Implementation of the Fisher Kernel Framework for Image Retrieval

Implementation based on the article *“Large-scale image retrieval with compressed fisher vectors”* written by Florent Perronin, Yan Liu, Jorge Sánchez and Hervé Poirier.

FINAL DEGREE PROJECT

AUTHOR Santiago Herrero Bajo

ADVISOR Prof. Hongyu Li 李宏宇

Shanghai, China September 2012

Implementation of the Fisher Kernel Framework for Image Retrieval

Implementation based on the article *"Large-scale image retrieval with compressed fisher vectors"* written by Florent Perronin, Yan Liu, Jorge Sánchez and Hervé Poirier.

AUTHOR Santiago Herrero Bajo
ADVISOR Prof. Hongyu Li 李宏宇

Final thesis of course ending submitted at the School of Software Engineering – Tongji University – in order to graduate as a Telecommunications Engineer (ETSETB – UPC – Barcelona, Spain)

Focus area: Image processing

Tongji University – Shanghai, China
September 2012

Acknowledgments

• To my advisor: Prof. Hongyu Li, for giving me the opportunity to implement this project in Shanghai and letting me be a small part in student Wang Yi Master project.

• To Wang Yi for allowing me to help him with the front-end development of his project.

• To my girlfriend for the moral support in moments where I couldn't find what was the next step to take.

• To my roommates and Shanghai friends for making me feel like having a Chinese family.

• To my family for their never ending long-distance support.

Abstract

This project aims to implement one of the different approaches existing nowadays for the resolution of the problem of large-scale image retrieval: the Fisher Kernel Framework. This approach was first introduced by Tommi S. Jaakkola and David Haussler in [14], and later on implemented by Florent Perronnin and Christopher Dance in [6].

More specifically, the project focus on this implementation as it is presented in the paper *"Large-scale image retrieval with compressed fisher vectors"* [1] written by Florent Perronin, Yan Liu, Jorge Sánchez and Hervé Poirier during their work at Xerox Research Center Europe (XRCE). To accomplish this, we make use of the Yael library, which

The project starts by introducing the problem of large-scale image search from a general point of view. Following to the introduction of the problem, the Fisher Kernel Framework (FKF) is explained in more detail. In the next section, this information is used to compare this approach with the alternative existing solutions and once this comparison is addressed, the drawbacks of the FKF are analyzed and the different solutions to palliate these disadvantages are carefully explained. Finally, a few of the different possible searching algorithms are described.

Proceeding to this more theoretical section of the project, comes the experimental section. In it, the scope of the implementation is detailed together with the description of the experimental setup. Finally, the whole process from beginning to the final search result is explained step-by-step with clear focus on the programming code as a helper for a better understanding. Two different programs are shown: first, a user interface that visually retrieves image from a small database and displays them; second, a test software where the framework will be tested under different contexts to test its performance.

Objectives

The personal objectives of this project are the following:

- Expose myself to the realization of a practical complex problem without having a set path to follow, only with the help of my advisor in the case of a strong need of advice.
- Have a first approach to the field of Image Processing, which I did not have the chance to cover during my undergraduate studies at Universitat Politècnica de Catalunya (UPC).
- Immerse myself in a different culture, which I did previously know, and practice my Chinese language during the seven months that the realization of this project has taken.

TABLE OF CONTENTS

1 THEORETICAL APPROACH

1.1 INTRODUCTION AND HISTORY OF CONTENT-BASED IMAGE RETRIEVAL

Image retrieval development can be first tracked back to the late 1970s in a conference held in Florence about Database Techniques for Pictorial Applications. At its beginning, image retrieval was based on manually creating text annotations that were used on traditional text-based database management search systems. These annotations organized images by giving it a topical or semantical meaning, facilitating their organization and the navigation within the databases. Manually annotating images was soon found to be non optimal because of the inaccurate process that is a person relying on their sight and judgment to create an image signature. That is subject to the person subjectiveness, it is context-sensitive and is very likely to be incomplete.

In the early 1990s and together with the growing development of Internet and improved digital image sensor hardware and software, the application potential of image database management techniques was considerably boosted and applied in many different fields such as:

- Education
- Medicine
- Industries

Text-based image retrieval approaches were left back to move on to content-based image retrieval techniques. It was in 1992 when the National Science Foundation of the United States organized a workshop to debate the following steps to take on this problem. It was of general consensus that a much more efficient way to characterize and represent images would have to be based on the properties inherent to the images themselves. Since then the amount of papers, research groups and commercial and academic retrieval systems have been written and development in a large amount. During the past decade, remarkable progress has been made in both theoretical research and system development. Nevertheless, the problem is far from solved and there are still many challenging research problems that are still under development. It has to be noted that the content-based techniques are based on a query-by-example framework, where a query image is provided and the objective is to retrieve similar images within the database.

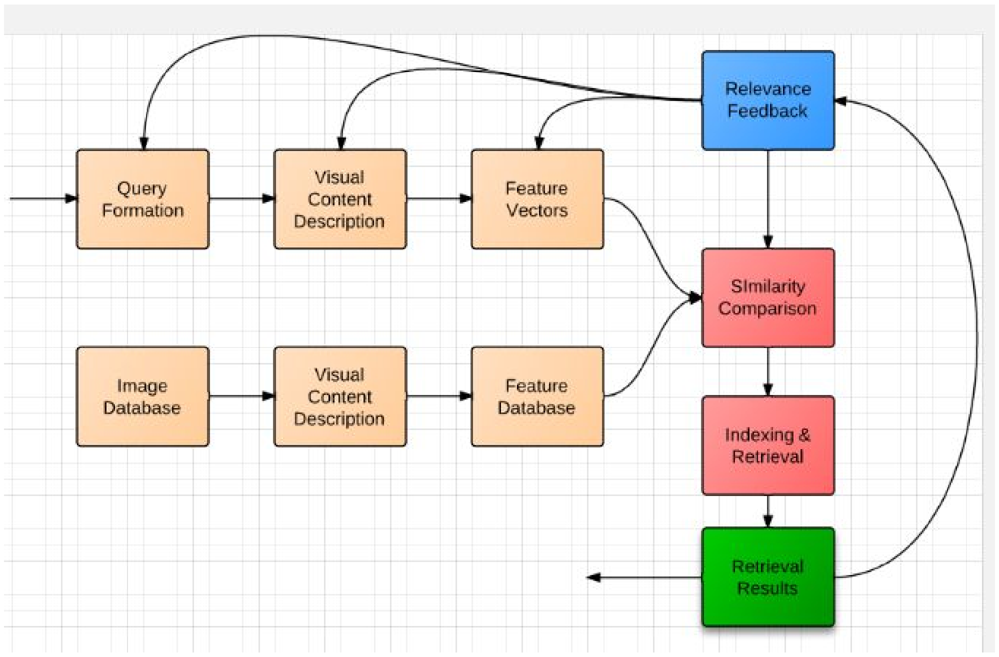
Content-based image retrieval, uses the visual characteristics of an image like:

- Color
- Shape
- Texture
- Spatial layout

to represent and index the image. In typical content-based image retrieval systems the process is the following:

1. The images in the database are characterized by extracting multi-dimensional feature vector descriptor vectors, forming a feature database.
2. The similarities between the feature vectors corresponding to the query image and the rest of the database are computed.
3. Retrieval is performed by means of an indexing scheme, which efficiently provides a way to search within the image database.
4. In recent techniques, user's feedback is added in order to improve the retrieval process by adding user's perception and semantic meaning.

The following figure better describes this process:



About the feature descriptors obtained directly from the inherent characteristics of the images, they are generally computer in two steps:

1. First interest points in the image are identified.
2. The values for the descriptors are computed by using the characteristics of the image around each point of interest.

This process can be handled in various ways and currently there are many different descriptor schemes. Their objective is to be robust to rotations, translations of objects in images, changes in color and compression and scale changes. Some of the current schemes are:

- SIFT
- GIST
- RDTQ
- PCA-SIFT
- Eff descriptors

and many more, each of which is better suited for different image processing approaches including image retrieval.

Once the feature database is obtained by using one of the many descriptor schemes available, these vectors have to be compared and these difference have to be measurable. There are also a large variety of similarity computation methods to measure the 'distance' between the query image feature vectors and the rest. Euclidean distance, Hamming distance, asymmetric distance computation systems, using codes for the non-query image vectors such as with Spectral Hashing, etc. are just a few of the possibilities available. In the end, the objective is to obtain a reliable distance measurement system that efficiently computes the real inherent visual differences and similarities between the database of images.

Once measured the distance between the query example image and the rest of the database feature vectors is computed, retrieval has to be computed by means of an indexing system that allows us to search within the database those images with smaller 'distance' with the query image. In this area the variety also doesn't fall short. The most common and straight-forward indexing system is to use inverted lists to sort the images by distance and avoid having to visit the whole database for the retrieval process.

At this point, the retrieval problem arrives to its conclusion and it is now when the performance of all of the different possibilities that exist in every single of the steps taken during the problem will be shown. In a content-based image retrieval problem, there are many constraints to be considered when finding a suitable implementation, among which are:

- **Search accuracy**

  It is the most important of the constraints to address since it will ultimately dictate whether a solution is valid or not. It is determined by the number of similar obtained images within all the images obtained after the search. There are various tools to measure search accuracy, two of which are mAP and recall@R, both introduced in section 2.5.

- **Efficiency and memory usage**

  These two constraints are closely related. Memory usage defines the amount of memory space needed to perform the search. It is obviously desirable to employ as less memory space as possible, since if the database is large enough the system requirements could be too high. A solution that visits the less amount of memory during the search is also said to be effective.

- **Search time**

  In order to apply an implementation to a real situation is it extremely desirable that the response time between the image query selection and the end of the search process is as small as possible, no matter the conditions on which the implementation is running on (one core computer, very-large database, etc.).

- **Scalability**

  A good implementation has to be robust to scalability, meaning that no matter if the size of the database is very large, the search performance should still be efficient, accurate and efficient.

These are just some of the most important constraints that have to be taken on account, but the importance of each of them is subject to the context and application where the implementation is meant to be running. Nevertheless, these clearly determine which of the different approaches succeed.

In this project we focus on one of this approaches: the Fisher Kernel Framework. This framework is used to address the problem of image categorization, inherent to the image search problem. Other alternatives are also going to be mentioned in order to better allocate where does FKF stand among the sometimes more commonly used solutions for the image search problem. First, though, it is necessary to analyze in depth the implementation that will be the focus of this project, as it was firstly introduced in [14].

1.2 The Fisher Kernel Framework for image retrieval

1.2.1 Introduction

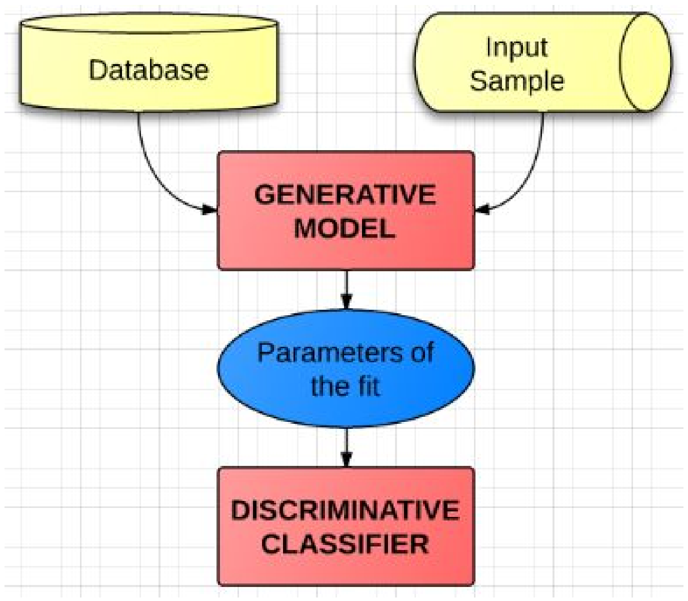
To address the problem of image retrieval it is necessary to analyze the problem of image categorization on the first place. Image categorization is a problem consisting in assigning one of more labels to an image according to its semantic content. This problem implies a several amount of complexity since there are many factors that have to be taken on account such as:

- Object and scene variations
- Changes in viewpoint
- Changes in lighting
- Occlusion

Therefore, this problem still remains open and there are different models that are being applied to find the ideal classifier that better results provide.

In the paper "*Exploiting generative models in discriminative classifiers*" [14], an ideal classifier is defined as one that includes the capabilities of treating missing information, which generative models have, mixed with the advantages discriminative models provide, in terms of classification performance. Tommi S. Jaakkola and David Haussler defend that combining both generative and discriminative methods to address the problem of classification leads to better results without adding too much computation complexity. It is in this context where kernel functions are introduced in order to achieve the powerful combination of the generative and discriminative models.

Here is the schema of the model proposed in the article:



The kernel function is used as a "similarity" measure between two feature vectors ϕ_{X_i} and ϕ_{X_j} previously extracted by means of a generative model and mapped with the set of examples X . Nevertheless, only kernel functions that accomplish to be positive semi-definite are suitable because, according to Mercer's theorem, such a kernel can be represented as an inner product in the feature space.

$$K(X_i, X_j) = \phi_{X_j}^T \phi_{X_j}$$

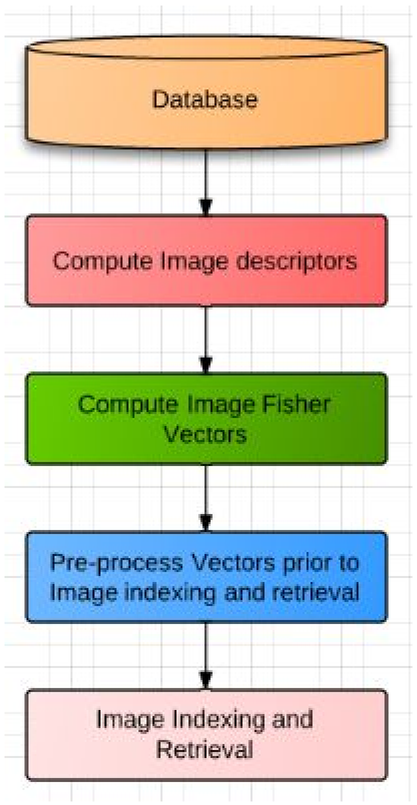
Being able to represent a kernel function as an inner product in the feature space is equivalent to defining an Euclidean metric space in which the distances between the feature vectors are calculated directly with by means of the kernel function:

$$\|\phi_{X_i} - \phi_{X_j}\|^2 = K(X_i, X_i) - 2K(X_i, X_j) + K(X_j, X_j)$$

These distances can be used for similarity computation purposes as it will later be shown in the implementation section of this project, making the kernel functions a very straight-forward method for image classification.

The next step after proving the suitability of Tommi S. Jaakkola and David Haussler proposal of kernel functions, comes finding a valid kernel function for the purpose. Here is where the Fisher Kernel Framework is introduced. It is out of the scope of this project to demonstrate how Fisher Kernel functions were derived from a generative model, but it is noted that this approach arises basically from the use of Fisher scores as features from the set of examples.

1.2.2 The Fisher Kernel Framework


Database
Compute Image descriptors
Compute Image Fisher Vectors
Pre-process Vectors prior to Image indexing and retrieval
Image Indexing and Retrieval

The Fisher Kernel Framework proposes using a gradient vector derived from a probability density function as a generative model that generates the signal, in order to later on feed it to a discriminative model in charge of the categorization process. This proposal is known to have done a successful job in a certain number of applications such as:

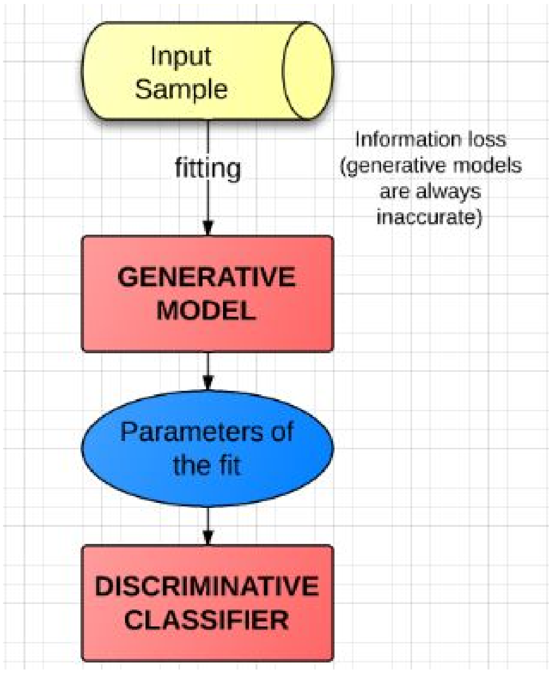
- Categorization
- Dimensionality reduction
- Saliency detection
- Audio indexing
- Speakers recognition

Nevertheless, it has yet not been used for image categorization. In [6] this framework is used for this purpose with the following characteristics:

1. The input signals are the images.
2. The generative model used is a GMM (Gaussian mixture model) and is in charge of the extraction of the low-level features of the images, also called the visual vocabulary.

$$p(x) = \sum w_i p_i(x)$$

As seen in [14], this is a graphical representation of the described proposal. We will continue by getting into more detail in the mathematical side of the Fisher Kernel Framework basics.


Input Sample
fitting
Information loss (generative models are always inaccurate)
GENERATIVE MODEL
Parameters of the fit
DISCRIMINATIVE CLASSIFIER

Let X be a sample corresponding to a single input image and $X = \{x_t, t = 1 \ldots T\}$ be the set of low-level feature vectors extracted from it. X has been generated by a GMM (a probability density function p) with parameters $\lambda = \{w_i, \mu_i, \Sigma_i, i = 1 \ldots N\}$ and is described by the following gradient vector:

$$G_\lambda^X = \nabla_\lambda \log p(X/\lambda)$$

w_i, μ_i and Σ_i correspond to the weight, mean vector and covariance matrix of Gaussian i respectively and N the number of Gaussians. w_i represents the frequency of the Gaussian i, also known as word in the visual vocabulary, μ_i the mean of the Gaussian and Σ_i the variation around the mean. Covariance matrices are assumed to be diagonal and σ_i^2 will represent the variance vector. In [1] and, therefore, in this project, the focus remains only on the derivatives respect to the mean.

No matter if the length of X is variable, the gradient vector is always a fixed length vector. The size depends only on the λ parameters. This vector describes both the contribution of the parameters λ or, more specifically, the direction towards which the parameters ought to be modified in order to best fit the data. The kernel of this gradient is:

$$K(X, Y) = G_\lambda^{X\prime} F_\lambda^{-1} G_\lambda^Y$$

where F_λ is the Fisher information matrix of p as suggested in [14]:

$$F_\lambda = E_{X\ p}[\nabla_\lambda \log p(X/\lambda) \nabla_\lambda \log p(X/\lambda)']$$

this matrix is used to normalize the gradient vector, which is needed if we are going to follow by using a discriminative classifier that makes use of the inner-product. The resulting normalized gradient vector results:

$$F_\lambda^{-1/2} \nabla_\lambda \log p(X/\lambda)$$

Nevertheless, in practice, the cost of computing the matrix and inverting it is so high that F_λ is normally approximated by the identity matrix, therefore, no normalization is performed. In the paper subject to analysis in this project [1] this normalization is only a whitening of the dimensions $L_\lambda = F_\lambda^{-1/2}$ because the diagonal closed-form approximation of [6] is used.

Continuing with the mathematical development, we see that because F_λ is symmetric and positive definite, it has a *Cholesky* decomposition:

$$F_\lambda = L_\lambda{}' L_\lambda$$

the kernel function, as it was said in section 1.2.1, can then be represented as:

$$K(X, Y) = \boldsymbol{G}_\lambda^X{}' \boldsymbol{G}_\lambda^Y \text{ where } \boldsymbol{G}_\lambda^X = L_\lambda G_\lambda^X$$

$\boldsymbol{G}_\lambda^X$ will be known as the Fisher vector of X , and the kernel function as the dot-product between the Fisher vectors of X and Y .

For the Gaussian mixture model (GMM) both supervised and unsupervised training methods are valid. However, for the sake of simplicity, we use unsupervised training using a large number of images and the Maximum Likelihood Estimation (MLE).

1.2.3 Application of this framework for image signature computing

Once the basics of the Fisher Kernel Framework have bet detailed it is necessary to prove why Fisher vectors are suitable for performing image signatures and, consequently, valid for image categorization applications. To demonstrate this, it is necessary to find the mathematical expression of the Fisher vector $\boldsymbol{G}_\lambda^X$ of an image X as it is done in [6].

We define $\gamma_t(i)$ as the soft assignment of the descriptor x_t with the Gaussian i. After some derivations the gradient G_i^X, which as it was said in the previous section the gradient with respect to the mean μ_i of Gaussian i, becomes:

$$G_i^X = \frac{1}{\sqrt{w_i}} \sum_{t=1}^{T} \gamma_i(i) \left(\frac{x_i - \mu_i}{\sigma_i} \right)$$

G_λ^X is the concatenation of the G_i^X vectors for $i = 1 \ldots N$.

Once the expression of G_λ^X is found, it can be best proved one of the main reasons why Fisher vectors are suitable for image categorization: **Fisher vectors discount the influence of the descriptors that are likely to occur in any image (named background descriptors)**. In other words, Fisher vectors discount the influence of the descriptors with high $p(x_t)$.

Since the descriptors obtained from the images are high-dimensional, γ_t has a peaky distribution:

$$\left\{ \begin{matrix} \gamma_t(i) & \approx & 1 & \\ \gamma_t(j) & \approx & 0 & j \neq i \end{matrix} \right\}$$

Therefore, we can approximate:

$$p(x_t) \approx w_i p_i(x_t)$$

It can be observed that for a descriptor to be likely to occur in any image (have high $p(x_t)$):

- w_i has to be high, in other words, i has to be a frequent visual word.
- $\|\frac{x_i - \mu_i}{\sigma_i}\|^2$ has to be small, which happens when x_t and μ_i are close together.

If we observe equation:

$$\boldsymbol{G}_i^X = \frac{1}{\sqrt{w_i}} \sum_{t=1}^{T} \gamma_i(i)\left(\frac{x_i - \mu_i}{\sigma_i}\right)$$

in case where the two conditions for $p(x_t)$ to be high occur, we can see in this formula that its impact is discounted by the factor:

$$\frac{1}{\sqrt{w_i}} \approx 0$$

Consequently, it is clear that Fisher vectors discount the influence of the descriptors that are likely to occur in any image which means that: **Fisher vectors descriptors describe an image by what it makes it different from the others**.

1.2.4 Measuring similarity

Once it has been proven that Fisher vectors discount the influence of the commonly found descriptors, it is time to analyze the similarity computation that takes place in the categorization process.

As in [1] the following notations are introduced:

$$w_i^X = \frac{1}{T} \sum_{t=1}^{T} \gamma_t(i) \quad \rightarrow \quad \textit{proportion of descriptors X soft} - \textit{assigned with visual word i}$$

$$\mu_i^X = \frac{\sum_{t=1}^{T} \gamma_t(i) x_t}{\sum_{t=1}^{T} \gamma_t(i)} \quad \rightarrow \quad \textit{average of the descriptors of X weighted by their probability}$$

$$\textit{of being assigned with Gaussian i}$$

$$\delta_i^X = \frac{\mu_i^X - \mu_i}{\sigma_i}$$

Applying these notations to $\boldsymbol{G}_i^X$ we obtain:

$$\frac{1}{T}\boldsymbol{G}_i^X = \frac{w_i^X}{\sqrt{w_i}}\delta_i^X$$

Therefore, the kernel function can be rewritten as:

$$K(X,Y) = \boldsymbol{G}_\lambda^X{}'\boldsymbol{G}_\lambda^Y = \alpha\cdot\sum_{i=1}^{N}\frac{w_i^X w_i^Y}{w_i}\delta_i^X{}'\delta_i^Y \rightarrow \textit{where } '\alpha' \textit{ is a scalar}$$

Analyzing this new rewritten version of the kernel function it is concluded in [1] that:

- The first term in this new rewritten version of the kernel function ($\frac{w_i^X w_i^Y}{w_i}$), does the function of discounting the effect of background descriptors as it is the multiplication of the frequencies of visual word i in X and Y divided by the average frequency of occurrence of i in any image.
- The second term $\delta_i^X{}'\delta_i^Y$ is large if δ_i^X and δ_i^Y vectors have similar direction and a large norm.
  - δ_i^X and δ_i^Y vectors have similar direction if the descriptors of X and Y assigned to Gaussian i have the same average.
  - δ_i^X and δ_i^Y have a large norm if μ_i^X and μ_i^Y are significantly different from μ_i . Having a large norm indicates that the descriptors of X assigned to Gaussian i are unlikely to be background descriptors.

As it was said in section 1.2.1, being able to represent the Fisher kernel function as an inner-product, and proven that the resulting formula from the dot-product has very desirable properties for image categorization, allows us to define an Euclidean metric space in which the distances are directly calculated by means of the kernel function and used for similarity purposes. This simplifies the implementation process as it will be seen in the implementation section of the project.

1.2.5 Alternatives to Fisher Vectors for Image retrieval

Even though the theoretical effectiveness of using Fisher vectors to address the problem of image categorization has been proved, it is not the most common solution. The most widely used one is the bag-of-visual words (BOV) which will be summarized in this section for comparison purposes. In addition, other solutions such as a non-probabilistic Fisher Kernel (VLAD) introduced in [10] has also been proved to have successful results [2].

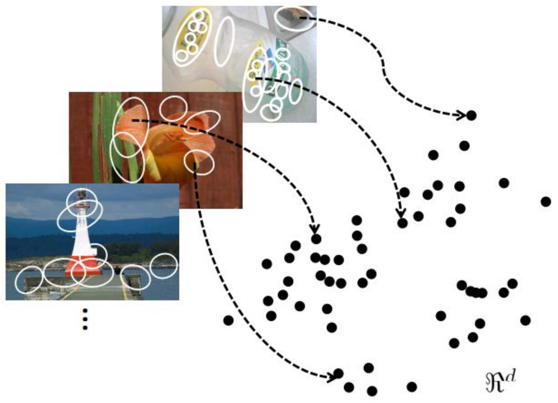
1.2.5.1 BOV

The bag-of-visual words is the most common solution applied to image categorization problems. This approach is based on a visual vocabulary, or codebook of centroids, obtained by k-means clustering. The descriptors obtained from the images are assigned to the closest of the centroids from the codebook. Finally, a vector with the dimension equal to the number of centroids is obtained together with a histogram showing the number of descriptors assigned to each visual word.

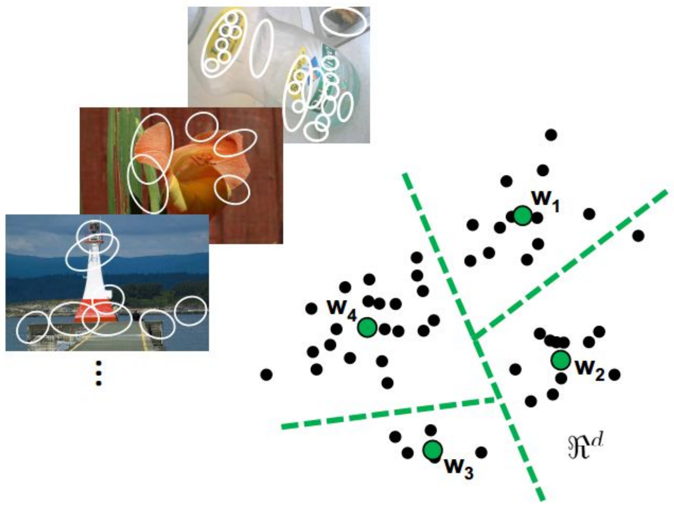
The vector is subsequently normalized, commonly using Euclidean normalization. *Td-idf* scoring, is used to discount the influence of background descriptors and euclidean distance is used for measuring similarity. Finally, several proposed solutions exist to improve the quality of this solutions.

Here is a much more detailed step-by-step explanation of the process as it is presented in [21]:

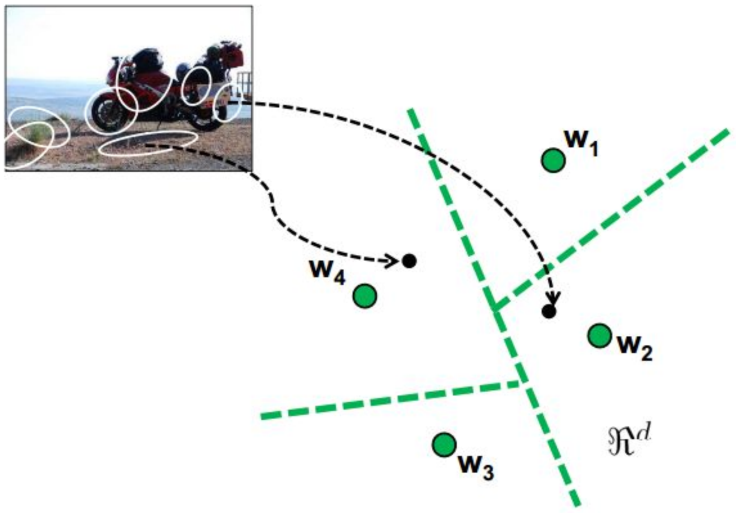
1. A database of training images is used to extract its features and and fill the feature space.



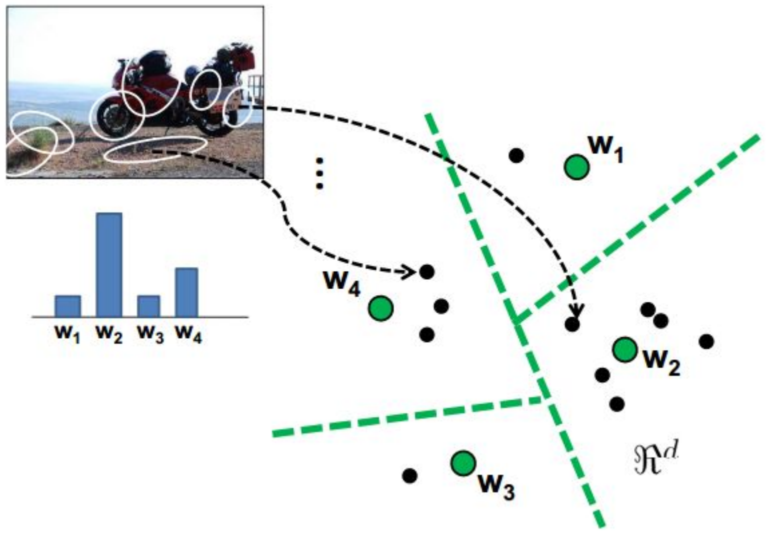
2. The extracted features are clustered in order to obtain a discrete number of visual words.



3. Given a new image, it's features are extracted and each of them is mapped to the nearest visual word, obtaining a fixed length vector the size of the pool of visual words that we are working with.



4. The histogram containing the information of the amount of descriptors mapped with each visual word is generated.



The reason why BOV is a popular solution for indexation and categorization problems can be explained by the following two characteristics:

1. The descriptors obtained from the images that are used in this representation are

powerful descriptors such as SIFT or GIST. These accomplish to represent the images characteristics very well and have a great impact on the final search accuracy of the whole process (SIFT descriptors have a better performance for categorization of similar images taken from different points of view [13][16])

2. The similarity between image vectors can be computed through simple standard distances. This also facilitates the usage of robust classification methods but it has to be noted that for BOV requires using non-linear classifiers.

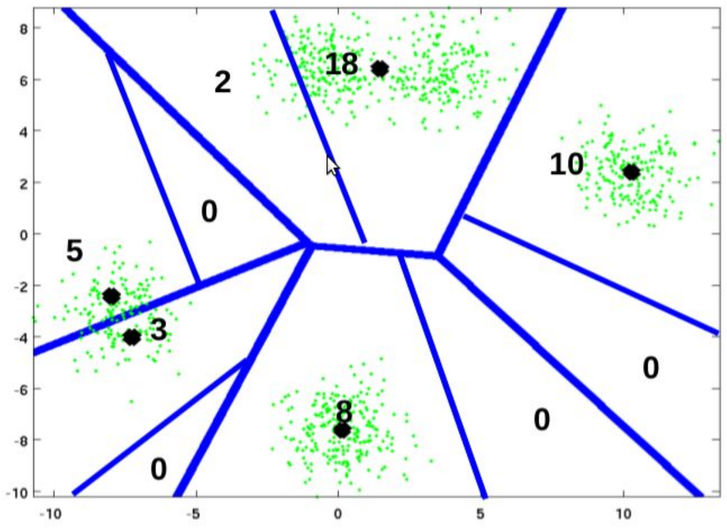
1.2.5.2 Comparison between Fisher Vectors and BOV

Given that BOV is by now the default solution for image indexation and categorization problem, it is time to draw a comparison between it and the proposed Fisher Kernel Framework in [6]. This is the way to detect which are the areas of improvements do Fisher Vector meet and make it a better solution.

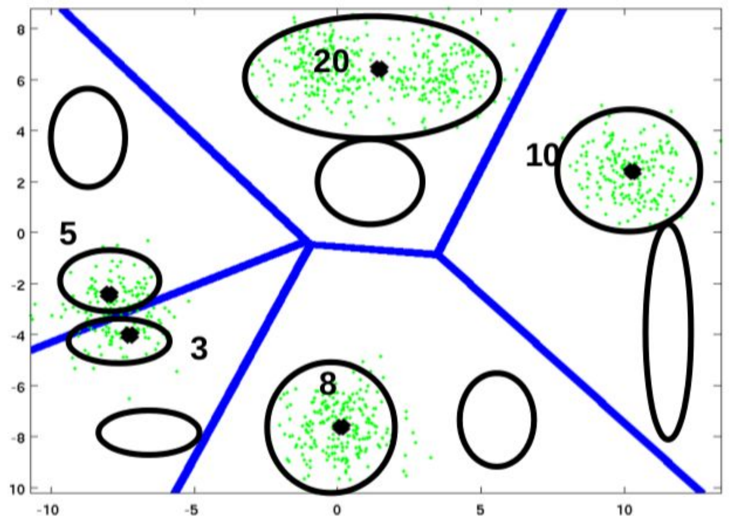
In [4], the following main points of similarity of difference are drawn:

1. Both BOV and Fisher Kernel Framework are systems based on a visual vocabulary.

2. Fisher vectors are based on a GMM clustering while BOV is based on k-means clustering.



BOV k-means clustering



Fisher Vectors GMM clustering

3. The image signature extracted by BOV is smaller than the Fisher Vector one for the same amount of image features. This is because BOV only encodes 0-order statistics while the Fisher Kernel Framework also encodes 1st and 2nd for the same vocabulary size. This enables to characterize images with very high-dimensional vectors that contain more information than the ones generated by BOV. For this reason, the Fisher Kernel Framework better accomplishes the fourth of the constraints mentioned in

section 1.1: scalability, because with a small dictionary it can perform better than BOV. This is the biggest advantage of Fisher vectors over BOV, despite the fact that they are less sparse.

4. BOV requires non-linear classifiers while Fisher vectors can be classified using linear ones. Consequently, Fisher vectors require less computation complexity in the classification process.

5. Both solutions discount the influence of background descriptors.

Besides these theoretical advantages, some previous papers have proved by experimentation that Fisher vectors outperform BOV under the same conditions. This is the case of [2] where Fisher vectors have been tested in different conditions (number of centroids *K* and different binarization techniques that will be discussed in following sections). The following table shows the results of one of this experiments, where search accuracy is measured by the mAP score, which counts the precision of the search for each query and averages it by the number of queries. Measurements have been done on the Holidays database of [15].

Method	**Holidays mAP score (%)**
BOV K=20000	43.7
BOV K=200000	54.0
Fisher Vector K=64 spectral hashing, 128 bits	39.4
Fisher Vector K=8 binarized	46.0
Fisher Vector K=64 binarized	57.4
Fisher Vector K=64 ADC 16x8 (D'=96)	50.6
Fisher Vector K=256 ADC 256x10 (D'=2048)	63.4

Despite having shown that Fisher Vectors are suitable for categorizing images, it is also clear that the image vectors obtained in this framework are high-dimensional and dense. This implies heavy image signatures that might be a problem at the time of using them for image categorization. This was the major drawback for the implementation of Fisher vectors for this problem, however, a series of compression and binarization techniques can be applied prior to the image search to speed up the process and without significant loss of performance. In the following section these techniques will be described.

1.3 Improving Fisher Vectors for image retrieval

As it has been proven in the previous sections, the Fisher Kernel Framework obtains high-dimensional feature vector database. To apply those to solve the problem of image retrieval many techniques have appeared to lighten the vectors in order to make it feasible to use the for this purpose. Among the many different techniques that have been tested to improve the performance of Fisher vectors, this project is going to focus on the ones mentioned on [1] for the sake of simplicity. Nevertheless, in this section some more will be introduced.

1.3.1 Principal Component Analysis

As the Fisher Kernel Framework characterizes images with very high-dimensional vectors it creates a very heavy signature. To reduce the dimensionality of the vectors, [1] applies Principal Component Analysis to project the image vectors and reduce their dimensions. [1] reports that applying this process has a positive impact on the accuracy of the whole process.

In a nutshell, PCA aims to orthogonally project into a new coordinate systems in order to reduce the correlation between vectors and, as an inherent result, reduce its dimensionality.

The final result is a set of projected vectors with equal or less components than the original ones. The mathematical steps behind this process with be listed now but not detailed because in the scope of this project, the important conclusion that has to be drawn from applying PCA to Fisher vectors is that it is a way to compress the image vectors without significant loss of information.

PCA step-by-step:

1. Collect the data subject to be projected
2. Subtract the mean
3. Calculate the covariance matrix if not available previously
4. Calculate the eigenvectors and eigenvalues of the covariance matrix
5. Selecting the components that will remain and build the feature vectors
6. Derive the new dataset of vectors

After having compressed the original set of Fisher vectors, in [1] these undergo two normalization processes.

1.3.2 L2 normalization

L2 normalization is a technique that aims to eliminate the dependence that Fisher image signature with ω which is the proportion of image-specific information contained in the image. In [5] G_λ^X is rewritten after some mathematical computations as:

$$G_\lambda^X \approx \omega \nabla_\lambda E_{x \sim q} \log u_\lambda(x)$$

after having split p split into two parts:

$$p(x) = \omega q(x) + (1-\omega) u_\lambda(x) \qquad 0 \leqslant \omega \leqslant 1$$

- u_λ : a background image-dependant part
- q : an image-specific part which follows an image-specific distribution q

As it can be seen in the rewritten version of G_λ^X there is a dependence with ω . This can cause that two images that show the same object together with different background information end up having different signatures. To eliminate this dependence it is necessary to L2-normalize the Fisher vectors. To do so, the original expression of the Fisher Kernel,

$$K(X,Y) = \boldsymbol{G}_\lambda^{X}{}'\boldsymbol{G}_\lambda^{Y}$$

is substituted by:

$$\frac{K(X,Y)}{\sqrt{K(X,X)K(Y,Y)}}$$

[5] reports that L2-normalization leads to large improvements

An aside that need to be noted by applying L2-normalization prior to the dot-product similarity process we are causing not to use dot product similarity on Fisher vector but actually use cosine similarity (i.e. the dot product between the L2 normalized vectors). The analysis that was done 1.2.4 remains valid. The only difference cosine-similarity brings is that it guarantees that the first retrieved image is indeed the query image, which is a positive outcome.

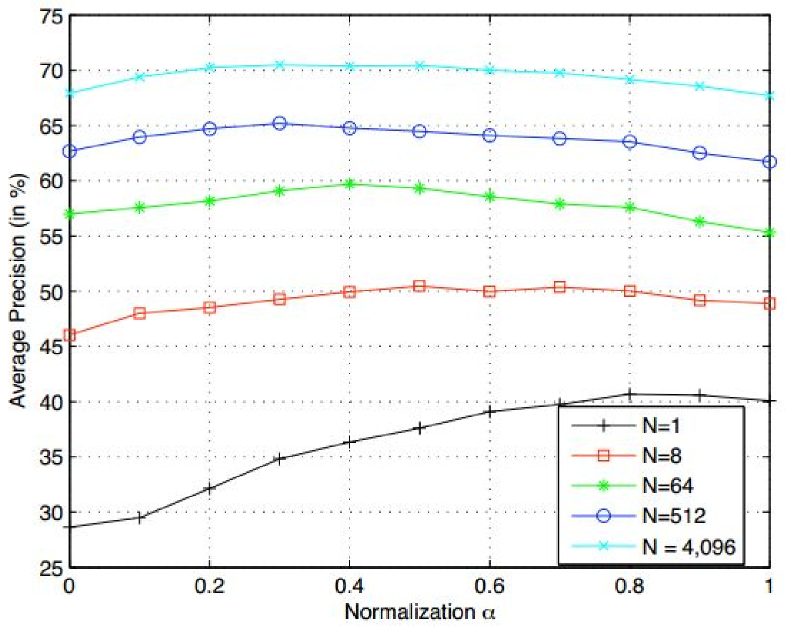
1.3.3 Power normalization

Power normalization is a tool used to reduce how sparse a vector representation is. In the context of this project, we use cosine-similarity for the image search. The fact is that both dot-product and L2 distance are non optimal for measuring similarity between sparse vectors. Therefore, the need of applying power normalization to “unsparsify” the vectors prior to computing similarity is clear.

In the Fisher Kernel Framework we are using GMM to generate image signatures. The number of Gaussians used can vary but, by increasing the number of Gaussians we are also causing the resulting Fisher vectors to be more sparse (fewer image descriptors are assigned to each Gaussian or visual word). With L2-normalization this side effect can be eliminated. To achieve this [5] proposed applying independently in each component the following function:

$$f(z) = sign(z)|z|^{\alpha} \qquad 0 \leqslant \alpha \leqslant 1$$

The optimal value of α depends upon the number of Gaussians. Nevertheless, [1] reports that setting $\alpha=0.5$ leads to near-optimal results independently of the number of Gaussians. This is better illustrated by the following graphic contained in [1], where *N* refers to the vocabulary size:



The following graphic extracted from [5] shows the effect of varying the amount of Gaussians used and of the power normalization on how sparse the Fisher vector representation is:

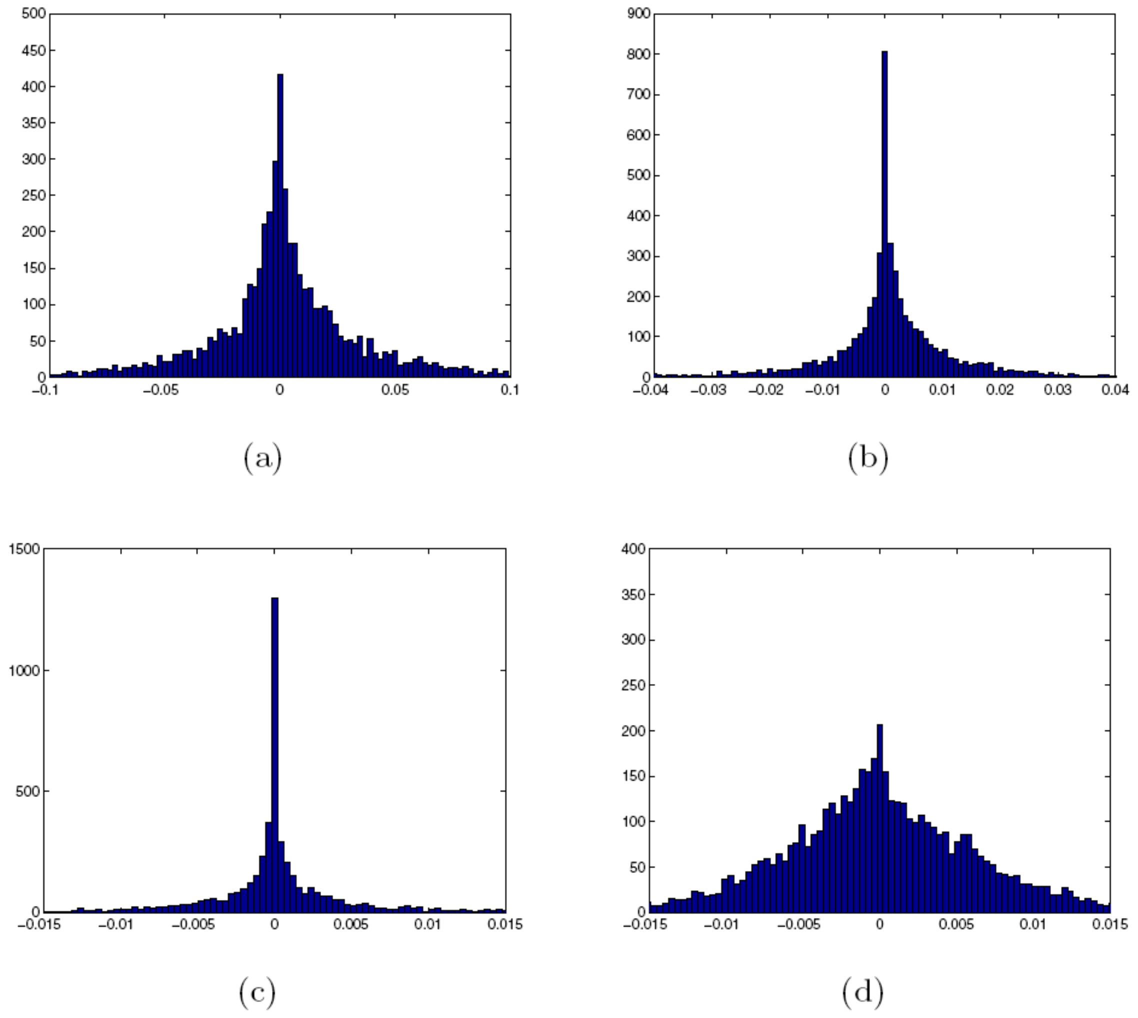


Fig. 1. Distribution of the values in the first dimension of the L2-normalized Fisher vector. (a), (b) and (c): resp. 16 Gaussians, 64 Gaussians and 256 Gaussians with no power normalization. (d): 256 Gaussians with power normalization ($\alpha = 0.5$). Note the different scales. All the histograms have been estimated on the 5,011 training images of the PASCAL VOC 2007 dataset.

1.3.4 Discrete Cosine Transform

This technique is useful for minimizing feature vector size in content-based image retrieval problems. It is closely related to the discrete Fourier transform, being a separable linear transformation. For an input image A and output image B, the technique is mathematically described as follows:

$$B_{pq} = \alpha_p \alpha_q \sum \sum A_{mn} \cos \frac{\pi(2\mathrm{m}+1)}{2\mathrm{M}} \cos \frac{\pi(2\mathrm{n}+1)}{2\mathrm{N}}$$
$$0 \leqslant p \leqslant M-1$$
$$0 \leqslant q \leqslant N-1$$
$$\alpha_p = \begin{Bmatrix} 1/\sqrt{M} & ,p=0 \\ \sqrt{2/M} & ,1 \leqslant p \leqslant M-1 \end{Bmatrix} \quad \alpha_p = \begin{Bmatrix} 1/\sqrt{N} & ,q=0 \\ \sqrt{2/N} & ,1 \leqslant q \leqslant M-1 \end{Bmatrix}$$

M and N being respectively the row and column size of A.

1.3.5 Walsh Transform

This technique defines a matrix formed by a set of rows $W_j \ for \ j=0,\ldots,N-1$ which have the following properties as defined in [22]:

- W_j takes ob the values +1 and -1.
- $W_j[0] = 1$ for all j.
- $W_j W_j^T = 0$ for all $j \neq k$ and $W_j W_j^T = N$ for all $j = k$.
- W_j has exactly j zero crossings for $j=0,\ldots,N-1$
- Each row W_j is even or odd with respect to its midpoint.

The feature vectors are subsequently transformed using this matrix.

1.3.6 Kekre's Transform

As explained in [22], this transform is an NxN matrix given by:

$$K_{NxN} = \begin{bmatrix} 1 & 1 & 1.. & 1 & 1 \\ -N+1 & 1 & 1.. & 1 & 1 \\ 0 & -N+2 & 1.. & 1 & 1 \\ \vdots & \vdots & \vdots\ \vdots & \vdots & \vdots \\ 0 & 0 & 0.. & 1 & 1 \\ 0 & 0 & 0.. & -N+(N-1) & 1 \end{bmatrix}$$

The feature vectors are subsequently transformed using this matrix.

1.3.7 Selection of techniques for the context of this project

In the Fisher Kernel Framework, the focus lies in the simplicity of the similarity computation when as simple dot-product operation between feature vectors can be an efficient way to measure the 'distance' between them. In order to preserve this important characteristic, it is key that this is maintained through the normalized Fisher vectors.

As explained in the previous subsections, applying power normalization does modify this characteristic. L2 normalization though changes it by converting it to a cosine-similarity problem (dot-product between L2 normalized vectors). Nevertheless, after these two normalization processes the similarity computation remains as simple as the Fisher Kernel Framework stresses to do.

At this time, there are available a large range of techniques to preprocess vectors before applying them to content-based image retrieval problems. The Discrete Cosine Transform (DCT), the Walsh Transform and the Kekre's Transform are just a few of them. In this project power normalization and L2 normalization were chosen because they were proven mathematically in [1] and in section 1.3.2 to preserve the properties of not normalized Fisher

Vectors and to yield good retrieval performance improvements in reducing memory usage and computing time without an impact on accuracy.

1.4 SEARCH AND INDEXATION

Once the Fisher vectors of the image database have been computed and the necessary modifications to compress them and optimize their future performance have been applied, it is time to actually compare the vectors with a query image and search which ones are found to be more similar. To achieve this there are also different procedures with their own success rate. In this section they are going to be briefly introduced.

1.4.1 ANN using Hamming embedded distance for similarity

In this project implementation section, this will be the used approach: approximate nearest neighbor search using Hamming embedded distances for similarity. This approach consist on what was already explained: the dot-product (or cosine similarity) between the Fisher vectors. Nevertheless, in [1] it is shown that for $\alpha \rightarrow 0$ the α normalized Fisher vectors converge to a ternary representation that can be transformed into an equivalent binary encoding after some mathematical binarization processes. The process is mathematically represented below, for mathematical interpretation we relate to [1]:

- w_i^X and δ_i^X are binarized and we obtain b_i^X and u_i^X :

$$w_i^X = \frac{1}{T}\sum_{t=1}^{T} \gamma_t(i) \rightarrow b_i^X = 1 \; if \;\; w_i^X > 0; \quad otherwise \;\; b_i^X = 0$$

$$\mu_i^X = \frac{\sum_{t=1}^{T} \gamma_t(i) x_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\delta_i^X = \frac{\mu_i^X - \mu_i}{\sigma_i} \rightarrow u_i^X = 1 \; if \;\; w_i^X > 0; \quad otherwise \;\; u_i^X = 0$$

- We recover the previous representation of the Fisher vector:

$$\frac{1}{T}\boldsymbol{G}_i^X = \frac{w_i^X}{\sqrt{w_i}}\delta_i^X$$

- The similarity measured as the dot-product of the Fisher vectors can be rewritten as:

$$K(X,Y) = \boldsymbol{G}_\lambda^X{}'\boldsymbol{G}_\lambda^Y = \alpha\cdot\sum_{i=1}^N \frac{w_i^X w_i^Y}{w_i}\delta_i^X{}'\delta_i^Y = \sum_{i=1}^N b_i^X b_i^Y\left(D - 2\text{Ha}\left(u_i^X, u_i^Y\right)\right)$$

- As we are using the cosine-similarity, we end up by normalizing the kernel function:

$$\frac{K(X,Y)}{\sqrt{K(X,X)K(Y,Y)}}$$

$Ha(x,y)$ stands for the Hamming distance between x and y .

Finally, once the similarity has been computed, an inverted list is created containing the most similar image found up to the less similar one. The size of this inverted list can be defined shorter than the whole amount of images for the sake of saving memory space and to avoid making an exhaustive search along all the similarities computed.

1.4.2 Spectral Hashing

This algorithm is directly applied to the L2 normalized Fisher vectors. It creates a binary encoding for each of the vectors, except the query ones. These binary codes are the ones that Spectral Hashing uses to compute the distance weighting them by the Gaussian kernel in a similar manner as in the previous section.

The difference lies in the way these binary codes are generated. In spectral hashing [11] the codes are obtained using the eigenfunctions of continuous Laplacian operators.

2 IMPLEMENTATION

Once the context and the theory of the image search problem has been introduced, it is time to proceed to describe the implementation realized within this project.

2.1 INTRODUCTION

As mentioned in the abstract, this project focuses on the Fisher Kernel Framework applied to the image search problem as it is defined in the article *Large-scale image retrieval with compressed fisher vectors"* [1] written by Florent Perronin, Yan Liu, Jorge Sánchez and Hervé Poirier during their work at Xerox Research Center Europe (XRCE). In this article, Fisher Vectors are shown to be a suitable solution once the drawbacks of them being high dimensional and dense, are solved.

To do so, the article talks about the previously analyzed projection and normalization techniques, as well as different search algorithms. More specifically, it describes the following ones:

- PCA projection of the Fisher Vectors.
- L2 Fisher Vector normalization.
- Power normalization
- Local Sensitive Hashing (LSH)
- Spectral Hashing (SH)

The scope of the implementation realized in this project will focus specifically on some of these points rather than in all the previously mentioned in section 1 of the project. In [2] all these different approaches are tested and the conclusions of this paper show that the setup that obtains better results is: PCA projected vectors followed by power and L2 normalization.

Therefore, the final program of this project will implement this setup.

It has also to be noted that among the constraints that have to be taken in account in the image categorization problem and described in section 1.1:

- Search accuracy
- Efficiency and memory usage
- Search time
- Scalability

the implementation will basically be focused on search accuracy. Nevertheless, search time will be used to observe the scalability constraint as well as a measure to support qualify the impact of using PCA projecting on the Fisher vectors. Specially because, in addition to implementing the prototype described in the article, this will also be tested with five different database sizes. The result of this tests will allow us to analyze the accuracy performance variability when adding different database images and also to prove the usefulness of projecting the Fisher vectors through the analysis of the search time related to the database size.

In conclusion, at the end of this implementation **we will have two different programs:**

1. A user interface where, using only the Holidays dataset, we will be able to see the retrieved images resulting from applying the Fisher Kernel Framework both using and not using PCA projecting over the image Fisher vectors.

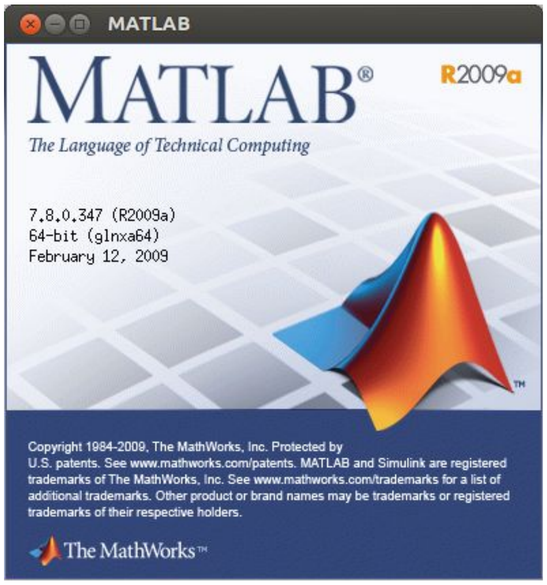
2. A test software where the framework is tested under 4 different database sizes, obtaining both timing results as well as the ROC curve also for not PCA projected vectors and PCA projected vectors.

2.2 EXPERIMENTAL SETUP

In this section we will describe the database and different on-line libraries and resources have been used in this project before describing more specifically the programming code.

2.2.1 SOFTWARE USED

This piece of software has been implemented using MATLAB R2009a by MathWorks Inc. for Linux 64-bits. All the toolboxes have been installed together with a compiler for *mex* files.



```
7.8.0.347 (R2009a)
64-bit (glnxa64)
February 12, 2009
```

Copyright 1984-2009, The MathWorks, Inc. Protected by U.S. patents. See www.mathworks.com/patents. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

2.2.2 YAEL LIBRARY

The implementation of this project is built completely around a library called *Yael* created by Hervé Jegou and Matthijs Douze and property of INRIA. It is available for download at https://gforge.inria.fr/projects/yael/ under a CeCILL license.

The library aims to compute efficiently basic operations like exhaustive nearest neighbor search function and others, in three different interfaces:

- C
- Python
- Matlab

Yael also allows to execute the more efficient implementations of some functions in the C interface by means of *.mex* files that can be compiled within Matlab. These functions are not directly implemented in $\textit{The MathWorks}^{TM}$ software.

2.2.3 Database

The database used for the project is the Holidays database provided in [15]. It contains a total of 1491 images, 500 of which are query images and 991 are a set of correct retrieval results for each of the queries. The descriptors are presented in *.siftgeo* format and all the information about this database and helper functions to work with it are available at http://lear.inrialpes.fr/people/jegou/data.php#holidays. The images include a large variety of scenes types and are diverse in rotations, viewpoints and illumination changes, in order to test the robustness of the systems that use it for image categorization.

The Holidays dataset will be the center of this implementation but to better test this prototype another database will be used: the Flickr60K database which contains 60000 images (also available for download on the same link provided for the Holidays dataset). It is also stored in the *.siftgeo* format, which makes it easier to work with. This database will be split and concatenated with the Holidays database in order to obtain 4 different database sizes:

- 1491 images from Holidays database.

- 1491 images from Holidays database, plus first 5000 images from Flickr60k database.
- 1491 images from Holidays database, plus first 10000 images from Flickr60k database.
- 1491 images from Holidays database, plus first 20000 images from Flickr60k database

2.2.4 FEATURE EXTRACTOR (SIFT)

The feature extractor used to obtain the descriptors of the Holiday dataset is a modified version of the software created by Krystian Mikolajczyk. The code is available on-line for download and use and available in the software attached with this project.

The type of descriptors obtained from the images are the SIFT descriptors. These are very powerful and have good performance under variety of viewpoints and rotation context. GIST are also a powerful type of descriptors but are outperformed by SIFT descriptors under the conditions of the Holidays dataset.

2.3 MEASUREMENT OF SEARCH PRECISION

In order to compute how valid is our implementation it is necessary to use some tools that measure the search precision obtained. We have available the following tools available, in the context of this implementation:

- **mAP**

  This tool provides a single-feature measure of quality across recall levels by counting the precision of the search for each query and averaging it by the number of queries (500 in our implementation).

  If a set of relevant documents for an information need $q_j \in Q \; is \; \{d_1, \ldots, d_{m_j}\}$ and

R_{jk} *is the set of ranked retrieval results starting at the top result until you get document* d_k

then:

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Precision(R_{jk})$$

1.

Precision is measured by the amount of relevant images retrieved over the total of retrieved images.

$$Precision = \frac{(relevant\ items\ retrieved)}{(items\ retrieved)}$$

- **recall @ R**

  This tool measures the rate of relevant images that are ranked in the top *R* positions. It is an important tool for two reasons. First, for a small *R* it indicates if this method provides a relevant shortlist for a subsequent precise direct image comparison. Second, it indicates the empirical distribution of the rank of true positives. The recall stands for the number of relevant retrieved images over the amount of total relevant images.

$$Recall = \frac{(relevant\ items\ retrieved)}{(relevant\ items)}$$

- **The ROC curve ("Receiver Operating Characteristics")**

  It plots the true positive rate (sensitivity or recall) against the false positive rate (or 1-specificity). This curve always goes from the (0,0) coordinate in the graph to the (1,1) coordinate. A good ROC curve has to steeply climb until reaching the horizontal asymptote value of one (sensitivity).

  In any test there are four possible outcomes:

- True positives (TP): retrieved relevant images
- True negatives (TN): non retrieved non relevant images
- False negatives (FN): relevant images that aren't retrieved
- False positives (FP): non relevant images that are retrieved

Therefore, the number of positives and the number of negatives equal to:

$$\begin{aligned} P &= TP+FN \\ N &= FP+TN \end{aligned}$$

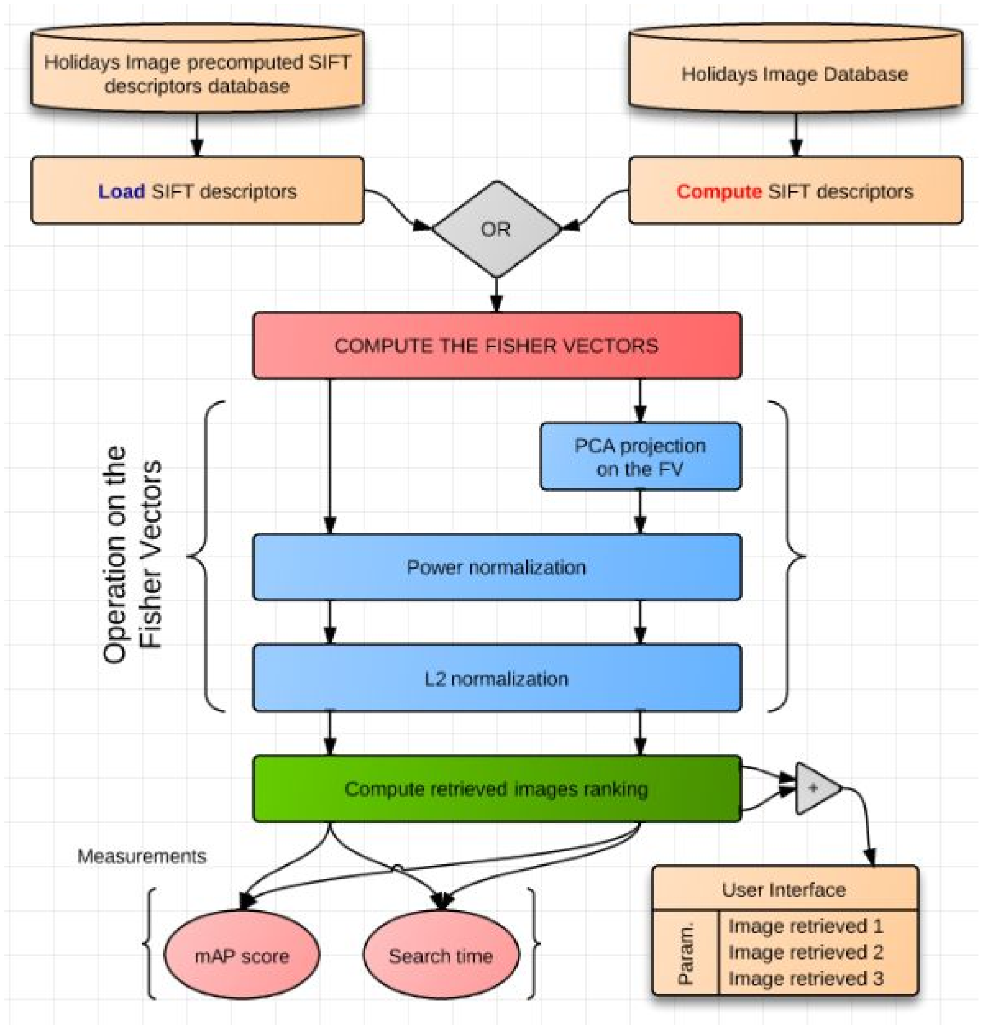
and sensitivity and 1-specificity equal to:

$$\begin{aligned} sensitivity\ (TPfraction) &= \frac{TP}{P} \\ 1-specificity\ (FPfraction) &= \frac{FP}{N} \end{aligned}$$

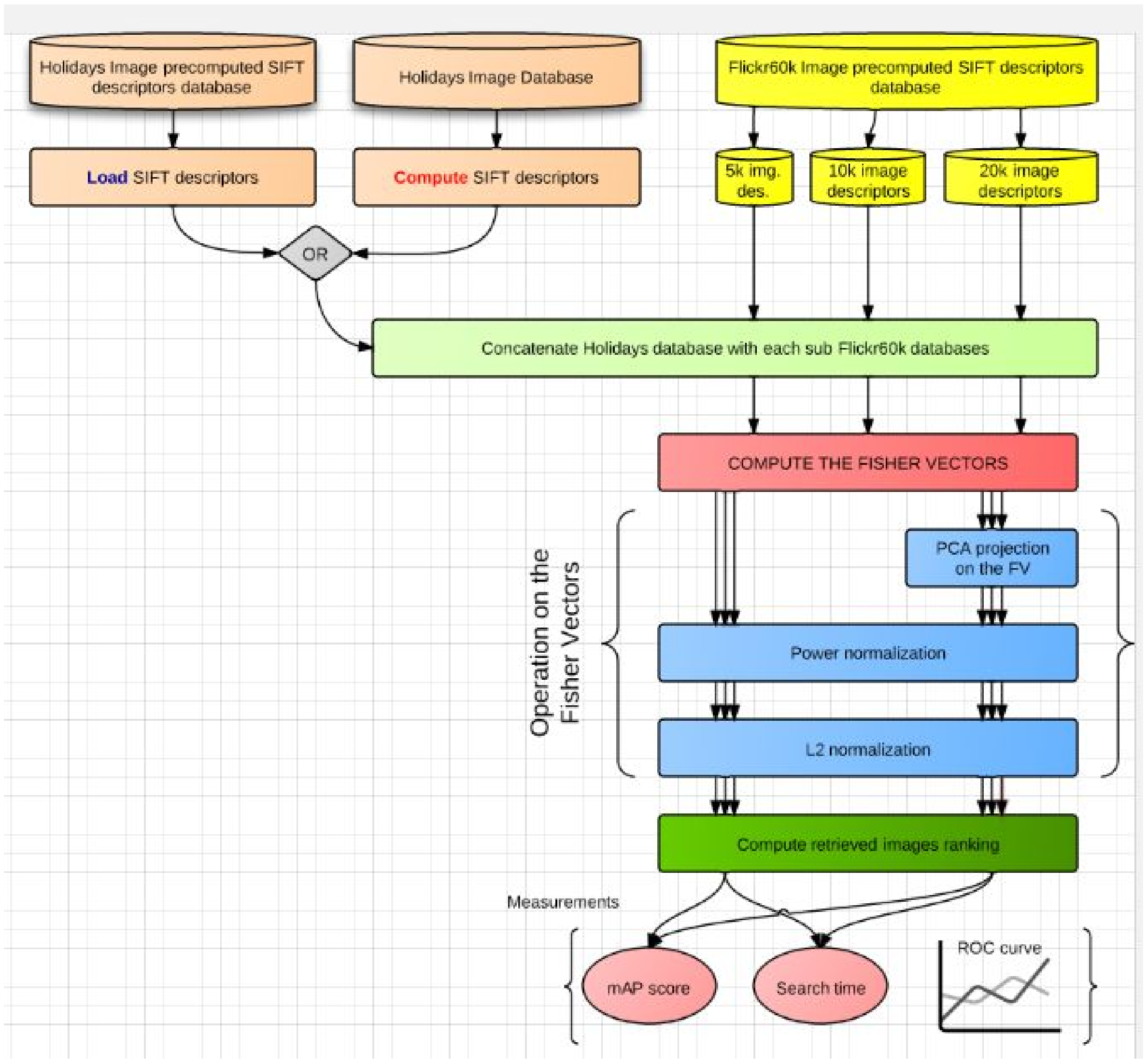
2.4 Implementation and User Interface

Next, it is time to describe the user interface implementation together with the "back-end". The approach to solve the process is procedural and this section will be divided into the small different parts according to the different purpose of each code snippet, and in order of execution. The following flowcharts better describes it.

- **User interface software**



- **Test software**


Holidays Image precomputed SIFT descriptors database
Holidays Image Database
Flickr60k Image precomputed SIFT descriptors database
Load SIFT descriptors
Compute SIFT descriptors
5k img. des.
10k image descriptors
20k image descriptors
OR
Concatenate Holidays database with each sub Flickr60k databases
COMPUTE THE FISHER VECTORS
Operation on the Fisher Vectors
PCA projection on the FV
Power normalization
L2 normalization
Compute retrieved images ranking
Measurements
mAP score
Search time
ROC curve

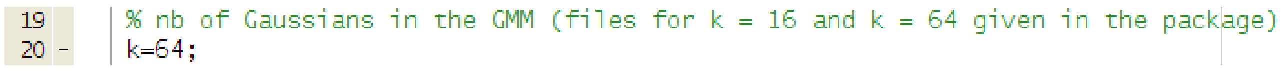
The user interface code that will be presented in this section will be mainly contained within the *fisher.m* file. It is in this file where the necessary Yael library functions are triggered in the correct order. In the last subsection, the focus will be in the user interface designed for this project, which corresponds with the *gui_fisher.m* file.

The test software where the different database sizes will be applied to the Fisher Kernel Framework is contained in the *test_fisher.m* file, to set the workspace environment, and the *test.m* file to run the tests.

2.4.1 Setting up the data and Loading the database

To begin with the implementation of the user interface the parameters of the image categorization process have to be set. Specifically the next ones:

1. **Number of Gaussians**



```
% nb of Gaussians in the GMM (files for k = 16 and k = 64 given in the package)
k=64;
```

In the Yael package we have the choice of 16 and 64 Gaussians. In this case, we chose 64 because, as seen in the following graphic from [6], for unsupervised learning Fisher reaches almost optimal performance around 64 Gaussians.

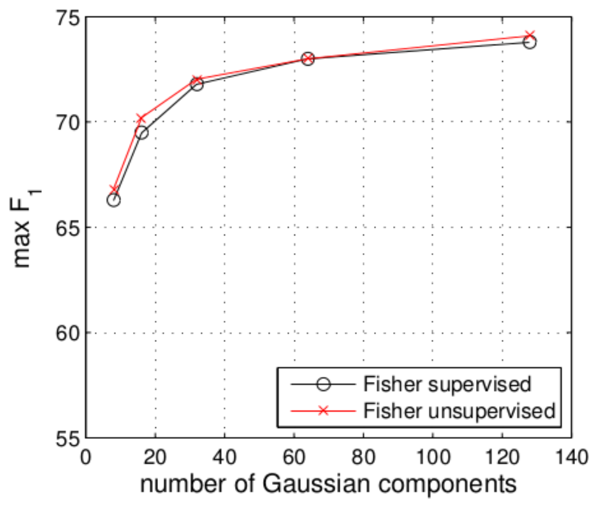


Figure 2. Performance of the Fisher kernels as a function of the number of Gaussian components on our in-house database. Performance plateaus after 128 Gaussians.

2. **GMM parameters**

   These are directly read from the *sift_pca64_k64.gmm* provided in [15].

```
% GMM parameters
f_centroids = sprintf('%s/sift_pca64_k%d.gmm', dir_data, k);
```

3. **Load the provided precomputed Fisher vectors**

```
% Precomputed Fisher descriptors
f_precomputed = sprintf('%s/holidays_fisher_k%d.fvecs', dir_data, k);
```

   In case we want to speed up compiling during the development time of this project, INRIA also provides a file containing the precomputed Fisher vectors of the Holidays database. This considerably reduces the time of execution. Nevertheless, in the final implementation of this project, the vectors are directly computed from the images every time we execute the program.

4. **PCA Matrix used to apply Principal Component Analysis**

```
% PCA matrix for the Fisher vectors
f_pca_proj = sprintf('%s/fisher_k%d_pca_matrix.fvecs', dir_data, k);
```

   This file is also provided by INRIA in [15].

5. **Set up the value of the number of dimensions remaining after PCA**

```
% nb of dimensions to keep after Fisher descriptor is projected
dd = 128;
```

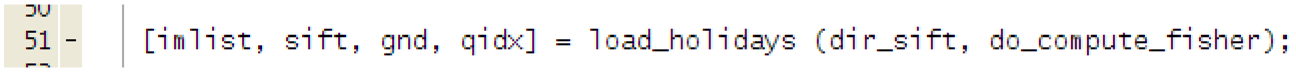
   The dimensions of the Fisher vectors will be reduced from 4096 to 128.

6. **Size of the shortlist of nearest neighbors of the query image. In other words, the amount of retrieved images we want to save**

```
% during evaluation, how many results per query to keep
shortlistsize = 1000;
```

This value is not very important and can be changed at any time. In the final user interface implementation, despite having stored 1000 retrieved results, only the first 5 will be displayed.

Once the parameters have been set, the Holidays database is loaded.



```
[imlist, sift, gnd, qidx] = load_holidays (dir_sift, do_compute_fisher);
```

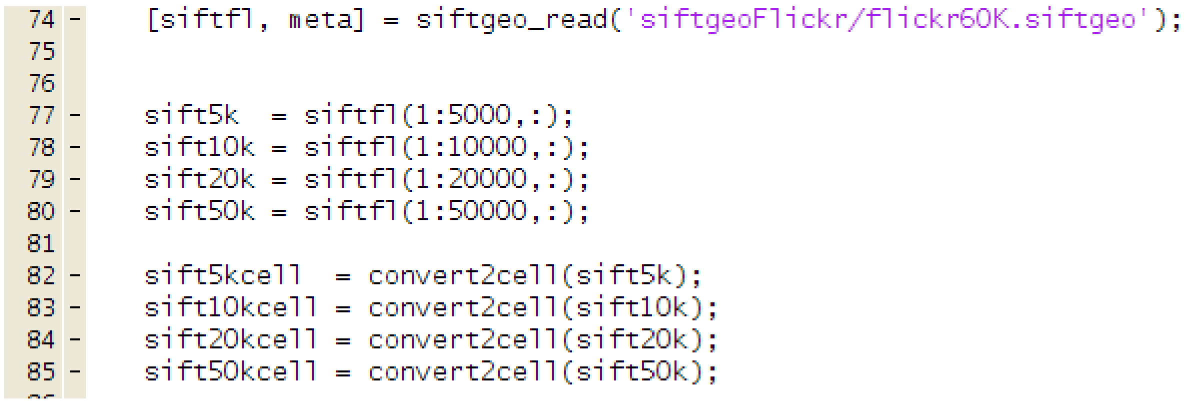
For this purpose, we use the provided function that deals with reading the file containing the database information. The parameters that are returned from this functions are the following:

Variable	**Value**	**Description**
imlist	< 1 x 1491 double>	List of image names (numbers)
sift	< 1491 x 1 cell >	List of cells each of one containing the SIFT descriptors of every image
gnd	< 500 x 1 cell >	Groundtruth: first value of each cell is the query number, next are the corresponding matching images
qidx	< 1 x 500 double >	The query images identifiers

These parameters will come in handy both for the computation of the Fisher vectors and for the evaluation of the obtained results. Especially the groundtruth, which contains the information of what we should obtain in case the program worked perfectly.

NOTE: for a more detailed information on how load_holidays function works, the reference manual of Yael package in [15].

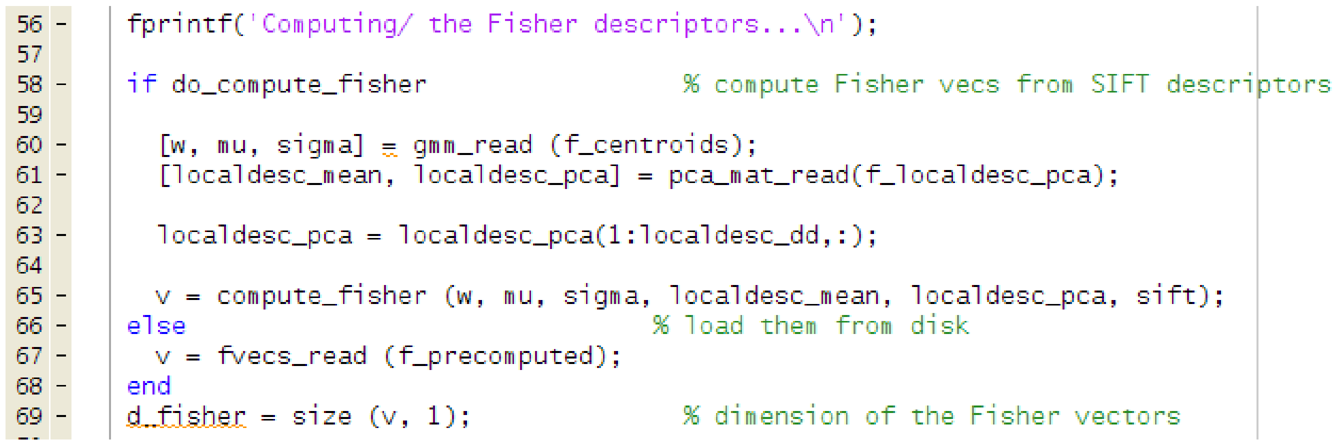
For the test software, in addition to the Holidays dataset, the Flickr60k dataset is also loaded and used to build the 5 different test databases as follows:



```
[siftfl, meta] = siftgeo_read('siftgeoFlickr/flickr60K.siftgeo');


sift5k  = siftfl(1:5000,:);
sift10k = siftfl(1:10000,:);
sift20k = siftfl(1:20000,:);
sift50k = siftfl(1:50000,:);

sift5kcell  = convert2cell(sift5k);
sift10kcell = convert2cell(sift10k);
sift20kcell = convert2cell(sift20k);
sift50kcell = convert2cell(sift50k);
```

2.4.2 Computing the Fisher Vectors

Once the workspace has been populated with all the necessary parameters to compute the Fisher vectors, we can proceed to yet again make use of the Yael package to compute them:



```
fprintf('Computing/ the Fisher descriptors...\n');

if do_compute_fisher                % compute Fisher vecs from SIFT descriptors

  [w, mu, sigma] = gmm_read (f_centroids);
  [localdesc_mean, localdesc_pca] = pca_mat_read(f_localdesc_pca);

  localdesc_pca = localdesc_pca(1:localdesc_dd,:);

  v = compute_fisher (w, mu, sigma, localdesc_mean, localdesc_pca, sift);
else                               % load them from disk
  v = fvecs_read (f_precomputed);
end
d_fisher = size (v, 1);             % dimension of the Fisher vectors
```

As we can see, a conditional is added in case it is preferable to use the precomputed descriptors when we are in development time.

The process is quite straight-forward (see flowchart in the beginning of section 2.4.):

1. We read the parameters of the centroids that form the visual vocabulary, previously

trained in the *yael_gmm.m* function.

2. We read the mean and PCA matrix stored in another of the provided files.
3. We compute the Fisher vectors by providing:
    - The GMM parameters
    - The mean and PCA matrix
    - The array of cells containing the Holidays database image descriptors
4. We pre-process the Fisher Vectors prior to compute the retrieved images in two different ways.
5. We compute the ranking of images.
6. We obtain some measures and the user interface.

The result is a matrix containing the Fisher vectors of the 1491 images of the database arranged in columns. As it was explained in the theoretical approach section, these vectors are not directly amenable to the image categorization problem unless they undergo a series of projections(optional) and normalizations.

For the test software the implementation is identical but the process has to be repeated for each of the databases, which considerably increases the execution time. (See flowchart at the beginning of section 2.4.)

2.4.3 APPLYING POWER NORMALIZATION

As described in section 1.4.2, this normalization consist in applying the following formula to the Fisher vectors:

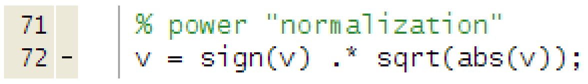
$$f(z) = sign(z)|z|^{\alpha} \qquad 0 \leqslant \alpha \leqslant 1$$

also, as concluded in [2], a nearly optimal solution is always reached by setting $\alpha = 0.5$

which can be rewritten as:

$$f(z) = sign(z)\sqrt{|z|} \qquad \alpha = 0.5$$

Therefore, this is easily translated into code in the following way:

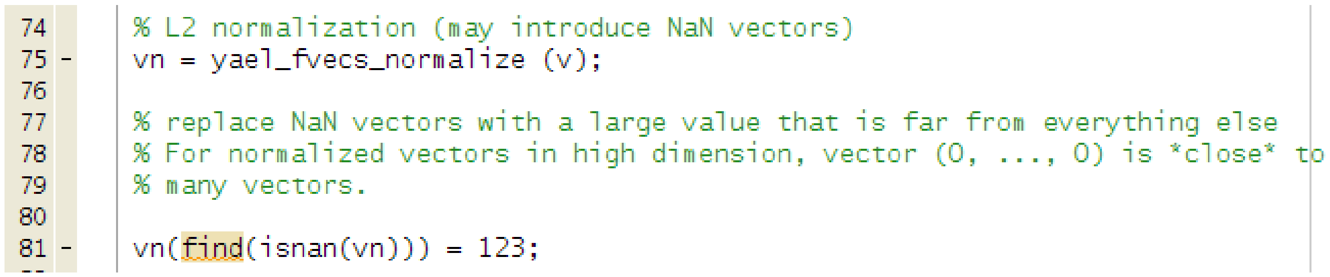


```
% power "normalization"
v = sign(v) .* sqrt(abs(v));
```

We override the resulting vectors with the original ones since this normalization will always be applied.

2.4.4 L2 NORMALIZATION

For L2 normalization Yael package also provides a better and more efficient C function that can be used in Matlab through the corresponding *.mex* file. The code to implement this is simple, however, "not-a-number" results (or NaN) may be introduced in the normalization process, and this has to be corrected by replacing these values with a high number.



```
% L2 normalization (may introduce NaN vectors)
vn = yael_fvecs_normalize (v);

% replace NaN vectors with a large value that is far from everything else
% For normalized vectors in high dimension, vector (0, ..., 0) is *close* to
% many vectors.

vn(find(isnan(vn))) = 123;
```

At this point, we already have the Fisher vectors ready to be applied to the problem of image categorization. From here on, the program will split in two different paths:

1. Directly performing a nearest neighbor search on these vectors to obtain the retrieved images and, subsequently, measure the performance of this process by using the

mAP tool described in section 2.3. This will be called the f"Full Fisher" implementation.

2. Prior to the nearest neighbor search, apply the PCA matrix to the vectors in order to reduce their dimensionality. Once projected, proceed to obtain the retrieved images following the same process as in the first path. This will be called the "Fisher + PCA implementation".

2.4.5 FISHER WITHOUT PCA PROJECTIONS

For this path of the implementation we make use of two functions:

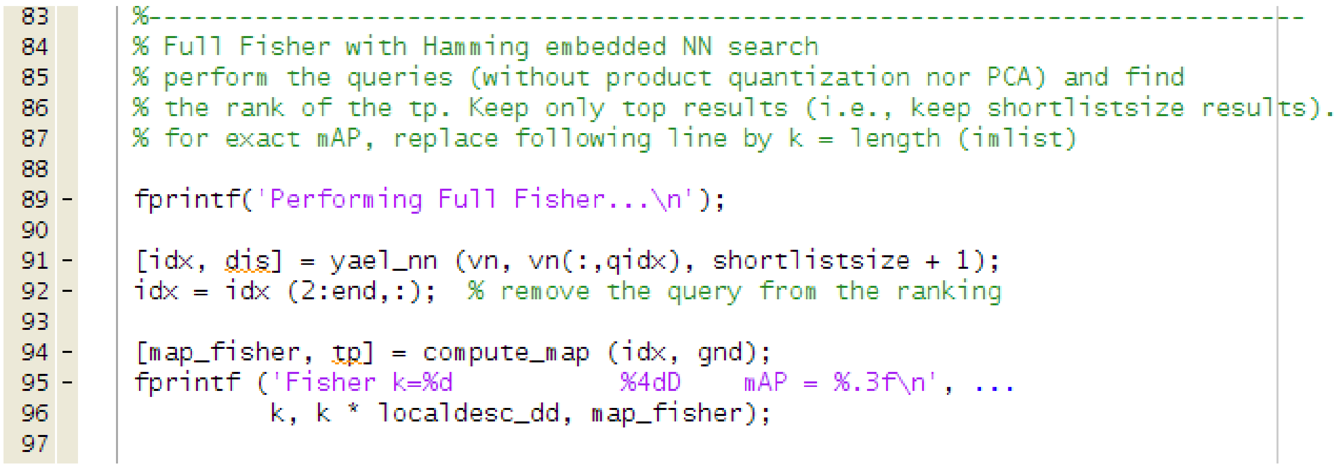
- **yael_nn**
  This functions perform the nearest neighbor search as it is presented in section 1.5.1. For it to work, we have to provide it with the normalized Fisher vectors of the whole database as well as a sub-matrix of these vectors containing only the ones corresponding to query images. Also, we can denote the amount of retrieved images to return.
  As a result, we get the following parameters:

Variable	**Value**	**Description**
idx	<1001 x 500 int32>	The vector index of the "shortlistsize" nearest neighbors
dis	<1001 x 500 single>	The corresponding square distances of these vectors to the query images

Note that we eliminate the first row of *idx* because, since we used L2 normalization, it is guaranteed that this row corresponds to the query image itself and, consequently, needs to be taken away from the *idx* ranking.

- **compute_map**

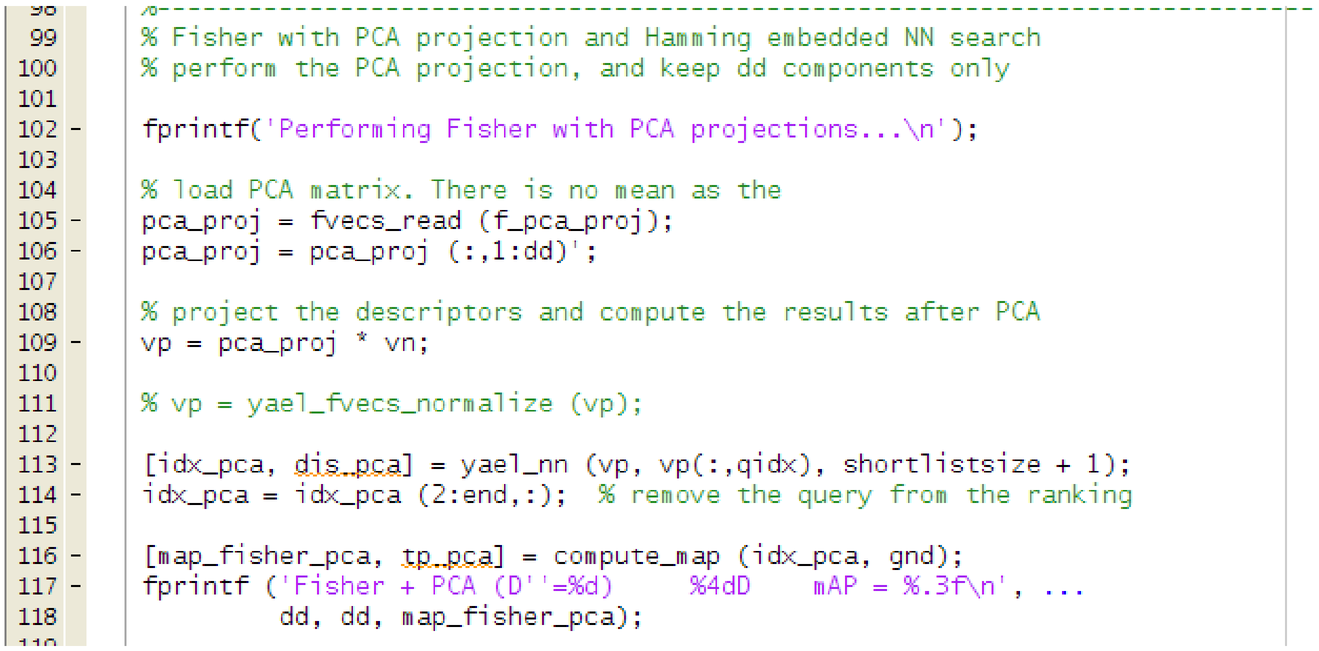
  This function computes the mAP score by taking the provided *groundtruth* containing the ideal retrieval information and comparing it to the *idx* variable that contains the experimental retrieval results.



```
%--------------------------------------------------------------------------
% Full Fisher with Hamming embedded NN search
% perform the queries (without product quantization nor PCA) and find
% the rank of the tp. Keep only top results (i.e., keep shortlistsize results).
% for exact mAP, replace following line by k = length (imlist)

fprintf('Performing Full Fisher...\n');

[idx, dis] = yael_nn (vn, vn(:,qidx), shortlistsize + 1);
idx = idx (2:end,:);  % remove the query from the ranking

[map_fisher, tp] = compute_map (idx, gnd);
fprintf ('Fisher k=%d           %4dD    mAP = %.3f\n', ...
         k, k * localdesc_dd, map_fisher);
```

2.4.6 FISHER + PCA

For the Fisher + PCA implementation, we also make use of yael_nn and compute_map. Nevertheless, the vectors we provide these functions with are a previously projected version of them. To achieve this, the PCA matrix is loaded and multiplied with the power and L2 normalized Fisher vectors and a reduce dimensionality matrix of image vectors is obtained.



```
%-----------------------------------------------------------------------
% Fisher with PCA projection and Hamming embedded NN search
% perform the PCA projection, and keep dd components only

fprintf('Performing Fisher with PCA projections...\n');

% load PCA matrix. There is no mean as the
pca_proj = fvecs_read (f_pca_proj);
pca_proj = pca_proj (:,1:dd)';

% project the descriptors and compute the results after PCA
vp = pca_proj * vn;

% vp = yael_fvecs_normalize (vp);

[idx_pca, dis_pca] = yael_nn (vp, vp(:,qidx), shortlistsize + 1);
idx_pca = idx_pca (2:end,:);  % remove the query from the ranking

[map_fisher_pca, tp_pca] = compute_map (idx_pca, gnd);
fprintf ('Fisher + PCA (D''=%d)     %4dD    mAP = %.3f\n', ...
         dd, dd, map_fisher_pca);
```

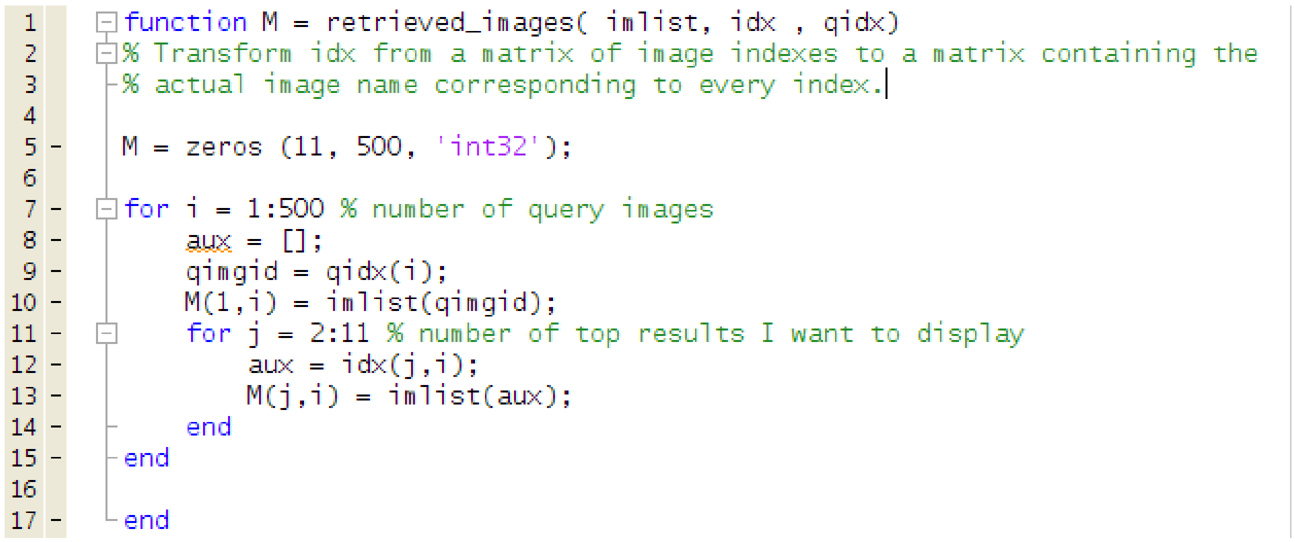
2.4.7 User Interface

At this point, we have the following results over the Holidays database:

- mAP scores for Full Fisher and Fisher + PCA implementations.
- An index and distance matrices containing the ranking of retrieved images for each image query and the distances of those retrieved images to the query itself.

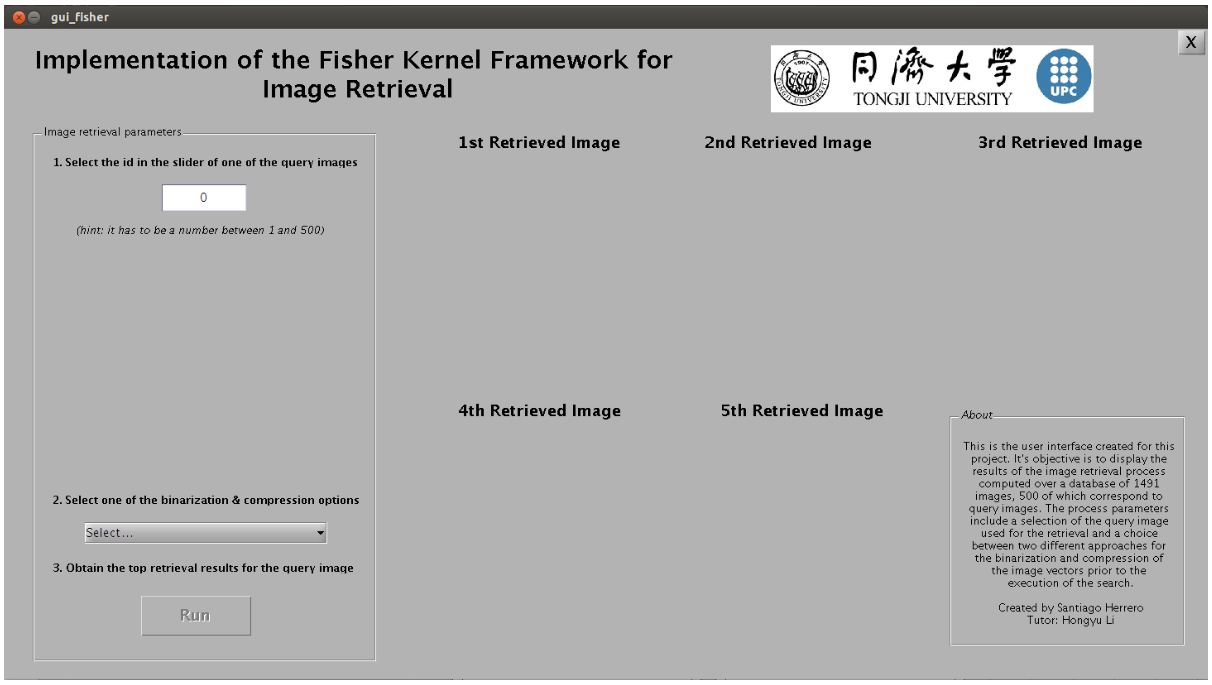
In order to better display the performance of this program, it was a need to build a friendly user interface. First though, we need some variable or matrix that contains the name of the images ranked by their proximity with the query images so that we can easily display those images by grabbing its name from the matrix. To achieve this we use *retrieved_images* function.

With this function we obtain the matrix of ranked image names that we could cast from a user interface to display the retrieved images easily.



```
function M = retrieved_images( imlist, idx , qidx)
% Transform idx from a matrix of image indexes to a matrix containing the
% actual image name corresponding to every index.

M = zeros (11, 500, 'int32');

for i = 1:500 % number of query images
    aux = [];
    qimgid = qidx(i);
    M(1,i) = imlist(qimgid);
    for j = 2:11 % number of top results I want to display
        aux = idx(j,i);
        M(j,i) = imlist(aux);
    end
end

end
```

Given both M and M_pca matrices, corresponding to the Full Fisher implementation and the Fisher + PCA one respectively, a user interface is created using the built-in Matlab GUI tools. The final appearance of the GUI is as follows:



Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

0

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

Select...

3. Obtain the top retrieval results for the query image

Run

1st Retrieved Image

2nd Retrieved Image

3rd Retrieved Image

4th Retrieved Image

5th Retrieved Image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

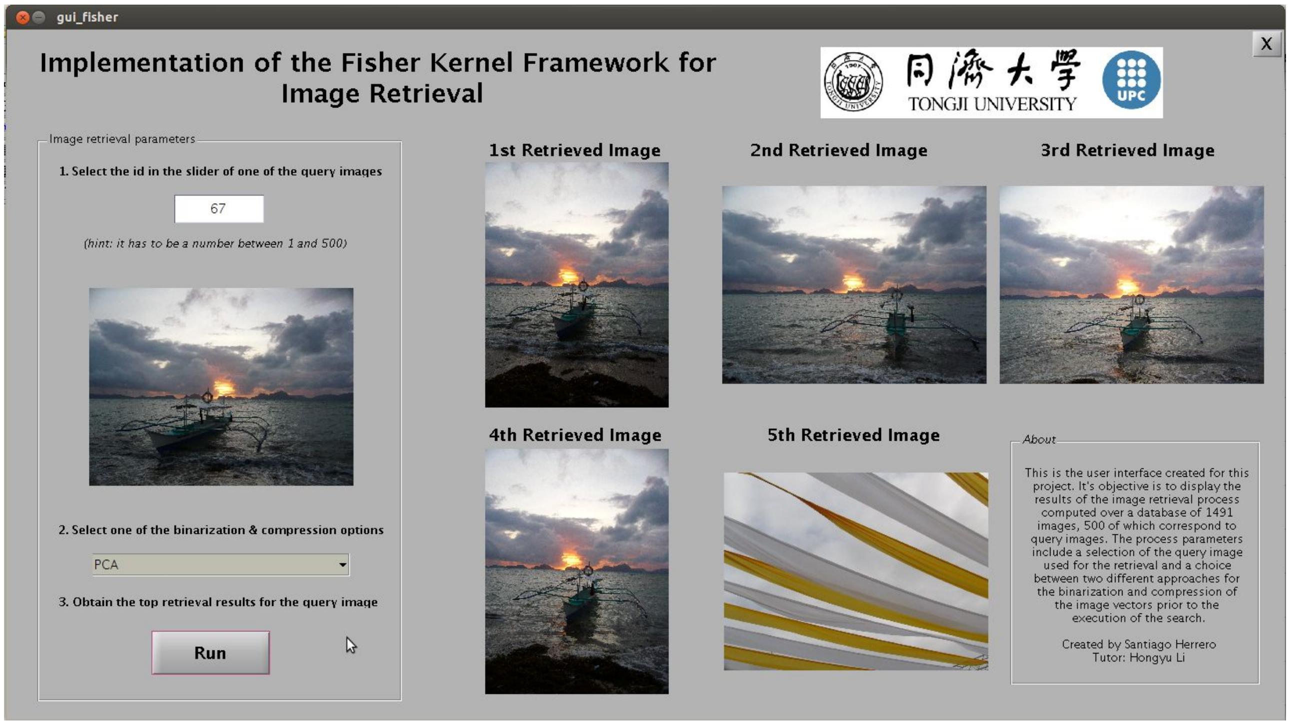
On the left, we have a box where the user has to set all the necessary parameters to run to search. These are:

1. An edit text where an index (going from 1 to 500) has to be introduced to select among the 500 query images available in the Holidays dataset.

2. A select list where we have to select either "Simple" corresponding to applying the nearest neighbor search directly to the normalized Fisher vectors, or "PCA" corresponding to do so with the projected and normalized Fisher vectors.

Once these two parameters are correctly set the "RUN" button is toggled and we can trigger the image categorization problem. Here we display four examples of the results obtained:

- **EXAMPLE 1:** Query image 67 and Fisher + PCA



Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

67

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

PCA

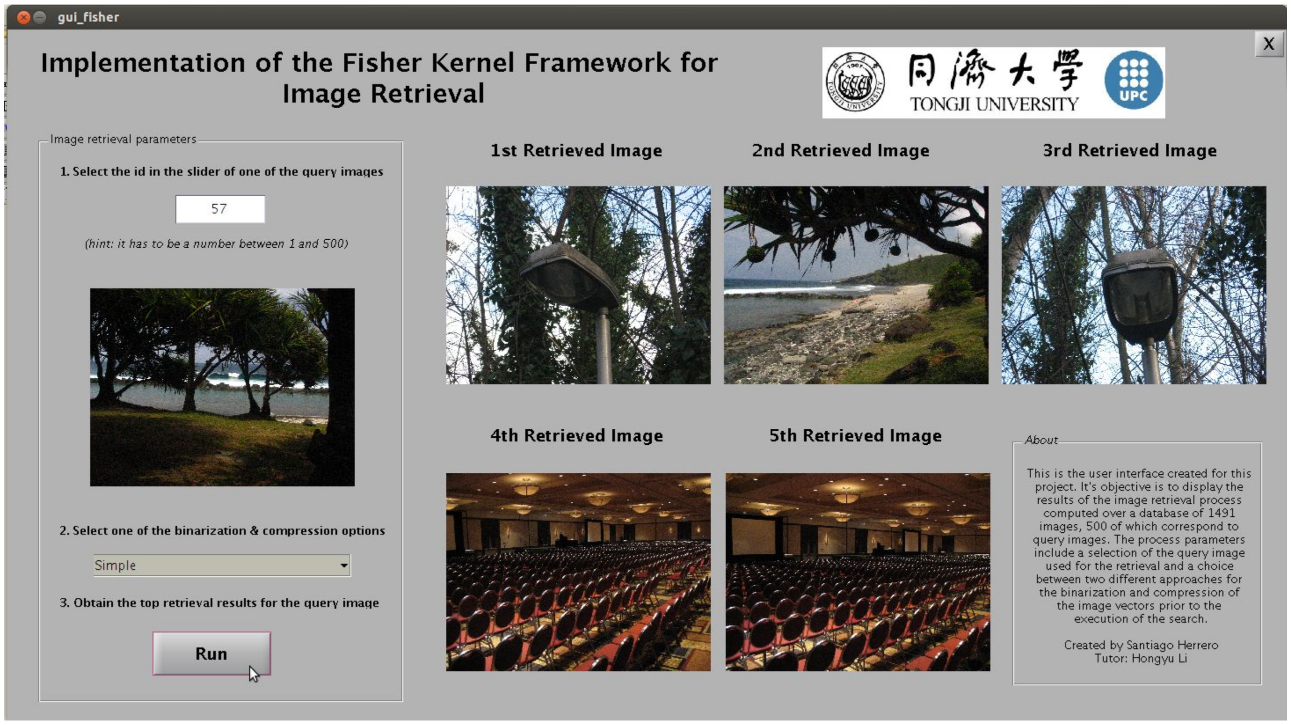
3. Obtain the top retrieval results for the query image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

- **EXAMPLE 2:** Query image 57 and Full Fisher



Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

57

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

Simple

3. Obtain the top retrieval results for the query image

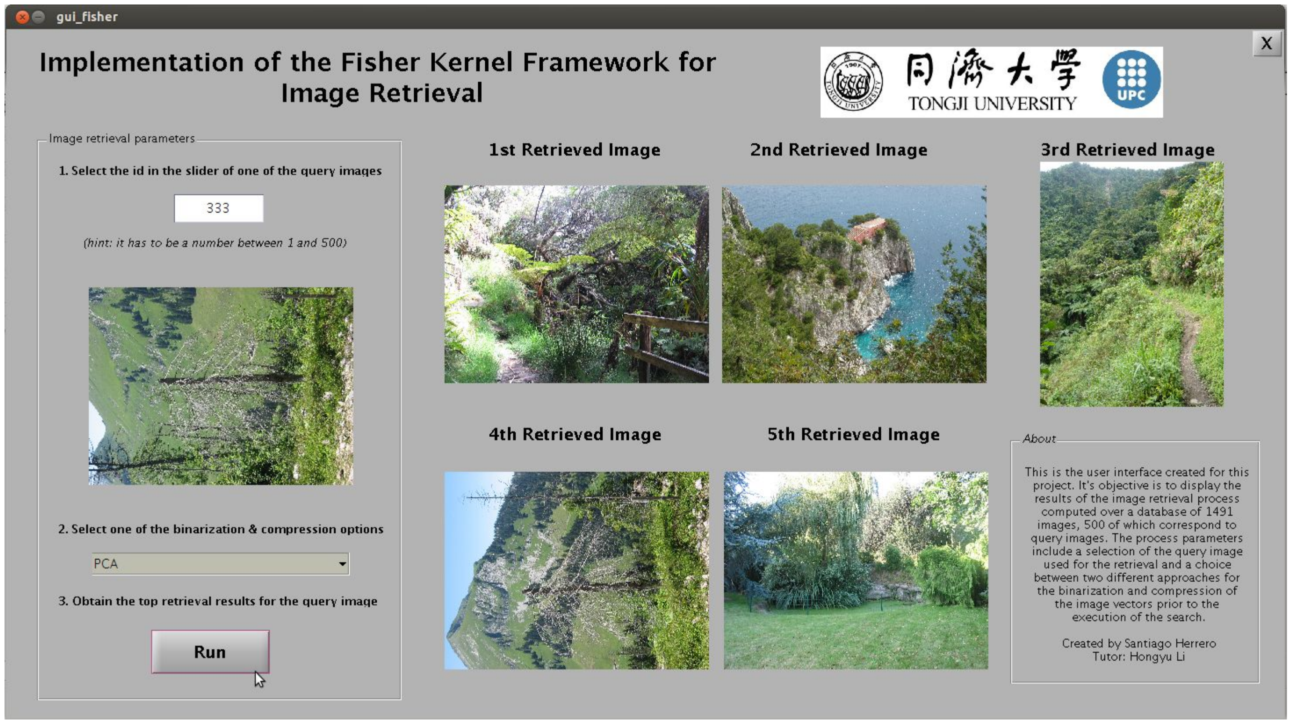
1st Retrieved Image | 2nd Retrieved Image | 3rd Retrieved Image | 4th Retrieved Image | 5th Retrieved Image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

- **EXAMPLE 3:** Query image 333 and Fisher + PCA



Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

333

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

PCA

3. Obtain the top retrieval results for the query image

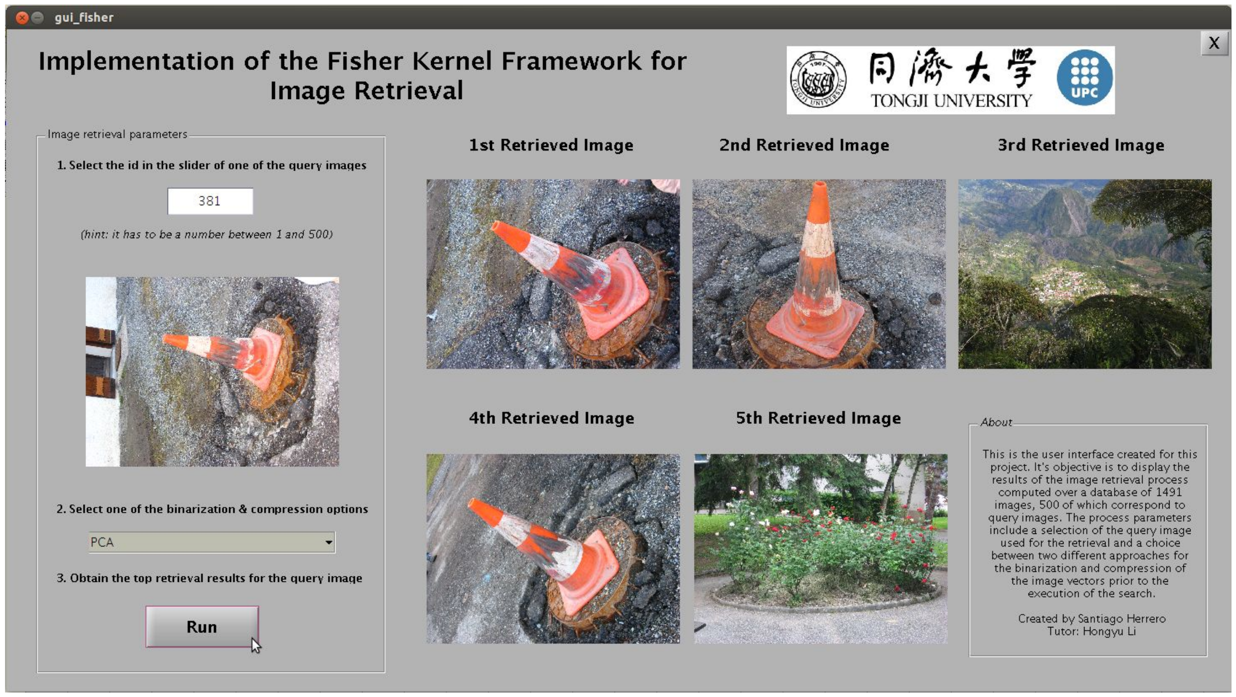
1st Retrieved Image | 2nd Retrieved Image | 3rd Retrieved Image | 4th Retrieved Image | 5th Retrieved Image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

- **Example 4:** Query image 381 and Fisher + PCA



gui_fisher

Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

381

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

PCA

3. Obtain the top retrieval results for the query image

Run

1st Retrieved Image

2nd Retrieved Image

3rd Retrieved Image

4th Retrieved Image

5th Retrieved Image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

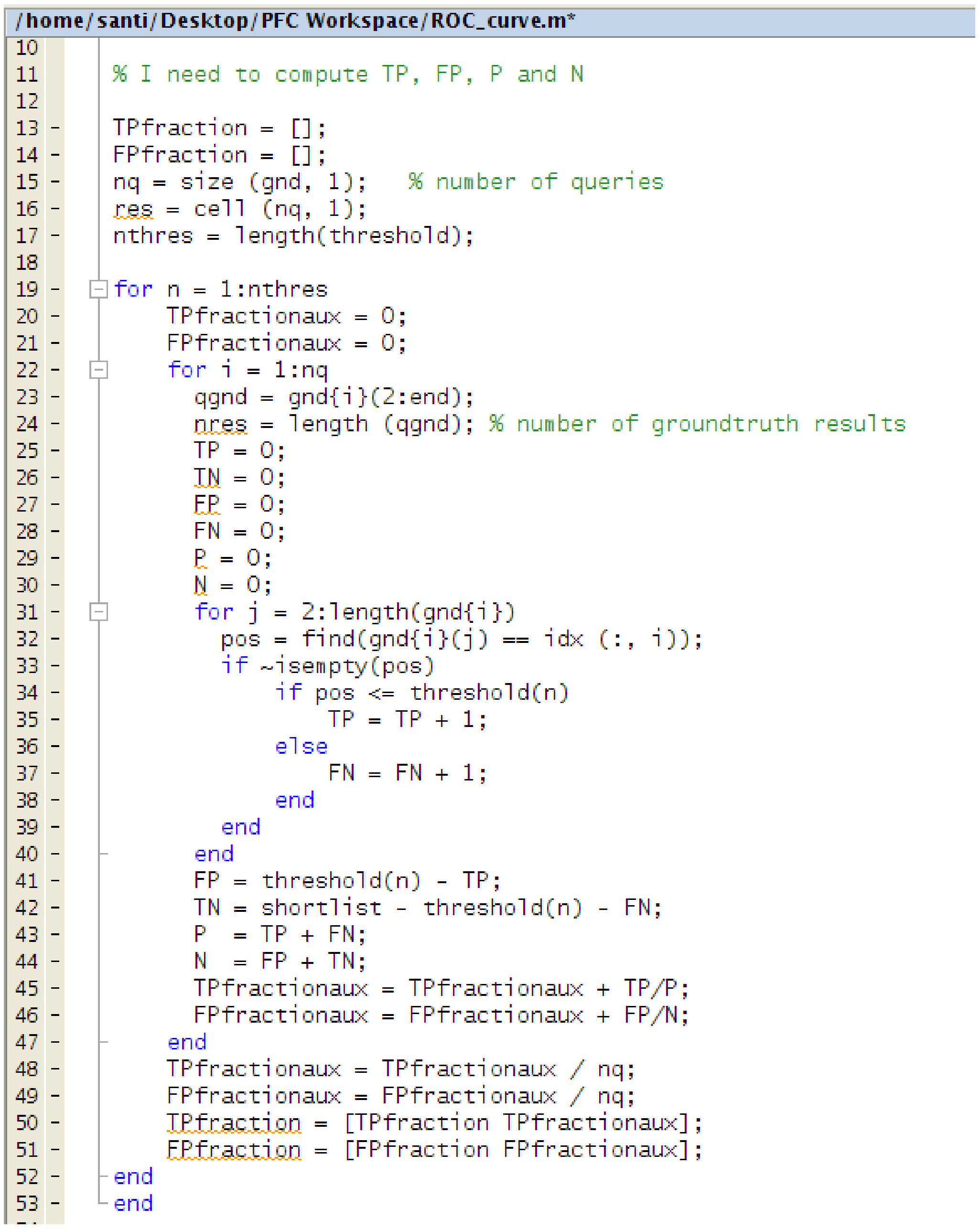
Created by Santiago Herrero
Tutor: Hongyu Li

It has to be noted that the user interface only uses the Holidays database because we have the real images as well as the descriptors. For the Flickr60k database we only have the descriptors for the images and, therefore, we will only use it for testing purposes.

2.4.8 Test Software

For the test software we have the following:

- mAP scores for Full Fisher and Fisher + PCA implementations over 4 different databases.
- An index and distance matrices containing the ranking of retrieved images for each image query and the distances of those retrieved images to the query itself, for the 4 databases.
- The ROC curve representations.

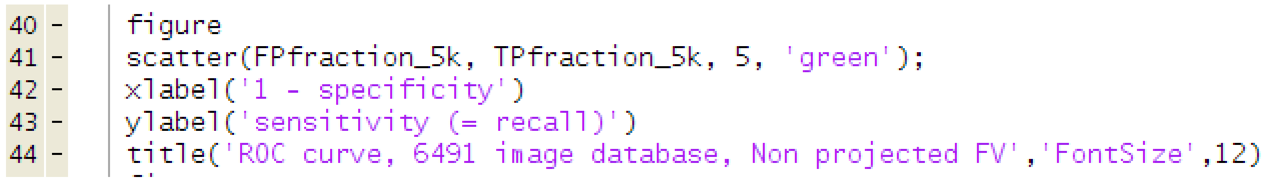


/home/santi/Desktop/PFC Workspace/ROC_curve.m*

```matlab

% I need to compute TP, FP, P and N

TPfraction = [];
FPfraction = [];
nq = size (gnd, 1);   % number of queries
res = cell (nq, 1);
nthres = length(threshold);

for n = 1:nthres
    TPfractionaux = 0;
    FPfractionaux = 0;
    for i = 1:nq
      qgnd = gnd{i}(2:end);
      nres = length (qgnd); % number of groundtruth results
      TP = 0;
      TN = 0;
      FP = 0;
      FN = 0;
      P = 0;
      N = 0;
      for j = 2:length(gnd{i})
        pos = find(gnd{i}(j) == idx (:, i));
        if ~isempty(pos)
            if pos <= threshold(n)
                TP = TP + 1;
            else
                FN = FN + 1;
            end
        end
      end
      FP = threshold(n) - TP;
      TN = shortlist - threshold(n) - FN;
      P  = TP + FN;
      N  = FP + TN;
      TPfractionaux = TPfractionaux + TP/P;
      FPfractionaux = FPfractionaux + FP/N;
    end
    TPfractionaux = TPfractionaux / nq;
    FPfractionaux = FPfractionaux / nq;
    TPfraction = [TPfraction TPfractionaux];
    FPfraction = [FPfraction FPfractionaux];
end
end
```

In this case, we don't have available the real images of the Flickr60k database so there is no need to build the matrix, as in the previous section, that will allow us to visually show the retrieval results. Instead, we create a snippet of code that will provide us with the ROC curve representation. The code is the following:

With this, we obtain the *x* and *y* vectors of the ROC curve corresponding to *FPfraction* and *TPfraction* respectively. We can proceed to represent them as follows (example of only one of the graphs):

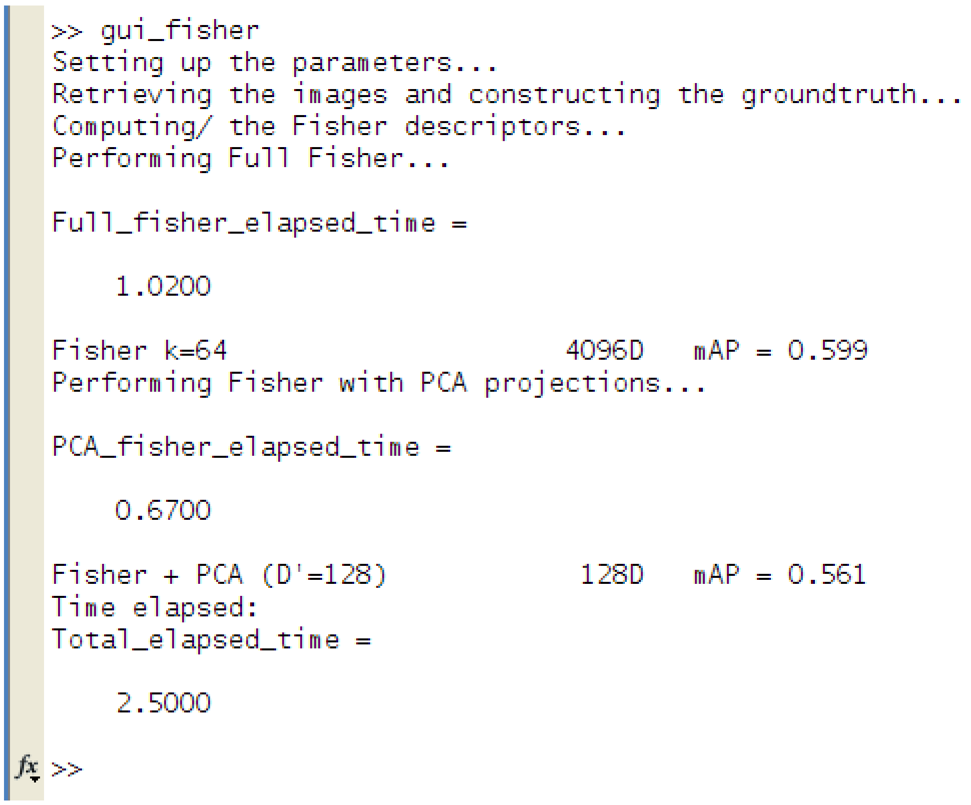


```
figure
scatter(FPfraction_5k, TPfraction_5k, 5, 'green');
xlabel('1 - specificity')
ylabel('sensitivity (= recall)')
title('ROC curve, 6491 image database, Non projected FV','FontSize',12)
```

At this point all the code has been briefly presented. If more insight is needed the code is available at the attached file together with this project. Now it's time to look at the results.

2.5 RESULTS

When we run any permutation of query image and binarization option in **the user interface**, both full Fisher and PCA Fisher implementations are computed. Therefore, we obtain the mAP score for both of them. The output is the following one:



```
>> gui_fisher
Setting up the parameters...
Retrieving the images and constructing the groundtruth...
Computing/ the Fisher descriptors...
Performing Full Fisher...

Full_fisher_elapsed_time =

    1.0200

Fisher k=64                    4096D   mAP = 0.599
Performing Fisher with PCA projections...

PCA_fisher_elapsed_time =

    0.6700

Fisher + PCA (D'=128)           128D   mAP = 0.561
Time elapsed:
Total_elapsed_time =

    2.5000

>>
```

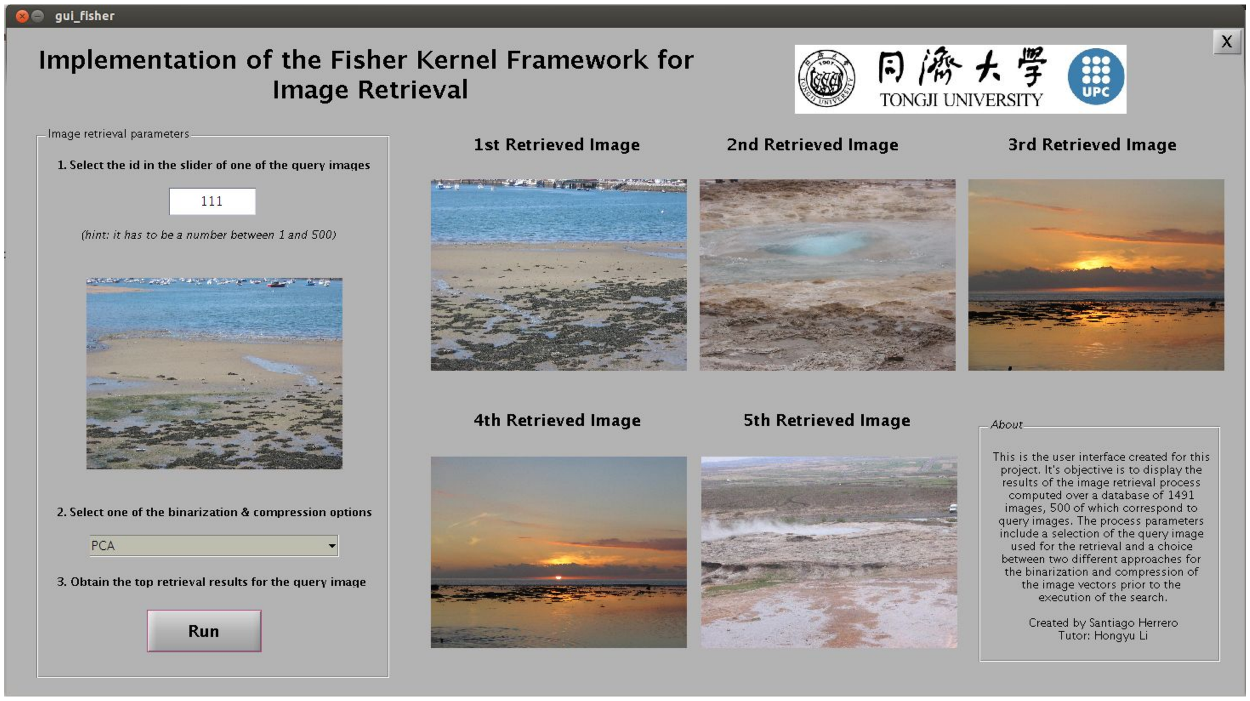
The information obtained is better presented in the following table:

	Holidays database (1491 images)	
	SEARCH SPEED	mAP score
No PCA projection	1.02 s	0.599
PCA projection	0.67 s	0.561

The analysis of this table will be done by comparing these values after the test software. For the user interface it is of interest to comment the visual results obtained first.

The following conclusions can be drawn through analyzing the output retrieved images for our 500 queries (examples to prove so are provided):

1. The framework yields better results with images representing landscapes rather than defined objects because its sensitiveness to illumination changes.



Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

111

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

PCA

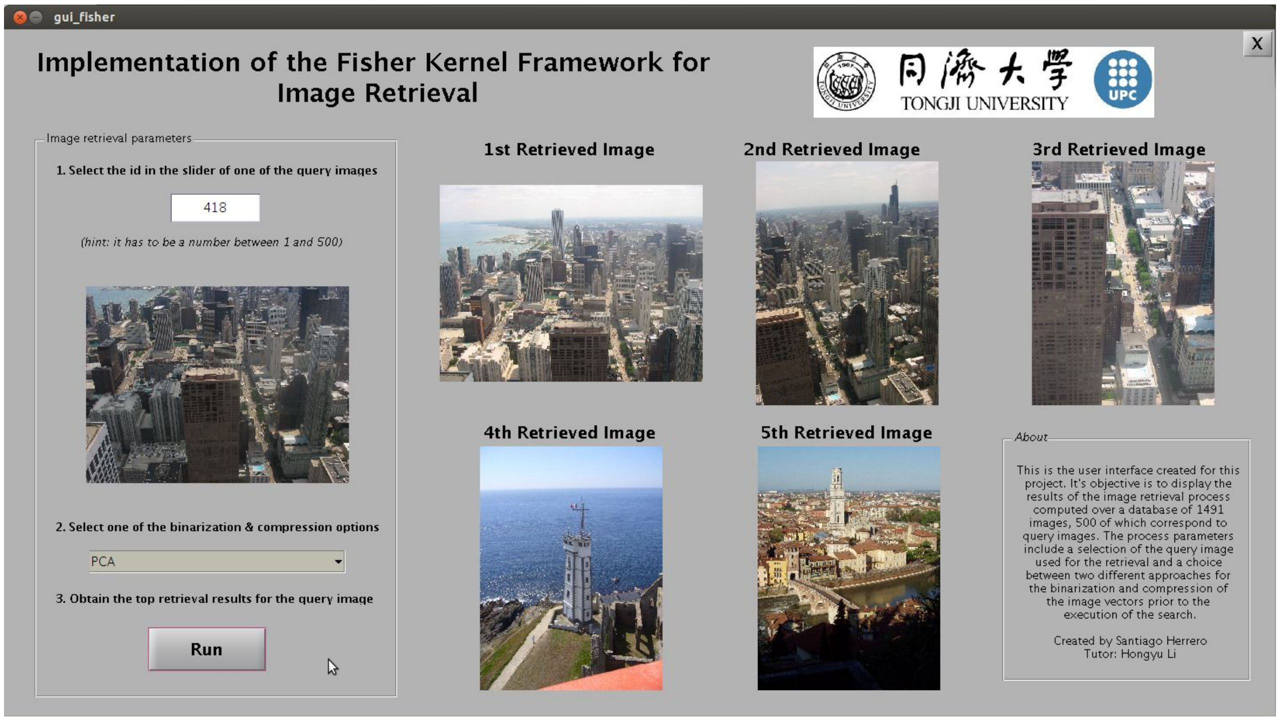
3. Obtain the top retrieval results for the query image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

2. It also retrieves better images when the query represents an image with a repetitive pattern such as the texture of a sea or sand, or a city skyline.



Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

418

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

PCA

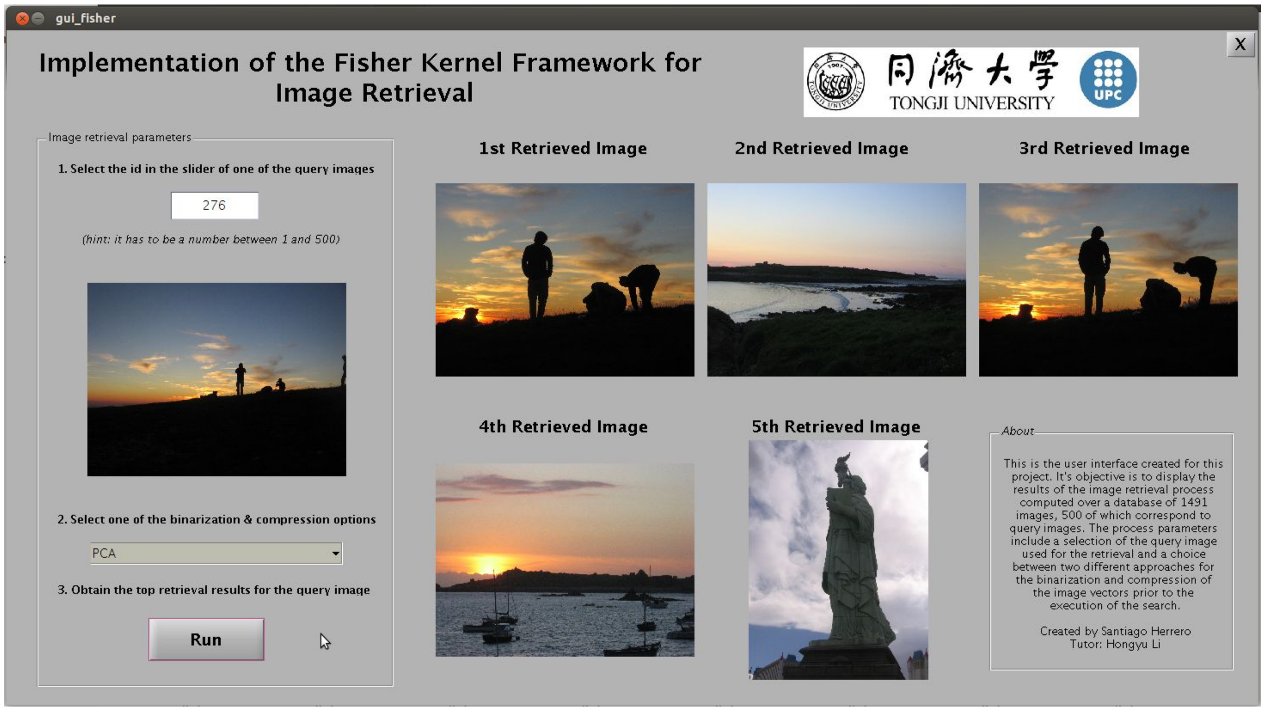
3. Obtain the top retrieval results for the query image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

3. The framework seems to prioritize reproducing similar **sharp** shapes or color contrasts. Therefore, it is sensitive to illumination changes in similar images and obtains bad results for almost identical images with severe illumination changes. Once the shape of an object in an image isn't defined very clearly, the prototype yields suboptimal results. The following two images show how when shapes are sharp and clear results are good (first of the images) and when illumination blurs the image results are not the expected ones (second image together with expected results).



gui_fisher

Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

276

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

PCA

3. Obtain the top retrieval results for the query image

Run

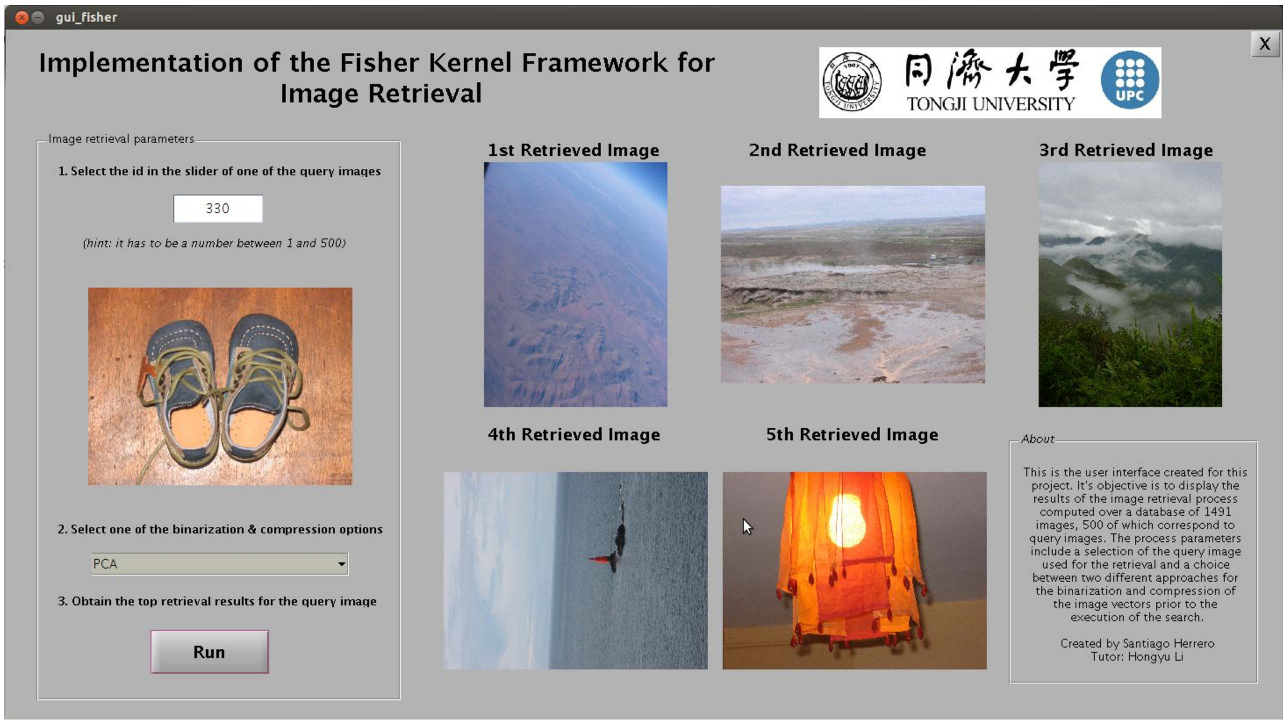
1st Retrieved Image | 2nd Retrieved Image | 3rd Retrieved Image | 4th Retrieved Image | 5th Retrieved Image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

Bad results due to illumination changes:



gui_fisher

Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

330

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

PCA

3. Obtain the top retrieval results for the query image

Run

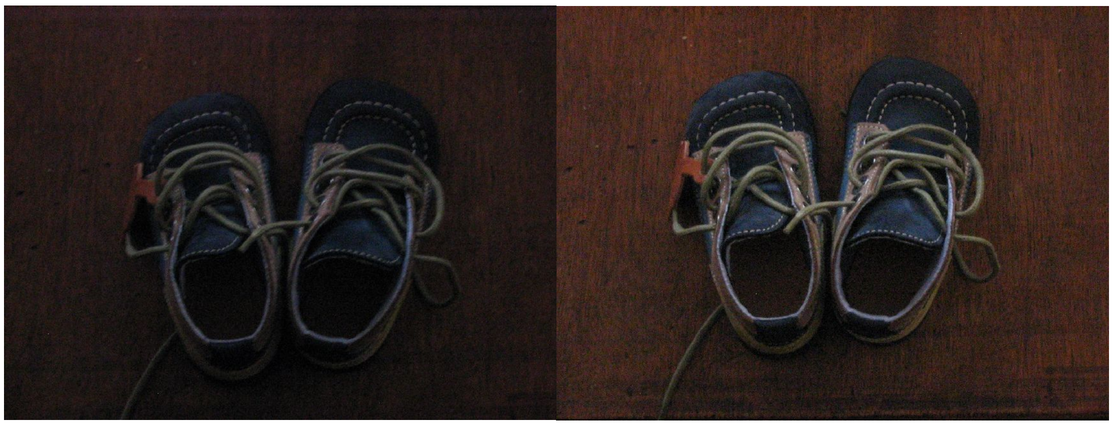
1st Retrieved Image | 2nd Retrieved Image | 3rd Retrieved Image | 4th Retrieved Image | 5th Retrieved Image

About

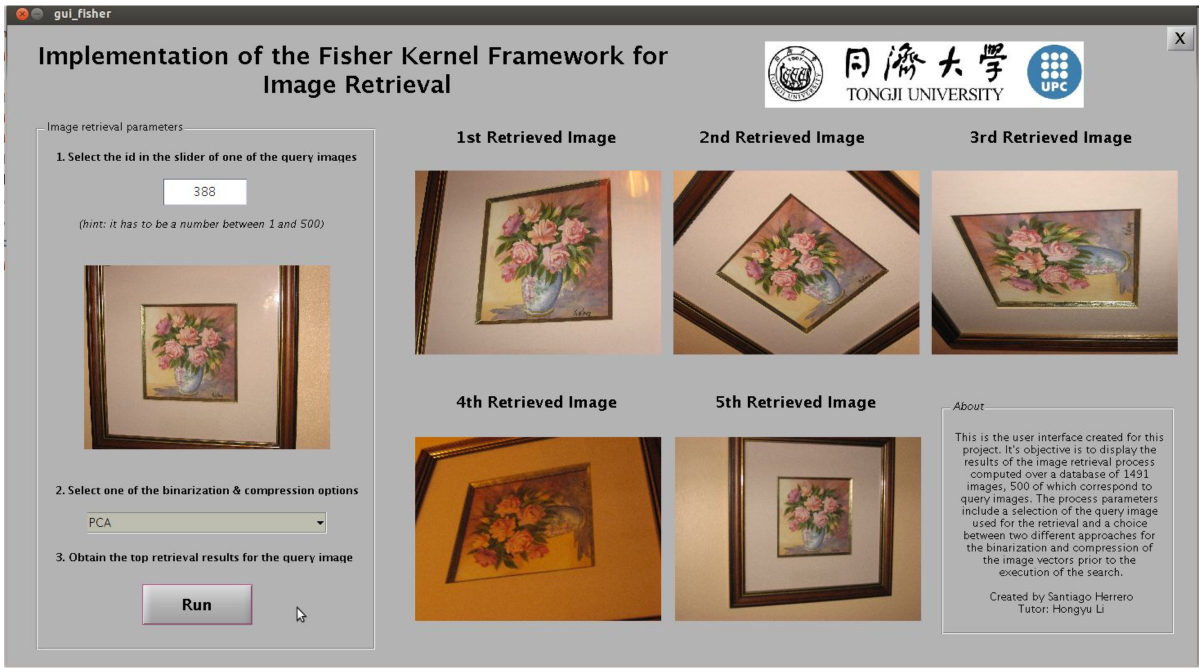
This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

Expected two retrieved images with severe illumination changes:



4. The prototype is fairly robust to rotation changes on images. In other words, it retrieves images showing the same object in different angles (advantage of SIFT descriptors).



Implementation of the Fisher Kernel Framework for Image Retrieval

Image retrieval parameters

1. Select the id in the slider of one of the query images

388

(hint: it has to be a number between 1 and 500)

2. Select one of the binarization & compression options

PCA

3. Obtain the top retrieval results for the query image

About

This is the user interface created for this project. It's objective is to display the results of the image retrieval process computed over a database of 1491 images, 500 of which correspond to query images. The process parameters include a selection of the query image used for the retrieval and a choice between two different approaches for the binarization and compression of the image vectors prior to the execution of the search.

Created by Santiago Herrero
Tutor: Hongyu Li

The reason for some of these errors can be because even after the normalization process,

G_{λ}^{X} still doesn't completely discount image-specific information contained in the image.

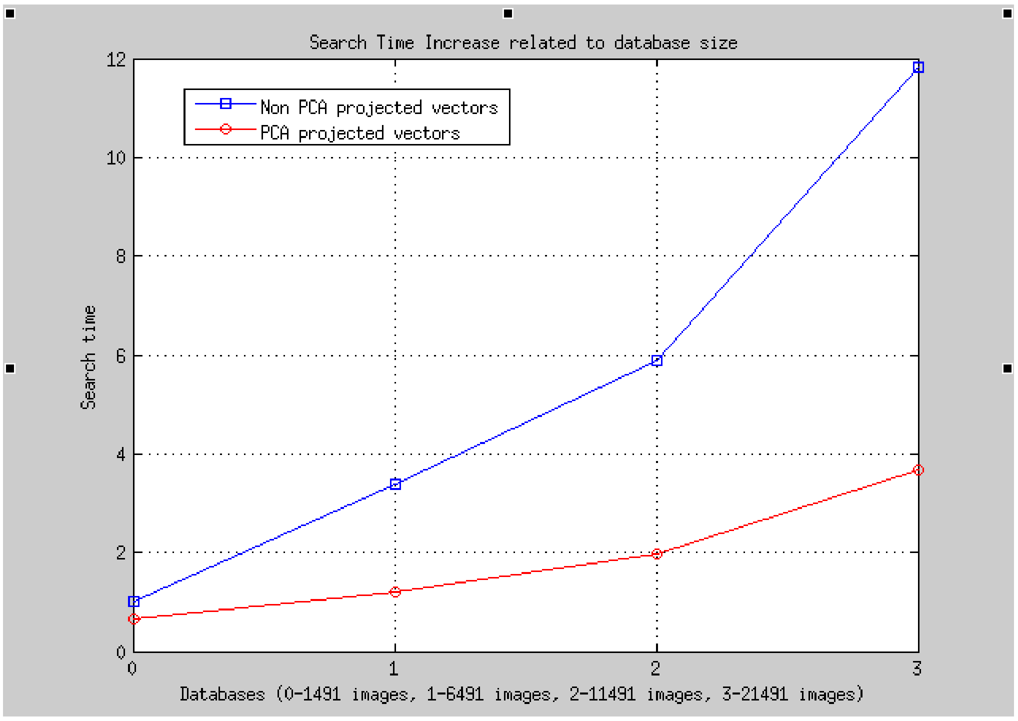
When **we run the test software**, after having executed the *test_fisher.m* file where all the databases are properly created and their Fisher vectors computed, we obtain the following:

- Elapsed time for both the projected and non projected Fisher vector implementation on each of the databases (3 databases of >5000, >10000, >20000 and >50000 images). So, a total of 8 time measures.
- mAP score for each of the two processes applied to the 4 databases.
- The ROC curve for the each of the two processes applied to the 4 databases.

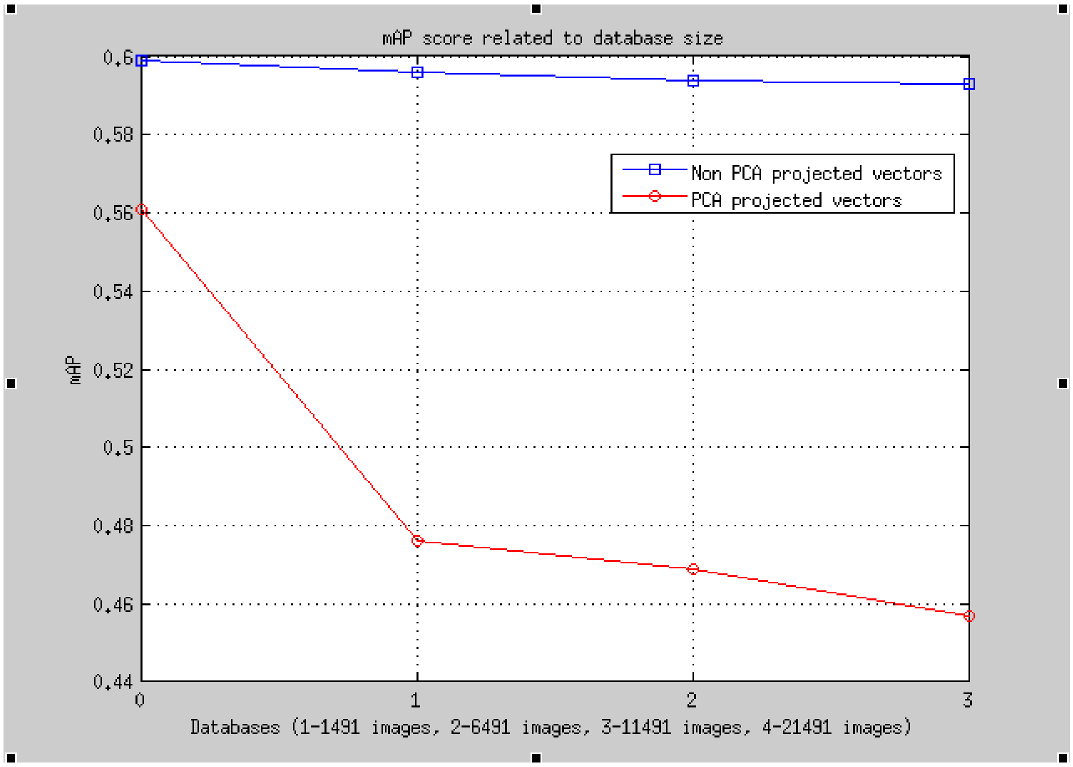
```
>> test(v, v5k, v10k, v20k, v50k, gnd, qidx, imlist);
Setting thresholds...
Concatenating the databases...
Setting up the parameters...
Normalizing the vectors...
Performing Fisher ranking...

Fisher_no_projection_elapsed_time =

    3.3700

Fisher k=64              4096D     mAP = 0.596
Performing Fisher with PCA projections ranking...

Fisher_projected_elapsed_time =

    1.2000

Fisher + PCA (D'=128)          128D     mAP = 0.476
Setting up the parameters...
Normalizing the vectors...
Performing Fisher ranking...

Fisher_no_projection_elapsed_time =
    5.8900

Fisher k=64               4096D     mAP = 0.594
Performing Fisher with PCA projections ranking...

Fisher_projected_elapsed_time =

    1.9700

Fisher + PCA (D'=128)          128D     mAP = 0.469
Setting up the parameters...
Normalizing the vectors...
Performing Fisher ranking...

Fisher_no_projection_elapsed_time =

   11.8300

Fisher k=64               4096D     mAP = 0.593
Performing Fisher with PCA projections ranking...

Fisher_projected_elapsed_time =

    3.6700

Fisher + PCA (D'=128)          128D     mAP = 0.457
```

The data is better presented in the following tables and graphs:

	SEARCH SPEED			
	Database size			
	1491 images	6491 images	11491 images	21491 images
No PCA projection	1.02 s	3.37 s	5.89 s	11.83 s
PCA projection	0.67 s	1.20 s	1.97 s	3.67 s



	mAP score			
	Database size			
	1491 images	6491 images	11491 images	21491 images
No PCA projection	0.599	0.596	0.594	0.593
PCA projection	0.561	0.476	0.469	0.457



As it can be seen, applying Principal Component Analysis on the Holidays database vectors (database of 1491 images) doesn't have a very negative impact on the mAP score, though it does reduce the search time by 34.31%. Nevertheless, we can conclude the following according to these figures:

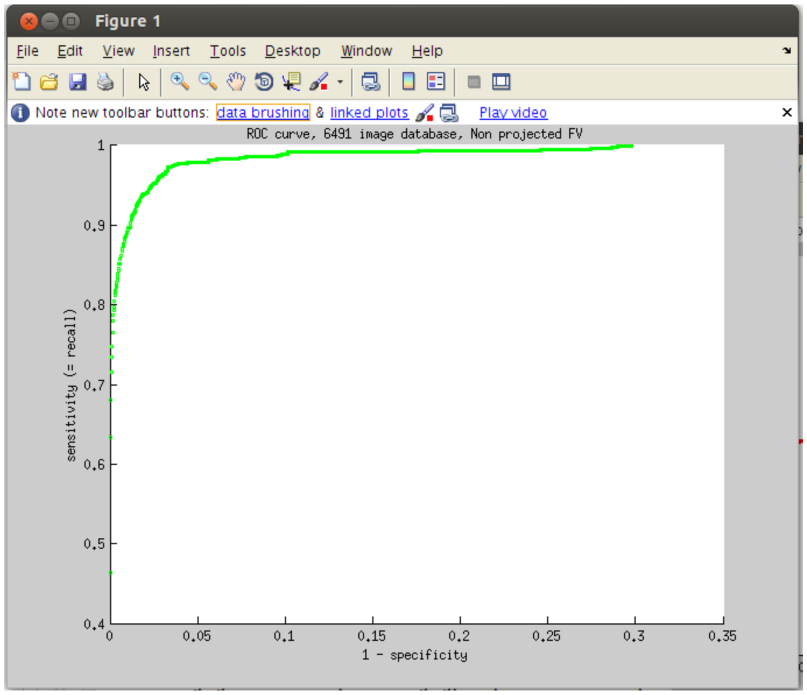
- Applying PCA makes the system more robust against the increase of the database size, since the search time only increases by 5.47 while the database size is multiplied by 14.41. Whereas if no PCA projection is applied for the same database size multiplication the search time increases by 11.59. Therefore for a database formed by over 20000 images the search time is over three times bigger without having projected the vectors than by having projected them.

- Applying PCA though has a much bigger impact on the mAP score as the database gets bigger. [1] reports great mAP score results for very-large databases but the results obtained in this prototype are not the expected ones.

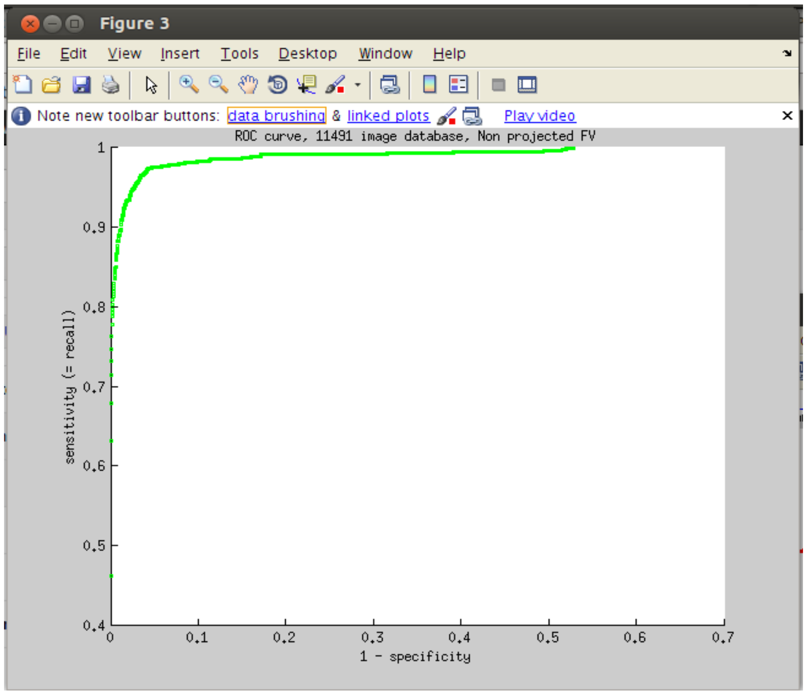
- Adding the Flickr60k descriptors as distracting descriptors doesn't have any impact on the search done with unprojected Fisher vectors although the search time is almost unacceptable when having a database of over 10000 items.

Another of the measures obtained in the test software is the ROC curve for every database size and for projected and not projected Fisher vectors. Here are the results, first for Non projected Fisher Vectors (3 graphs) followed by the projected Fisher Vectors (3 graphs):

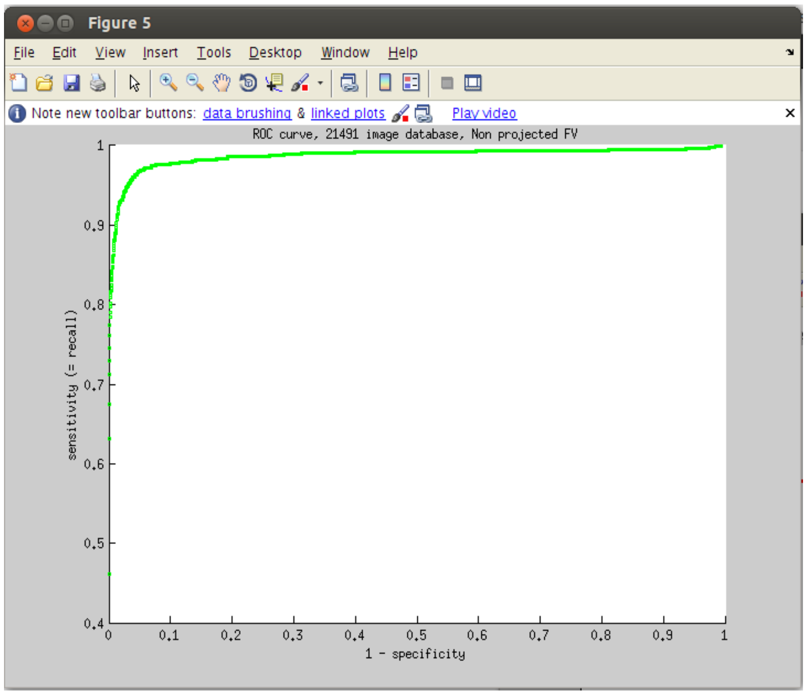
- 6491 images database:



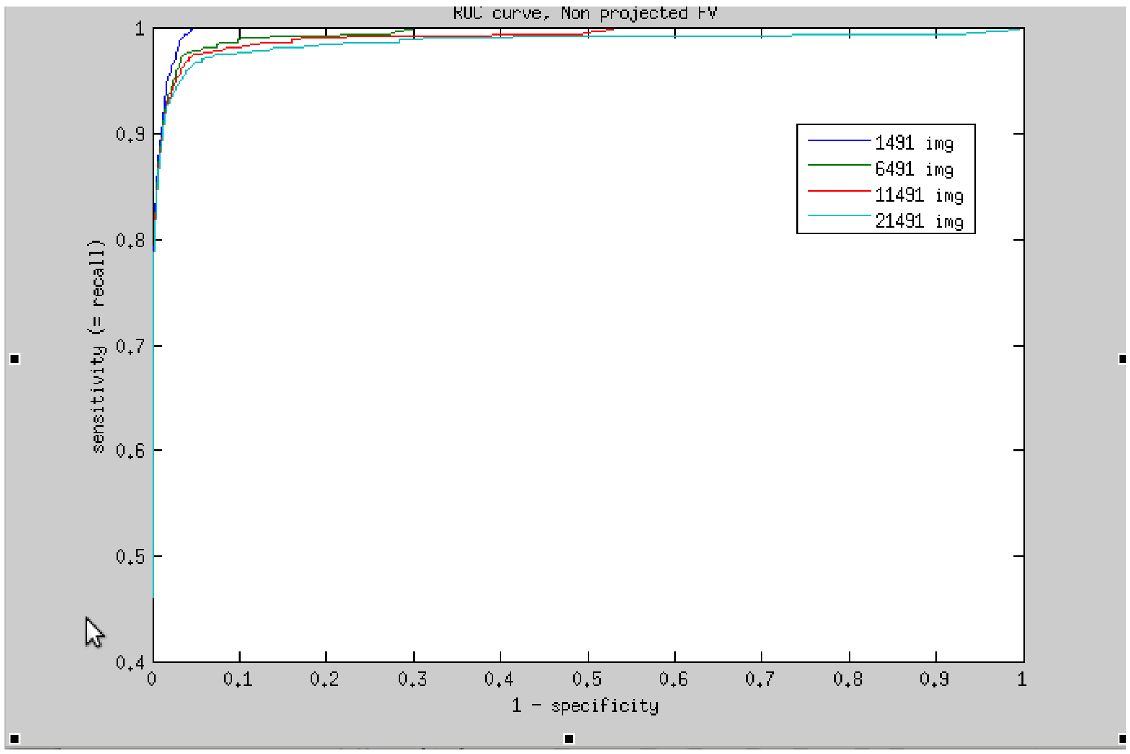
- 11491 images database:



- 21491 images database:

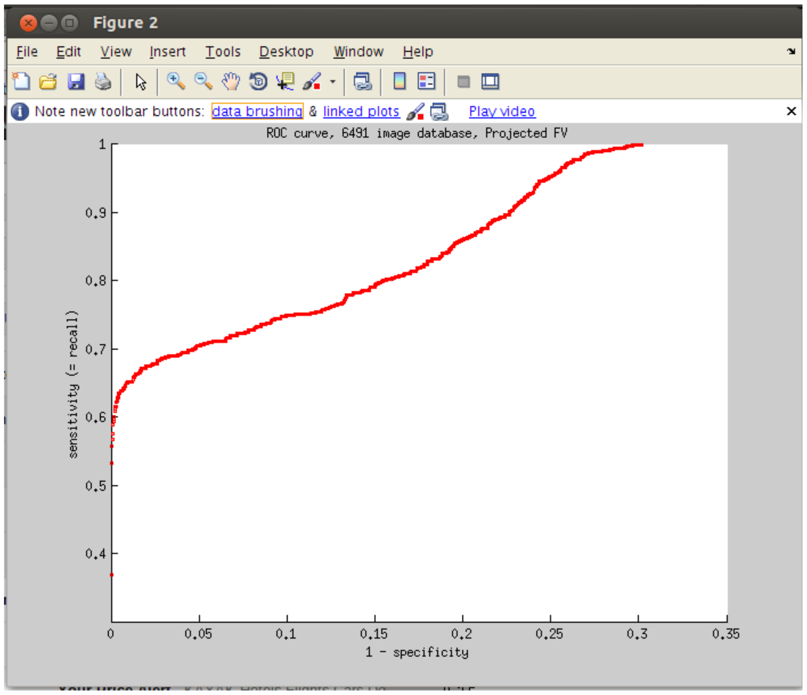


The results are coherent considering the mAP score results obtained previously, because mAP is equivalent to computing the area under the ROC curve. As it can be seen, there is near to no impact on the ROC curve performance when the database increases size. The shape of the curve shows that the prototype works well since it steeply increases in a very fast manner at the beginning of the graphic. If we overlap these three functions together with the one corresponding to the implementation that only uses the Holiday dataset the differences are shown to be minimum in search accuracy performance:

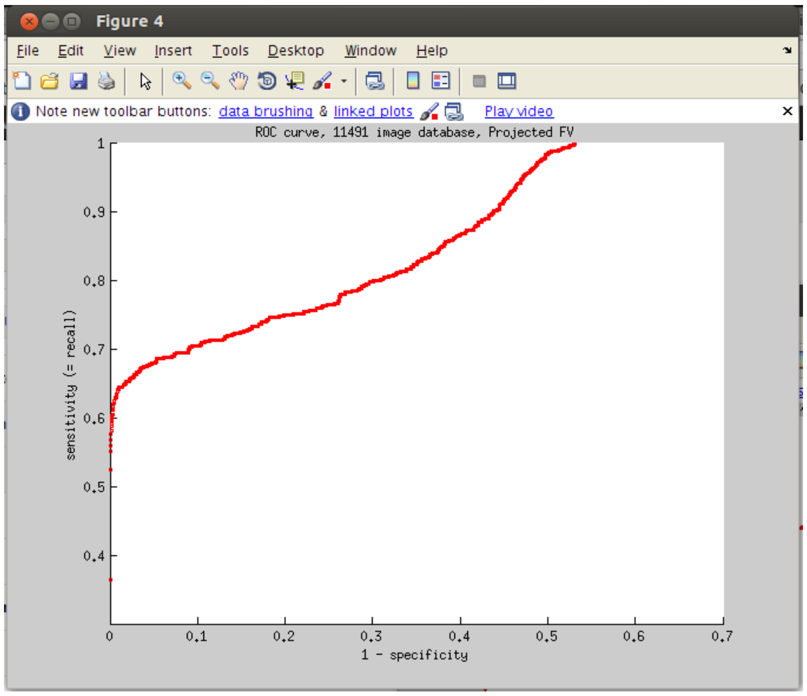


For the projected vectors the ROC curves obtain different results:

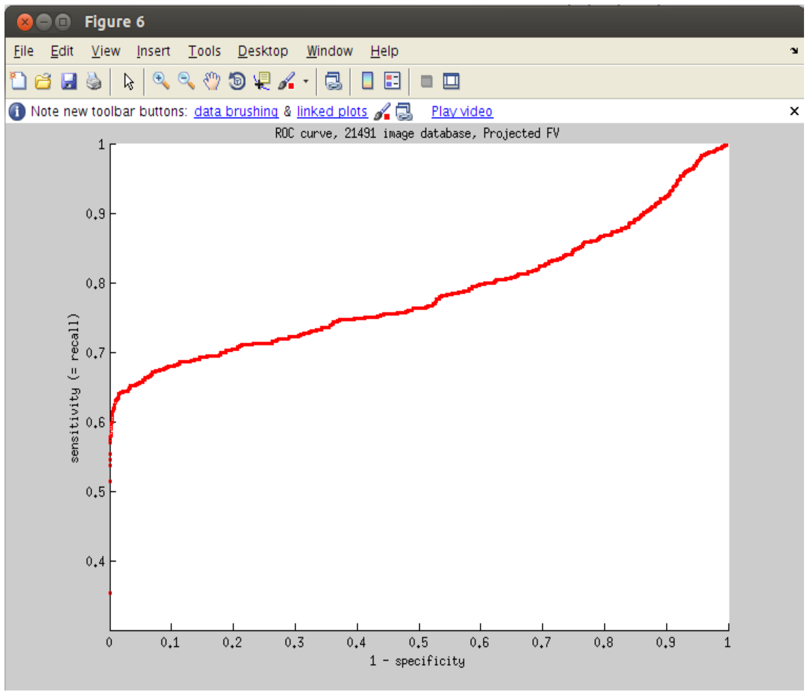
- 6491 images database:



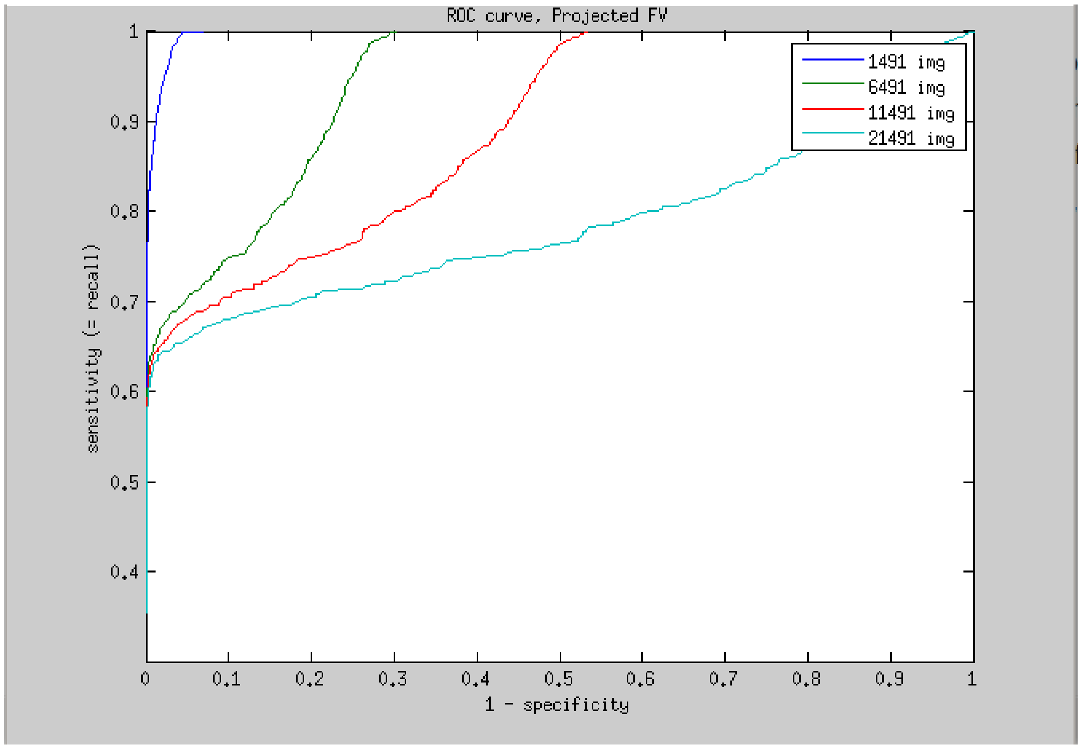
- 11491 images database:



- 21491 images database:



As it can be seen, the curve doesn't grow as steeply as it is desirable. This was previously shown by the low mAP score obtained in the tables but is more visually clear looking at these curves. In these three curves the differences are not big, but when we plot them on top of the ROC curve corresponding to the implementation that only uses the Holiday database, the differences come clear:



Here it is shown that prior to adding distractor features from the Flickr60k database the performance in terms of the ROC curve was very good, since the curve increased steeply towards the sensitivity=1 asymptote. But, once the database size increased it had a notable negative impact on the accuracy in each of the image addition processes.

3 CONCLUSIONS & FURTHER WORK

After having explained in detail the Fisher Kernel Framework and implemented it in a real image categorization problem the conclusion that is reached is that, according to the results obtained, Fisher vectors manage to create powerful signatures that characterize images but what make them different. This is shown in the user interface implementation where the results for the majority of query images are acceptable, even if they fail to properly categorize images containing specific objects. Nevertheless, in the test software it is shown that the current application accuracy is not robust when the Fisher Vectors are projected using PCA and the databases increase in size, despite that it also shows that the search time doesn't increase much when the database sizes are considerably boosted.

It has been explained how the Fisher Kernel Framework outperforms the more commonly used BOV in the problem of image categorization. The different normalization techniques, together with the application of principal component analysis as well as different techniques for indexation of the vectors, make this framework very amenable for this problem. Consequently, we conclude that Fisher vectors should become a more common technique than it is now in the foreseeable future.

From the objectives that I had in the beginning of the project I can say that I accomplished them all. Realizing this thesis was a very good introduction to the image processing field. First, by reading teaching material on the subject in general to familiarize myself with the techniques and terminology. Second, by helping on the front-end development of student Wang Yi master thesis and, therefore, having access to his programming implementation of a few image retrieval techniques. Finally, by applying myself one of these techniques to build this project.

As a final reflexion it is time to oversee on which way this project could be extended in the future. The user interface implementation only offers two of the techniques described in the theoretical approach section. Therefore, more of those could be applied starting by Spectral Hashing and also by using different kinds of descriptors such as the GIST. This way a platform to compare all these solutions could be built.

4 BIBLIOGRAPHY

[1] F. Perronin, Y. Liu, J. Sanchez and H. Poirer. Large-scale image retrieval with compressed fisher vectors. In CVPR, 2010.

[2] H. Jégou, M. Douze, C. Smith and P. Pérez. Aggregating local image descriptors into compact codes. In CVPR, 2010.

[3] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Product quantization for nearest neighbor search. In IEEE Transactions on Pattern Analysis & Machine Intelligence, 2011.

[4] Jakob Verbeek. Slides: Fisher vector image representation. 2012

[5] Florent Perronin, Jorge Sánchez, and Thomas Mensink. mproving the Fisher Kernel for large-scale Image classification. In Xerox Research Centre Europe (XRCE).

[6] F. Perronnin and C. Dance. Fisher Kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[7] Josip Krapac, Jakob Verbeek, and Frédéric Jurie. Modeling spatial layout with fisher vectors for image categorization. LEAR team, INRIA Grenoble Rhône-Alpes, France. GREYC, Université de Caen, France.

[8] Florent Perronin and Jorge Sanchez. Slides: Compressed Fisher Vectors for LSVR. XRCE@ILSVRC2011.

[9] Josip Krapac, Jakob Verbeek, and Frédéric Jurie. Spatial Fisher Vectors for Image Categorization. In *INRIA*, 2011.

[10] H. Jégou, M. Douze, and C. Schmid. Aggregating local descriptors into compact image representation. In *CVPR*, 2010.

[11] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008. http://www.cs.huiji.ac.il/~yweiss/SpectralHashing/.

[12] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[13] M. Douze, H. Jégou, H. Singh, L. Amsaleg, and C. Smith. Evaluation of GIST descriptors for web-scale image search. In *CIVR*, 2009.

[14] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.

[15] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.
http://lear.inrialpes.fr/~jegou/data.php

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. In International Journal of Computer Vision, 2004.

[17] H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *ICCV*, 2009.

[18] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. In *International Journal of Computer Vision*, 2005.

[19] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[20] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *CVPR*, 2009.

[21] Kristen Grauman and Bastian Leibe. Visual Recognition. Excerpt chapter from Synthesis lecture draft: Visual Recognition.

[22] Dr. H. B. Kekre, Sudeep D. Thepade, Akshay Maloo. Performance Comparison of Image Retrieval Using Fractional Coefficients of Transformed Image Using DCT, Walsh, Haar and Kekre's Transform.

[23] Selim Aksoy and Robert M. Haralick. Feature Normalization and Likelihood-based Similarity Measures for Image Retrieval. University of Washington, Seattle.

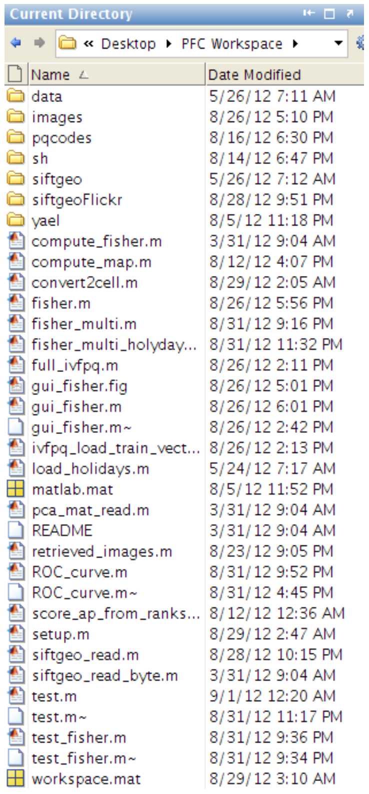
[24] Herwig Lejsek, Fridrik H. Ásmundsson, Björn Thór Jónsson, Laurent Amsaleg. Scalability of Local Image Descriptors: A Comparative Study. At IRISA-CNRS.

[25] Dr. Fuhui Long, Dr. Hongjiang Zhang and Prof. David Dagan Feng. Chapter 1: Fundamentals of content-based image retrieval.

APPENDIX: Configuring workspace

Step-by-step configuration:

1. Set up a system with Linux 64-bits environment (32-bit environment have problems executing certain Yael library functions unless some modifications of the library makefiles are made).
2. Download the Yael library from the link https://gforge.inria.fr/projects/yael/. This project was developed using version v277.
3. Install the packages required listed in the 'readme.txt' file shipped with the library.
4. Follow the instructions for setting up the environment listed in http://lear.inrialpes.fr/src/inria_fisher/.
5. Download the Flickr60k descriptors from http://lear.inrialpes.fr/people/jegou/data.php#holidays.
6. Install MATLAB R2009a by MathWorks Inc. or a similar version.
7. Organize the folder system for the project workspace as shown in the figure:



Name	Date Modified
data	5/26/12 7:11 AM
images	8/26/12 5:10 PM
pqcodes	8/16/12 6:30 PM
sh	8/14/12 6:47 PM
siftgeo	5/26/12 7:12 AM
siftgeoFlickr	8/28/12 9:51 PM
yael	8/5/12 11:18 PM
compute_fisher.m	3/31/12 9:04 AM
compute_map.m	8/12/12 4:07 PM
convert2cell.m	8/29/12 2:05 AM
fisher.m	8/26/12 5:56 PM
fisher_multi.m	8/31/12 9:16 PM
fisher_multi_holyday...	8/31/12 11:32 PM
full_ivfpq.m	8/26/12 2:11 PM
gui_fisher.fig	8/26/12 5:01 PM
gui_fisher.m	8/26/12 6:01 PM
gui_fisher.m~	8/26/12 2:42 PM
ivfpq_load_train_vect...	8/26/12 2:13 PM
load_holidays.m	5/24/12 7:17 AM
matlab.mat	8/5/12 11:52 PM
pca_mat_read.m	3/31/12 9:04 AM
README	3/31/12 9:04 AM
retrieved_images.m	8/23/12 9:05 PM
ROC_curve.m	8/31/12 9:52 PM
ROC_curve.m~	8/31/12 4:45 PM
score_ap_from_ranks...	8/12/12 12:36 AM
setup.m	8/29/12 2:47 AM
siftgeo_read.m	8/28/12 10:15 PM
siftgeo_read_byte.m	3/31/12 9:04 AM
test.m	9/1/12 12:20 AM
test.m~	8/31/12 11:17 PM
test_fisher.m	8/31/12 9:36 PM
test_fisher.m~	8/31/12 9:34 PM
workspace.mat	8/29/12 3:10 AM