kn:a] K-means and hierarchical clustering tutorial. www.autonlab.org/tutorials/kmeans09.pdf.

[kn:b] Mathworks wavelet toolbox. http://www.mathworks.com.

[kn:c] The wavelet digest. http://www.wavelet.org.

[Koh89] T. Kohonen. *Self-Organization and Associative Memory, third ed.* 1989.

[KR90] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. 1990.

[ME95] X. Xu M. Ester, H. Kriegel. A database interface for clustering in large spatial databases. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 1995.

[NH94] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the International Conference on Very Large Data Bases*, pages 144–155, 1994.

[PB98] C. Reina P.S. Bradley, U. Fayyad. Scaling clustering algorithms to large databases. In *Proceedings of the International Conference on Knowledge discovery and Data Mining*, 1998.

[Pla04] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 109–116, 2004.

[RD08] Jia Li James Z.Wang Ritendra Datta, Dhiraj Joshi. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, April 2008.

[RK97] J. Shepherd R. Kurniawati, J. Jin. Techniques for supporting efficient content-based retrieval in multimedia databases. *Australian Computer Journal*, 29:122–130, 1997.

[RZ05] A. Shashua R. Zass. A unifying approach to hard and probabilistic clustering. In *Proceedings of the International Conference on Computer Vision*, 2005.

[SBP02] J. Malik S. Belongie and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.

[SS01] Linda G. Shapiro and George C. Stockman. *Computer Vision*. 2001.

[TZ97] M. Livny T. Zhang, R. Ramakrishnan. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, June 1997.

[TZ06] Petra Neumann Mark S. Hancock Sheelagh Carpendale Torre Zuk, Lothar Schlesier. Heuristics for information visualization evaluation. In *Proceedings of the AVI Workshop on Beyond time and errors: novel evaluation methods for information visualization*, pages 1–6, 2006.

[WR02] Jorg A. Walter and Helge Ritter. On interactive visualization of high-dimensional data using the hyperbolic plane. In *Proceedings of the International Conference on Knowledge discovery and Data Mining*, pages 123–132, New York, 2002.

REFERENCES

[YXC05] Homa Atabakhsh Yang Xiang, Michael Chau and Hsinchun Chen. Visualizing criminal relationships: comparison of a hyperbolic tree and a hierarchical list. *Decision Support Systems*, 41(1):69–83, November 2005.