Relevant Image Search Engine (RISE)

by

Franco Segarra Querol

Final Master Thesis submitted to

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Thesis Adviser: Roberto Paredes Palacios

Valencia, November 2010

1

Abstract

Digital image retrieval has attracted a surge of research interests in recent years. Most existing Web search engines usually search images by text only. They have yet to solve the retrieval tasks very effectively due to unreliable text information. Until now, general image retrieval is still a challenging research task. In this work, we study the methodology of cross-media retrieval and its effects on a tailor made visual search engine.

Content Based Image Retrieval is a very active research topic which aims improving the performance of image classification. This work shows how to build a content based image retrieval engine. Later on several experiments consisting in changing some parameters and adding new retrieval techniques will show how they affect the global system.

Contents

Chapter 1

Introduction

The number of images in internet is increasing rapidly due to the presence of digital cameras and social networks. Since the speed on the Internet connection has raised, it's easier to access and share digital media such as photos or videos. In the modern age it is now common-place for private individuals to own at least one digital camera, either attached to a mobile phone, or as a separate device in its own right . The ease with which digital cameras allow people to capture, edit, store and share high quality images in comparison to the old film cameras, coupled with the low cost of memory and hard disk drives, has undoubtedly been a key driver behind the growth of personal image archives. Furthermore, the popularity of social networking websites such as Facebook and Myspace, alongside image sharing websites such as Flickr (see Figure 1.1) has given users an extra incentive to capture images to share and distribute amongst friends all over the world [42]. For this reason the correct classification and storage of images in the net is a studied fact nowadays. A normal person is able to find a picture between a small amount of them but when the quantity increases to thousands it can be a very hard effort. Computers are able to help humans in this task by approaching through several ways. An example of this approach is the textual description of images using text-based retrieval engines to complete the duty. But it is not feasible to make the annotations of all images manually, and on the other hand it is not possible to translate some images and specially abstract concepts in words. There are other types of automatic annotation based on web page description or context. Due to the rich content of images and the subjectivity of human perception no textual description can be absolutely complete or correct [14]. In order to refine the user search other image searching technique is needed:CBIR (Content Based Image Retrieval). "Content-based" means that the search will analyze the actual contents of the image. The term "content" in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. Given an image this engine will return the most similar images attending to different requirements. This method is object of study world wide and there exist a great number of databases for this need.

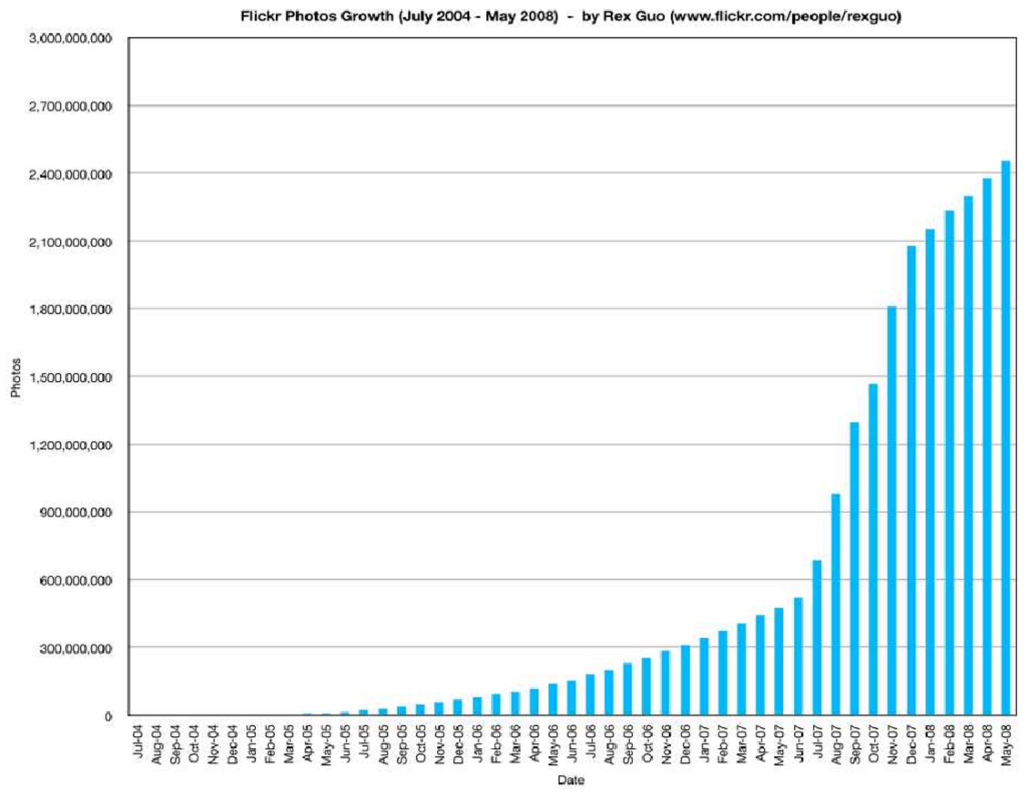


Figure 1.1: Flickr image growth over the past years.

Sometimes this systems may need the help of humans (users) in the process of selecting the relevant documents from an initial set. This method is called Relevance Feedback. Relevance feedback is a feature of some information retrieval systems. The idea behind relevance feedback is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query. The problem of search and retrieval of images using relevance feedback has attracted tremendous attention in recent years from the research community. A real-world-deployable interactive image retrieval system must (1) be accurate, (2) require minimal user-interaction, (3) be efficient, (4) be scalable to large collections (millions) of images, and (5) support multi-user sessions. For good accuracy, it needs effective methods for learning the relevance of image features based on user feedback. Efficiency and scalability require a good index structure for retrieving results. The index structure must allow for the relevance of image features to continually change with fresh queries and user feedback [41].

A correct solution of this problem can lead to a solution in many different fields. Some applications where is applied and requested nowadays are security; image comparison for corporate security and law enforcement investigations, media archives; retrieve and manage visual assets, scientific imaging; retrieve and classify images with respect to specific visual content, art collections, medical diagnosis, architectural and engineering design and many more. In CBIR, there are, roughly speaking, two different main approaches: a discrete approach and a continuous approach [13]. (1) The discrete approach is inspired by textual information retrieval and uses techniques like inverted files and text retrieval metrics. This approach requires all features to be mapped to binary features; the presence of a certain image feature is treated like the presence of a word in a text document. (2) The continuous approach is similar to nearest neighbor classification. Each image is represented by a feature vector and these features are compared using various distance measures. The images with lowest distances are ranked highest in the retrieval process.

Image retrieval procedures can be divided into two separate options: query-by-text (QbT) and query-by-example (QbE). In QbT, queries are texts and targets are images. That is, QbT is a cross-medium retrieval. In QbE, queries are images and targets are images [30]. The text based approaches apply regular text retrieval techniques to images annotations or descriptions. The content-based approaches apply image processing techniques to extract image features and retrieve relevant images [67]. The goal of the photographic retrieval task is to find as many relevant images as possible from an image collection given a statement describing a user information need. When text retrieval and image retrieval cooperate when searching a unique solution it is called fusion or multi modal retrieval. The fusion between image retrieval and text retrieval can happen in many different moments, and depending on these we can have very different solutions. In other chapters some techniques will be explained and finally one of them will be selected to be used on the final version of the engine.

1.1 State of the Art

Among the first content based image retrieval systems that were available were the QBIC system from IBM [25] and the Photo book system from MIT [48]. QBIC uses color histograms, a moment based shape feature, and a texture descriptor. Photo book uses appearance features, texture features, and 2D shape features. Another well known system is Blob world [5], developed at UC Berkeley. In Blob world, images are represented by regions that are found in an Expectation-Maximization-like (EM) segmentation process. In this systems, images are retrieved in a nearest-neighbor-like manner, following the continuous approach to CBIR. Other systems following this approach include SIMBA [59], CIRES [31], SIMPLIcity [65], and IRMA [35]. Existing commercial image search engines, such as Google image search, Lycos, AltaVista photo finder, use text-based image retrieval technique, i.e. without considering image content and relying on text or keywords only, to look for images on the web. The retrieval model for text retrieval is different from image retrieval. Images are represented by low-level features like color, texture, region-based, shape-based descriptors or salient point descriptor and more. Meanwhile, text documents are represented by the term weight features.

In fact, recently Google Labs has launched an application for Google Images called

Similar Image Search based on CBIR [1] probably using local features, also FIRE Flexible Image Retrieval Engine by Thomas Deselears works similarly as what it is done in this work [16], Flickr has an application named Xcavator [2], INRIA has a project named Imedia based on a visual search engine and soon the rest of search engines will follow the same path (Yahoo!,Altavista). Some other engines such as VAST, use a semantic network and relevance feedback based on visual features to enhance keyword-based retrieval and update the association of keywords with images [32].

1.2 Goal of this work

The purpose of this work is to build a cross-media retrieval search engine web demo named RISE (Relevant Image Search Engine) with images annotated automatically. In this engine the user will start by introducing a query and will end with a certain number of images relevant to its query. The different possibilities in feature extraction, distance methods and retrieval techniques will be tested and explained. When building the web demo we will explain how to automatically download and tag the images. This images are saved in a huge database. We will talk about the importance of having this database tidy and clean so that the user can easily query it. In this demo the user will help in the process of selecting the correct images (Relevance Feedback).

One of the most interesting issues in multimedia information retrieval is to use different modalities (e.g. text, image) in a cooperative way. We will test this different modalities with relevance feedback presenting a Multimodal Relevance Feedback approach. Our goal is to explore how the output of an image retrieval system in cooperation with the annotated text may help the user. The hypothesis is that two images which are visually similar share some common semantics. Finally we will present several experiments demonstrating how different the queries can be and how using several modalities helps the system perform with a greater precision.

Chapter 2

Image and Text Representation

In image retrieval there exist many different types of features one can extract from the pictures. For a web demo like ours with a great amount of images in the database, it is important to select a method which is fast and accurate. Features can be categorized into the following types:(a) color representation, (b) texture representation, (c) local features and (d)shape representation. In previous work [56], (a), (b) and (c) are implemented and explained. The different distance method functions used to compare images or text are greatly influenced by the representation, causing a difference in the performance of the system. Next we will present some representations and we will choose the best option for the web demo.

2.1 Color Histograms

In order to represent an image, one of the most common approaches is color histogram representation. Each image can be represented as a vector where each position is the quantity of color in the image. Color Histograms are perhaps one of the most basic approaches as well as the baseline for many image retrieval systems. Despite its simplicity they obtain a high precision for some of the databases used, as we will observe later on. Histograms are independent from size of images since every histogram is normalized previously.

2.1.1 Unquantified

This type of color histograms are the easiest to obtain but not the most accurate. Once the image is built in matrix form, we will construct a vector where first 256 positions correspond to red, next 256 to green and last 256 to blue. Therefore the vector will have a total of 768 positions. Purple color for example, has a red, green and blue component so first step is detecting the value of each of this 3 colors. Next step will be to examine the color matrix and assign a position in the final vector for each of the pixels. For the purple pixel in the example, the red, green and blue component is selected and its value in the vector is increased. If all the values in the vector are added they will be 3 times more than the total number of pixels in the image. Last step will be to normalize the vector dividing by the total number of pixels, this way the vector is no longer exposed to different size of images and now will have a measure of proportion. For the rest of the text, this feature will be named as Histogram.

This is equal to getting three different color histograms separately and putting one after the other. This kind of histograms have a main problem: each color component is treated separately from the others and this may lead to erroneous results.

2.1.2 Quantified

As we previously saw, treating color histograms individually can lead to several problems. By making multi-dimensional quantified histograms this problems disappears completely and it helps in making a more trusted vector representation. In order to understand this concept much better it is important to see the color map as a cube, because there are three primary colors (red, green, blue), an object that represents the colors must be a three-dimensional object. Red is represented in the height dimension, blue in the width dimension and green in the depth dimension.

The cube has 256 possible values per dimension which allows over 16 million different colors (16777216). The most accurate possible vector representation occurs when each position corresponds to a color, but this leads to a huge vector where mainly all the positions will be zeros(sparse vector). A variation in any of the pixels makes a complete different color although visually is nearly impossible to say. This leads to the color space partitioning method. The process consists in partitioning each color into a number chosen by the user and then counting the number of pixels in this range. Perceptually, the color space is divided so it no longer consists on 256 values per dimension. Each dimension is sliced and it now forms a color space with a lower number of colors than before as in Fig.2.1.

This way all the values from the same range are considered as the same color. When choosing a pixel from the original image it is located on the new color space and its new value is assigned. When making multi-dimensional quantification, this color space is reduced and vector is less sparse.

2.1.3 Example

For 8 quantification we divide 256 color space in 32 parts each of size 8(positions 0 to 7). For example for pixel (17 19 16) we will add 1 to position (2x32x32)+(2x32)+2 since 17 falls into range 2 (16 to 23) and so does 19 and 16. From now on Q8 will be used to name the histograms with 8 quantification, Q16 for 16 quantification and so

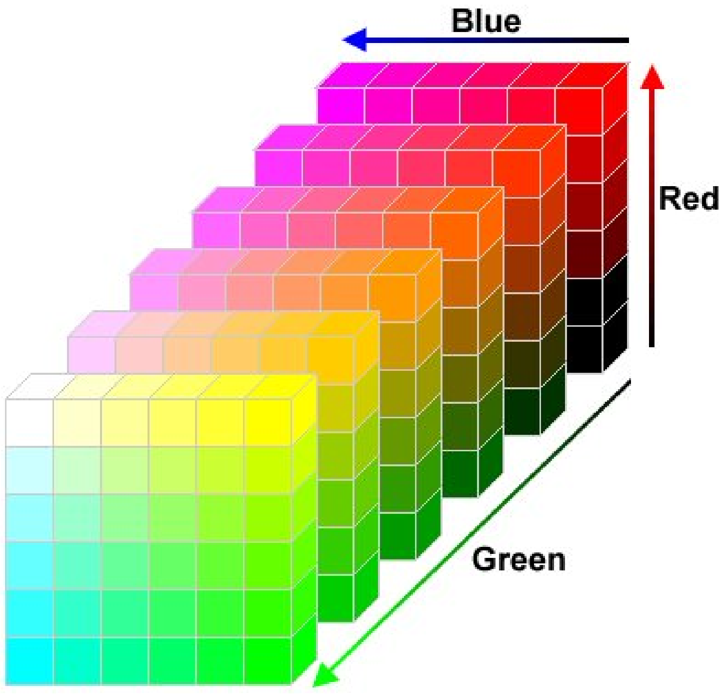


Figure 2.1: Sliced representation of color space for quantified histograms.

Quantification	Vector Length
1	16777216
8	32768
16	4096
32	512
64	64

Table 2.1: Vector Size when Partitioning.

on.

$$Position = \sum_{x=0}^{2} \lfloor \frac{V[x]}{q} \rfloor \left(\frac{256}{q} \right)^{2-x} \tag{2.1}$$

Where q in Eq.2.1 stands for the range of values of each color and x for the position of red, green and blue in each pixel. In this case we do take advance of color properties to create the vector since what this more or less means, is that each different color is made out of q values. The quantification number is very important since it can lead to a sparse vector in case the chosen number is really small or to misclassification and confusion if it is too big as seen in Table 2.1.

The Table 2.1 is really important, since we are not only looking at precision but also at computational time. The user wants its query to be answered as soon as possible.

2.2 Tamura Features

Texture is a key component of human visual perception and like color it is an essential feature when querying image databases [28]. It is very hard to describe texture in words but we can say that texture involves informal qualitative features such as coarseness, smoothness, granularity, linearity, directionality, roughness and regularity [29]. After several studies authors determined that the only meaningful and computable features where coarseness, directionality and contrast. All the different textures stated by Tamura can be seen in Fig.2.2.

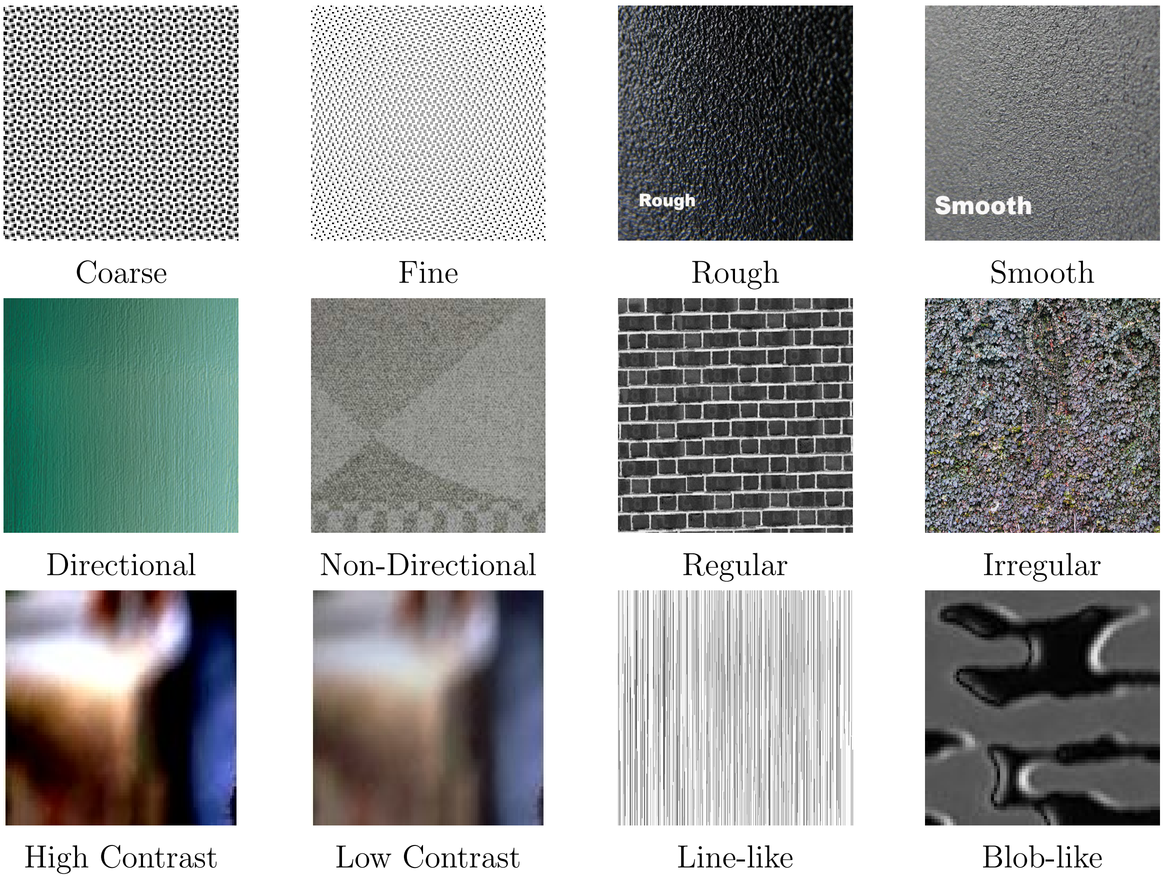


Figure 2.2: Example of Different Textures.

2.3 Histogram Layout

As a particular case of Color Histograms, this representation extracts information of certain parts of an image. This is, it divides the image in different regions and creates sub images. Later on, an independent histogram is extracted from each of this sub images. The main goal of this system is to use position information of images as well as color.

We can observe how throughout this process there are many adjustments that can be made. The number of sub images is definitely very important. Given a photo we have to decide in how many sub images it is going to be partitioned. For example, given a photo if it is divided in 9 sub photos and we take a look at each one of them, we may not appreciate any object if the main photo is too zoomed. In the other hand, a more detailed image may show complete different objects or scenes in each one of the sub images. Therefore this means that each picture may need a different partition number for it does not exist a fixed number of subimages for all the pictures. A more visual example explaining how the number of images affect the final precision can be seen in Fig. 2.3.

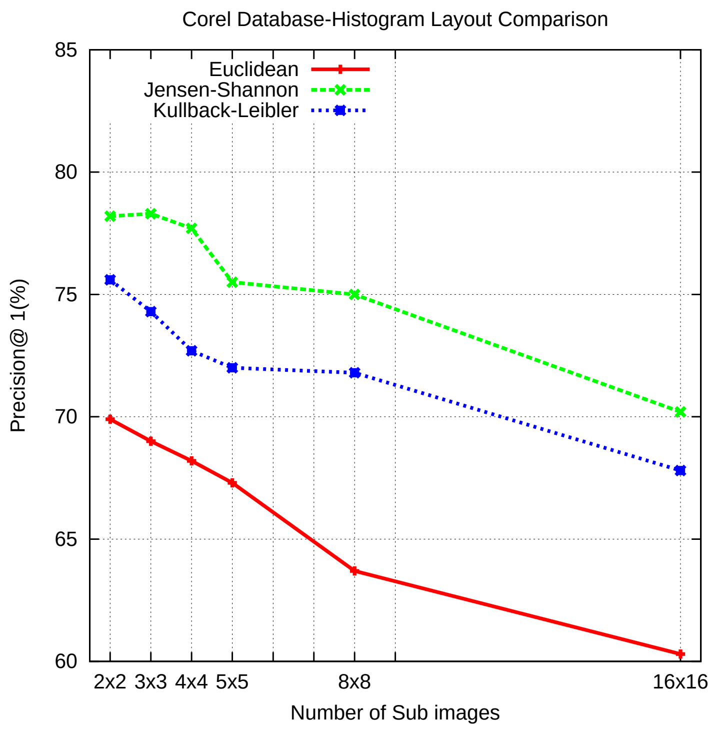


Figure 2.3: Precision@1 for Histogram Layout using Corel Database as seen in [56]. Each number in the x-axis represents the total number of subimages. The different lines represent different distance measures.

A first approach to this method will be to simply divide each image in different sub images and extract the color histogram of each one of them and create a large vector with all the histograms of the image. Then, knowing the number of sub images, compare this large vector of histograms with all the others. What this means is that each of the sub images will only be compared to the one on the same position in different images.

2.4 Local Features

In a similar way to Histogram Layout, Local Features extract partial information of an image. But in this case much more detail is extracted. The main goal of this feature is to compare the different details that appear in all the images.

In this way there is no longer interest in the whole image but in very small regions. So if this small regions are mainly present in other image as well it is probable that they pertain to the same class. This will happen in cases like taking a photo of the same object from different angles, light, position, distance, or cameras, where as before it was not similar at all. In this case the important information relies on those little parts of the image.

This type of descriptor is more common in object recognition task. Object Recognition and CBIR are closely related fields and for some retrieval tasks, object recognition might be the only feasible solution [16]. This features are part of invariant features. A feature is called invariant with respect to certain transformation if it does not change when these transformations are applied to the image. The transformations considered here are rotation, scaling and translation.

The classification process with local features is a two step process: the training phase and the testing phase. In the training phase, local features are extracted from all of the training images, resulting in a huge amount of local features. Then PCA dimensionality reduction is applied. Although precision results for local features is higher than any other method, it is a 2 step process and it makes the comparison slower [56].

2.5 Local Color Histograms

Similar to what is done in local feature's approach, this time the intent is to select a portion of the image and later on extract the color histogram of this portion. What is obtained here is a bit different than before for local feature. By obtaining a histogram there is no importance on the position of each pixel and color so the comparison is between the colors of the parts of an image. The window size therefore, can be a bit bigger since the histogram is invariant to position, and the size of each patch (local feature) will also be less. Before the window size of 11x11 created an output of 363 bins while now the vector will be formed by 64 bins. Quantified Histograms are used to extract the main features of each patch and 64 Quantification was chosen to be the final size of the histogram since the results are really close to those with 512 bins and it weighs much less.

2.6 GIST Descriptor

The GIST descriptor was initially proposed in [45]. The idea is to develop a low dimensional representation of the scene, which does not require any form of segmentation. The authors propose a set of perceptual dimensions (naturalness, openness, roughness, expansion, ruggedness) that represent the dominant spatial structure of a scene. They show that these dimensions may be reliably estimated using spectral and coarsely localized information. The image is divided into a 4-by-4 grid for which orientation histograms are extracted. Note that the descriptor is similar in spirit to the local SIFT descriptor [38]. GIST descriptor is more focused to scene recognition.

2.7 TF-IDF

The TF-IDF representation (term frequency-inverse document frequency) is a well-known method for text-based retrieval [27]. In general, a document and a query can be represented as a term frequency vector $d = (x_1, x_2, ..., x_n)$ and $q = (y_1, y_2, ..., y_n)$ respectively, where n is the number of total terms, x_i and y_i are the frequency (counts) of term t_i in the document vector d and query vector q, respectively. In a retrieval task, given a document collection C, the IDF of a term t is defined by $log(N/nt)$, where N is the total number of documents in C, and n_t is the number of documents that contain the term t. For the TF-IDF representation, all terms in the query and documents vectors are weighted by the TF-IDF weighting formula,

$$d = (tf_d(x_1)idf(t_1), tf_d(x_2)idf(t_2), ..., tf_d(x_n)idf(t_n)) \tag{2.2}$$

and

$$q = (tf_q(y_1)idf(t_1), tf_q(y_2)idf(t_2), ..., tf_q(y_n)idf(t_n)) \tag{2.3}$$

For a simple TF-IDF retrieval model, one simply takes $tf_d(x_i) = x_i$. In our case we are going to use something similar to the TF-IDF representation. It uses the counts of the different terms but it does not use the inverse document frequency since we do not have all the vocabulary to find this value. This experimental representation is explained later on.

Chapter 3

Relevance Feedback

Relevance feedback enables the user to iteratively refine a query via the specification of relevant items. By including the user in the loop, better search performance can be achieved. Typically, the system returns a set of possible matches, and the user gives feedback by marking items as relevant or not relevant. In this particular case, the user starts his query with an example image and is then presented with a set of hopefully relevant images; from these images the user selects those images which are relevant and which are not (possibly leaving some images unmarked) and then the retrieval system refines its results, hopefully leading to better results after each iteration of user feedback [47].

In RISE the user selects the images he/she likes to end up with a bunch of similar images. Those images marked by the user are considered as relevant. On the other hand, those images that remain unmarked are considered non relevant. This is a really important fact since many authors consider another state in the image process; neutral. We thought to delete this state so images can only be relevant or non relevant. Relevant and non relevant images are stored for each user query. Let U be the universal set of images and let $C \subset U$ be a fixed, finite collection of images. The initial query image proposed by the user is $q \in U$. We assume the user has in mind some relevant set of images $R \subset C$. This set is unknown and the systems objective is to discover n images of it, among the images in C. The interactive retrieval process starts with the user proposing a particular query image, $q \in U$. Then the system provides an initial set $X \subset C$ of n images that are similar to q according to a suitable distance measure. These images are judged by the user who provides a feedback by selecting which images are relevant (and, implicitly, which are not relevant). Such feedback information is used by the system to obtain a new set of images X and the process is repeated until the user is satisfied, which means that he/she considers all images X to be relevant.

At any step of this process, let the user feedback be denoted by $F = (Q^{+} \bigcup Q^{-}) \in C^{m}$, where $m \geq n$ is the number of images supervised by the user , $Q^{+} \subset R$ are the images that the user has considered to be relevant and $Q^{-} \subset CR$ are the images that the user has considered to be non-relevant. Let $C_F = CF$ be the set of the images in the collection that have not been retrieved. Usually the initial query is considered to be in the set of relevant images, $q \in Q^+$.

To optimize the user experience, we need to maximize the probability that the images in X are relevant according to F . That is, the images in X should be similar to the images in Q^+ (and may also be similar among each other) and different from images in Q .

$$X = \arg\max_{X \in C^n} \Pr(X|C, q, F) \tag{3.1}$$

Simplifying the notations as explained in [47], the “relevance” of an image X could be computed as:

$$Relevance(X) = \frac{\sum_{q \in Q^+} d(q,x)^{-1}}{\sum_{q \in F} d(q,x)^{-1}} \tag{3.2}$$

3.1 Distance Methods

It is necessary to have a certain measure to tell the quality of the system and to compare several images or words and establish which is the most similar image or word given a query. As it will be shown, distance methods play an important role in classification since some may be more suitable for certain features than others.

3.1.1 Euclidean Distance

Euclidean distance or L_2 is the most used distance and consist of the straight distance between 2 points. The representation of all the points that have the same distance to a particular point in 2-Dimension is a circle, as seen in Fig. 3.1, at 3-Dimension a sphere and so on...

$$d_2(x,y) = \sqrt{\sum_d (x_d - y_d)^2} \tag{3.3}$$

3.1.2 L_1

Euclidean distance calculates squared differences, therefore it may depend too much on those points far away separated. L_1 is more equitative since it only adds differences in absolute value.

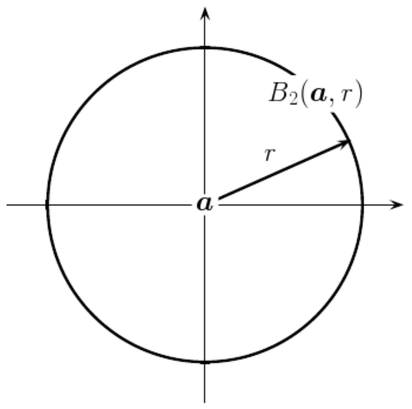


Figure 3.1: In 2 dimension, Euclidean distance has a circular appearance.

$$d_1(x,y) = \sum_d |x_d - y_d| \tag{3.4}$$

3.1.3 L_∞

L_∞ only takes to account the maximum difference of 2 points in absolute value. This distance will not be used but it is important to understand the relationship between L_1 and L_2.

$$d_\infty(x,y) = \max_d |x_d - y_d| \tag{3.5}$$

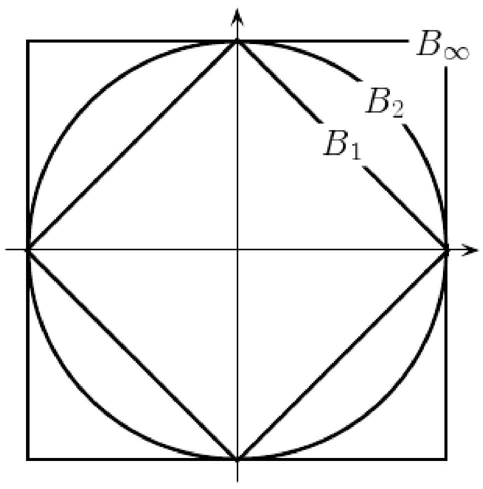


Figure 3.2: Relations amongst L_1, L_2 and L_∞ distance [7].

3.1.4 Jensen-Shannon

This method is commonly used in probability and statistics, and measures the similarity between 2 distributions. In this case we will use it in histograms to determine the degree of likeness. There are several approaches of implementing this algorithm in case it has a 0 denominator. Some authors choose to discard that point in histogram, while others smooth to 0.001. Both Jensen-Shannon and Kullback-Leibler techniques will be to smoothed in 0-denominator case for the rest of the work.

$$d_{JSD}(H,H') = \sum_{m=1}^{M} H_m \log \frac{2H_m}{H_m + H'_m} + H'_m \log \frac{2H'_m}{H_m + H'_m} \tag{3.6}$$

Where H and H' are the histograms compared and m is the number of bins in the vector.

3.1.5 Kullback-Leibler

Kullback-Leibler is the non-symmetric approach of Jensen-Shannon.

$$d_{KL}(H,H') = \sum_{m=1}^{M} H_m \log \frac{2H_m}{H_m + H'_m} \tag{3.7}$$

3.1.6 Distortion

This type of distance is used when comparing Histogram Layouts. Histogram Layout's goal is to compare different small images at different positions that may have similar appearances. But the distances viewed so far do not let the comparison of small regions located at different positions. This method compares each sub image with the ones near it and selects the smallest distance. It has no sense to compare all the sub images appearing in a photo with all the other ones on another image since the cost of the process will increase. By the way this solution allows similar objects or scenes to appear in different positions on another image but still make possible to classify them to the same class. Both histograms (color and texture) did not recorded any position of the pixels and the comparison was made with respect to the colors or texture appearing in the whole image. Histogram Layout extract color characteristics of a concrete zone in the image and thus it was necessary to allow certain freedom when comparing different portions of the images.

3.1.7 Visual Word Distance

This distance was implemented further on and it is based on a threshold method. It aims to be very fast in calculating the distance between 2 vectors and obtaining a good result. It works for Local Feature vectors only since there is no sense in using it in the rest of implemented methods. Once a threshold is defined, this algorithm takes both vectors and compares position by position each one of them as in Fig.3.3. If both positions compared are greater than the threshold defined it counts 1. Therefore what here is really happening is that it is counting how many visual words are in common between 2 images. The result will be positive and greater if it has many of them in common. But the system works with distance so it is necessary to translate this answer for the retrieval to work properly by making it negative. Therefore the more patches in common, the less distance between them.

A variation of this method is used in the textual retrieval engine. It works the same way but instead of having visual words (different patches) we have real weighted keywords. The final result contains how many keywords match in both annotations.

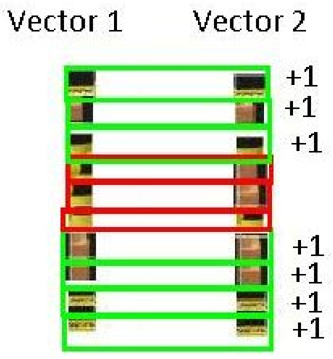


Figure 3.3: Example of the visual word distance applied to 2 different vectors made up of different local features

Upon all the different measures, Jensen-Shannon is the best for almost every different visual feature extraction technique as it is explained on my final degree thesis [56]. Therefore the distance methods used for RISE are: word distance for the textual retrieval engine and Jensen-Shannon distance for the image retrieval engine.

3.2 Evaluation Measures

Let the database $\{x_i, ...x_n\}$ be a set of images represented by features. Given a query image q each database image x_i will be measured given a distance method $d(q, x_i)$. Then the database is sorted according to the lowest distance first. To evaluate CBIR there are 2 main measures; precision P and recall R.

$$P = \frac{\text{NR}}{\text{TR}} \tag{3.8}$$

$$R = \frac{\text{NR}}{\text{TI}} \tag{3.9}$$

Where NR stands for number of relevant images retrieved, TR for total number of images retrieved and TI for total number of relevant images. Precision and recall are two widely used statistical classifications. Precision can be seen as a measure of exactness or fidelity, whereas recall is a measure of completeness. Precision vs. Recall can be explained as it follows:

Example: Asking for 10 objects, we obtain 6 correct results (from the same class) and 4 wrong results. The desired class has 8 elements, so 2 wanted object did not show up! We have a precision of 60 percent (6 out of 10) and a recall of 75 percent (6 out of 8). It becomes clear that the higher the recall, the lower the precision (if we want to obtain all of the interesting results, we get quite a few wrong ones mixed in).

Precision and recall values are usually represented in a precision-recall graph $R \rightarrow P(R)$ summarizing $(R, P(R))$ pairs for varying numbers of retrieved images. The most common way to summarize this graph into one value is the mean average precision.

The average precision AP for a single query q is the mean over the precision scores after each retrieved relevant item:

$$AP(q) = \frac{1}{N_R} \sum_{n=1}^{N_R} P_q(R_n) \tag{3.10}$$

Where R_n is the recall after nth relevant image was retrieved, N_R is the number of relevant images for the query. The mean average precision MAP is the mean of the average precision scores of all the queries.

$$MAP(q) = \frac{1}{|Q|} \sum_{q \epsilon Q} AP(q) \tag{3.11}$$

In other words, to obtain the MAP for a given query all the distances to the rest of images have to be ordered using for example quick sort algorithm. Once this is made, you have to find all the relevant images and divide by the position each one is found, add all of them and divide by the total number of relevant images.

Having 5 relevant images for a query, we will have complete precision when in the ordered vector of images the first 5 positions are the relevant images.

$$\frac{1}{5} \sum \frac{1}{1} + \frac{2}{2} + \frac{3}{3} + \frac{4}{4} + \frac{5}{5}$$

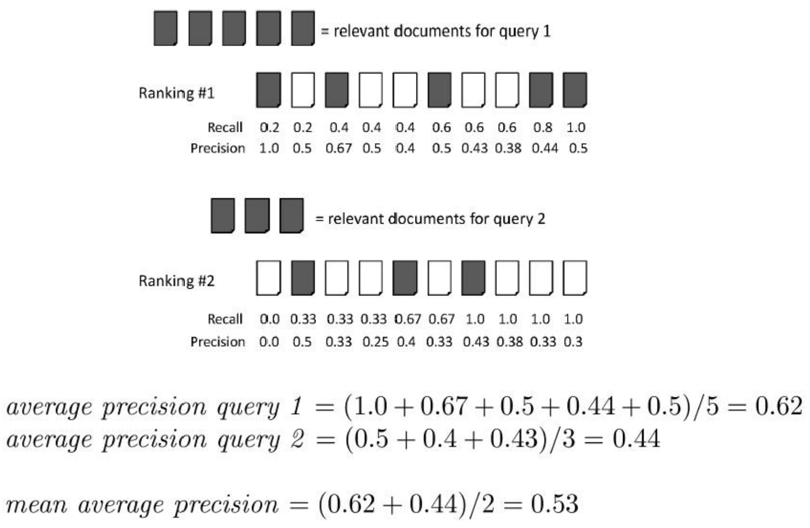

= relevant documents for query 1
Ranking #1
Recall 0.2 0.2 0.4 0.4 0.4 0.6 0.6 0.6 0.8 1.0
Precision 1.0 0.5 0.67 0.5 0.4 0.5 0.43 0.38 0.44 0.5
= relevant documents for query 2
Ranking #2
Recall 0.0 0.33 0.33 0.33 0.67 0.67 1.0 1.0 1.0 1.0
Precision 0.0 0.5 0.33 0.25 0.4 0.33 0.43 0.38 0.33 0.3
average precision query 1 = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62
average precision query 2 = (0.5 + 0.4 + 0.43)/3 = 0.44
mean average precision = (0.62 + 0.44)/2 = 0.53

Figure 3.4: Example explaining both mean average precision (MAP) and average precision (AP).

The main advantage of using this metric is that it contains both precision and recall aspects and it is sensible to the entire ranking.

Precision@Q is a measure of precision. Given a query image and a number of images Q, precision is obtained dividing the number of relevant images retrieved between the total number of images obtained Q.

$$Precision@Q = \frac{1}{|Q|} \sum_{q \epsilon Q} \begin{cases} 1 \text{ if the most similar image is relevant} \\ 0 \text{ otherwise} \end{cases} \tag{3.12}$$

In this case precision is measured, although it is very common to measure error by doing $1 - P(Q)$ where the most frequent Q number is 1, in order to retrieve the most similar image for each query.

Chapter 4

Multimodal Relevance Feedback

Most of research prototypes for web image retrieval use the interactive relevance feedback technique to integrate the keywords and visual features. They rely on the users to complete the combination retrieval and thus add heavy burden to the users. Therefore, the approach that can improve the retrieval accuracy of a search engine without any user feedback is valuable [32].

One of the most interesting issues in multimedia information retrieval is to use different modalities (e.g. text, image) in a cooperative way. The fusion between several methods usually leads to better results in precision. Some queries may be solved by just using visual information. If for example we think about "tigers", the visual part is really important and therefore the visual retrieval technique will perform much better. In other cases, visual information may not help in any case and the textual techniques applied to the annotations will help solving the problem. In many other cases the fusion between this 2 methods will obtain a higher precision. Moreover visual and textual information are usually orthogonal, when the tagging is unsupervised as in the present work, this is, with visual features is easy to find similar looking images. With textual information we are able to find semantically similar images, but in order to find this similarity is important to have a clean complete annotation for each image.

A recent tendency to fuse visual and textual features has been observed in different evaluation tracks such as TRECVID and ImageCLEF , with the belief that these sources of information more than competing are complementary, and that the actual problem may be reduced to finding a way of adequately fusing them. This kind of fusion is a sort of multimedia information retrieval, and it can be performed either as early or late fusion. Research on these two directions has already been developed, but current performance of these methods remains poor, showing the need of research to find better fusion alternatives and to select better individual relevant models.

We talked about the fusion between several methods. Depending on when the fusion occurs on the process we have several approaches:

4.1 Fusion by Refining

The first approach is really obvious and was already implemented in the past version of RISE (see Fig. 4.1). The user introduces a text query in the web browser. Images have small captions or annotations, therefore the system searches for those images that textually correspond with the query. Then the system provides the user with visual examples and the user selects those images he considers relevant. This is a multimodal approach since first text information is used in the first iteration and then for the next iterations only visual information is needed. This type of fusion is also called fusion by refining, since textual information is only used to provide a certain amount of pictures and later by the use of visual techniques we refine the results.

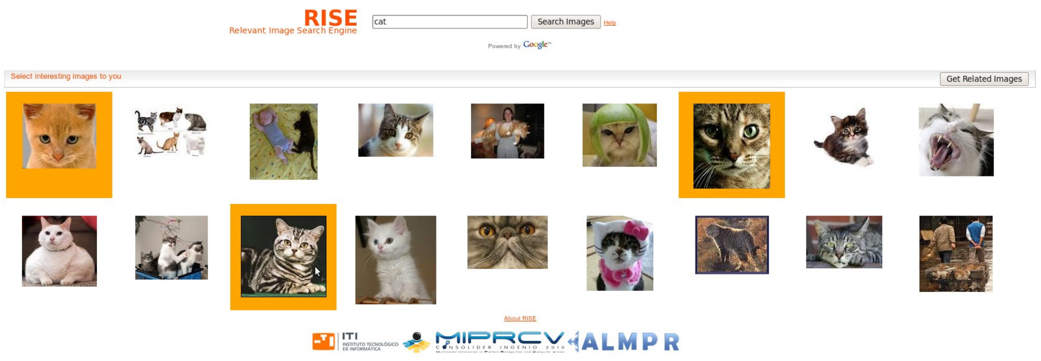


Figure 4.1: Previous RISE version with several images selected as relevant.

4.2 Early Fusion

It is a supervised learning process in where images are trained manually and classified into different classes. This type of fusion is commonly used in automatic annotation problems. It consists in linking image features with semantic concepts. After each image is elected into a class, a binary classifier is trained to detect the class. When a new image comes, the visual similarity to each class is computed . More or less, early fusion tries to discover the statistical link between visual features and semantic concepts using unsupervised learning methods [64].

There is a common task we participated in named ImageCLEF 2010. In this event, we had 57 classes and a set of 9000 training images correctly annotated. In this automatic image annotation task we were supposed to extract strong relations between visual features and keywords. Then we trained with a classifier and annotated the test images. We tested with very different visual features (GIST, SIFT, Color Histograms, Texture, Concatenations...) and different classifiers. The results were not as expected and we ended in an average position. We learned that early fusion is a hard approach due to some reasons:

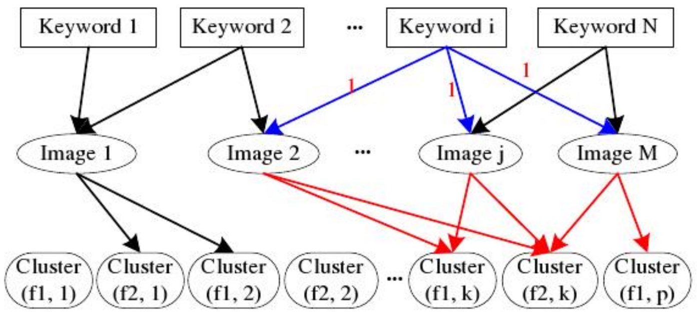


Figure 4.2: Typical architecture in Annotation Based Image Retrieval systems with early fusion, where the keywords are combined with low level visual features.

1)

The annotations of the images in ImageCLEF corpus often contain keywords that are not strongly associated with particular visual features. They correspond to abstract concepts. Examples of such keywords are “friendship”, “north” or “tournament”.

2)

Even if there are some relationships between keywords and visual features, these relationships may be difficult to be extracted because there are huge amount of possible visual features. In fact, visual features are continuous. Even if we use some discretization techniques, the number is still to high to be associated to some keywords. For example, for a set of images associated with the keyword “water”, one would expect to extract strong relationships between the keyword and the texture or color. However, water in many images may only take a small portion or region of the image. There may be many other objects in the image making it really difficult to isolate the typical features of “water”.

4.3 Late Fusion

Late fusion is the approach chosen for RISE. This is motivated by the hypothesis that two images with a very strong visual similarity should share some common semantics [34]. Late fusion of independent retrieval methods is the simpler approach and a widely used one for combining visual and textual information for the search process. Usually each retrieval method is based on a single modality, or even, when several methods are considered per modality, all of them use the same information for indexing/querying (see Fig. 4.3). The latter reduces the diversity and complementariness of documents considered for the fusion, as a consequence the performance of the fusion approach is poor [24]. In multimedia image retrieval, the sources of information are visual features extracted from the image and textual features in the form of associated captions. These sources of information have been mostly used individually and separately. Textual features have proved to be more effective for this task than their visual counterpart, and systems based only on these features tend to significantly outperform systems based merely on visual features, which perform poorly. However, a problem generally found in both cases is the lack of generalization, which makes systems fail with varied sets of queries. A more specific diagram showing this behavior for only visual and textual retrieval engines can be seen in Fig. 4.4.

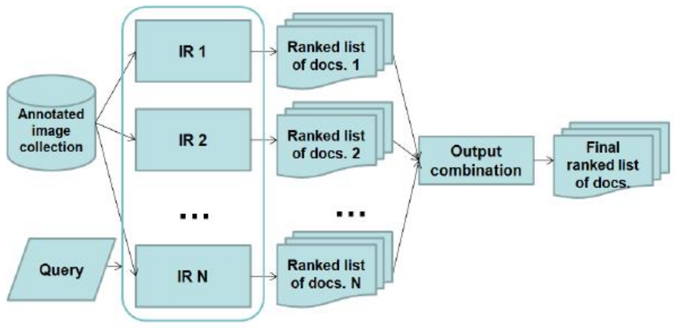


Figure 4.3: Graphical General Diagram showing Late Fusion for heterogeneous methods. The output is combined for obtaining a single list of ranked documents.

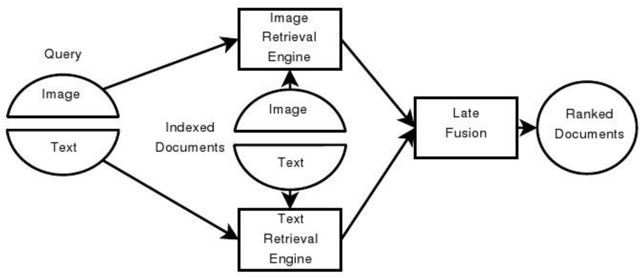


Figure 4.4: Late Fusion example as implemented in RISE2.

Each time a query q is sent, the N different retrieval methods work separately. The result for each of the retrieval engines is a set of N ranked list of documents or images. The information of the N ranked lists is used for obtaining a single list of ranked documents, which is returned to the user in response the the query q. The final list is obtained by assigning a score to each document appearing in at least one of the N lists. A high score value indicates that the document is more likely to be relevant to query q. Documents are sorted in decreasing order of their score and the top$k$ documents are considered for the final list of ranked documents. For this work we consider a simple (yet very effective) score based on a weighted linear combination of the documents rank through the different lists. The proposed score takes into account redundancy of documents and the individual performance of each retrieval method. Diversity and complementariness are bring to play by the heterogeneousness of the considered IRMs

(Independent Retrieval Methods), while redundancy is considered through the use of several IRMs per modality. We assign a score to each document d_j in at least one of N lists $L\{1, ..., N\}$ as described by the following Equation :

$$Score(d_j) = (\sum_{i=1}^{N} 1_{d_j \in L_i}) \times \sum_{i=1}^{N} (\alpha \times \frac{1}{\psi(d_j, L_i)}) \tag{4.1}$$

Where i indexes the N available lists of documents; $\psi(x, H)$ is the position of document x in ranked list H; 1_a is an indicator function that takes the unit value when a is true and α_i, with $\sum_{k=1}^{N} \alpha_k = 1$, is the importance weighting for IRM i. α_i's allow including prior knowledge into the retrieval process in the form of the confidence we have on each IRM. If we have only 2 IRMs (visual an textual engines only) we have that one will be multiplied by α and on the other by $1 - \alpha$ since $\sum_{k=1}^{2} \alpha_k = 1$. Respecting the idea in the general equation presented above, we can use a simpler equation for just 2 heterogeneous IRM's (text retrieval and image retrieval):

$$Rank = \alpha R_v + (1 - \alpha) R_t \tag{4.2}$$

Where R_v is the visual ranking and R_t is the textual ranking.

Documents appearing in several lists at the top positions will receive a higher score, while documents appearing in few lists or appearing at the bottom positions most of the times will be scored low. Eventually, only relevant documents will be kept. The more relevant a document is to a query the higher will be its position in the final list. As we can see this is a simple and intuitive way of merging the output of IRMs proved to be very useful in practice. A more graphical version can be seen in Fig. 4.5

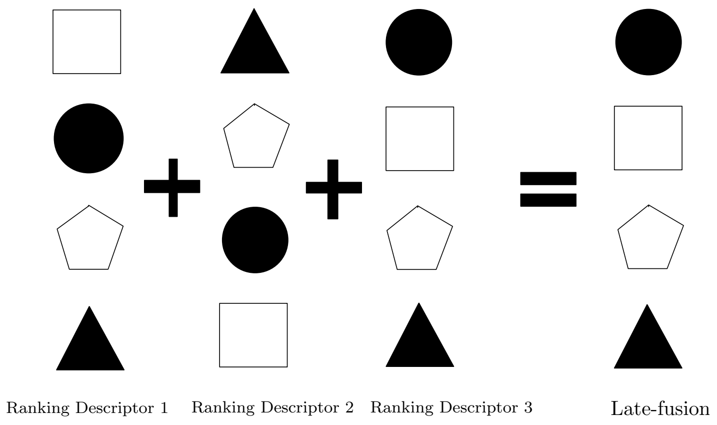
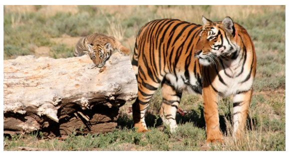


Figure 4.5: Example of late fusion ranking algorithm.

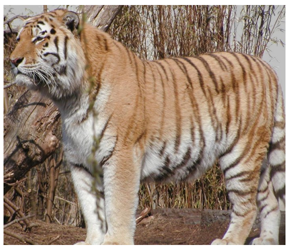
The α appearing in the main formula is really important since depending on this value we give more importance or credibility to the results obtained from one retrieval engine or another. One of the main experiments will be to observe how changing the α value affects the final precision in the system.

4.4 Intuitive Example

The following example shows how visual retrieval and text retrieval techniques behave by themselves.



tiger, global, team, brandt, sourcing, services, portfolio, home, product, contact, shanghai, cost



tiger, den, tigeren, kan, det

Figure 4.6: Example of visually relevant images.

In Fig. 4.6 one can observe how similar the images look. In fact these 2 images are considered relevant to the visual retrieval algorithm explained in previous section. For the human eye is easy to see that both pictures contain tigers with similar brownish-nature background. But for the textual retrieval algorithm it is really difficult to determine this 2 pictures are relevant to one another. If we observe the annotations we can see how one of them contains enterprise words such as "sourcing" or "portfolio", mainly because it corresponds to an image extracted from the "about page" of a China exporting and manufacturer enterprise (http://www.tigerglobal.co.uk). The other annotation contains words in german or danish collected from Flickr Database. It is impossible for the textual retrieval engine to determine with these 2 annotations if they are relevant to one another. This demonstrates it is a visual query. A query that can be solved with the visual engine better than with the text engine, since mostly, all the tigers look really alike and the scene we are looking for has similar colors and backgrounds. But what if the pictures are really different between each other? What if we are searching more for semantic concepts instead of visual concepts? Here is an example Fig. 4.7:

In the textual example one can observe how different the images look. In fact, one picture is taken in the beach with light colors and people in dark shirts with kites and in the other the people wear white shirt and stand on green grass. In the other hand the descriptions from each of the images look really similar. Although they are different words we do have some of them that match. Words like "team", "members", "championship" or "season" appear in both pictures. Showing this examples, we are trying to visually show the problem in CBIR. At one point, visual features can not help any more, even with new feature extraction techniques, there is a point in which a system needs other type of information to raise the overall precision. By introducing textual techniques and making a multimodal retrieval engine we intend to increase this precision and therefore provide better results to the user.

team, flame, kite, world, members, sport, competition, flying, championship national, season, man, festival precision, france, person, event

team, golf, fordham, season, score, championship, record, final, match, members, round squad, tournament, morgan, mike, finished play, golfers, federation, club

Figure 4.7: Example of textually relevant images.

4.5 Proposed Approach: Linear Fusion

At a given point, visual retrieval engine and text retrieval engine need to cooperate in order to obtain a higher precision. But there is a main problem when using late fusion, the α in the previous equation of Late Fusion needs to be fixed. This means that each time we use the late fusion approach we have to establish which α value we would like. α is a value between 0 and 1 which states how important the results are from each of the retrieval engines, in our case, text engine and visual engine.

As we can see on the previous intuitive example, depending on the type of query we may need a different α value. This means there does not exist a unique value that works for all the queries. Some queries may need high values (visual) while other may need very low values (textual). Intuitively, we may predict that most of the queries need visual information as well as text information, leading to an average α value far from the extremes.

We propose to select the value of the α by means of solving this maximization problem:

$$\arg\max_{\alpha} \sum_{x \in R} \sum_{y \in NR} Rank(y) - Rank(x) \tag{4.3}$$

Where $Rank$ is the late fusion of both rankings R_v and R_t, and R and NR is the set of relevant and non relevant images respectively. As it was explained before, the final ranking in the late fusion approach is made by all the rankings of the different methods, $Rank = \alpha R_v + (1 - \alpha) R_t$.

The idea is for each iteration the system proposes a set of images which later on the user selects as relevant or non relevant. Without disturbing the user in the process, we can compare and see how good was our prediction. This way the value is updated with the users intention. Without the user knowing, he or she participates in the process of helping the system to show more precise images.

Given the ranking of the visual engine R_v and text engine R_t, we will have the most similar images at the top of each ranking. For example for the visual ranking R_v the 10 most visual similar images will be located in positions 0 to 9. Therefore the approach tries to leave the relevant images at the top positions and the non relevant at the lower positions.

$$\arg\max_{\alpha} \sum_{x \in R} \sum_{y \in NR} \alpha(R_v(y) - R_v(x)) + (1-\alpha)(R_t(y) - R_t(x)) \tag{4.4}$$

In the above equation we pretend to choose the α where the difference between the relevant images and the non relevant images is higher. This is for each α tested, we want each relevant image to be ranked in the first positions. Moreover we also want the difference of this relevant image with the non relevants to be as high as possible. This equation intuitively searches for the area under the ROC curve, trying to make the difference between the non relevant images and the relevant images very high by ranking them as far as possible. Given the user selection as test and our previous hypothesis as training, we can work out a new α which maximizes this result by making an exhaustive search throughout the different α values in $\{0, 1\}$.

Chapter 5

RISE Database

In order to test the different proposals it is necessary to build an image engine. Building an engine is a long task with several difficulties. Previously, the demo was built over the Google Database, therefore nothing was kept in the system (www.rise.iti.upv.es). We had problems in accessing all the times that we wanted to those images and this made the system fail continuously. Moreover accessing the images and converting them to our format and comparing them to see which where most similar was a process too heavy to be done online. So the idea was to build a system where we kept all the information, all the images converted and therefore suit faster the user queries. For the database to be important it is necessary to have a large amount of images correctly labeled. For this process to be done there are two clearly separated parts: downloading the images and labeling those images.

5.1 Downloading the Images

First of all it is necessary to design a database in paper and decide the tables and fields of the future system.

Once this is done it is important to decide how the information is going to be downloaded. This process is a little bit more difficult. In this case there were several approximations.

The Database was build in MySQL. Now the problem was, in order to have a large amount of information kept in this database we need an automatic process of downloading the images and labeling them since doing this process manually is not efficient. Labeling the images will be studied in the next section. For each image downloaded (around 1500 per query) we make a thumbnail of it and convert the original picture to our feature vector. This vector has histogram layout features. Remember histogram layout consisted in dividing each image in smaller images and extract a histogram from each on of them. In this case we divide each image in 9 smaller parts

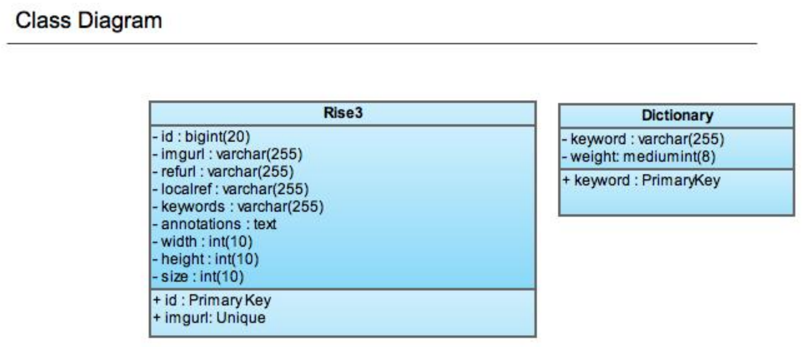


Figure 5.1: Database elements.

and we extract a quantified histogram of size 64 for each one of them. This makes a 576 bin vector. Instead of saving all positions we decided to only keep the position and the value. This representation saves some space since it is a sparse vector.

Keeping the thumbnail saves a lot of space and makes the engine go fast when retrieving images. Downloading an image is simple with the "wget" command in Linux, but, where to start downloading images from? As it is well known, Google uses a spider that crawls through internet and downloads the images from each web, then it follows all the links in that page and goes to another page continuing with this process. But in our case we decided to use the information kept by the 3 greatest search engines in the net, Bing, Google and Yahoo. The three of them have image search engines already working with weighted images, so why not use this information? The idea was, given a word, search this word with the three different engines, retrieve the results and download the correspondent images.

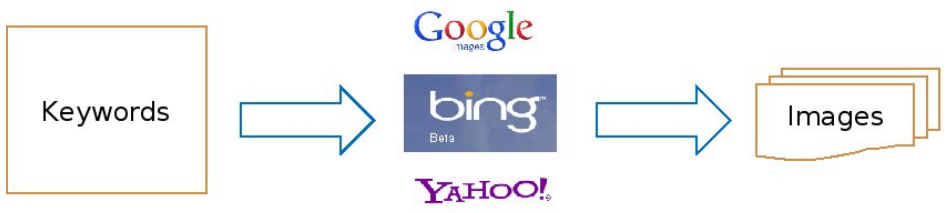


Figure 5.2: Downloading process.

But another question aroused:which word or words will we use to retrieve the different images? This is an important decision since each word will act as a seed. This is, each word in a list of words or keywords is queried upon the 3 engines. Then all the images dealing with the query will be downloaded. The first approximation was to download all images in the english dictionary. But the dictionary has many similar words that don't have a real visual meaning such as "along" or "alongside". So we created a list of stop-words to prevent the meaningless words to participate as seeds.

Finally the list of keywords was enriched with the 1000 most common english nouns, resulting in a particular dictionary.

But a new problem appeared, Google affirms that the brand names are the most queried words in internet. Unless the brands are “apple” or “jaguar”, the dictionary won’t have those words. Some examples of brands not appearing in dictionary may be “toyota” or “iphone”. This particular case will be solved on the following section.

5.2 Labeling the Images: Related Work

A critical point in the advancement of content-based retrieval is the semantic gap, where the meaning of an image is rarely self-evident. The aim of content-based retrieval systems must be to provide maximum support in bridging the semantic gap between the simplicity of available visual features and the richness of the user semantics. The main question that needs to be answered in this concept is: how does one effectively map the extremely impoverished low level data representation of the image to the high level concepts so easily understood by human beings? This is the reason of the semantic gap issue in Computer Vision (see Figure 5.3).

The context in which an image appears can be abstracted from the containing HTML document using a method known in the information retrieval community as Latent Semantic Indexing (LSI). LSI works by statistically associating related words to the conceptual context of the given document. This structure is estimated by a truncated singular value decomposition (SVD) [6].

When we retrieve images based on annotations, the quality of the annotations should be taken into account. We assume that manually assigned annotations are usually more reliable than automatically assigned ones. Because of the cost, however, annotations are sometimes assigned automatically. Two types of methods are frequently used to assign textual information to images. One method is based on information extraction techniques. For example, some textual information corresponding to images on the WWW can be extracted from their surrounding texts or anchor texts linked to the images. If the extraction rules are carefully designed, the acquired annotations may be relevant to the images. However, because there are usually exceptions in the data that are not consistent with the assumptions, the extracted annotations may contain noise. The other method is based on classification techniques. The goal of this method is to link visible image features to words but it is really difficult to capture content and contextual information from images that do not have any associated image features. The development of procedures for assigning keywords to a given image is an active research topic [30].

Another interesting technique is tested by Google. It is called Image Labeler (see Figure 5.4), in this web service users compete against each other to assign the most relevant labels to randomly selected images from Googles index. Given that the search

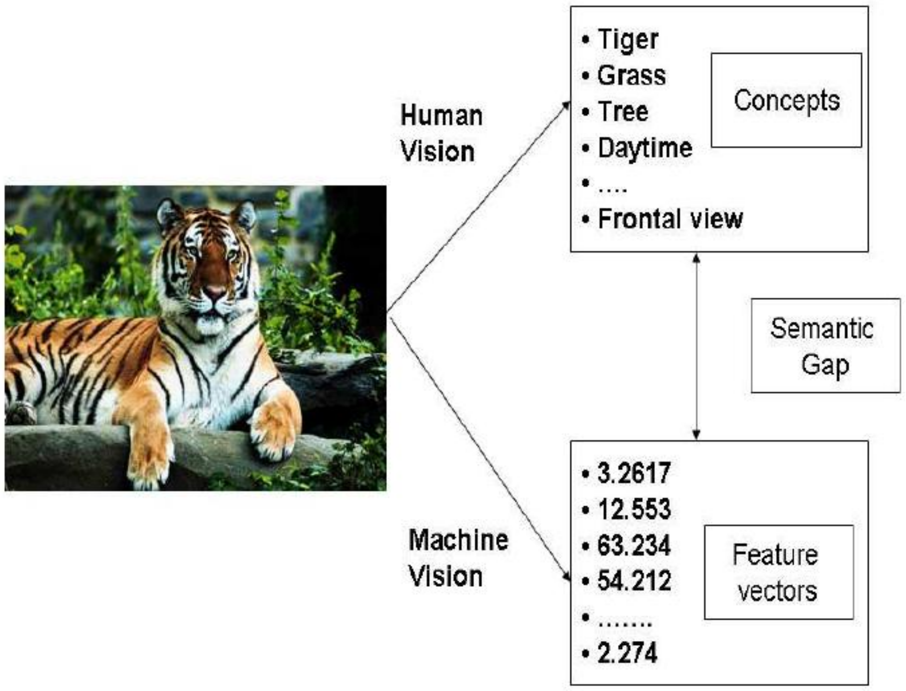


Figure 5.3: An illustration of the semantic gap problem. Here we have an image of a tiger at rest. The concepts likely to be noticed by a human looking at the image are shown in the box at the top right. On the bottom right we have the feature vector representation of the same image, detailing properties such as position, color, shape and so forth.

giant is using this manual means of image tagging demonstrates the difficulty inherent in the automated image tagging process particularly with regard to scaling those models suggested in the literature to multi-million scale image libraries. A great deal of work needs to be completed before the models of the research literature can be migrated as robust and scalable technologies to the commercial world.

So now we have got a great amount of images saved in the system but we still don't have a way we to access them easily. An image has to be labeled with the main concepts appearing in the photo as explained before, but without human aid is difficult to have a precise description. The idea is to make an automatic process based on information extraction techniques over the WWW, where the images are tagged by a computer without the human interaction, calling this process Unsupervised Image Tagging.

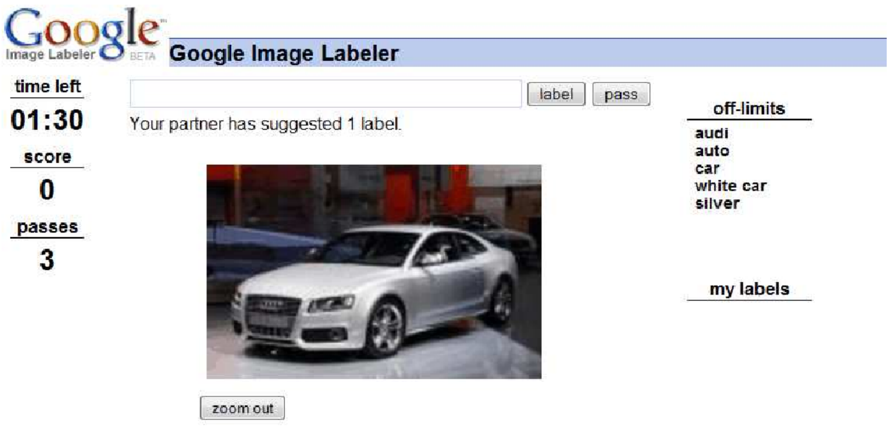


Figure 5.4: Car labeling in Google Image Labeler.

5.3 Proposed Approach: Unsupervised Image Tagging

The process starts once all the images are downloaded, each image has associated a web reference. This web reference will help us in the image tagging. The main code was done in Python and a library called Beautiful Soup. Beautiful Soup is a Python HTML/XML parser designed for web scraping. This module helps in stripping the HTML tags out of the web.

For each of the web addresses, we collect all the information with Beautiful Soup. The first part of the process consists in deleting from the web all those unnecessary elements such as tables or scripts. Next step is to remove all the HTML tags that may be contained to end with a full-plain text array. All words in the same HTML document may not be equally relevant to the document context. Hence words appearing with specific HTML tags are given special importance by assigning a higher weight as compared to all other words in the document. The system assigns different weights to the words appearing in the title, headers and in the “alt” fields of the “img” tags along with words appearing in the main URL or thew image URL (see Table 5.1). These weight values have been fixed according to the likelihood of useful information that may be implied by the text. Weighting selectively the words appearing between various HTML tags helps in emphasizing the underlying information of that document. A dictionary is created with the term frequency of each word that appears in the web. We decided to use some of the search engine’s information. Depending on which page the image is retrieved originally, more points can be added to the word frequency dictionary . This is, for example, if we search “cat” in Google Images, the first picture shown is more important than the last photo shown in the 10th page.

If the keyword satisfies any of this conditions, more points will be added to the term frequency dictionary. Before writing each word into the database, each of them are

HTML Tags	Weight
URL	4
Img URL	4
Title in Image	2
Alt in Image	3
Src in Image	4

Table 5.1: Word weights based on HTML tags.

lower cased and all the unwanted characters are removed. This is a long process but it is necessary to think that not all the web pages follow the same structure, therefore it is mandatory to go through all this steps in order to avoid dirty annotation. Only those words which weights are above a threshold (initially defined) are written into the database. These words are written with their term frequency since later on, this number will help to determine how important an image is to a concept. The goal is accomplished if by reading the annotations, one can picture mentally how the image looks like.

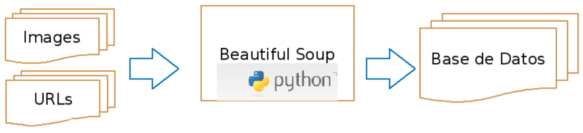
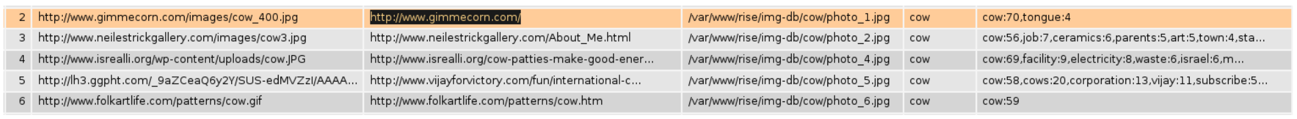


Figure 5.5: Labeling Process.

Each time an image is inserted into the database several fields are added:

ID	Unique number referenced by the database.
Img URL	This URL is where the image is saved originally in the web server.
Refurl	Base path where the image comes from. For example “http://www.cat-pictures.co.uk/cat.jpg” comes from “http://www.cat-pictures.co.uk/”.
Localref	Image path in local disk.
Keywords	Seed word used during the process.
Annotations	Process previously defined. Note that the seed word may not be in the first place
Width, Height, Size	Original image information.

Table 5.2: Word weights based on HTML tags.

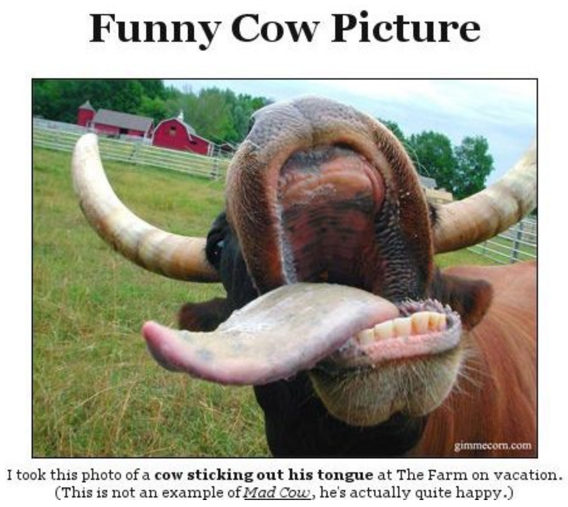


2	http://www.gimmecorn.com/images/cow_400.jpg	http://www.gimmecorn.com/	/var/www/rise/img-db/cow/photo_1.jpg	cow	cow:70,tongue:4
3	http://www.neilestrickgallery.com/images/cow3.jpg	http://www.neilestrickgallery.com/About_Me.html	/var/www/rise/img-db/cow/photo_2.jpg	cow	cow:56,job:7,ceramics:6,parents:5,art:5,town:4,sta...
4	http://www.isrealli.org/wp-content/uploads/cow.JPG	http://www.isrealli.org/cow-patties-make-good-ener...	/var/www/rise/img-db/cow/photo_4.jpg	cow	cow:69,facility:9,electricity:8,waste:6,israel:6,m...
5	http://lh3.ggpht.com/_9aZCeaQ6y2Y/SUS-edMVZzI/AAAA...	http://www.vijayforvictory.com/fun/international-c...	/var/www/rise/img-db/cow/photo_5.jpg	cow	cow:58,cows:20,corporation:13,vijay:11,subscribe:5...
6	http://www.folkartlife.com/patterns/cow.gif	http://www.folkartlife.com/patterns/cow.htm	/var/www/rise/img-db/cow/photo_6.jpg	cow	cow:59

Figure 5.6: Database view after inserting a couple images.

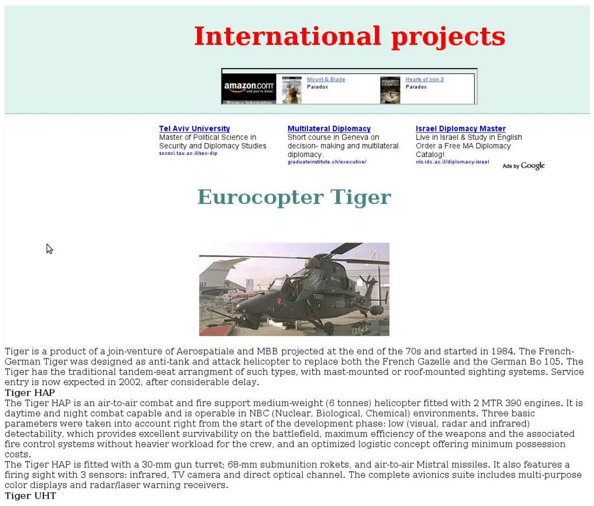
As it was questioned before, what happens with the brand names and all those words that don’t appear in the dictionary? Well this words will not be used to start querying the different engines but this does not mean they will not appear in the database. While scraping the web page, we don’t select only the words that appear in our word list, we select all the words and only delete those appearing in the stop word list. Words like "Volkswagen" or "google" will not be used as seeds since they are not in the english dictionary, but if they appear in other web pages, they will end up as annotations.

The main goal of this automatic tagging process is to have a full database with images and annotations. These annotations have to be really descriptive and accurate since the queries initially will be made using this information. If an image can be pictured while looking at the annotations, this will mean it has rich information about how the image looks like.


Funny Cow Picture

I took this photo of a **cow sticking out his tongue** at The Farm on vacation. (This is not an example of *Mad Cow*, he's actually quite happy.)

cow:70, tongue:4


International projects

Tel Aviv University
Master of Political Science in Security and Diplomacy Studies
socsci.tau.ac.il/sec-dip

Multilateral Diplomacy
Short course in Geneva on decision- making and multilateral diplomacy.
graduateinstitute.ch/executive/

Israel Diplomacy Master
Live in Israel & Study in English Order a Free MA Diplomacy Catalog!
ris.idc.ac.il/diplomacy-israel

Ads by Google

Eurocopter Tiger

Tiger is a product of a join-venture of Aerospatiale and MBB projected at the end of the 70s and started in 1984. The French-German Tiger was designed as anti-tank and attack helicopter to replace both the French Gazelle and the German Bo 105. The Tiger has the traditional tandem-seat arrangment of such types, with mast-mounted or roof-mounted sighting systems. Service entry is now expected in 2002, after considerable delay.

Tiger HAP

The Tiger HAP is an air-to-air combat and fire support medium-weight (6 tonnes) helicopter fitted with 2 MTR 390 engines. It is daytime and night combat capable and is operable in NBC (Nuclear, Biological, Chemical) environments. Three basic parameters were taken into account right from the start of the development phase: low (visual, radar and infrared) detectability, which provides excellent survivability on the battlefield, maximum efficiency of the weapons and the associated fire control systems without heavier workload for the crew, and an optimized logistic concept offering minimum possession costs.

The Tiger HAP is fitted with a 30-mm gun turret; 68-mm submunition rokets, and air-to-air Mistral missiles. It also features a firing sight with 3 sensors: infrared, TV camera and direct optical channel. The complete avionics suite includes multi-purpose color displays and radar/laser warning receivers.

Tiger UHT

tiger:16, air:11, missiles:6, tank:4, mountain:2

Figure 5.7: 2 web pages and their annotations after the automatic image tagging process.

In above Fig. 5.7 one can observe how the annotations resume the web page topic. The left web page shows a cow with a big tongue. The final annotation has this exact 2 keywords. The other web page talks about different helicopters, one in particular named "tiger". The annotation of the second web, has some keywords such as "missile" or "tank" which makes us think that this web talks about a different type of tiger than the ones we are normally used to. This disambiguation is really important since many words are polysemic (have different meanings although words are written the same way). If we did not have images annotated correctly we will never know if the engines work correctly or the descriptions are not really precise.

Chapter 6

Experiments

Before doing the experiments it is mandatory to have a corpus to simulate the users relevance feedback. For this reason we select a set of 21 queries with 200 images per query. A brief description of each query can be found at Table 6.1.

Query	Description
Banana	Real yellow bananas. Cartoon or drawn bananas are considered not relevant.
Baseball1	Baseball balls.
Baseball2	Baseball players in the baseball pitch.
Bike1	Catalog motorbikes but not only with white background.
Bike2	Catalog bicycles. People can appear but the bike appears completely.
Bird	Birds flying.
Car1	Real cars with no people or other cars around. Car appears completely.
Car2	Car engines.
Corn	Corn pictures with no human hands.
Cow	Real cow on mountain, land or grass.
Gun	Catalog guns with white background.
Hat	People wearing hat and looking at the camera. Hats by themselves are not relevant.
Horn	Trumpets.
Lake	Panoramic landscape pictures with or without houses around the lakes.
Rain	Pictures where one can appreciate rain falling.
Snake	Not only snake heads but also some body must appear in the picture.
Team	Group of people.
Tiger1	Full body real tigers.
Tiger2	Tiger Woods by himself. Either playing golf or not, but alone.
Tiger3	Tiger sneakers or shoes.
Volcano	With lava or fire around.

Table 6.1: Queries and their descriptions as they appear in the test corpus.

We have a set of images, we know which of them are relevant or irrelevant to a topic, and we have the descriptions for each image. This way we can create a user feedback automatically. This will avoid us repeating the whole process manually for each change applied in any algorithm or the features. In the experiment we will fake a user who wants N images to be seen at a time. Therefore in each iteration he will see N images and judge if those are relevant or not ($Precision@N$), depending on the criteria specified for each query. Depending on the number of images shown N, we will have different results.

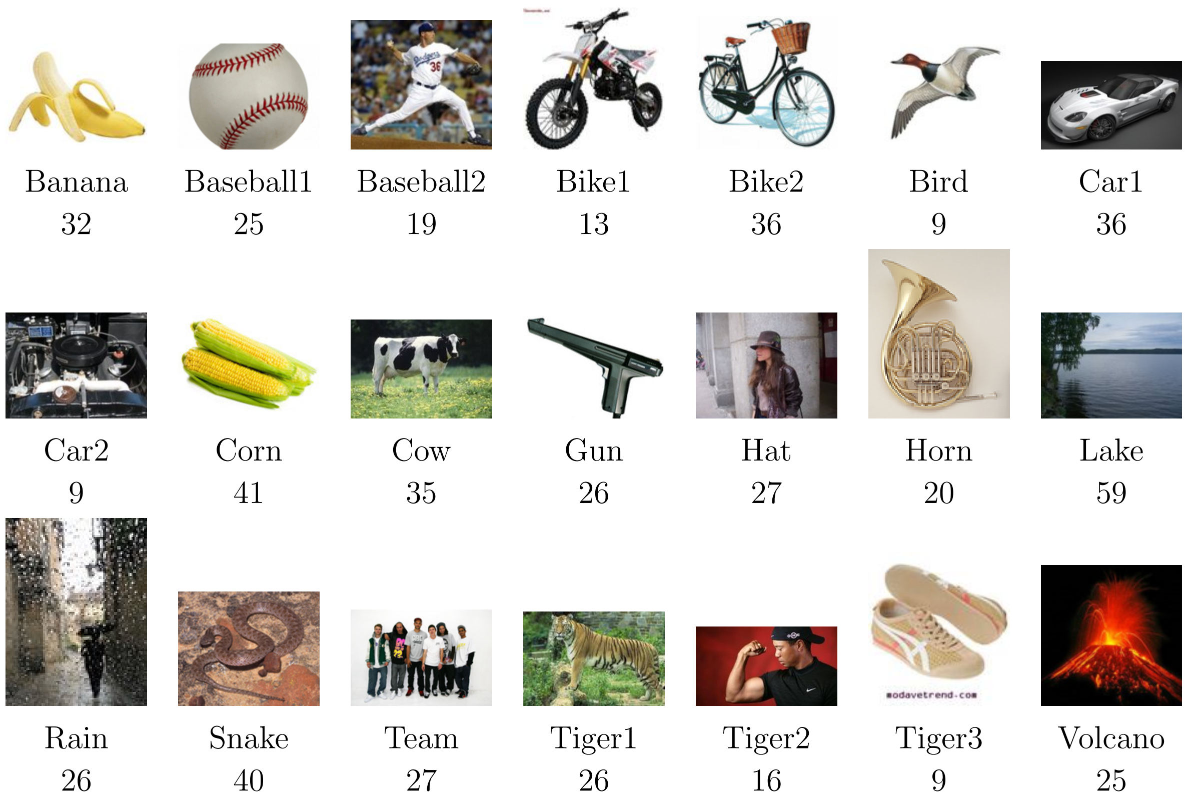


Figure 6.1: Example of relevant images (and total number of relevants) from each of the 21 queries in the test corpus.

6.1 Results for each Concept

The first experiment we did was to test how the different values of α affected the system. As explained on Section 3.3, this α is a parameter between $\{0, 1\}$ which is in charge of giving less or more importance to each retrieval engine. If $\alpha = 1$ (Visual Percentage=100%) it means we "trust" completely the visual engine while if $\alpha = 0$ (Visual Percentage=0%), the final ranking will only depend on the textual engine. Our hypothesis is that the values in between $\{0, 1\}$ will raise the overall precision of the system. This hypothesis is based on the examples we gave before, where depending on the type of query (visual or semantic) we had different results using different retrieval engines.

For iteration 0 (Table. 6.2) we have the same value for all the different α. If we remember, this iteration is only text based. The user introduces a query and the system searches for the words matching this query. No visual information is collected or needed therefore there is no slope by changing α's value. Some queries have a precision equal to 0, this is because none of the images present in the initial query are relevant to the user.

For the following iterations the parameter α does affect the precision, so the results are shown using plots showing the precision with respect to the Visual Percentage. As

Query	Precision@20 (%)
Banana	10
Baseball1	10
Baseball2	10
Bike1	0
Bike2	10
Bird	0
Car1	30
Car2	0
Corn	10
Cow	0
Gun	10
Hat	20
Horn	10
Lake	0
Rain	0
Snake	10
Team	0
Tiger1	0
Tiger2	0
Tiger3	0
Volcano	10
MP@20	6.19

Table 6.2: Iteration 0 of User Feedback, showing all the concepts.

we said earlier, α is a value between 0 and 1, where Visual Percentage (%) is a value between 0 and 100; $\alpha = \frac{VisualPercentage}{100}$.

All the 21 different concepts $C = 21$ don't fit in the same graph, for this reason we have 2 graphs showing the same iteration dividing 10 concepts on one graph and 11 on the next one. For each concept c we have a $Precision@N$ value. In both of the graphs we can see there is a line named "MP@N" where N is initially 20. The mean precision for all the concepts is $\frac{1}{|C|}\sum_{c\in C} Precision@N(c)$.

This is the first iteration (Fig. 6.3 and Fig. 6.2) in which both retrieval engines cooperate in the final result. The reason of the slopes for being so squared is the Precision@20. When measuring this way it is very common to have this type of lines and values. Some of the queries perform better with only the text retrieval engine ($\alpha = 0$) and others better with the image retrieval engine ($\alpha = 1$). But what is interesting for the thesis is to observe if the collaboration of both methods helps in obtaining more relevant images. At iteration 1, very little queries obtain its maximum precision value on points not located at the extremes. This is caused due to the little information present in the system. Only 1 iteration has been done therefore we only have 20 previous images saved on the system. In this 20 previous images we can have some relevants or perhaps all of them are non relevant as explained before, but no matter this fact, we have information about what the user wants or does not want to see. As the iterations pass by, this information increases and the algorithms "learns" the user needs. It is expected to see the precision grow as the user iterates the query.

In Fig. 6.5 and Fig. 6.4, 2 of our hypothesis have been confirmed. This iteration has higher mean precision as the MP@20 line shows. Another important fact is that at this point, more queries have found its maximum precision value at intermediate α

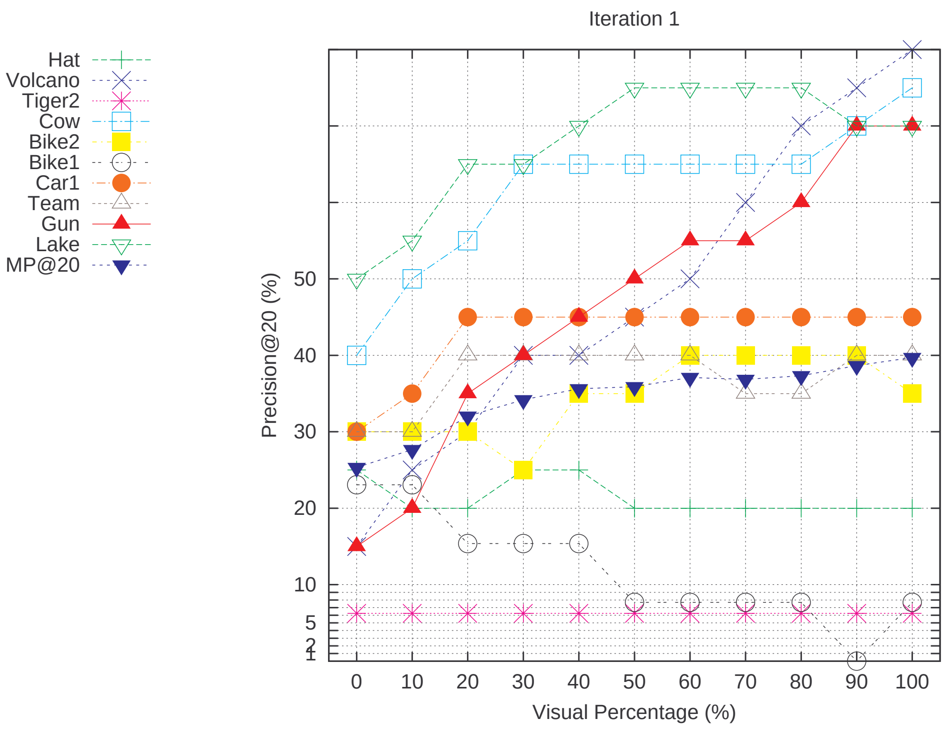


Figure 6.2: Iteration 1 of User Feedback, first 10 concepts.

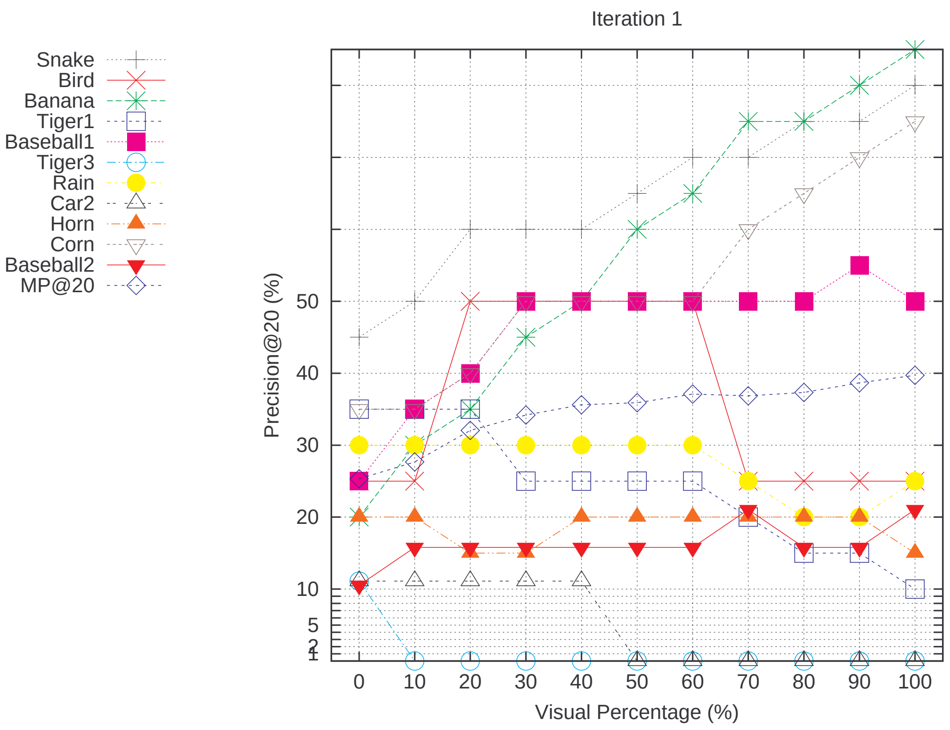


Figure 6.3: Iteration 1 of User Feedback, showing last 11 concepts.

values. Both engines are collaborating and "sharing" information with one another and this is making some difficult concepts to achieve a higher value than in the extremes. At iteration 1, the mean precision@20 line had its maximum at $\alpha = 1$ value. In this

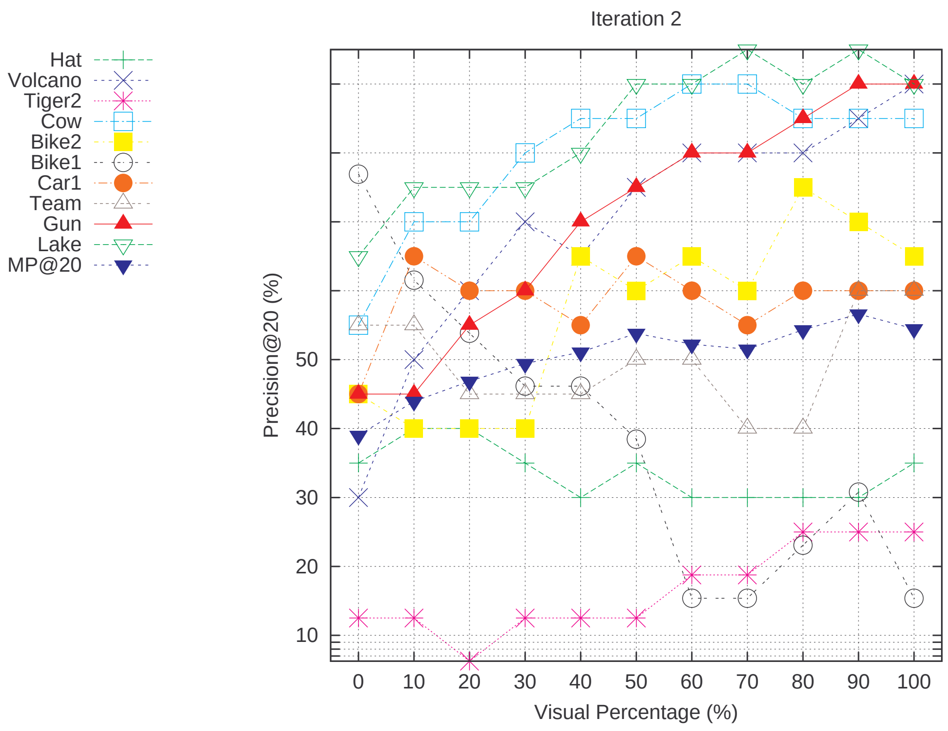


Figure 6.4: Iteration 2 of User Feedback, showing first 10 concepts.

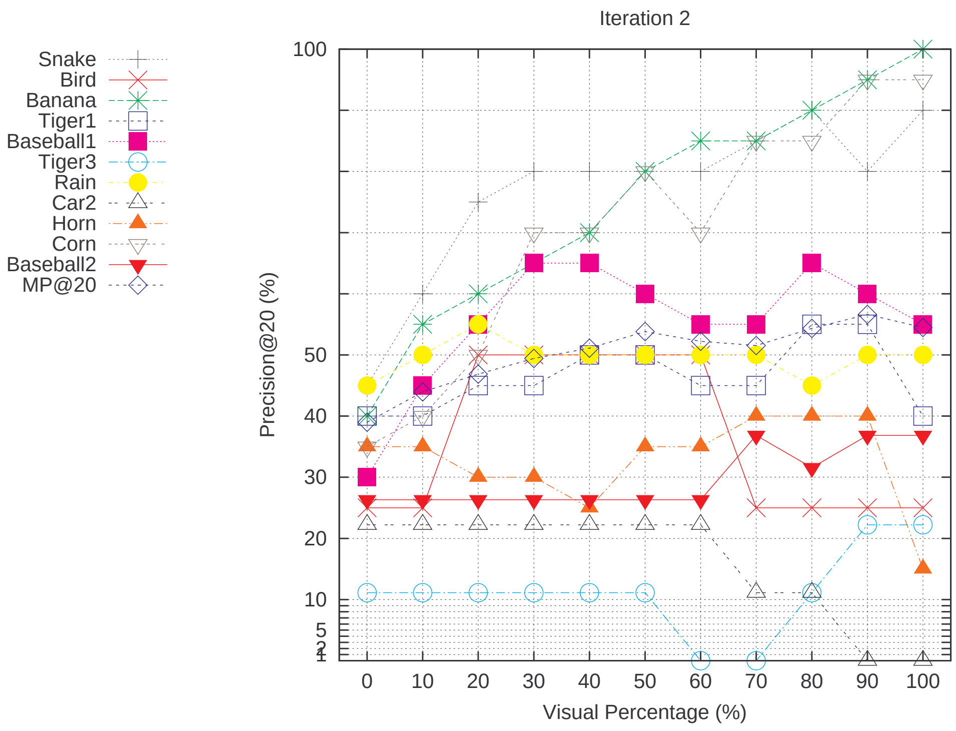


Figure 6.5: Iteration 2 of User Feedback, showing last 11 concepts.

iteration, the value $\alpha = 0.5$ has beaten the visual retrieval precision and the maximum value is not found on the extreme but found at $\alpha = 0.9$.

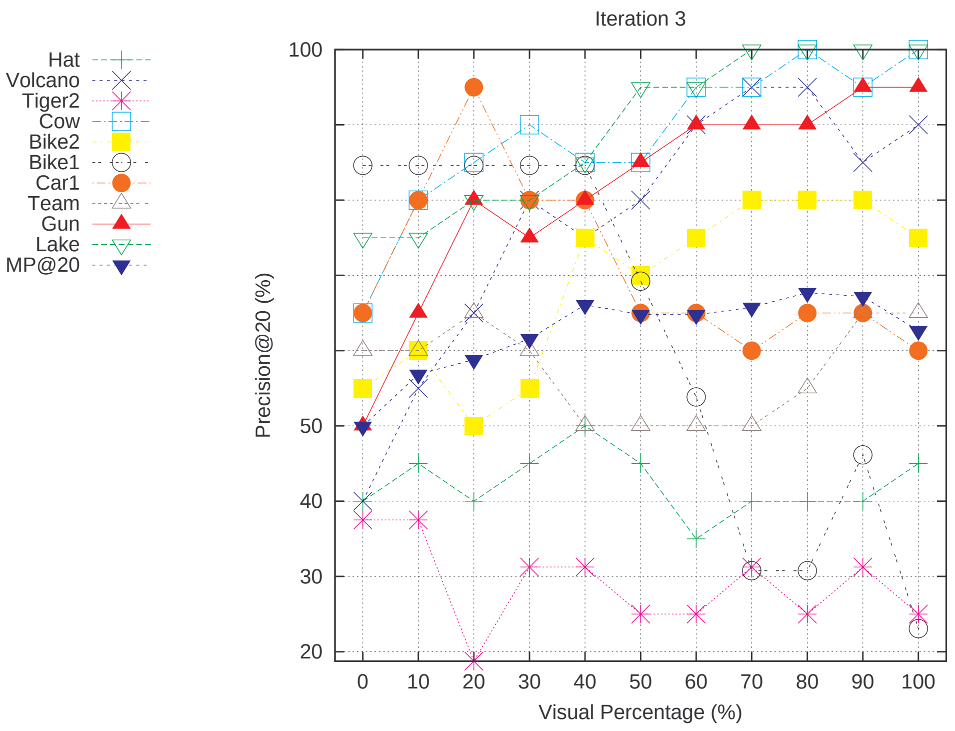


Figure 6.6: Iteration 3 of User Feedback, showing first 10 concepts.

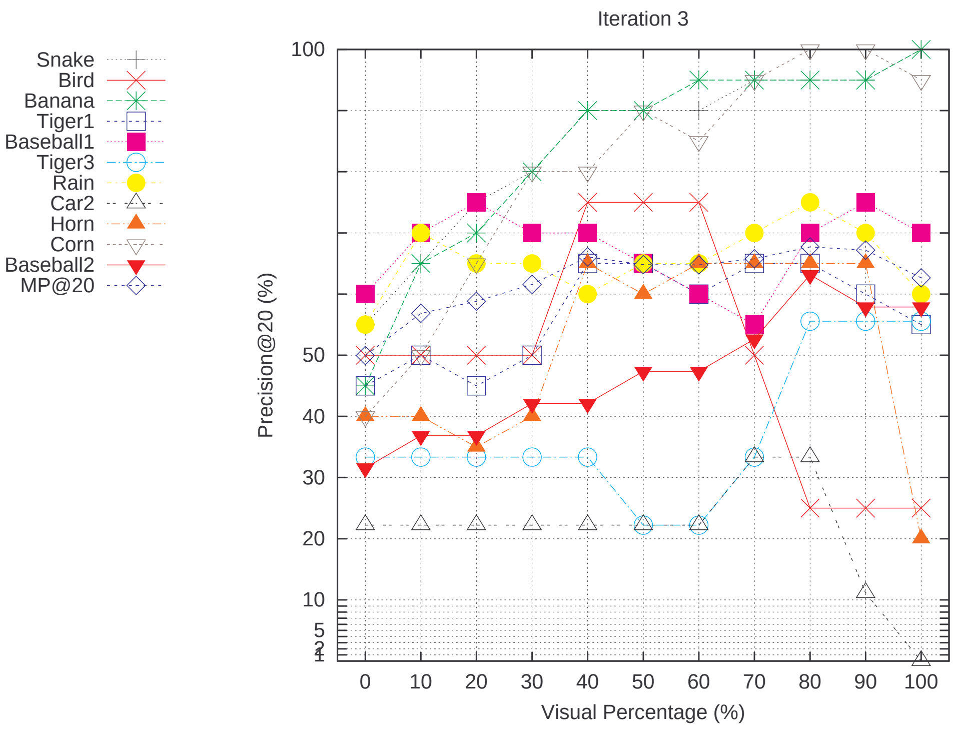


Figure 6.7: Iteration 3 of User Feedback, showing last 11 concepts.

For the Fig. 6.7 and Fig. 6.6, many things have changed. Precision in $\alpha = 1$ is clearly being passed by α values between 0.4 and 0.9. Highest precision is obtained in $\alpha = 0.8$. If we take a look at the queries in the corpus, we can observe how mainly of them are visual queries. This fact associated with the fact that visual features perform better than textual, gives us a slightly shifted global maximum. Moreover we can observe how at $\alpha = 0.4$ we have a high precision although is not enough to become the best value.

6.2 Results with Average Values

So far we have seen how different concept queries behave with different α throughout several iterations. This type of graphs help to ilustrate the problem that different concepts have different needs. In the other hand, with this graphs it is difficult to compare the mean precision in each iteration since we have many lines and things going on in such a little space. In the following section we will compare the mean precision values. This value shows more clearly the imporovement in each iteration and the variation for each α.

In the following Fig. 6.8 we can see how the mean precision raises as the iterations pass by. Also we can see how the slope changes from clearly ascendant in iteration 1 to a sort of parabola-like with the global maximum in the center values.

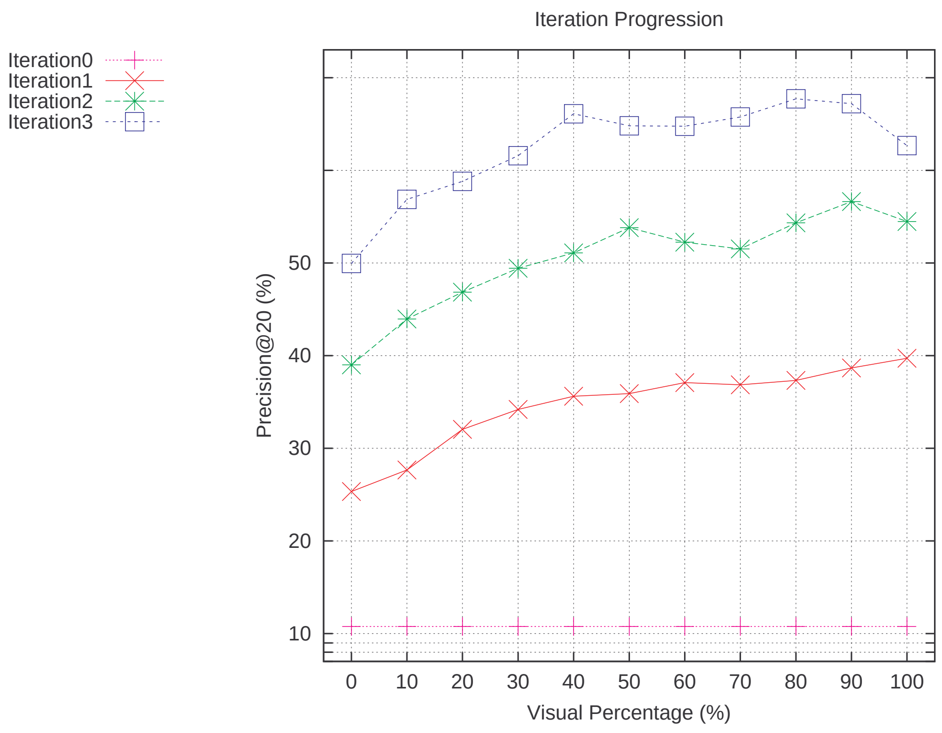


Figure 6.8: Mean Precision@20 Iteration Progression.

Another interesting experiment consists in reducing the number of images shown to the user in each iteration and see how this affect the mean precision slope. The number of images shown corresponds to the N of the $Precision@N$. So far we have assumed the user wants to see 20 images (Precision@20), but it is common to have around 10 images per iteration.

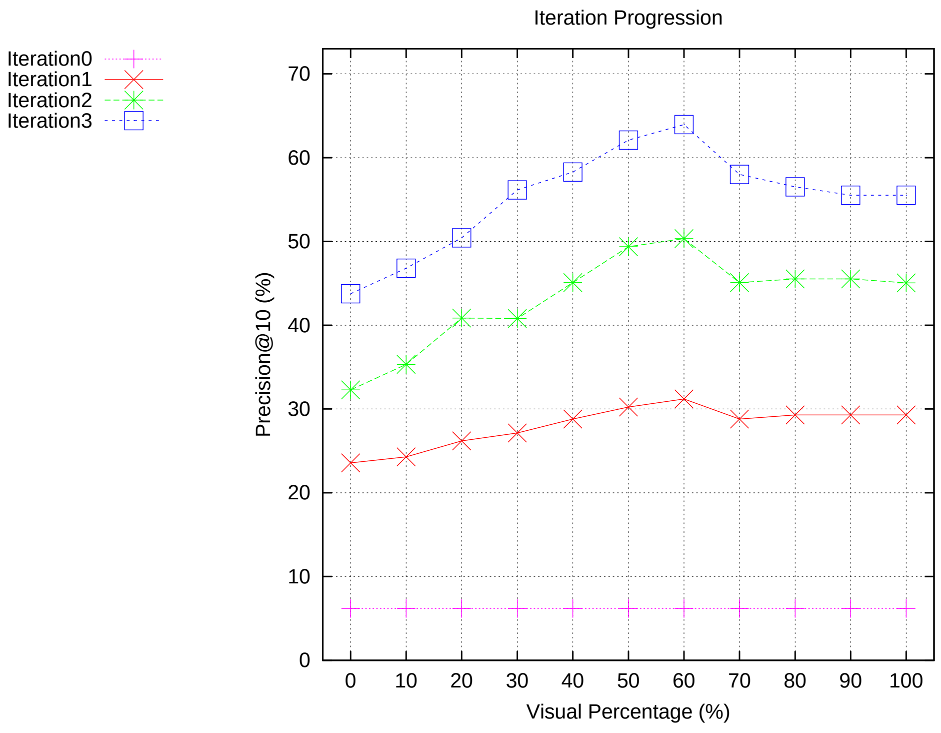


Figure 6.9: Mean Precision@10 Iteration Progression.

We can observe how the slope at all iterations is much more centered than before (Fig 6.9). Before the highest precision was obtained at $\alpha = 1$ for the first iteration and at $\alpha = 0.8$ for the last iteration. This was a little bit higher than expected since we thought that the best precision should be found at $\alpha = 0.5$. But now with Precision@10 the best values are found always at the center of the α values. Since now we have less images per iteration the selection of relevant or non relevant is a far more important issue. This fact may have made the system find the higher precision values at the center.

The next experiment consisted in quantifying how much better this precision is in comparison to full text retrieval and full visual retrieval. For this experiment we got the maximum values for each of the queries in every α. This is, for query 1, the highest value may happen at $\alpha = 1$ and for query 2 it may happen at $\alpha = 0.5$. This number will show us if there is any room for improvement. What we are trying to calculate if implementing another solution (Linear Fusion) has sense.

By looking at Fig. 6.10, we can observe how starting from the same point on iteration 0, all the other iterations, the best possible α, clearly over performs both retrieval engines by separate. We know that best values occur for iteration 3 around $\alpha = 0.8$, but the maximum for every query is not necessarily at that point. Therefore, this situation has opened a new problem: how can we automatically establish an α value for each type of queries?

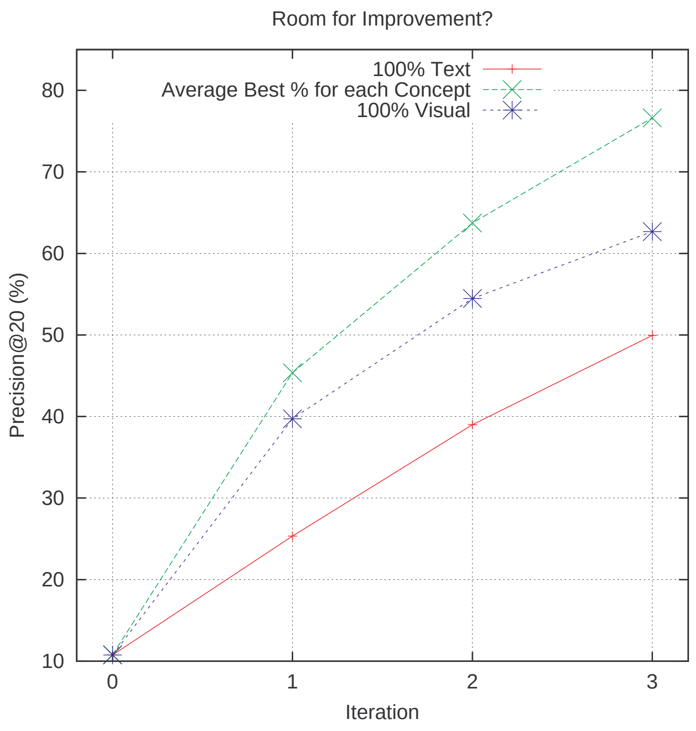


Figure 6.10: This graph shows if there is room for improvement while measuring Precision@20.

As it happened before, it is interesting to see how different this graph will look like with Precision@10 measured instead of Precision@20.

By taking a look at Fig. 6.11 we observe there are lots of similarities with previous Fig. 6.10. We still have much room for improvement. In fact, if we take a look there is much more room for improvement when measuring Precision@10 than when measuring Precision@20. In one hand, Precision@20, we have around 12 percent of improvement while on the other hand, Precision@10, we have almost 20 percent. Of course, best overall precision is obtained when we measure Precision@20, but it is not that much higher than the one measured by Precision@10.

Now we know that we do have room for a new approach and that it does make sense implementing one. As said before this approach consists of a method that finds automatically the best possible α for each type of query; we named this approach linear fusion. We only need to save the proposed rankings from the visual and text engine for the previous iterations. The user will select which images he considers relevant and which images not. With this information we can test how far away was the hypothesis from the text engine and the visual engine, and which α value will maximize the late fusion of both rankings.

In Fig. 6.12 we can see how the approach works. It was expected to perform worst than the average best percentage for each query. Remember this value means finding the best possible α throughout all possible values and all the relevant images. In our

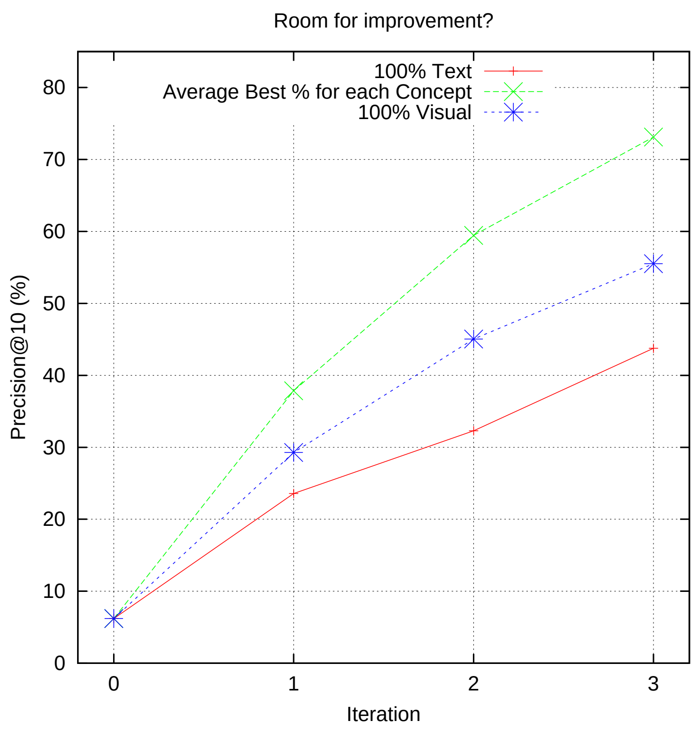


Figure 6.11: This graph shows if there is room for improvement while measuring Precision@10.

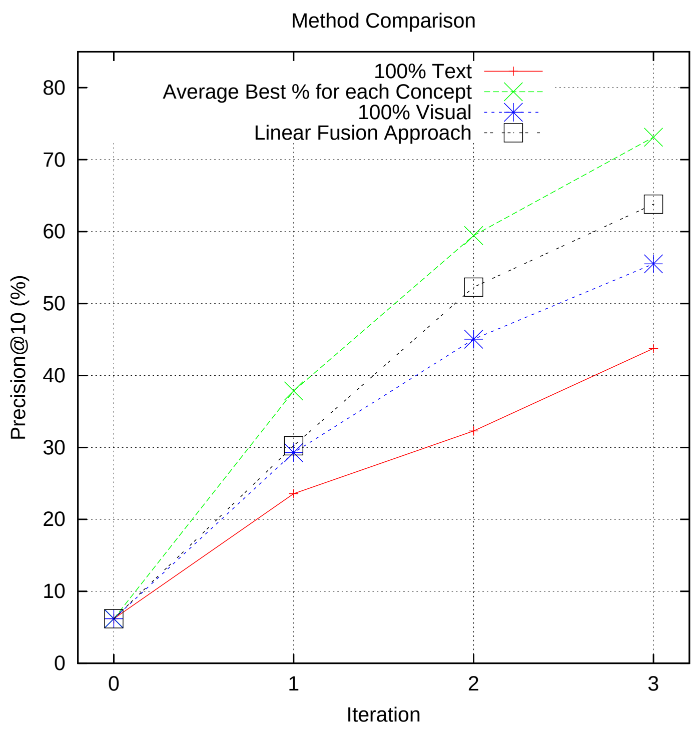


Figure 6.12: This graph shows the automatic α approach.

case we do not have all the relevant images of a concept. We only know the relevant and non relevant images for one iteration. It was expected that the approach works better than the visual engine retrieval approach. Otherwise it will make no sense to have the linear fusion approach.

An important fact about this linear fusion method is that it has no memory. The user can change its mind while looking at a query or concept. First he can start looking at “apples”, choosing real, eatable apples. Then maybe he gets tired of these and starts selecting as relevant “Apple” the company. The α value is calculated for each iteration so if the user changes its mind, the online value will detect this issue and next images shown will have the new α value learned. This gives the user much more flexibility in the results.

Chapter 7

Conclusions and Future Work

In this thesis we show how to build a content based image retrieval engine. First we start defining a set of image features and text features that will later be applied in the process. We have discussed the different ways of downloading the images and creating a database with images and annotations. This annotations were obtained using different methods and finally saved to the database. It is important to have a well organized database with well annotated captions since the user's first iteration is only text based. Moreover we have proposed a set of approaches. The first one, late fusion, mixes the visual ranking and the text ranking depending on a α value. This is a fix value we need to know before calling the method. Some experiments were done to prove that depending on the type of the query a different α is needed. If we took the best possible α for each query, we would have a much more precise system. This took us to the next approach; linear fusion. Our second proposed approach learns the user needs and the type of query the user is searching for and calculates an α. This method is called Linear Fusion since it is obtained by maximizing the result of Late Fusion by making an exhaustive search throughout the different α values in $\{0, 1\}$. In the previous experiments we demonstrate that this value performs much better than the other retrieval engines approaches separatedly. The last of our proposed approaches was the unsupervised image tagging method. This method consisted in scanning a web page and obtaining a particular annotation composed of a weight and a term. Remember this weight is obtained using several ranking techinques. The final result is a database with annotations and references to each image.

There are many different work-lines we can follow in the future. First of all we can speed up the download of images. Right now this process runs around 10-12 images per second. The system is capable of running much faster but threading in Python has many drawbacks. We have encountered many problems dealing with this issue but we have managed downloading around 5 million images. Threads in Python don't run completely parallel and processes are too heavy and slow. Perhaps another computer language can perform a better solution. This solution will speed the process and the amount of images in the database.

Another future work option might deal with the annotation of images. Right now all the web page is downloaded and stripped. All the words in the web are counted and we create a dictionary with each different term. Depending on where does the term appear, we propose a certain weight. This weighting scheme as well as the stripping process might be changed. An option may be to download only the terms surrounding the image we want to download. The problem is that depending on the web page designer, this space might be empty or full with useless information. This is one of the problems we found when working with webs. Each web page has a different structure and doing a script that can work with all the different type of webs was a really difficult task. If we could only capture those words surrounding an image we could weight each term with its distance to the image. The idea is that the terms close to an image usually have a strong relationship with it. This way we may have another type of weight which determines the importance of each term depending on the distance.

Before we talked about how other search engines such as Google or Yahoo work. I think an important step is to add more functions to RISE. At this moment RISE does not save any information on where does the user click. This is, if I select last couple images of "tiger" query as relevant it would be interesting to use this information for future "tiger" requests. Perhaps another user thinks this images are relevant, then if a third user searches for the same query, this images will need to show up first. User intentions is what have made the big search engines be really precise. When we search in Google depending on where in the world we are, we have different results for the same query. At some point, image and text features are not enough. User intentions might be the next step. We could save how many times a user has selected an image as relevant, and for next users, this image will show earlier in the process. This new method might be applied in the late fusion approach. This way we will have 3 heterogeneous methods of finding a ranking: visually, textually and user intention.

When building the system we have discovered that visual retrieval is not enough for a content based retrieval engine to work precisely. We decided to add some text measure for those abstract queries where visual features have no effect. But now its time to think further. Perhaps text and image retrieval is not enough and we may need to add some other measures to keep on improving the system. In fact, this other measure needs to be user information. Although computer pattern recognition is each time more and more precise, at some moment the user information is vital. So far we have a completely automatic process when a user searches for a particular image. In the near future, other users will collaborate with this automatic process in the search of a query with further better results.

Bibliography

[1] Google similar image search. `http://similar-images.googlelabs.com/`. 8

[2] Xcavator. `http://www.xcavator.net/Photo-Search?mode=set_tags&search_tags=dog&x=16&y=5`. 8

[3] Carmen Alvarez, Ahmed Id Oumohmed, Max Mignotte, and Jian yun Nie. Toward cross-language and cross-media image retrieval. In *Proceedings of the 5 th Workshop of the Cross-Language Evaluation Forum. CLEF 2004, Springer: Proceedings Multilingual Information Access for Text, Speech and Images*, page 676687. Springer, 2004.

[4] Romaric Besancon and Christophe Millet. Merging results from different media: Lic2m experiments at imageclef 2005.

[5] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Color- and texture-based image segmentation using em and its application to image querying and classification. 24(8):1026–1038, August 2002. 7

[6] Marco La Cascia, Saratendu Sethi, and Stan Sclaroff. Combining textual and visual cues for content-based image retrieval on the world wide web. In *In IEEE Workshop on Content-based Access of Image and Video Libraries*, pages 24–28, 1998. 35

[7] Alfons Juan Ciscar. Introduction to pattern recognition-distance based methods, October 2008. 19

[8] Alfons Juan Ciscar. Introduction to pattern recognition-feature extraction, October 2008.

[9] Alfons Juan Ciscar. Introduction to pattern recognition-unsupervised learning, October 2008.

[10] Alfons Juan Ciscar, Roberto Paredes, and Enrique Vidal. Learning and perception-introduction, February 2008.

[11] A. Popescu H. Le Borgne P.-A. Moëllic D. Myoupo. Visual reranking for image retrieval over the wikipedia corpus. In *Working notes for the CLEF 2009 Workshop*, 2009.

[12] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, 2008.

[13] A. P. de Vries and T. Westerveld. A comparison of continuous vs. discrete image models for probabilistic image and video retrieval. In *Proceedings of the International Conference on Image Processing (ICIP)*, Singapore, October 2004. 7

[14] Thomas Deselaers. *Features for Image Retrieval.* Diploma, RWTH Aachen University, Aachen, Germany, 12/2003 2003. 5

[15] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Classification error rate for quantitative evaluation of content-based image retrieval systems. In *ICPR*, volume 2, pages 505–508, Cambridge, UK, 23/08/2004 2004. IEEE, IEEE.

[16] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for image retrieval – a quantitative comparison. In *DAGM*, volume 3175 of *LNCS*, pages 228–236, Tübingen, Germany, 30/08/2004 2004. Springer, Springer. 8, 14

[17] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for image retrieval: An experimental comparison. *Information Retrieval*, 11:77–107, 03/2008 2008.

[18] Thomas Deselaers and Henning Müller. Tutorial on medical image retrieval, aug 2006.

[19] Thomas Deselaers, Roberto Paredes, Enrique Vidal, and Hermann Ney. Learning weighted distances for relevance feedback in image retrieval. In *International Conference on Pattern Recognition 2008*, Tampa, Florida, USA, 08/12/2008 2008.

[20] Thomas Deselaers, Tobias Wey, Daniel Keysers, Wolfgang Macherey, Hermann Ney, and Lehrstuhl Für Informatik Vi. Fire in imageclef 2005: Combining content-based image retrieval with textual information retrieval. In *In Workshop of the Cross–Language Evaluation Forum (CLEF 2005), Lecture Notes in Computer Science*, pages 652–661. Springer, 2005.

[21] Thomas Deselaers, Tobias Weyand, Daniel Keysers, Wolfgang Macherey, and Hermann Ney. Fire in imageclef 2005: Combining content-based image retrieval with textual information retrieval. In *CLEF Workshop 2005*, volume 4022 of *LNCS*, pages 652–661, Vienna, Austria, 21/09/2005 2006. Springer, Springer.

[22] Osama El Demerdash, Leila Kosseim, and Sabine Bergler. Image retrieval by intermedia fusion and pseudo-relevance feedback. In *CLEF'08: Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, pages 605–611, Berlin, Heidelberg, 2009. Springer-Verlag.

[23] Hugo Jair Escalante, Jesús A. Gonzalez, Carlos A. Hernández, Aurelio López, Manuel Montes, Eduardo Morales, Luis E. Sucar, and Pineda Luis Villase.

Annotation-based expansion and late fusion of mixed methods for multimedia image retrieval. In *CLEF'08: Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, pages 669–676, Berlin, Heidelberg, 2009. Springer-Verlag.

[24] Hugo Jair Escalante, Carlos A. Hérnadez, Luis Enrique Sucar, and Manuel Montes. Late fusion of heterogeneous methods for multimedia image retrieval. In *MIR '08: Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 172–179, New York, NY, USA, 2008. ACM. 27

[25] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3:231–262, 1994. 7

[26] Georgy Gimel'farb. Cbir texture features, 2006.

[27] Steven C. Hoi. Cross-language and cross-media image retrieval: An empirical study at imageclef2007. pages 538–545, 2008. 15

[28] P. Howarth and S. Rüger. Robust texture features for still-image retrieval. *IEE Proceedings - Vision, Image, and Signal Processing*, 152(6):868–874, 2005. 12

[29] Winston H. Hsu. Texture and shape for image retrieval – multimedia analysis and indexing, oct 2007. 12

[30] Masashi Inoue. On the need for annotation-based image retrieval. *Information Retrieval in Context (IRiX): A Workshop at SIGIR 2004*, pages 44–46, July 2004. 7, 35

[31] Qasim Iqbal and Jake K. Aggarwal. Cires: a system for content-based retrieval in digital image libraries. In *ICARCV*, pages 205–210, 2002. 7

[32] Hai Jin, Wenbing Tao, and Aobing Sun. Vast: Automatically combining keywords and visual features for web image retrieval. *International Conference on Advanced Communication Technology (ICACT)*, 3:2188–2193, 2008. 8, 25

[33] Hyun Sook Kim and Young Kyu Yang. An automatic classification technique for indexing of soccer highlights using neural networks. *MG&V*, 10(4):467–487, 2001.

[34] Caroline Lacoste, Jean-Pierre Chevallet, Joo-Hwee Lim, Diem Thi Hoang Le, Wei Xiong, Daniel Racoceanu, Roxana Teodorescu, and Nicolas Vuillemenot. Inter-media concept-based medical image indexing and retrieval with umls at ipal. In *CLEF*, pages 694–701, 2006. 27

[35] Thomas M. Lehmann, Mark Oliver Güld, Thomas Deselaers, Daniel Keysers, Henning Schubert, Klaus Spitzer, Hermann Ney, and Berthold Wein. Automatic categorization of medical images for content-based retrieval and data mining. *Computerized Medical Imaging and Graphics*, 29(2):143–155, 2005. 7

[36] Stefanie Lindstaedt, Roland Mörzinger, Robert Sorschag, Viktoria Pammer, and Georg Thallinger. Automatic image annotation using visual content and folksonomies. *Multimedia Tools Appl.*, 42(1):97–113, 2009.

[37] Haibin Ling and Kazunori Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(5):840–853, 2007.

[38] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91, November 2004. 15

[39] Nicolas Maillot, Jean-Pierre Chevallet, and Joo-Hwee Lim. Inter-media pseudo-relevance feedback application to imageclef 2006 photo retrieval. In *CLEF*, pages 735–738, 2006.

[40] Nicolas Maillot, Jean-Pierre Chevallet, and Joo-Hwee Lim. Inter-media pseudo-relevance feedback application to imageclef 2006 photo retrieval. In *CLEF*, pages 735–738, 2006.

[41] D. Mehta, E.S.V.N.L.S. Diwakar, and C.V. Jawahar. A rule-based approach to image retrieval. volume 2, pages 586–590 Vol.2, Oct. 2003. 6

[42] Sean Moran. Automatic image tagging, 2009. 5

[43] Nhu-Van Nguyen, Jean-Marc Ogier, Salvatore Tabbone, and Alain Boucher. Text retrieval relevance feedback techniques for bag of words model in cbir. In *International Conference on Machine Learning and Pattern Recognition (ICMLPR09)*. IEEE Computer Society, 2009.

[44] Eric Nowak and Frederic Jurie. Learning visual similarity measures for comparing never seen objects. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[45] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 2001. 15

[46] Roberto Paredes, Thomas Deselaer, and Enrique Vidal. A probabilistic model for user relevance feedback on image retrieval. In *Proc. of MLMI*, pages 260–271, 2008.

[47] Roberto Paredes, Thomas Deselaers, and Enrique Vidal. A probabilistic model for user relevance feedback on image retrieval. In *MLMI*, pages 260–271, 2008. 17, 18

[48] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases, 1994. 7

[49] Mika Rautiainen, Timo Ojala, and Tapio Seppänen. Analysing the performance of visual, concept and text features in content-based video retrieval. In *MIR '04: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 197–204, New York, NY, USA, 2004. ACM.

[50] César Reyes, Maria Luisa Durán, Teresa Alonso, Pablo G. Rodríguez, and Andrés Caro. Behaviour of texture features in a cbir system. In *HAIS '08: Proceedings of the 3rd international workshop on Hybrid Artificial Intelligence Systems*, pages 425–432, Berlin, Heidelberg, 2008. Springer-Verlag.

[51] S. E. Robertson and K. Sparck Jones. Simple, proven approaches to text retrieval. Technical report, 1997.

[52] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. *Computer Vision, IEEE International Conference on*, 0:59, 1998.

[53] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, November 2000.

[54] Juan Carlos Caicedo Rueda. Extracción de características para recuperación de imágenes por contenido. Master's thesis, Universidad Nacional de Colombia, email: jccaicedoru@unal.edu.co, 2008.

[55] N. Sebe, M. S. Lew, X. Zhou, T. S. Huang, and E.M. Bakker. The state of the art in image and video retrieval. In *in Int. Conf. on Image and Video Retrieval*, pages 1–8. Springer-Verlag, 2003.

[56] Franco Segarra. Content based image retrieval. Master's thesis, Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, July 2009. 9, 13, 14, 21

[57] Franco Segarra, Luis A. Leiva, and Roberto Paredes. A relevant image search engine with late fusion: Mixing the roles of textual and visual descriptors. In *IUI Demo Paper*, 2010.

[58] Jonathon Shlens. A tutorial on principal component analysis. December 2005.

[59] Sven Siggelkow, Marc Schael, and Hans Burkhardt. Simba - search images by appearance. In Bernd Radig and Stefan Florczyk, editors, *DAGM-Symposium*, volume 2191 of *Lecture Notes in Computer Science*, pages 9–16. Springer, 2001. 7

[60] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. PAMI*, 22(12):1349–1380, 2000.

[61] Lindsay I. Smith. *A tutorial on Principal Components Analysis*, February 2002.

[62] Amanda Spink. Term relevance feedback and query expansion: relation to design. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 81–90, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[63] Y. Sun, H. Zhang, L. Zhang, and M. Li. Myphotos – a system for home photo management and processing, 2002.

[64] Trong ton Pham, Nicolas Eric Maillot, Joo hwee Lim, and Jean pierre Chevallet. Abstract latent semantic fusion model for image retrieval and annotation. 26

[65] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:947–963, 2001. 7

[66] James Z. Wang, Nozha Boujemaa, Alberto Del Bimbo, Donald Geman, Alexander G. Hauptmann, and Jelena Tešić. Diversity in multimedia information retrieval research. In *Proc. of MIR*, pages 5–12, 2006.

[67] Chen Zhang, Joyce Y. Chai, and Rong Jin. User term feedback in interactive text-based image retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 51–58, New York, NY, USA, 2005. ACM. 7