

UNIVERSITÄT KAISERSLAUTERN

**Evaluation of Parameter Influence and
Dimensionality Reduction for CBIR
Systems using Local Features, TF-IDF
and Inverted Files**

by

Sebastian Palacio

A thesis submitted in partial fulfillment for the
Bachelor degree

in the
Department of computer science

April 2010

Declaration of Authorship

I, SEBASTIAN PALACIO, declare that this thesis titled, 'USING TF-IDF AS AN IMPROVEMENT METHOD FOR IMAGE RETRIEVAL ENGINES' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a bachelor degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

“I know that you believe you understand what you think I said, but I’m not sure you realize that what you heard is not what I meant.”

Robert McCloskey

UNIVERSITÄT KAISERSLAUTERN

Abstract

Department of computer science

Bachelor degree

by Sebastian Palacio

Although there has been extensive work on content-based image retrieval, there has been a lack of benchmarks and in-depth analysis that quantify the impact of widely used techniques such as PCA, inverted files or TF-IDF in terms of quality (i.g. similarity) and speed gain. This thesis describes benchmarks that measure the variations in performance experienced by a CBIR system when such techniques are included. This work also studies the behaviour of a CBIR engine when PCA is being applied on early stages of the feature extraction procedure in order to preserve as much information as possible and keep its structure as intact as possible while addressing the curse of dimensionality and computational speed issues. Major conclusions that can be drawn from the experimental results are that TF-IDF provides a low-cost and very stable boost in performance and that the use of PCA does not necessarily imply a drop in performance but it can actually help to reduce the influence of noise in the data and therefore improving quality.

Acknowledgements

Whenever I reach a milestone in life, I realize how little my effort would have meant without the support of so many people. I'd like to start by thanking my mother Angela and my father Luis whose unconditional support is the foundation of all that I've achieved so far. In the academic field, I'd like to thank all the good teachers I've had the honor to meet for they have reaffirmed my passion for what I do; Graciela Villegas, Juan Guillermo Lalinde, Jose Luis Montoya, Wolfgang Effelsberg, Gregor Schiele, Richard Sueselbeck and Carlos Mario Jaramillo are just some of them. I also wish to thank the bad teachers for they gave me the strength and the courage to carry on, even when I was not feeling comfortable with the conditions. I would like to thank AIESEC for giving me the opportunity to expand my frontiers and providing me with the means to find the way to Kaiserslautern and to the DFKI. I wish to thank my friends for the extracurricular company, and for keeping me sane. I'd like to thank Sebastián Castillo for showing me that no problem is too hard to not give it a try, to Daniel Acosta for always being there and Carlos Bercianos for feeding me and taking care of me when I forget to do it myself. I'd like to give a special mention to Prof. Thomas Breuel and all the people at the (former) IUPR group at the DFKI-KL for making me feel welcome among them, for sharing their knowledge, teaching me and for giving me the opportunity to be a part of the group; to Christian Schulze, thank you for believing in my skills and for encouraging and supporting me since I arrived in Kaiserslautern. Finally I also want to make a special mention to Prof. Klaus Madlener for his tremendous help in getting me enrolled at the University of Kaiserslautern.

To all of you, thank you very much.

Contents

Declaration of Authorship	ii
Abstract	vi
Acknowledgements	viii
List of Figures	xii
List of Tables	xiii
Abbreviations	xiv
1 Introduction	1
1.1 Motivation	1
1.1.1 State of the Art of Image Retrieval	1
1.1.2 Keeping the Balance: Scalability vs. Reliability	2
1.2 Motivation for using TF-IDF and Patch-Based Approaches	3
2 Background Theory	5
2.1 Image Features	5
2.2 Local Features	7
2.3 Invariant Features	7
2.4 Distance Measures	8
2.5 Theory and Algorithms	9
2.5.1 Comparison with Text Retrieval Methods	11
2.5.2 Inverted File	15
2.5.3 Term Frequency - Inverse Document Frequency	15
2.5.4 Principal Component Analysis	16
2.6 Previous work	17
3 Materials and Methods	21
3.1 Dataset: INRIA’s Holiday data set	21
3.2 MoViMoS	21
3.3 Baseline Establishment	22

4	Results	25
4.1	Baseline	25
4.2	Optimizing the TF-IDF Measure	25
4.3	Getting the Best of the Two Worlds	26
5	Discussion	29
5.1	Baseline	29
5.2	Optimizing TF-IDF	30
5.3	Combining Analysis	33
6	Conclusions	35
7	Further Work	37
A	Definitions and other results	39
A.1	Mean Average Precision (mAP)	39
A.2	Distance Measure Comparison	40
	Bibliography	41

List of Figures

1.1	Basic CBIR Procedure	2
1.2	Basic CBIR Querying Procedure	2
1.3	Similarity comparison	4
2.1	Color histogram	5
2.2	Local color histogram	6
2.3	Texture map	6
2.4	Blob and Edge detection	7
2.5	Local Features	8
2.6	Distances	9
2.7	Keypoint detection	11
2.8	Difference of Gaussians	11
2.9	Keypoint descriptor	12
2.10	Text retrieval	13
2.11	Codebook generation	14
2.12	CodebookMatching	14
2.13	Inverted file	15
2.14	Patch-based method and TF-IDF	16
2.15	PCA	17
3.1	Holiday DS	22
3.2	MoViMoS baseline system	23
4.1	Baseline Results	26
4.2	TF-IDF experiments	26
4.3	Combination experiments	27
5.1	Noisy patches	30
5.2	Image descriptor structure	31
5.3	Sparse clustering	32
5.4	PCA clustering	33

¹Image of Lena has been taken from: <http://www.cs.cmu.edu/~chuck/lennapg/>

²All other external images have been taken from www.freepixels.com,
www.freedigitalphotos.net and www.designpacks.com

List of Tables

A.1 Distance comparison	40
-----------------------------------	----

Abbreviations

TF-IDF	T erm F requency - I nverse D ocument F requency
PCA	P rincipal C omponent A naly S is
CBIR	C ontent- B ased I mage R etrieval
IF	I nverted F ile (or I nverted I ndex)
visword	V isual W ord
SIFT	S cale- I nvariant F eature T ransform
SURF	S peeded- U p R obust F eature
NN	N earest N eighbor

Chapter 1

Introduction

In this chapter, a brief overview of Content-Based Image Retrieval, along with the issues and challenges that this field presents to science, is going to be conducted as well as the description of the contributions this thesis makes.

1.1 Motivation

The field of Content-Based Image Retrieval (CBIR) is one of the most interesting and captivating research areas in computer science now days. It has a lot of potential in a large number of scenarios and it seems very promising in terms of the results researchers are obtaining. This work is going to focus on some approaches and techniques used in this field for which detailed analysis have been missing, and combine them in a way so that the benefits they provide can be maximized, resulting in a more robust and effective system due to the increase in quality and tolerance to noise.

The idea is to analyse the way CBIR systems are currently being built, the methods that are being combined and the order in which they are implemented. Then a test scenario is proposed based on the results obtained by the other CBIR engines and then the tested system is going to be slightly altered to see if that produces a variation on quality measures, namely time consumption and accuracy of the results being displayed to the end user.

1.1.1 State of the Art of Image Retrieval

In its very essence, the procedures followed by a CBIR engine can be described by a generic method, regardless of their particular algorithms and techniques. [Figure 1.1](#) illustrates the basic procedure to build a CBIR engine. It simply scans the image describing certain characteristics or *features* and then it represents them using efficient and meaningful structures. This structures are then stored together in a database so that they can be retrieved later.

Once the database is loaded with the feature representation of the processed images, it is possible to ask or *query* the system to get the most similar images it has indexed using an image that is not contained in the database. The system will then scan the image

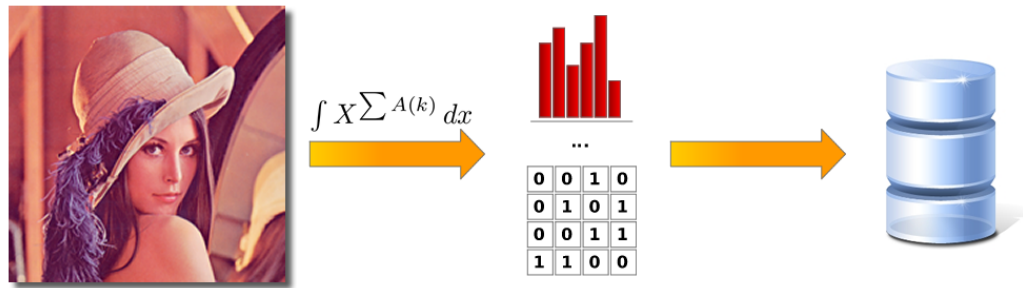


FIGURE 1.1: General procedure used by CBIR engines to build their DB. Properties are extracted from an image, represented in a standard way and then stored in a database.

that served as a query, extract the same kind of features as from the reference images and compare them separately. The images with the lowest distance are considered to be the most similar images (also known to as the *hits*).

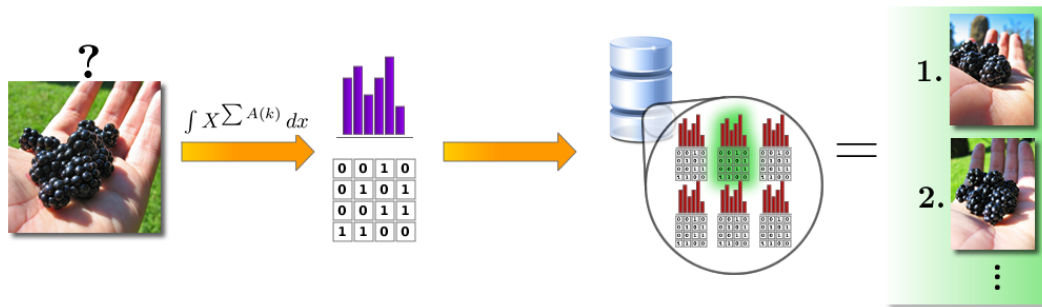


FIGURE 1.2: Querying procedure once the DB is loaded. A query image is processed the same way as the images in the database. Then the data from the query image is compared against the one of the images in the database and the results are sorted according to the similarity metric in use.

Now the most interesting and critical part of a CBIR engine is the kind of features that are going to be extracted for comparison (and the method by which this features are going to be extracted and stored). There are countless approaches to analyse the data within an image, all of them offering certain kinds of information about color, texture or shapes. Most of these methods are treating the image as waves, as a collection of pixels, or as n-dimensional histograms (in the case of grey scale images for instance); there are also analog techniques that borrow some principles of some other domains such as the text domain and treat images in a similar way words and documents can be indexed. Finding the right features and the way they have to be combined together is of critical importance to achieve good retrieval results. More details about the different features, their structure and advantages are mentioned in [chapter 2](#).

1.1.2 Keeping the Balance: Scalability vs. Reliability

Given the size of today's online image repositories [1–4] (e.g. Flickr, Facebook, Photobucket, ImageShack) and web-users' surfing preferences regarding waiting times and

quality of the information, an ideal CBIR engine would be able to operate on a nearly unlimited body of images and throw optimal results in a very short time. The problem of space, time and quality becomes an issue once the DB grows large enough. To save space in the DB, there are a lot of well known image compression algorithms but since the DB also has to store the features that were extracted for each image, techniques have to be developed in order to compress or to dismiss parts of the additional data that is further describing every image.

Using a naive approach, the more images there are in the database, the more comparisons the system has to perform in order to find the most similar image (also known to as the *nearest neighbor* or simply the *NN*). In an ideal environment, it would be preferable to store as much information as possible about an image but comparing all this data would slow the system down and it eventually would become unusable because of the time it would take to perform a single query. Sometimes CBIR engines implement algorithms that use the notion of *approximate NN* [5, 6] which basically consist on dividing the information in batches so the list of possible NN is quickly being narrowed down; they also can establish a particular criteria to truncate the searching cycle and return the best NN found so far. This would make the system faster but it obviously implies that sometimes the optimal solution is not going to be retrieved (i.e. affects the system quality wise causing a drop in performance).

One aspect of CBIR systems that has always been rather difficult to measure is the quality of the results they retrieve. The notion of “similarity” in the image domain hasn’t been formally established yet and at the end, only the human judgement can determine if an image is similar enough to another image based on the aim of the search. Sometimes some people might disagree about whether two images are similar or not. [Figure 2.14](#) tries to illustrate three examples of such scenarios: the first one compares an apple in the grass with a green rectangle and a red circle; although their shapes are similar, the amount of colors and the textures do not match at all. The second one compares a field of roses against just three roses (which correspond to a zoomed version of the image above). The last one shows an image of an island and a picture of a horse. How similar they really are?

This last aspect is better known as the *semantic gap*. The general notion of the semantic gap refers to the difference between the representation of a particular object in two or more languages and the information that is being omitted when using one or the other language. In the image domain, emphasis can be done on very different visual properties that need to be parameterized (refer to [chapter 3](#) for some of this properties) in a way that they can be linked back together and reflect the interpretation that a human being can give to an image.

1.2 Motivation for using TF-IDF and Patch-Based Approaches

As mentioned in [subsection 1.1.1](#), the amount of features that can be extracted from an image is very numerous and it is almost impossible to put them all together. One of the most successful and wide-spread features for image representation is the patch-based approach which basically finds all the most “interesting” areas within an image and then describes their shape and texture. This method is considered to be very reliable because

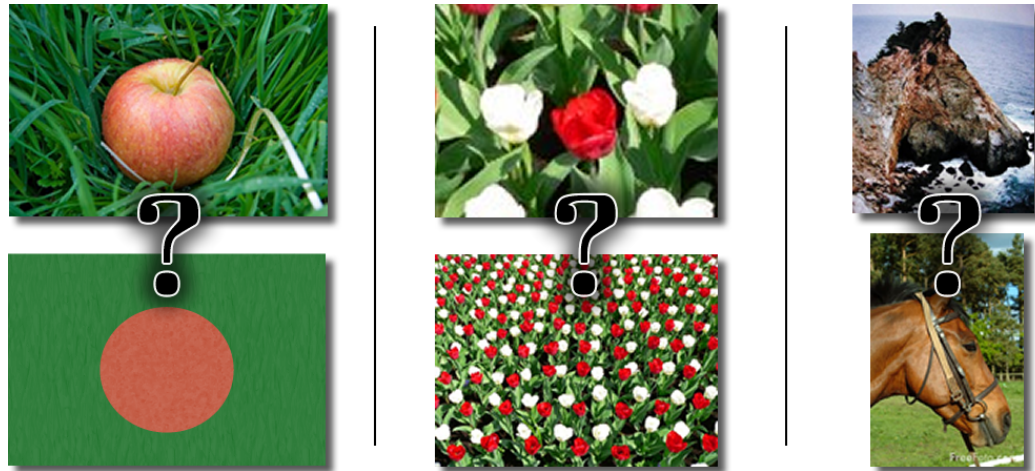


FIGURE 1.3: Sometimes not even humans can agree about the notion of similarity.

it can handle a wide range of different changes in the structure of the image (it doesn't take color into account). Rotation, scaling and *affine transformations* in general can be tolerated if within a certain range. It is normally being used as the base of all CBIR engines now days [7–13]. More details about how the patch-based approach works are giving in [chapter 2](#).

The TF-IDF scoring system is a method taken from the text domain to determine the relevance of certain words in a document. A basic example would be if two documents contain the word *doctor*; the first document is a medical report and the second one is a fairy tale. In the first one, the word *doctor* can appear multiple times but it won't give too much information about the actual content of the report whereas the occurrence of the same word within the second document would provide information that could be of relevance in classifying that particularly fairy tale (e.g. the existence of a character who is maybe a doctor).

This two techniques can be merged together by classifying the patches obtained by the patch-based approach according to some pre-defined vocabulary or *codebook*. Every occurrence of a particular vocabulary entry is then accumulated and a histogram of vocabulary entries is generated. Since the vocabulary is known in advance and it is restricted in the amount of *terms* it can hold, a TF-IDF score can be computed so that the terms in the vocabulary that aren't important enough can be dismissed or punished and the relevant ones are rewarded by getting a better score. This way, the patch-based approach gets a boost in performance by giving more or less importance to some parts of the descriptor it generates. Throughout the development of this thesis, this combination will show that performance is increased while keeping time and memory overhead to a minimum.

In this chapter, a generic description about CBIR has been provided, how it works and why is it interesting to try to further improve such systems and therefore, justifying the work presented on this thesis.

Chapter 2

Background Theory

In this chapter, there will be a general overview of some of the main techniques, methods and theories that underlie CBIR systems; the kind of information that can be extracted from an image and the way it is being represented in a computer. Finally there is a review of some previous work that has been done in the field of CBIR that closely relates to the ideas being developed in this thesis.

2.1 Image Features

An image can be viewed as more than a collection of separate pixels. There are countless relations that can be found among pixel values according to the scale, interpretation and even the notation being used for a particular image. Lets start by giving a very brief overview of the most commonly used elements that CBIR engines focus on while scanning an image.

Color

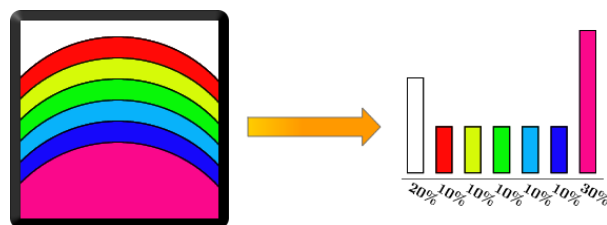


FIGURE 2.1: A global color histogram is created from the picture of a rainbow.

Color would be one of the most obvious things to focus on when analysing an image. On a global scale, an image can be represented as the occurrences of a color within the image, creating a histogram where every entry corresponds to a different color and the height of each entry would be the amount of pixels that have that particular color. [Figure 2.1](#) shows a simple example on an image of a rainbow and its corresponding global color histogram. Another approach to take advantage of color would be to segment the image and generate local histograms for each image segment as shown in [Figure 2.2](#).

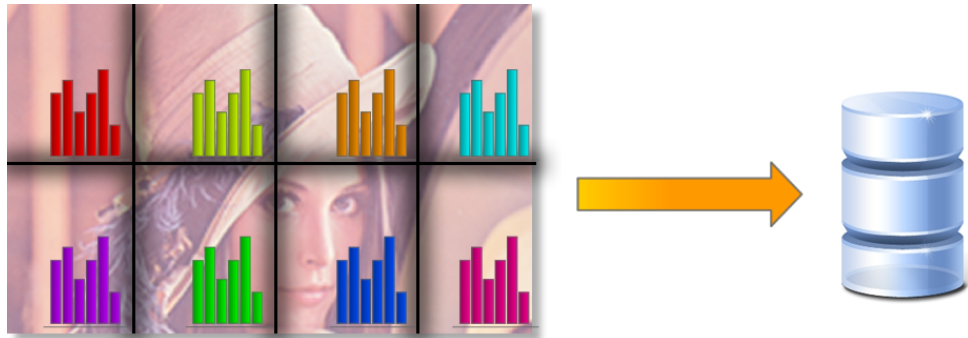


FIGURE 2.2: Several local color histograms can be build on segments of an image.

Texture

Almost all objects that are normally caught on images, contain certain local patterns. These different patterns occur due to the type and structure of material the objects are made of. The variations on the pixel intensities and the existence of patterns within those intensity variations are the ones that texture features are focusing on. This kind of “repetitive” change in certain parts of an image can be represented as gradient vectors and with those, texture can be indexed by creating a histogram just like the one used with the color features but this time, every bin will represent a certain kind of texture.

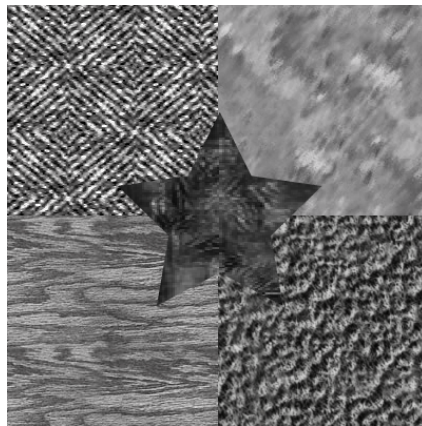


FIGURE 2.3: Although this is just a grey scale image, the shapes are clearly recognizable due to the difference in their texture.

Shapes

Shapes can be considered as a particular kind of texture. Since shapes also rely in the detection of edges, corners or boundaries, these methods look for strong intensity variations within a small area. This changes are then often represented as gradients and therefore the relation to texture.

Three of the main shape detector approaches are the *edge*, *blob* and *corner* detection. The edge detection is focused at finding strong variations in the intensities of an image; the corner detection works just as the edge detection but additionally, it looks for sudden changes on the direction of the edges; finally, blob detection operates on smooth images that can't be analysed by neither an edge nor a corner detector.



(a) Blob detection



(b) Edge detection

FIGURE 2.4: Examples of a blob detector (a) and an edge detector (b).

2.2 Local Features

As stated before, the local features have been one of the most successful methods that are being used in current CBIR systems. The idea of local features takes into account the relations between the data within a neighborhood. Properties such as geometry, color distribution or texture can be indexed to represent as much data from an image as needed. This way, a finer analysis can be done and more relevant information can be retrieved from a single image. Depending on the type of data being analysed in the image, local features provide a more robust way of collecting data. Figure 2.5 shows an example of two images that are completely different between each other but have very similar global color histograms. In this scenario, local features can give more information about the distribution of color among certain images and tell a potential system that the images are not as similar as they seem at first sight.

2.3 Invariant Features

With some frequency, images might contain information about three dimensional spaces. Everytime we take a picture of an object or a person, all the spacial information gets reduced or lost, making the recognition of 3D objects a much more challenging task. If a CBIR system wants to identify and retrieve images based on the appearance of a particular object, the system should be able to recognize the object even if the camera has a different angle, if it is at a different distance or even if partially occluded. For



FIGURE 2.5: In some scenarios, local features can provide more information to make a decision on whether two images are similar or not.

In this kind of scenario, invariant features are very useful because they provide robustness along a variety of changes in an image. The most interesting attributes of this kind of features are:

- Scale invariance
- Rotation invariance
- Tolerance to slight changes in the viewpoint
- Tolerance to noise and changes to the illumination

As explained in [14], invariant features can focus on the intensities of the images (i.e. processing a greyscale version of the image) as well as in their full color version. A very popular invariant feature developed by [15] makes emphasis on the texture of an image and it is able to maintain the consistency among transformed versions of the image (e.g. affine transformations). This algorithm, known as the *Scale-Invariant Feature Transform* (SIFT), exploits the gradients to describe an image as a set of local regions that are going to be considered as interesting within that image. Although there are many variations of it [16, 17], it's still not robust enough to work as the only image description technique but it does provide a baseline upon further enhancements can be built on.

In this work, features are going to be extracted from a data set of images and they will act as the base for further calculations in order to improve the behaviour of a CBIR system.

2.4 Distance Measures

So far, several methods to detect and represent image data have been described. Another aspect that influences how a CBIR engine sorts the result images of a query is the function that determines how similar the image descriptors in the database are. This is

a very critical part of the system since it is here where the semantic gap becomes more evident. Depending on the type of data being compared, an adequate distance measure (also known as *dissimilarity measure*) should be used so that the semantic gap can be shortened as much as possible. One aspect to consider when choosing the distance function is that it takes advantage of the structure of the data (e.g. histograms, scores, coordinates) and that it establishes a total order relation among the set of images in the database. So far, all features being described here can be represented as histograms and hence, the dissimilarity measures that are suitable for such data structures are (among others)[18]:

- Euclidean distance
- Manhattan distance
- Jensen-Shannon divergence
- Chi-squared distance (χ^2)

The Euclidean or L_2 distance just refers to the distance between two points in an euclidean space. The Manhattan distance (also known under other names such as the taxicab or L_1 norm) reproduces the distance a car would have to travel between a city of pure squared blocks, like a taxi in the city of Manhattan would have to and therefore the name. Another way to see it is the way a king can move in a chessboard. The Jensen-Shannon (also known to as *JSD*) is a distance measure for probability distributions. Departing from the fact that the image descriptors follow some kind of distribution, this function can be used to rank results in a CBIR system. Last, chi-squared distance is used as a correspondance measure to the distributions of the χ^2 with k degrees of freedom.

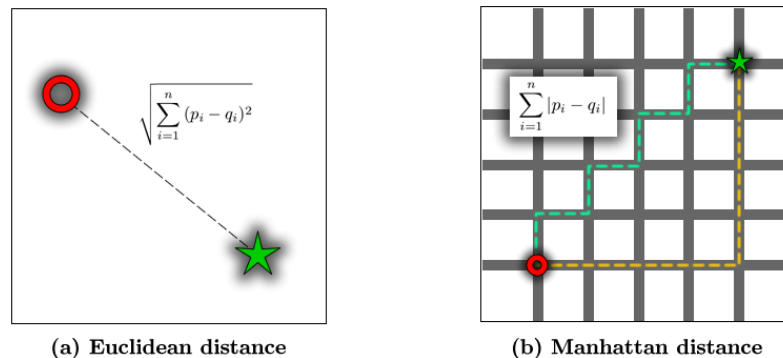


FIGURE 2.6: The dissimilarity measure used in a CBIR engine can cause an important impact on the results. The distance measure is the one that tries to breach the semantic gap.

2.5 Theory and Algorithms

In this section, there is a more detailed explanation about the algorithms to extract image features as well as an in-depth review of the theory underlying them and the structures they take advantage of.

Regardless of the implementation details and even some changes on the procedure, a patch-based feature extraction follows a simple line of execution that involves three steps:

image preparation: This is a pre-processing step to enhance the characteristics of the image. This processing stage can be skipped but in some cases it would greatly enhance both, time consumption during feature extraction and quality of the results in terms of similarity. In this stage the image can be scaled to a specific arbitrary resolution that meets the requirements of the system. Normally images will be resized to a resolution below the 1 megapixel threshold. This measure is rather empirical and it can vary depending on the requirements of the system. An image scaled to VGA or SVGA is still containing enough information about the texture to be described by the patch-based method and yet not too many pixels to be scanned. It's always good to keep the compromise between time and quality. That way the user doesn't get tired of waiting while the results being retrieved meet the expectations. Another adjustment that can be made prior to feature extraction is a light normalization and grey scale conversion. Even if this method is tolerant to illumination changes, the detection of the *interest points* will be faster and therefore improving time performance. This is because image normalization makes the noisy areas in the image more flat and hence the actual edges can be detected more easily. Some times, smooth areas such as a wall or a piece of paper can get noisy because of lossy compression algorithms or even by scaling procedures [19]. Three-band images can also be processed as proposed by [17] but it also adds overhead and it doesn't improve results in a significant way. Since there are many ways to embed color information to a patch-based search, considerations about the impact of color in such systems is out of the scope of this work.

interest point detection: During this stage, the system goes through the entire image looking for what is known to as *keypoints* or *interest points*. A point is considered to be interesting if it meets two basic conditions according to [15, 20, 21]:

- The surrounding area of this point presents a significant change in illumination (i.g. is in the middle of an area with a high contrast). Common areas for this are edges, corners or borders.
- This area is not prone to change too much in the presence of noise. To evaluate that, the potential area of the image is analyzed at different scales and then reaffirm that this is still being an interest area even when scaled. The scaling is done using a Gaussian distribution which blurs the image.

In order to find such points, [15] uses an iterative scaling and filtering method that tries to reproduce the way humans extract relevant information from images. It consists of passing the image through a couple of blurred images (using Gaussian filters) and then comparing them by subtracting their values. By doing this, the relevant parts of shapes and edges remain in the image whereas noisy parts and clutter get reduced. This process is done several times at different scales in order to assure the stability of the edges. An example of how such a filter looks like is shown in [Figure 2.8](#). In this example things like the face of the lady, her hat, the feathers and even parts of the texture of the hat are being kept. Glitter from the illumination and different shades of skin tones are no longer visible in the filtered image.



FIGURE 2.7: Example of a detected keypoint. It is represented using a vector that shows the direction on which the difference of intensities is pointing to and how strong that change is (norm of the vector)

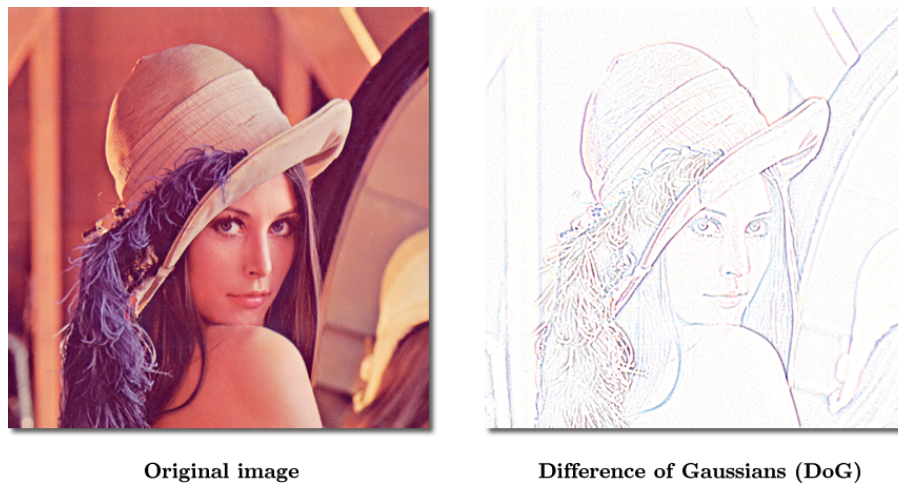


FIGURE 2.8: By applying a DoG, the image keeps the relevant information about the shapes and the edges that are going to serve as candidates to become interesting to the Interest Point Detector.

interest point descriptor: Once a series of keypoints has been detected in an image, the next step would be to represent those keypoints and their surrounding area in a meaningful and efficient way. There are several approaches [14–17] but for this work the *Scale-Invariant Feature Transform (SIFT)* descriptor was used. This SIFT descriptor takes the coordinates of every keypoint and takes an area of 16×16 pixels around that keypoint and then it assigns an intensity gradient vector to each pixel in that grid. The gradient is computed using a 4×4 subwindow. Then, all vectors are normalized and grouped together forming a matrix of gradient orientations (see Figure 2.9). Every vector on each 'star' is representing the accumulated impact of every single gradient that was first computed, to any of those main orientations. Finally, since their origin is the same, these 'stars' can be represented as an 8-bin histograms where each bin correspond to the magnitude of one of the main gradients.

2.5.1 Comparison with Text Retrieval Methods

Once the patch-based feature extraction has been applied on an image, there will be n descriptors with 128 dimensions, where n is the number of keypoints that have been detected in the image. Usually the quantity of keypoints ranges between 2000 and 5000

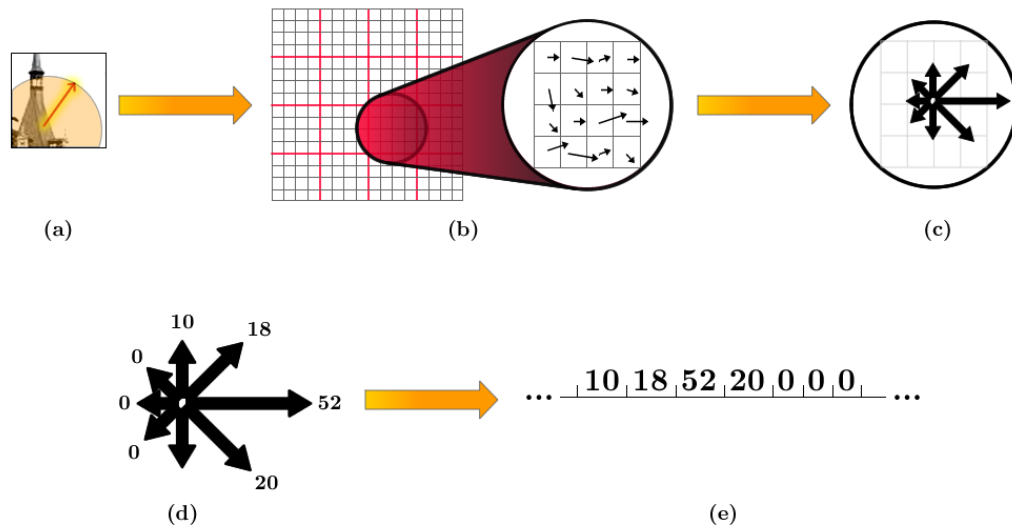


FIGURE 2.9: Process followed to represent a keypoint. (a) The keypoint is found. (b) Gradients on every pixel is computed within a 4x4 window. (c) Gradient normalization. (d) and (e) The magnitude of every gradient is put together to form the final numeric descriptor array. This process is then repeated for every keypoint of an image.

per image (when scaled to VGA¹) and therefore, the cost of storing all this information for each image would be even bigger than the cost of storing the image itself. For that reason, further processing should be done to compress these descriptors, still maintaining a meaningful structure for the comparison of the images.

Turning the attention into other fields where information retrieval has been successfully implemented, the text domain appears as one of the most reliable and studied areas where retrieval engines have succeeded (where Google presents itself as the most solid example of it). One of the most spreaded way to index text [13, 22] is as follows:

Given a set of documents, a word count is performed. A so called *stop list* prevents words that are very common (such as *the*, *of*, *from*, *to*, etc.) to reach the actual index. The word count for each document is then alphabetically stored as a list of numbers that correspond to the amount of occurrences of a particular word. Figure 2.10 shows an example with three documents that only contain one phrase each. If only the words *cook*, *dinner*, *kitchen* and *work* are to be considered for building the *descriptor* of each document, they will look like the ones in Figure 2.10 (b).

For this method to be useful and portable to the image domain, it will be necessary to have three elements:

¹Although the number of interest points obtained can be adjusted by manipulating a threshold, the default values on the original algorithm by [15] output these ranges. They can also be seen on the experiments carried out by [7]

Document 1: *When I went to the kitchen and cooked dinner, he was already on her way from work*

(a) **Document 2:** *They tried to work the way John told them but they didn't finish until dinner time.*

Document 3: *Cooking was no work for him. It was more a place to finish a good day.*

↓ ↓

(b)

	D1	D2	D3
<i>cook</i>	1	0	1
<i>dinner</i>	1	1	0
<i>kitchen</i>	1	0	0
<i>work</i>	1	1	1

FIGURE 2.10: (a) Three documents containing one phrase each. (b) Table of word count for three significant terms in the documents

- A set of terms that represent written words.
- A document-like structure where these terms can be grouped.
- A restricted and countable vocabulary of terms to look for.

At first sight, it can be assumed that keypoint descriptors can play the role of words and images are the structures to group them (i.g. the equivalent to a document). To build a vocabulary, it is first necessary to make some adjustments regarding the uniqueness of every descriptor. An example of this is being shown in Figure 2.10. As the words *cook* and *cooking* refer to the same action, they can be safely joined into the same root word *cook*. For this, a dictionary of standardized words is needed and the same concept can be applied to the image domain. The idea is to create a finite catalog of generic descriptors that reflect the *essence* of the descriptors. Such a structure is known to as the *codebook*.

To explain this concept, lets assume the descriptors are only 2-dimensional arrays:

Advantage can be taken of the fact that, since the numbers of the keypoints are describing a particular kind of texture, similar texture would be close together forming clusters (Figure 2.11 (a)). Then, an algorithm to find the centroids of these clusters is applied and from then on, every keypoint descriptor can be mapped to a cluster center and the descriptor will just count as an occurrence of that particular cluster (Figure 2.11 (d)).

This way, the information being extracted from an image, can be compressed and summarized in such a way that it is still meaningful and provides a system with enough data to be able to differentiate two different images.

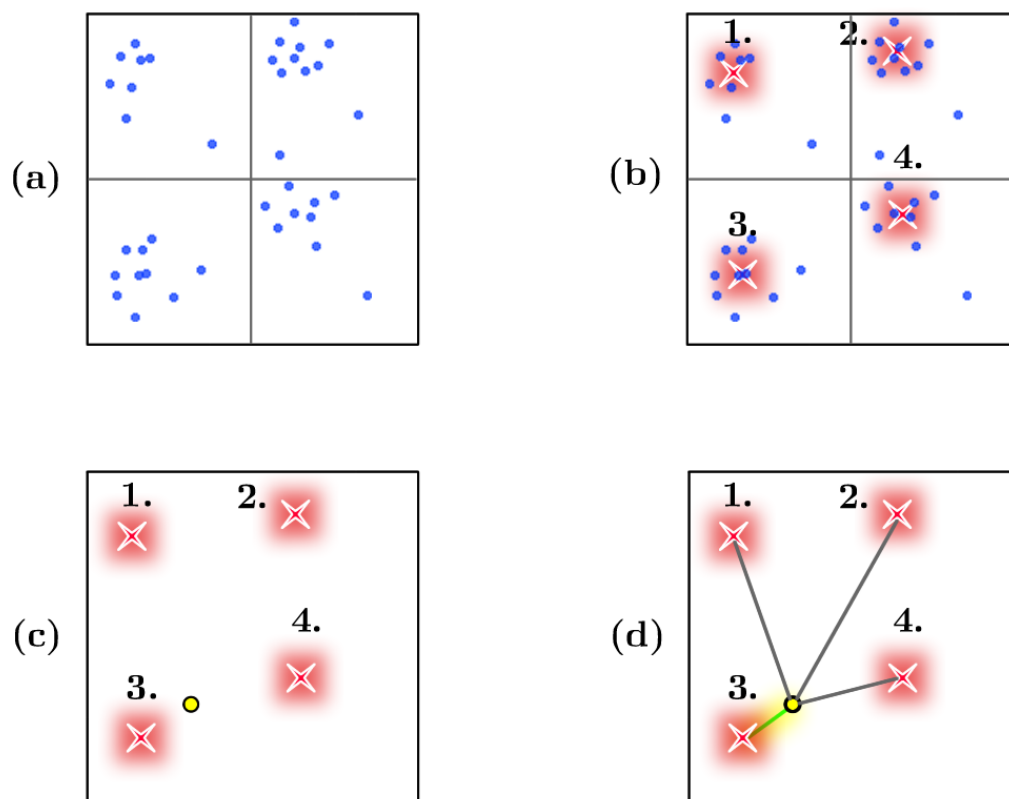


FIGURE 2.11: (a) 2D descriptors in space. (b) Center clusters are computed from the data. (c) A new point appears in space and needs to get a classification. (d) Distance is calculated to every cluster center and the closest one is chosen.

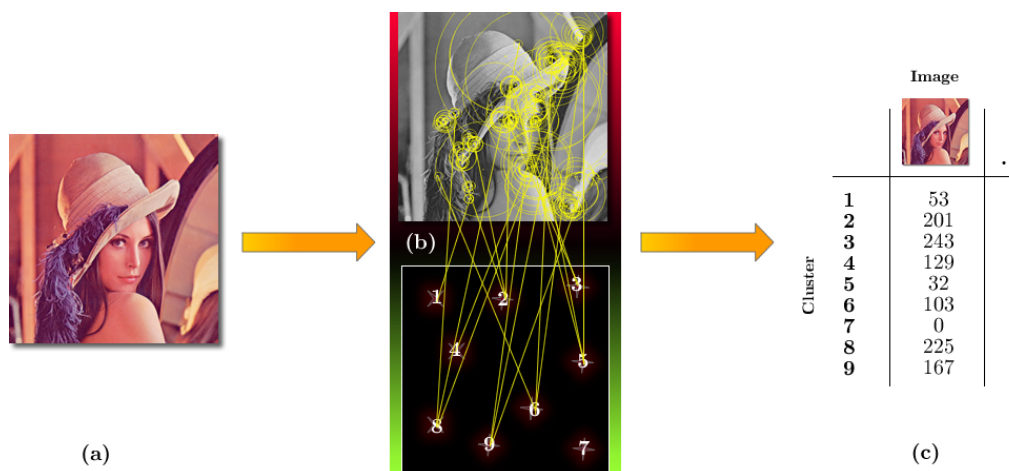


FIGURE 2.12: (a) The original image. (b) Keypoints are extracted and matched against the codebook. (c) The table with the histogram of visual words is generated and the image gets indexed.

2.5.2 Inverted File

Now that the descriptors of an image are being mapped to a generic codebook-based structure, each image is being represented as one numeric array of size s where s corresponds to the length of the codebook. Figure 2.12 is using a codebook of length 9 (i.e. there are 9 cluster centers in the complete feature space). For this example, this might work but given the diversity of the keypoint descriptors generated by the patch-based feature extraction, the codebooks need to grow much more in order for the clusters to be meaningful enough. In chapter 3 and chapter 4 it's being shown how the codebook size influences the retrieval performance.

The problem that arises when the codebook becomes too large is, again, that of the time and space consumption and hence it is necessary to address this issue. Fortunately this phenomena is also present in the text domain. Since even pocket dictionaries contain over a couple hundred different words which, transposed to the image domain, would mean a codebook and image descriptors of over 100000 entries, a master index is built to store all the information in the large descriptors in a compressed way and still not lose any information. This is achieved by analysing the structure of each descriptor and taking advantage of their growing sparsity as the number of entries increases. Refer to [7, 22–24] for further information. It is called *inverted file* or *inverted index* and consists of a list of all the documents that contain a specific term. Figure 2.13 illustrates an example of how an inverted file looks like when based in a regular index.














Regular index			Inverted file			
			Image			
						
Cluster	1	5	0			
	2	0	8			
	3	2	4			
	4	1	5			
	5	3	0			
	6	0	3			
	7	0	0			
	8	2	2			
	9	1	0			

FIGURE 2.13: The original index lookup table (a). Inverted file of the original index table (b).

2.5.3 Term Frequency - Inverse Document Frequency

As explained in [22], the *Term Frequency-Inverse Document Frequency* (TF-IDF) is another technique used in the text domain to improve the quality of the results retrieved. It is normally used as a weighting score to determine the importance of a codebook entry relative to the others. The TF-IDF is a product of two terms, the frequency of occurrence of a term in a document and the importance of that term compared to the rest of the

terms. Let i be the current image under investigation, j one term that occurs in i , $n_{i,j}$ is the number of occurrences of j in i and $n_{k,j}$ is the total number of occurrences of all terms in i . Then, given $|D|$ as the number of documents and $|\{d : t_i \in d\}|$ being the amount of documents containing term t , the TF-IDF score can be defined as follows:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (2.1)$$

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|} \quad (2.2)$$

$$tf-idf = tf_{i,j} \times idf_i \quad (2.3)$$

For example in a DB of soccer articles, the word *sport* or the word *soccer* are expected to get a lot of hits. These words are not going to be stopped by the stop list because they are not frequent in the English language but in this particular scenario, they won't add much discriminative information among the documents.

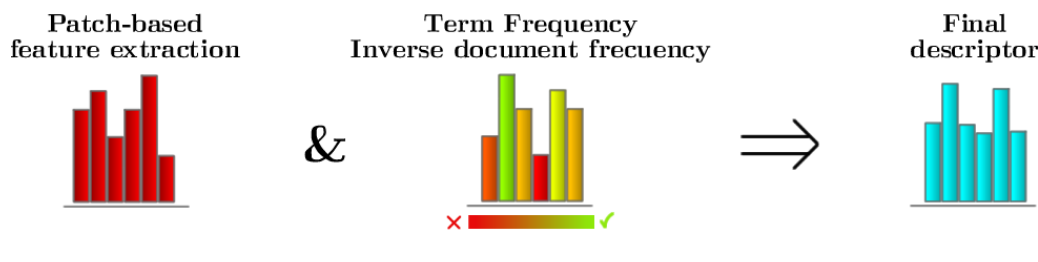


FIGURE 2.14: Improving reliability by combining the patch-based approach with TF-IDF scoring.

TF-IDF operates over the common words (or bins in the inverted file) giving them a low score so that they won't have a powerful influence when comparing a document containing this term. On the other hand, words that do not occur very often in a document but are rarely contained in the rest of the documents would get more relevance and thus, influencing the final rank of the document containing that term.

For the image domain, the same reasoning can be applied and instead of terms, keypoint descriptors which are commonly appearing in all the images in the DB are going to score lower when being compared against other images and vice versa for rare descriptors occurring in just a few images.

2.5.4 Principal Component Analysis

Principal Component Analysis (also known to as *the Discrete KarhunenLove Transform*, *Proper Orthogonal Decomposition* or *the Hotelling Transform*) is a data representation technique proposed by Karl Pearson in 1901 [25] that focuses on the differences and similarities of the data itself. It is well known because it helps to compress high dimensional data [26]. It basically identifies in which directions the data is varying the most. This directions should be orthogonal to each other (therefore they can be thought of to as dimensions) so that the data can be fully represented in terms of these new directions. The other less meaningful directions can be omitted so that they don't suppose an impact when making comparisons against other PCA filtered data. The information being

lost when some of the dimensions are ignored using PCA is not critical because they are the ones that contribute the least to locate data in space. Figure 2.15 shows the process of identifying the directions in which the data varies the most (b)-(c), and the final representation that fixes the non-relevant dimensions to minimize their influence (d)-(e).

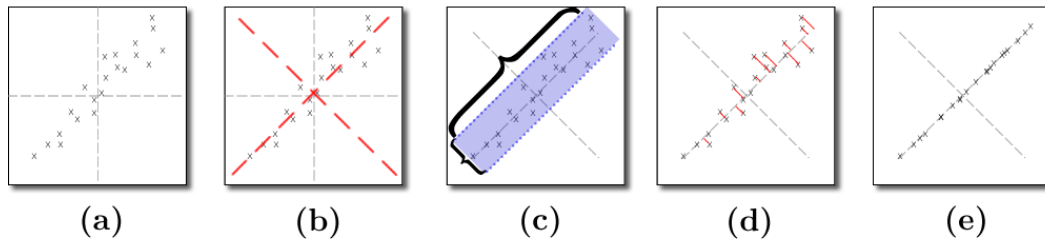


FIGURE 2.15: (a) The original data in a 2D space. (b) The main directions or *dimensions* in which the data varies. (c) The data varies more along one of the dimensions than the other. (d) The data is then represented along the most representative dimension. (e) Final data representation.

When applied to the image domain, given a high dimensional image descriptor (i.e. a histogram of keypoints based on a large codebook), the principal components of that particular histogram can be found and the others can be omitted, reducing the time to analyse the whole descriptor. This also helps to alleviate the curse of dimensionality by reducing the number of dimensions to be processed. More information about PCA can be found in [25, 26]. Examples of CBIR systems with PCA implementations can be found in [8, 14].

In this section there was a thoughtfully overview of the theory, algorithms and data structures that can be found in a CBIR engine. All of these concepts and structures are going to play a fundamental rol in the test scenarios of this work, since they are going to act as parts of a puzzle. Depending on the order and how are they going to be arranged and connected to each other, different outputs are going to be obtained. The idea is to find a way to put them so that it optimizes these outputs and therefore improve the overall performance of the system.

2.6 Previous work

In this section there is a brief summary of some of the work that has been developed in the field of content-based image retrieval. CBIR systems relying on the combination of several approaches to improve both time and quality performance have been implemented by [7]. They select a variety of techniques to address all of the issues that impact a CBIR engine (i.e. time efficiency, disk space consumption and high quality in the results) and propose a series of algorithms that can interact between each other to raise performance in all aspects. The use of Hamming signatures to improve clustering and the addition of a weak geometry consistency check to further enhance the stability of keypoints in an image are their main contributions to the field. They show an improvement of almost 100% (from 0.4 to 0.8 mAP in the best scenario) over their

baseline measurements when large a codebook with *multiple assignment* is used along with a combination of hamming distance weighting and weak geometry consistency.

Another system that aims to retrieve similar images relying on text retrieval methods is implemented by [13]. The system specializes on finding key objects along multiple frames from a video. By combining a patch-based approach (SIFT), a visual vocabulary built offline and taking advantage of the TF-IDF measure to further improve results, the system proves to be reliable enough for some test cases that only comprise objects within a 90 minute film. This restriction is mostly due to the amount of vector quantizations that the system has to make. They also adopted the concepts of spacial consistency and stop lists.

There is a similar CBIR engine proposed by [10] whose purpose is to compare two approaches to identify near identical video shots or images. One of them relies on hashed color histograms while the other uses SIFT local descriptors and a min-Hash algorithm for comparison. Although they lack of a ground truth for their experiments, they were able to deduce that the former approach is more efficient in terms of speed so they see a potential use of this technique for video retrieval. The latter shows more stability and reliability but looks inconvenient when it needs to be scaled up.

Other prototypes also make use of additional metadata to filter the results or provide further levels of refinement in the results as well as dimensionality reduction techniques to fight the curse of dimensionality and also improve speed. [8] builds a system that is able to recognize landmark images in a large body of images. They implemented an unsupervised landmark recognition engine by taking advantage of the SIFT descriptors, geotag information, region-graph clustering, PCA, and some other filters to clean up the model from non-landmark images. When the model is used for image content analysis it reaches 46% accuracy which supports also match the results obtained by [7].

[12] proposes another system that can detect objects and link them to Google Maps or Wikipedia articles without any human supervision. For that they need to classify the images they want to “mine” using a hierarchical clustering. This approach was taken because of its speed and robustness. They also rely on SURF descriptors to improve time performance and also use the TF-IDF measure to find the best tag for every cluster. Their results show that the system can reliably identify objects in the query images and locate them in a map or an encyclopedia.

Some other works have addressed the clustering issue which also presents some difficulties when being used in large scale. Since the codebook can grow indefinitely and the performance seems to get better as more bins are added to the codebook, handling big codebooks for NN matching becomes a serious problem. [9] explores a technique to perform an approximate NN matching through the definition of a hierarchy in the codebook. This approach can cut the NN search time to a logarithmic scale but still can deteriorate the quality if the right parameters (such as the depth and the width of the tree structure that represents the hierarchy) are not set correctly. They show experimentaly how an image retrieval engine can retrieve results in real-time by using the hierarchical clustering technique.

Alternatively the methods evaluated by [11] explore some other ways to fight the *curse of dimensionality* that arises when the vocabulary used for image matching, grows continuously. Approximate NN are compared against a flat vocabulary search. A benchmark between hierarchical and approximate visual clustering is performed showing that the

former can get much better results than the former and yet keeping the same cost in terms of processing time and storage use. They also took special information to refine the rankings although they point out that this should be enhanced if the system is aimed to serve a large body of images.

Chapter 3

Materials and Methods

In this chapter there is a brief description of the CBIR system that is used as the test server and the data that this CBIR engine is going to handle. There is also an explanation on some basic setup parameters of the CBIR engine that have to be fixed before starting the experiments.

3.1 Dataset: INRIA's Holiday data set

In order to be able to test all the proposed techniques to improve the retrieval of a CBIR engine, is necessary to have a set of images that recreates the conditions of an average data set. For all the experiments in this paper, the INRIA's holiday data set ¹ has been chosen. This provides a collection of high resolution images that contain different types of scenes (such as landscapes, under water, fire, man made, etc). All the images are grouped in 500 test cases where each group corresponds to one place or object and within each group, there are several images. The first one is going to be considered as the *query image* and the rest of the images in that group are the *result images*. These result images are taken from the same target but with a few variations such as angle, blurring, illumination, etc. That way, the robustness of the proposed methods to retrieve the images is being put to the test. [Figure 3.1](#) shows an example of how the data set looks like.

3.2 MoViMoS

MoViMoS is an extensible content retrieval engine developed by the MADM group at the DFKI. It is versatile about the types of data that it can handle and is easily extensible so that new functionality can be embedded with ease. This allows the combination of specific techniques to extract features from the contents it holds and merge them together. In this thesis, the MoViMoS platform was used with just images.

¹available for download at <http://lear.inrialpes.fr/~jegou/data.php>

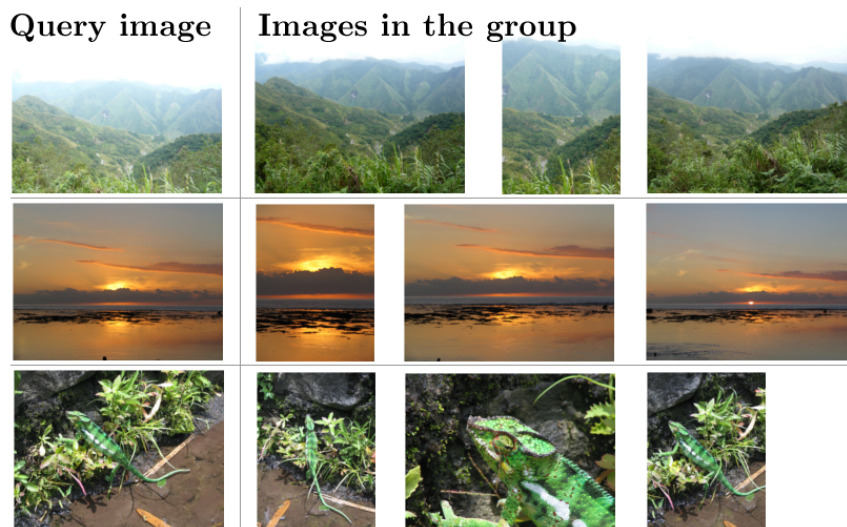


FIGURE 3.1: A sample of the Holiday data set.

3.3 Baseline Establishment

Before starting to evaluate experiments, the establishment of some base parameters is of priority importance in order to count with a baseline and compare the effects of the proposed techniques on such a system.

Given the amount of variables that a CBIR engine can have, it is almost impossible to evaluate and optimize all of them just based on experimentation. Therefore, some base parameters (that were not changed throughout all the experiments) have been set based on documentation and the work of others [7, 8, 13]. Here is a brief list of such variables:

- **distance measure:** L1 (Manhattan) distance. Although for feature comparison, others [7, 12, 15, 27] have used Euclidean distance, Manhattan has shown (experimentally) better performance than any other.
- **descriptor and detector:** SIFT feature extractor. Lately there have been other descriptors that claim to outperform SIFT descriptors such as SURF or Extended SURF but again, experiments have shown that SIFT still gets similar performance and yet not so bad at time consumption.
- **codebook size:** to measure the impact of a growing codebook, a range between 1000 and 15000 has been chosen. This is mainly due to CPU consumption issues and performance improvement rate.
- **performance measure:** mean average precision (mAP). This is a common and widely accepted metric to measure how good a generic data retrieval engine performs. It is based upon the concepts of *precision* and *recall* which basically reflect the amount of retrieved elements that are relevant to the search and the amount of relevant results that were actually retrieved. More information about this metric can be found in [Appendix A](#).

The setup of MoViMoS that was used for testing, follows these steps (note that the parts of the procedure in italics correspond to the stages of the process that are not part of the regular behaviour of a CBIR engine. These steps are the ones introduced to improve performance in the system. A visual description of the procedure can also be found in [Figure 3.2](#)):

1. Take a query image that is not in the data base.
2. Process the image using the SIFT descriptor, obtaining a set of descriptors.
3. *Run PCA over all the descriptors and reduce their dimensionality to get only the most relevant parts of each descriptor*
4. Match the patches against the codebook to obtain a single descriptor for the image.
5. Compare the resulting descriptor to the other pre-computed descriptors of the images in the data base. *One time, TF-IDF is used as the only distance measure, accumulating the scores of every partial TF-IDF result obtained on each cluster of the query image descriptor.*
6. Retrieve the results where the first image is the one with the lowest distance (or highest score) when compared to the query image.

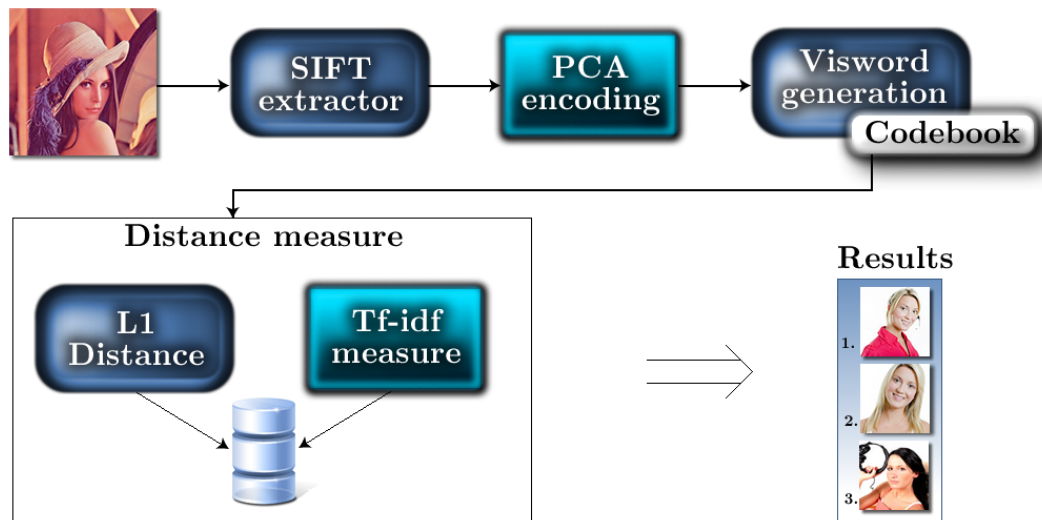


FIGURE 3.2: Setup for the baseline MoViMoS system. The stages in dark blue are the ones considered to be part of a regular CBIR engine. The stages in light blue are the ones proposed in this work.

This chapter went through the elements used for the experiments and the reasons for which these components were chosen, namely the data set of images, the CBIR system and also the base parameters that were fixed through the entire experimentation phase.

Chapter 4

Results

In this chapter, the results of the experimentation phase are being presented as well as the singularities that the data itself shows as the starting conditions of the experiments change. Three main scenarios are evaluated: what is the behaviour of the primary setup with just the base parameters, the process of TF-IDF optimization and the system response when both approaches are being combined.

4.1 Baseline

Several baselines need to be established before starting experimentation, namely the performance of a MoViMoS system configuration that recreates the standard procedure followed by any CBIR engine and also how good the TF-IDF score is when being the sole responsible for the ranking of the images. These results are shown in [Figure 4.1](#). Again, the performance is being measured using the mAP metric with an increasing codebook size so that the impact on more discriminative or *pure* data can also be seen.

4.2 Optimizing the TF-IDF Measure

In order to improve the performance shown by the TF-IDF scoring so that the impact on a patch-based system can be maximized, further adjustments can be done to the TF-IDF. At first, a naive approach to optimize the time and keep the performance is to only accumulate the TF-IDF scores of the bins in the image descriptor that have higher term frequencies (i.e. the most common kind of patches being found in the image). If only 10% of the bins were considered, then the time will be reduced (because the TF-IDF scoring procedure is being truncated) and still most of the information is being kept. Intuition also suggests that bins in the image descriptor with just one or two occurrences can often be generated due to noise or just a non-relevant part of the image.

Another approach that uses more sophisticated techniques consists on taking not only the most frequent bins but trying to get the most relevant ones. This can be done via PCA encoding as explained in [chapter 2](#). Keep in mind that the PCA is applied to the descriptors and not in the final image descriptor. Different levels of dimensionality reduction were tested to measure how much improvement or decrease this method

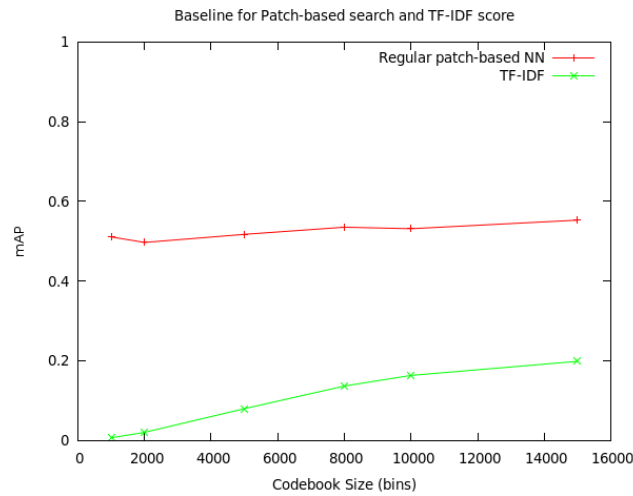


FIGURE 4.1: Mean Average Precision obtained in the baseline systems for a traditional CBIR engine (red) and for a system using TF-IDF as its only way to sort image candidates (green). The results of the former system are higher respect to the values obtained by the latter setup.

introduces. Given that the patch descriptors being generated have 128 dimensions, a PCA reduction of 50% and 25% were tested.

The results of the previous experiments are shown in Figure 4.2

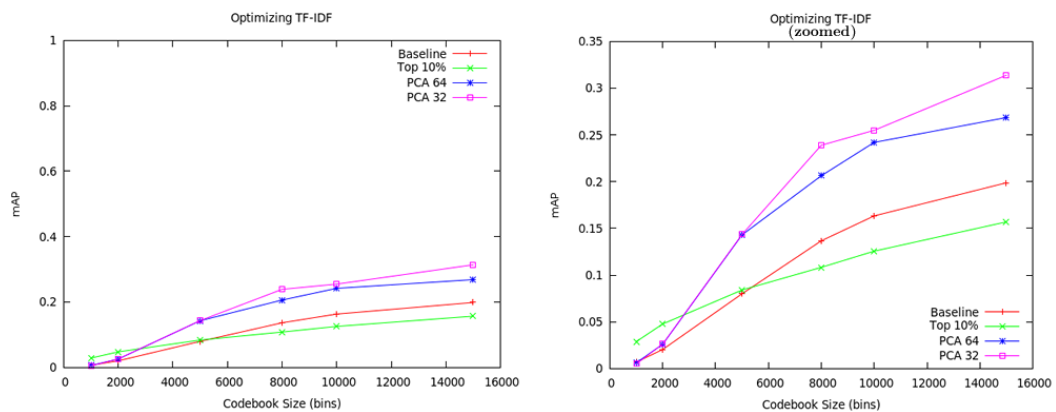


FIGURE 4.2: Results of the experiments with TF-IDF scoring.

4.3 Getting the Best of the Two Worlds

Once the optimal setup for the TF-IDF scoring has been detected, it is time to combine it with the basic CBIR engine and see how that mixture impacts performance. In order to determine which of the methods has to have more relevance over the other, an extra weighting parameter has been added. That way it can be tested if they are both good or if the first one is affecting the second one or if it is the other way around. Results

for this experiments are shown in Figure 4.3. Here, a combination of the regular patch based nearest neighbor and a TF-IDF weight has been implemented. They are also given different relevances to see how it affects performance.

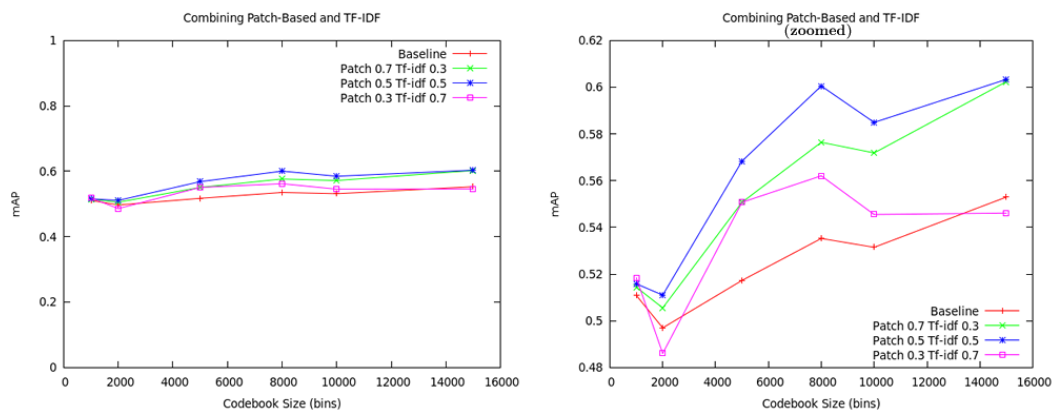


FIGURE 4.3: mAP on a system combining both TF-IDF and L1 distance.

In this chapter a display of the experiments was conducted, showing the charts that describe the responses of the system in terms of quality (mAP) on the different scenarios. It shows how a mixture of a base CBIR with a TF-IDF score can easily raise quality measurements.

Chapter 5

Discussion

In this chapter, an analysis on the results obtained in [chapter 4](#) is conducted and the behaviour of the system along the different scenarios is being supported by the concepts and theory explained in the previous chapters. Further discussions relate some aspects of these results with some particular characteristics of the data that is being treated along with the methods that were used.

5.1 Baseline

The curves described by both baselines are completely different from each other in terms of the values in the ordinate axis, yet this is due to the nature of the techniques being used here. Regarding the baseline of the regular patch-based search there is a very plain curve which suggest that the consistency of the clustering for this particular setup of the system was very good. Other factors such as the uniformity of the data in space also play an important role here and this uniformity can be safely assumed due to the variety of pictures in the dataset, as described in [chapter 3](#). In an optimal scenario, the data will form small clusters which at the same time will be grouped together in bigger sets and so on, as the group of stars in the solar system is just a part of the milky way which can also be grouped with other galaxies. On the other hand, there is a noticeable greater performance than that obtained by [7] using the same data. This is mainly caused by the use of the L1 distance instead of the Euclidean distance. This metric has shown to outperform the euclidean distance and even the Jensen-Shannon divergence metric. There are also some slight differences and hence of lesser impact between these two systems such as image scaling, image normalization and implementations of the SIFT descriptors.

Regarding the performance curve of the TF-IDF scoring system, it can be said that this results, although small, provide a hint of how much improvement (or decrease) can be achieved once the two methods are merged together. Even though it is not describing a significant steep curve, there is a clear evidence of how the size of the codebook impacts the performance of the TF-IDF score. This does not contradict the first statement about the clustering of the patch-based system. For this method, is more relevant the amount of keypoints that fall into a particular cluster (i.e. the distribution of keypoints within one image) and whether or not this particular cluster is relevant to the dataset.

5.2 Optimizing TF-IDF

After running the modified TF-IDF systems, different behaviours can be observed although the same basic principle of optimization was kept in mind.

The first naive approach, parts from the idea that noisy information is being extracted from the image, kept in the final descriptor and it remains as the descriptor bins that have just one or two entries. The belief on the existence of such noisy interest points is based on observations made to some images such as the one in [Figure 5.1](#). Since there could be many objects in an image that could end up being potentially interesting, the system identifies all of them. Therefore, by just taking the highest bins in the descriptor, the risk for this noise to be considered is being minimized and hence, lowering their impact on the results. Another consequence of forcing such truncation is that the time spent in the TF-IDF score calculation can be reduced. The results are not as expected which suggests that some of the assumptions were not accurate enough or that other factors are being underestimated. There is a clear drop in performance as a consequence of the information loss. Despite this reduction, consistency regarding the codebook growth is being kept, yet there is not a drastic difference when comparing it to the base line.

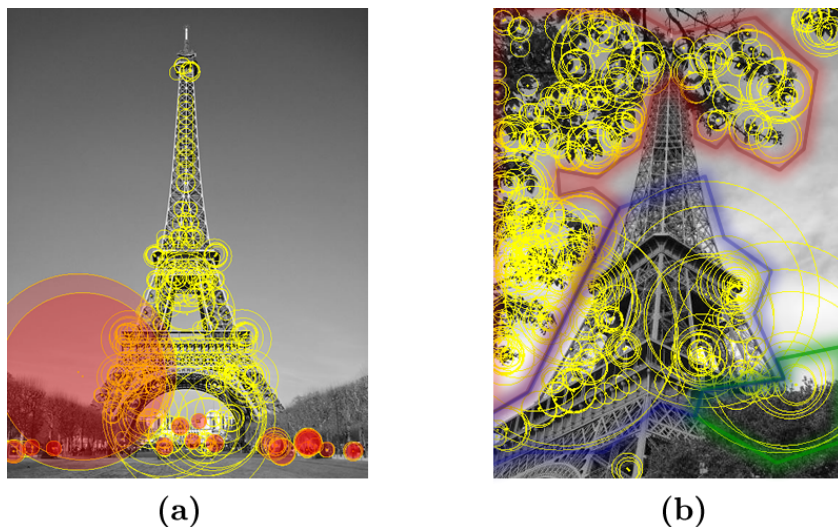


FIGURE 5.1: Observations made to images like these, suggest that there are interesting points that only introduce noise to the final image descriptor. The image of the Eiffel tower not only generates interest points of the tower but also the grass and the building behind (a). In part (b), even when the object of interest (blue) is strongly occluded (red), there are still noisy parts that could be discarded (green).

There are several factors that make this a bad approach for the problem, starting at the way TF-IDF works. This is a score that balances entries with a high count and entries that are more meaningful or provide more discriminative power. In this scenario, only the high bins are taken into account and most likely, they are not going to rate very high. On the other hand, the bins that get a low percentage of the keypoints can get a better score and although they won't score too high, they will contribute to get a better result. After taking a look at the structure of an image descriptor we can get a better insight of why this happens. [Figure 5.2](#) (a) shows a typical descriptor. In part (b) the descriptor is sorted by putting the highest bin in the center and the subsequent ones

on the side. In this last image it can be seen how much information is being lost. The most critical one is highlighted in green. This bins have the potential to provide more information to differentiate an image because it can grow due to the inverse document frequency. A positive aspect of this approach is that the time spent at accumulating the TF-IDF score for the whole descriptor gets cut although the compromise related to quality is decreasing as well.

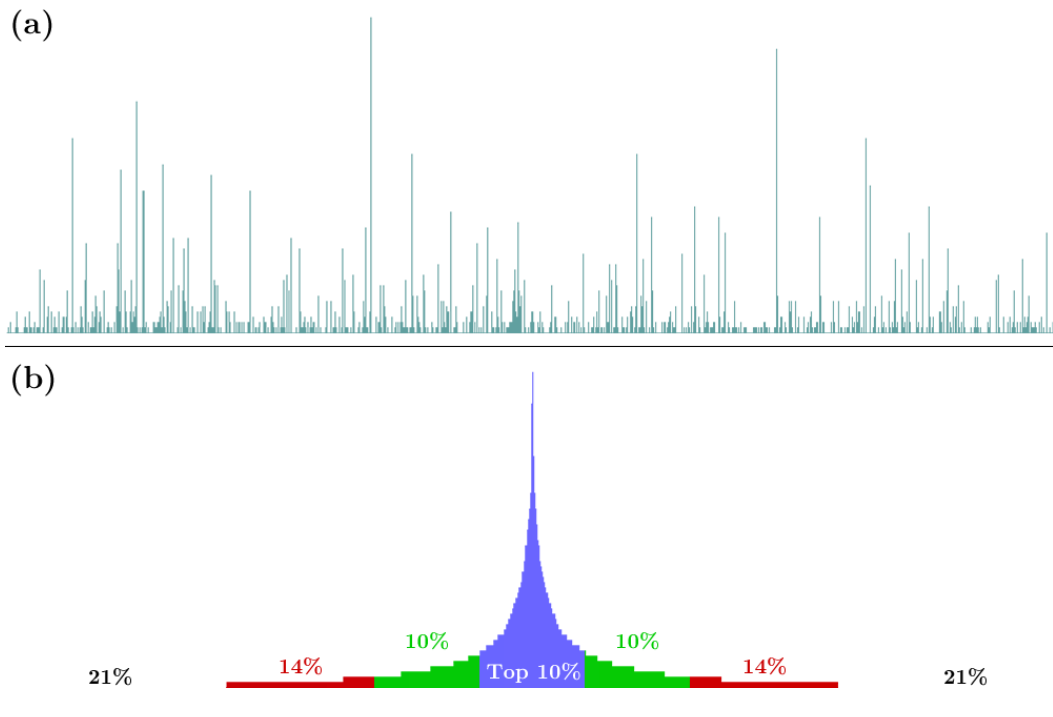


FIGURE 5.2: The structure of a 1k image descriptor. (a) Ordered the way the codebook is sorted. (b) Sorted by putting the highest bin in the center and the subsequent ones along the sides.

Now, when PCA is being used, a surprising rise in performance of around 7% is reached. Both PCA test runs show a steady tendency to keep on rising along with the codebook size. This behaviour suggests that the clustering of the codebook improved by setting the parts of the descriptors that were not varying too much to a constant value. Mathematically speaking, the 128 dimensional points in space are now being represented in a 64 and 32 dimensional hyper plane improving the chances of forming better clusters. Since the centroids of the clusters are calculated based on an Euclidean distance, the more dimensions there are, the bigger the distance that separates the points in a cluster from the centroid¹. Assuming that PCA keeps only the dimensions that are relevant to an image descriptor, all the other dimensions can be considered as noise. When summed up together, these noisy dimensions can influence the location of the centroids. This issue becomes more evident as the points become more sparse. Figure 5.3 shows that when dealing with sparse points in space, it is not too evident how the points should be clustered. When only the more relevant dimension is evaluated, then the clustering algorithm has better chances to find more stable clusters (i.e. after several runs of the clustering algorithm with different starting points, it ends up building the same clusters).

¹Assuming that the values of these dimensions is not 0

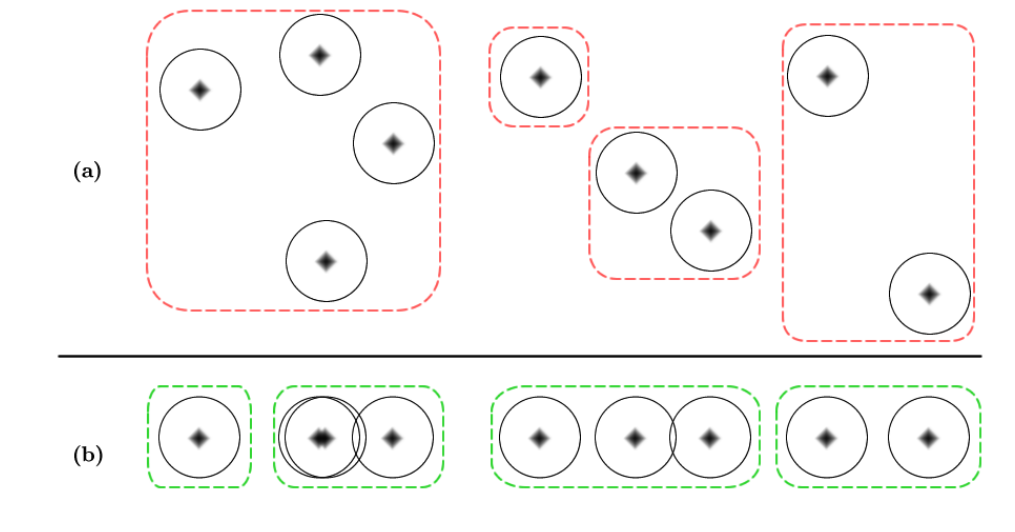


FIGURE 5.3: When performing clustering in a sparse space, PCA can help finding more stable clusters. A possible solution proposed when analysing all the dimensions available (a) and the one that is most likely resulting when PCA compression is used on 1 dimension (b). Note how in part (b) the clusters seem more compact and it is even visually easy to recognize them.

Figure 5.4 shows another example in a 2D scenario. Suppose there are just two dimensions available and the most meaningful one, is the x axis. In order for the data to be representable in 2D, the non-relevant dimension, y , will be set to zero. If 5 clusters are to be used to classify all the data, in the first scenario (part (b)), there are some clusters that were not chosen the best way because of small variations along the y axis that made the clustering algorithm to end up with a good, yet far from optimal clustering. In the other scenario (part (c)), since only the x coordinate is used to form clusters, they get grouped together in a better way and preserve the part of the information that is more meaningful. Note that the clustering algorithm also introduces other uncertainty variables (e.g. initialization of starting points, number of clusters, iterations) that could end up in small variations on the clustering results. Since the problem of exact data clustering is NP-hard [28], approximate clustering needs to be implemented for production CBIR systems. This means that the optimal solution cannot be computed in a reasonable time even when handling moderate amounts of data. Nevertheless the approximate method provides a good approximation in an acceptable time. Efforts to refine the approximation found by clustering algorithms have been proposed by [29, 30] (among many others) but the quality (i.g. the similarity of the results compared to the optimal solution) of the clusters is still depending on the nature of the data being processed. In the case of the PCA compressed data, the clusters can also present variations after several runs of the clustering algorithms but emphasis is made on the impact that PCA does over the data and the clustering itself.

The fact that the strongly reduced, 32 dimensional PCA performed equal or even better than the 64 dimensional one in some points, suggests that the amount of significant dimensions are actually less than 32, giving a compression ratio of over 75%. Still, the gain in performance was not too drastic between both PCA tests. This also shows that this method is strong and consistent enough so that it can make an impact in the patch-based search that ranked very high.

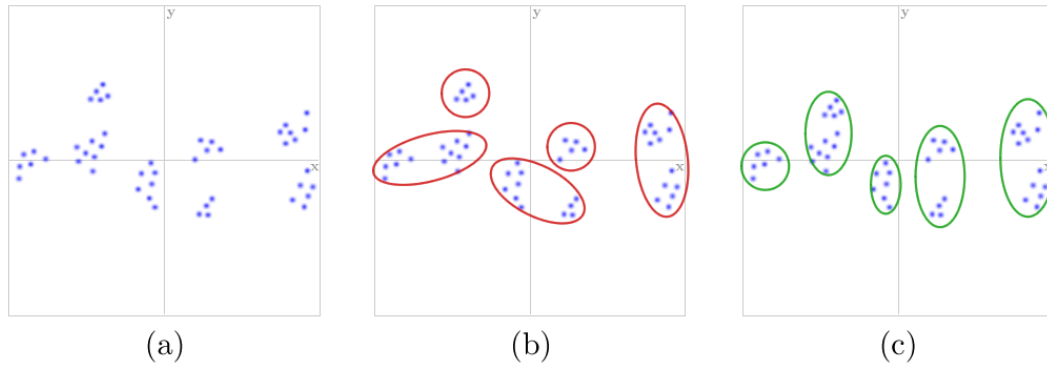


FIGURE 5.4: How PCA affects clustering. (a) The original points in a 2D space. (b) Regular clustering. (c) Clustering when only the X axis is considered.

5.3 Combining Analysis

After finding the optimal setup values for the base system and after tuning up the performance of the TF-IDF, both approaches are merged together to improve each other and raise the quality of the retrieved images. The results of the experiments show a reasonable improvement for the majority of the test runs. The weight of both sorting methods was varied to measure the impact on the results. As expected, the curve wasn't drastically altered even when the TF-IDF score was the dominant measure. This confirms the observation about the stability of the patch-based search and the TF-IDF. On the other hand similar results were obtained when the patch-based search was either equal or dominating the search ranking. This behaviour indicates that the TF-IDF is not only helping by high-ranking the correct results but also by discarding the false positives and preventing them to climb up to high in the list of candidates.

It is important to note that although this is not a vast improvement it's still worth using it because it helps making results less variable every time a new technique is added on top. Besides, thanks to the dimensionality reduction made by PCA, the cost of keeping the index structure in memory can go below 75% with PCA models of 32 dimensions and lower.

In this chapter the discussion on the results of the experimental phase took place. These results show that with little effort, an increase on quality measures can be achieved without sacrificing other aspects such as speed and memory consumption. Although there are many other parameters that can be adjusted in order to get better results (e.g. keep increasing the size of the codebook, augmenting the PCA compression ratio, make use of other different datasets, raise the noise level in the dataset), this work tried to focus on the parameters that seemed more promising at first. Due to time constraints, many other tests were left out of this work. These results prove many of the concepts underlying the test scenarios (e.g. the use of PCA, TF-IDF, dissimilarity measures, codebook clustering parameters) and show that it is still possible to keep on improving.

Chapter 6

Conclusions

This work went through some of the most well known approaches to extract information about images that can be efficiently stored and compared in order to sort them according to their similarities. There is an emphasis on two specific techniques that haven't been properly evaluated and documented until now, namely the patch-based approach and the TF-IDF scoring. While the former gives a steady base line to work upon, the latter provides an extra boost of performance while keeping times and disk overload to a minimum. Since there are many parameters that need to be adjusted, previous experiences were taken into account as well as some proper experimentation. That allows the system to gain in performance and stability regarding the results. For the experimentation stage, reasonable and well tested data sets were employed as well as a base CBIR engine that was adapted to work with the required data structures. Measures on the size of the codebook were also performed in order to prove the phenomena about coarse and fine clustering. At the end, the impact on performance when combining both approaches is being shown. The assumptions on the stability and the boost on performance are proven and an analysis on the reasons for this to happen were studied in detail. Some of the most important observations about the behaviour of the methods are:

- Term Frequency-Inverse Document Frequency (TF-IDF) can be used as an extra, easy to implement score to refine an already existing image search index but not as the primary criteria to sort images since the benchmarks of a system that relies only on TF-IDF are low compared to a regular nearest-neighbour search based on histogramized image features.
- Although the Euclidean distance is normally considered to sort a list of image candidates when using an approach based on invariant local features, the L1 norm has experimentally shown to outperform the Euclidean distance as well as all the other distance metrics that are normally used in this scenarios. Refer to [Appendix A](#) for a table comparing them.
- When the patch-based feature extraction and the TF-IDF are combined, an improvement of up to 10% was achieved on the Holiday image data set, using mean average precision (mAP) as the metric to measure accuracy on the results.
- Taking the highest bins of an image descriptor and discarding the rest, helps saving time during the calculation of a TF-IDF score but it also implies a drop in

performance (mAP) due to the loss of discriminative information that remains in the unprocessed part of the image descriptor.

- Implementing PCA in an early stage of the image feature extraction along with the use of TF-IDF in a CBIR system helps improving the clusters for patch-based features and therefore improving performance (mAP). An encoding of 50% and even of 75% using PCA on 128 dimensional SIFT descriptors, reports an improvement on performance that allows a CBIR system to output more refined results.

In a nutshell, the use of the TF-IDF score gives a boost in terms of quality of the results being retrieved while still maintaining a reasonable resource consumption.

Chapter 7

Further Work

Given the nature of the problem being addressed here and the way it can be tackled there could be countless directions in which image retrieval can turn. Based on the experience of this work and the structures that have been employed here, there are a couple interesting issues that can be addressed in future investigations.

- In a general scope, one must highlight the fact that the nature of the data is changing between the stages of the overall image extraction and retrieval. This means that the original essence of the data is being reinterpreted in a different way every time and bit by bit, fractions of the data structures are being lost. The idea would be to think of methods that are more data driven instead of having this kind of black-box behaviour.
- Starting at the clustering of the SIFT descriptors, they first are a concatenation of 8 bin histograms but then when being parsed against the codebook, this notion of gradient histograms is being lost (since the clustering stage just considers it as a 128 dimensional array or a point in a 128 dimensional space). One interesting thing that could be done is to try a similar clustering procedure that operates on the gradients of every SIFT descriptor and then build a catalog based on the clustered gradient histograms overall the image.
- Sticking to the idea that images can be treated as words, it would be interesting to introduce a more text-driven concept of distance. Until now, all the image descriptors are being compared against each other using metric distances such as Euclidean, L1, JSD, etc. These image descriptors can be adapted to better reflect the structure of a word or in this case a *term* and that way, a Levenshtein distance can be calculated.

Appendix A

Definitions and other results

A.1 Mean Average Precision (mAP)

The mAP is one of the most widely accepted metrics to measure the quality of the results given by an information retrieval system. Average precision (AP) basically considers not only the amount of correct results or *true positives* but also takes into account how high they rank. To have an overall impression of how a system is performing, several test queries have to be conducted in order to get an average precision for each try. Then, the mean of all of the average precision results is calculated and that number is going to reflect the mAP of the system.

To calculate the AP for a query, it is necessary to calculate two metrics at first: precision and recall. Precision is going to show the portion of the retrieved data (images in this case) that is actually relevant to the query. Recall refers to the portion of the relevant data that was retrieved. These definitions might seem confusing but [Equation A.1](#) and [Equation A.2](#) show both precision and recall in a more concrete way. It becomes evident that both metrics are rather similar but they refer to very different concepts.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (A.1)$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (A.2)$$

In terms of images in a CBIR system, one can express precision and recall as follows:

$$Precision = \frac{relevantImages \cap retrievedImages}{retrievedImages} \quad (A.3)$$

$$Recall = \frac{relevantImages \cap retrievedImages}{relevantImages} \quad (A.4)$$

Finally, AP can be expressed as [Equation A.5](#)

$$\frac{\sum_{i=1}^n Precision(i) \times Recall(i)}{\# \text{ of relevant Images}} \quad (A.5)$$

A.2 Distance Measure Comparison

Mean average precision calculated on the Holiday dataset with 1k, 2k and 5k image descriptors using SIFT. Using only local invariant features to describe an image, the Manhattan distance shows stability and better performance than the other distance measures in order to determine a nearest neighbor.

	1K	2K	5K
Manhattan	0.510884368086	0.497081527177	0.517297397186
Euclidean	0.489388177781	0.435519559112	0.337281821016
JSD	0.454907234219	0.402402774342	0.372502580653
χ^2	0.476934076966	0.438126018862	0.406113328638

TABLE A.1: Comparison of different dissimilarity measures in a high dimensional space using mAP.

Bibliography

- [1] Devin Coldewey. Facebook hits 10.000.000.000 photos good lord. Blog entry in CrunchGear, October 2008. URL "<http://www.crunchgear.com/2008/10/15/facebook-hits-10000000000-photos-good-lord/>".
- [2] Photobucket Staff. Fun stats. Blog entry in the Photobucket blog, March 2005. URL http://blog.photobucket.com/blog/2005/03/fun_statistics.html.
- [3] Heather Champ. "4,000,000,000". Blog entry in the Flickr blog, October 2009. URL <http://blog.flickr.net/en/2009/10/12/4000000000/>.
- [4] Erick Schonfeld. Who has the most photos of them all? hint: It is not facebook. Blog entry in TechCrunch, April 2009. URI: <http://techcrunch.com/2009/04/07/who-has-the-most-photos-of-them-all-hint-it-is-not-facebook/>.
- [5] Andrew Moore. An introductory tutorial on kd-trees. Technical Report Technical Report No. 209, Computer Laboratory, University of Cambridge, Pittsburgh, PA, 1991.
- [6] Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1000, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7822-4.
- [7] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 304–317, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88681-5. doi: http://dx.doi.org/10.1007/978-3-540-88682-2_24.
- [8] Yan-Tao Zheng, Ming Zhao, Yang Song, Hartwig Adam, Ulrich Buddemeier, Alessandro Bissacco, Fernando Brucher, Tat-Seng Chua, and Hartmut Neven. Tour the world: building a web-scale landmark recognition engine. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, Miami, Florida, U.S.A, June, 2009. doi: <http://doi.acm.org/10.1145/1291233.1291448>.
- [9] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: <http://dx.doi.org/10.1109/CVPR.2006.264>.
- [10] Ondřej Chum, James Philbin, Michael Isard, and Andrew Zisserman. Scalable near identical image and shot detection. In *CIVR '07: Proceedings of the 6th ACM*

- international conference on Image and video retrieval*, pages 549–556, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-733-9. doi: <http://doi.acm.org/10.1145/1282280.1282359>.
- [11] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [12] Till Quack, Bastian Leibe, and Luc Van Gool. World-scale mining of objects and events from community photo collections. In *CIVR '08: Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 47–56, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-070-8. doi: <http://doi.acm.org/10.1145/1386352.1386363>.
- [13] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, oct 2003. URL <http://www.robots.ox.ac.uk/~vgg>.
- [14] Thomas Deselaers. Features for image retrieval. Master’s thesis, Rheinisch-Westfälische Technische Hochschule Aachen, December 2003.
- [15] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91, November 2004.
- [16] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008. ISSN 1077-3142. doi: <http://dx.doi.org/10.1016/j.cviu.2007.09.014>.
- [17] G. J. Burghouts and J. M. Geusebroek. Performance evaluation of local colour invariants. *Computer Vision and Image Understanding*, 113:48–62, 2009. URL <http://www.science.uva.nl/research/publications/2009/BurghoutsCVIU2009>.
- [18] Jan Puzicha, Joachim M. Buhmann, Yossi Rubner, and Carlo Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1165, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0164-8.
- [19] Eric Brasseur. Gamma error in picture scaling. Unpublished technical report about the bad implementation of scaling algorithms, 2010. URL <http://www.4p8.com/eric.brasseur/gamma.html>.
- [20] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761 – 767, 2004. ISSN 0262-8856. doi: DOI:10.1016/j.imavis.2004.02.006. URL <http://www.sciencedirect.com/science/article/B6V09-4CPM632-1/2/7e4b5f8aa5a4d6df0781ecf74dfff3c1>. British Machine Vision Computing 2002.
- [21] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *ECCV (1)*, volume 2350 of *Lecture Notes in Computer Science*, pages 128–142. Springer, 2002. ISBN 3-540-43745-2. URL <http://dblp.uni-trier.de/db/conf/eccv/eccv2002-1.html#MikolajczykS02>.

- [22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008. ISBN 978-0-521-86571-5.
- [23] Nivio Ziviani, Edleno Silva de Moura, Gonzalo Navarro, and Ricardo Baeza-Yates. Compression: A key for next-generation text retrieval systems. *Computer*, 33:37–44, 2000. ISSN 0018-9162. doi: <http://doi.ieeecomputersociety.org/10.1109/2.881693>.
- [24] Justin Zobel, Alistair Moffat, and Kotagiri Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Trans. Database Syst.*, 23(4):453–490, 1998. ISSN 0362-5915. doi: <http://doi.acm.org/10.1145/296854.277632>.
- [25] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.
- [26] Lindsay I. Smith. A tutorial on principal component analysis. Tutorial, 2002. URL <http://kybele.psych.cornell.edu/~jedelman/Psych-465-Spring-2003/PCA-tutorial.pdf>.
- [27] Matthijs Douze, Hervé Jégou, Harsimrat Sandhawalia, Laurent Amsaleg, and Cordelia Schmid. Evaluation of gist descriptors for web-scale image search. In *CIVR '09: Proceeding of the ACM International Conference on Image and Video Retrieval*, pages 1–8, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-480-5. doi: <http://doi.acm.org/10.1145/1646396.1646421>.
- [28] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *WALCOM '09: Proceedings of the 3rd International Workshop on Algorithms and Computation*, pages 274–285, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-00201-4. doi: http://dx.doi.org/10.1007/978-3-642-00202-1_24.
- [29] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, 2002. ISSN 0162-8828. doi: <http://dx.doi.org/10.1109/TPAMI.2002.1017616>.
- [30] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-24-5.