INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Content-Based Image Classification

A Non-Parametric Classification

Paulo Manuel de Brito Ferreira

Dissertation for the achievement of the degree of master in

Electrical and Computer Engineering

October 2010

À memória do meu pai

Acknowledgments

The conclusion of this work represents the end of a cycle that was only made possible with the support of various people, to whom I express my gratitude.

First of all, I would like to thank Prof. Mário Figueiredo and Prof. Pedro Aguiar for having accepted to supervise my work and for their cooperation. I also thank them for having given me time to develop and define my research direction.

I have to thank Prof. José António Santos for the great patience and effort to rectify my English writings.

On top of all, I want to thank my family, especially my parents, Virgílio Ferreira and Maria Anselmo Ferreira, who did everything to give me the wholehearted support and the conditions necessary to my personal and academic growth.

I also thank my brother, Samuel M. Ferreira, for his encouragement, organization and methodology advice, that moved me. I show gratitude to my sister-in-law, Joana M. Ferreira, and my beloved niece and nephew, Maria M. Ferreira and Mateus M. Ferreira, for the joy they infected me with.

Obviously, I am grateful to the rest of my extended family for their unconditional support in difficult moments.

I thank all the lads with special highlights to my good friends Luís Oliveira, João Nuno Santos, Ivo Anastácio and Pedro Martins for their help and contribution to surpass all the difficulties associated with a master's course.

Finally, I would like to thank all the friends, teachers and institutions (IST and ESL) that throughout my life worried about my scholar success.

October 15, 2010

Abstract

The rise of the amount imagery on the Internet, as well as in multimedia systems, has motivated research work on visual information retrieval (VIR) systems and on automatic analysis of image databases.

In this work, we develop a classification system that allows to recognize and recover the class of a query image based on its content. Such systems are called Content-Based Image Retrieval (CBIR).

CBIR systems describe each image (either the query or the ones in the database) by a set of features that are automatically extracted. Then, the feature vectors are fed into a classifier.

In this thesis, the processes of image feature selection and extraction uses descriptors and techniques such as Scale Invariant Feature Transform (SIFT), Bag-of-Words (BoW) and Spatial Histograms (SP).

For the classifier, we employ the Naive Bayes Nearest Neighbor (NBNN) algorithm, which belongs to the category of non-parametric classifiers. We also present a brief description of other classifiers used in image classification.

In addition, our work herein described tests and compares the image-to-class and image-to-image distances, in order to decide which leads to better performance.

Keywords

Image classification, CBIR, feature extraction, classifiers, Naive Bayes Nearest Neighbor, image-to-class and image-to-image distances.

Resumo

O aumento da quantidade de imagens na Internet e dos sistemas de multimédia tem originado trabalhos de investigação sobre os sistemas de *visual information retrieval* (VIR) e na análise automática de base de dados de imagens.

Neste trabalho desenvolve-se um sistema de classificação que permite reconhecer e recuperar a classe de uma imagem através do seu conteúdo. Estes sistemas são designados por *Content-Based Image Retrieval* (CBIR).

Os sistemas CBIR descrevem cada imagem (consulta ou as da base de dados) através de um conjunto de características que são extraídas automaticamente. De seguida, os vectores de características são introduzidos num classificador.

Nesta tese, os processos de selecção e extracção das características das imagem utilizam descritores e técnicas tais como *Scale Invariant Feature Transform* (SIFT), *Bag-of-Words* (BoW) e *Spatial Histograms* (SP).

Em relação ao classificador, emprega-se o algoritmo Naive Bayes Nearest Neighbor (NBNN), que pertence à categoria dos classificadores não paramétricos. Apresenta-se também uma breve descrição de outros classificadores utilizados na classificação..

Além disso neste trabalho testam-se e comparam-se as distâncias imagem-a-classe e imagem-a-imagem do classificador NBNN, de modo a decidir qual delas lidera a uma melhor performance.

Palavras-Chave

Classificação de imagem, CBIR, extracção de características, classificadores, *Naive Bayes Nearest Neighbor*, distâncias imagem-a-classe e imagem-a-imagem.

Contents

List of Figures viii

List of Tables ix

List of Acronyms xi

Chapter 1

1. Introduction 1

1.1 Context 2

1.2 Motivation 2

1.3 Objectives and Contributions 6

1.4 Thesis Organization 10

Chapter 2

2. Image Retrieval 11

2.1 Evolution 12

2.2 Concept 12

2.3 Levels 14

2.4 CBIR Systems 15

2.5 Problems 16

2.5.1 Gaps 16

2.5.2 Object Retrieval Problems 17

Chapter 3

3. Feature Extraction 21

3.1 Image Representation 22

3.1.1 Global Representation 23

3.1.2 Local Representation 23

3.2 Features Descriptors 28

3.2.1 Scale Invariant Feature Transform 28

3.3 Bags-of-Word Model 30

3.3.1 Pyramid Representation 32

Chapter 4

4. Datasets 35

4.1 Object Classification Datasets 36

4.1.1 Caltech-101 36

4.1.2 Caltech-256 36

Chapter 5

5. Classifiers 39

5.1 Architecture 40

5.2 Unsupervised Learning 41

5.3 Supervised Learning 42

5.3.1 Naive Bayes Classifier 42

5.3.2 *k*-Nearest Neighbor 44

5.3.3 Decision Trees 45

5.3.4 Artificial Neural Network 48

5.3.5 Support Vector Machines 49

5.3.6 SVM-KNN 52

5.3.7 Naive Bayes Nearest Neighbor 52

Chapter 6

6. Implementation and Results 53

6.1 NBNN Distances 54

6.1.1 Distance Image-to-Class 54

6.1.2 Distance Image-to-Image 54

6.2 Implementation 55

6.2.1 Implementation Image-to-Class 55

6.2.2 Implementation Image-to-Image 57

6.3 Results 58

Chapter 7

7. Conclusions and Future Work 63

7.1 Conclusions 64

7.2 Future Work 65

References 67

Annex A 73

Annex B 75

Annex C 77

Annex D 81

Annex E 83

Annex F 87

List of Figures

Figure 1.1: CBIR in medicine application 3
Figure 1.2: CBIR in economy application 3
Figure 1.3: CBIR in crime application 4
Figure 1.4: CBIR in sport application 5
Figure 1.5: Images containing different object categories 7
Figure 1.6: Image classification system scheme 8
Figure 1.7: Results of an application in CBIR 9
Figure 2.1: CBIR concept 13
Figure 2.2: Sensory gap and semantic gap [13] 16
Figure 2.3: Illumination variation 18
Figure 2.4: Scale and size variation 18
Figure 2.5: Background Clutter 18
Figure 2.6: Viewpoint and pose variation 18
Figure 2.7: Occlusion, Truncation and Articulation 19
Figure 2.8: Intra-Class Variability 19
Figure 2.9: Inter-Class Variability 19
Figure 3.1: Example of global and local representation 22
Figure 3.2: Example of a Harris corner detector [21] 24
Figure 3.3: Example of Harris-Affine [22] 25
Figure 3.4: Example of Difference-of-Gaussian [3] 26
Figure 3.5: Example of MSER 26
Figure 3.6: Dense grid on an image 27
Figure 3.7: Interest points 28
Figure 3.8: SIFT descriptor computation 29
Figure 3.9: SIFT descriptor process 29
Figure 3.10: Vocabulary construction 31
Figure 3.11: BoW representation 31
Figure 3.12: Spatial pyramid representation [44] 32
Figure 3.13: Example of constructing a pyramid for $L = 2$ [44] 33
Figure 4.1: Caltech-101 categories 37

Figure 4.2: Caltech-256 categories 38
Figure 5.1: Design of classification system 40
Figure 5.2: *k*-Nearest Neighbor classification 44
Figure 5.3: Decision tree [13] 46
Figure 5.4: Space division [74] 47
Figure 5.5: Tree Construction [74] 47
Figure 5.6: MLP Structure 48
Figure 5.7: The basics of classification by SVM [13] 50
Figure 6.1: Class representation using image-to-class distance 56
Figure 6.2: CBIR system implementation using image-to-class distance 57
Figure 6.3: Class representation using image-to-image distance 57
Figure 6.4: CBIR system implementation using image-to-image distance 58
Figure 6.5: Number of classes for different levels of accuracy for image-to-class distance 59
Figure 6.6: Number of classes for different levels of accuracy for image-to-image distance 59
Figure B.1: k-means scheme [39] 75
Figure B.2: Simple procedure of k-means with Voronoi diagram 76
Figure C.1: Simple Simple Naive Bayes classification [49] 77
Figure D.1: A sample *kd*-tree [67] 82
Figure E.1: CBIR graphical interface 83
Figure E.2: Menu to introduce the image 84
Figure E.3: CBIR prepared to classify the query image 84
Figure E.4: Images of the similar class of query sample 85

List of Tables

Table 2.1: Developed CBIR systems 15

Table 6.1: Classification results by each class using image-to-class distance 60

Table 6.2: Classification results by each class using image-to-image distance 61

Table 6.3: Accuracy of CBIR using the image-to-class and image-to-image distances 62

Table F.1: Example of a result processing 87

List of Acronyms

ANN	Artificial Neural Network
BoW	Bag-of-Words
CBIR	Content-Based Image Retrieval
CV	Computer Vision
DoG	Difference-of-Gaussians
DT	Decision Trees
GLOH	Gradient Location and Orientation Histogram
HSV	Hue, Saturation, Value
HLS	Hue, Lightness, Saturation
IR	Image Retrieval
KNN	k-Nearest Neighbor
MLP	Multilayer *Perceptron*
MSER	Maximally Stable Extrema Regions
NB	Naive Bayes
NN	Nearest Neighbor
NBNN	Naïve Bayes Nearest Neighbor
PCA-SIFT	Principal Component Analysis SIFT
QBE	Query by Example
RBF	Radial Basis Function
RF	Random Forest
RGB	Red, Green, Blue
SIFT	Scale Invariant Feauture Transform
SL	Supervised Learning
SVM	Support Vector Machines
SURF	Speeded Up Robust Features
UL	Unsupervised Learning
VIR	Visual Information Retrieval

Chapter 1

1. Introduction

The number of digital images has grown astronomically, a consequence of the intense use of digital cameras, multimedia services and due to the storefront that the Internet turned into. Besides, in many areas, the use of image analysis has increased. Faced with this situation, the ability to classify images into semantic categories and objects (e.g. mountains, animals, humans, airplanes) is essential in order to manage and organize the collection of images on a database.

This chapter describes the thesis motivations and objectives. Moreover, it gives a brief overview of the structure and the approaches in image classification. The organization of the thesis is presented at the end of this chapter.

1.1 Context

Most image search engines are supported on metadata (e.g. file name, author, file data and file size). Naturally, these systems fail to provide meaningfull results in terms of what is usually pretended from an image query. Besides, manually labeling large databases of images is hardly feasible or very expensive.

In opposition, Content-based image retrieval (CBIR) [1] systems filter images based on their semantic content (e.g. objects, categories, relationships, and meanings), providing better indexing and giving more accurate results.

1.2 Motivation

Very often, when you are searching for an image on the world's largest database, the Internet, through search engines like Google and Yahoo!, you cannot find a type of the expected image. Whenever people want to see a specific image of a certain trip, party, event or a family member it is necessary for them to look at almost all the folders on their PC to find what they seek. Finding an image on a database is an often complicated task, due to the excessive amount of irrelevant records, caused by the huge volume of images available. Beside this, image classification is related directly and indirectly to various areas and to personal, social and professional applications.

Medicine

In medicine, doctors when examining x-rays, magnetic resonance imaging or ultrasounds scans need tools to provide them easy access to other similar cases - Figure 1.1. New applications can be created to overcome barriers in the field of ophthalmology, to help blind people in the visualization of objects and obstacles.

Journalism, Television and Advertising

In the publishing industry (e.g. newspapers, magazines, books), images are indispensable to illustrate news stories and articles. These pictures are stored in large archives that grow every days. In marketing and television it is necessary to look for old advertisements or old news stories in video archives.

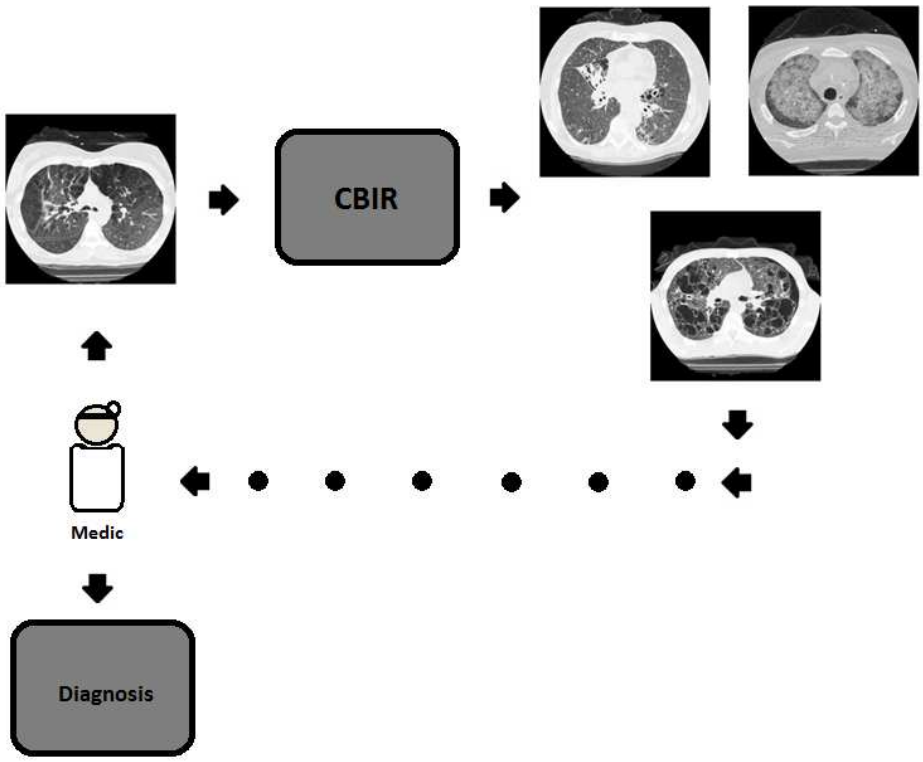


Figure 1.1: CBIR in medicine application.

Economy

In economy, analysts need quick access to stock charts over a specified period, or find charts with similar situations that have occurred - Figure 1.2.

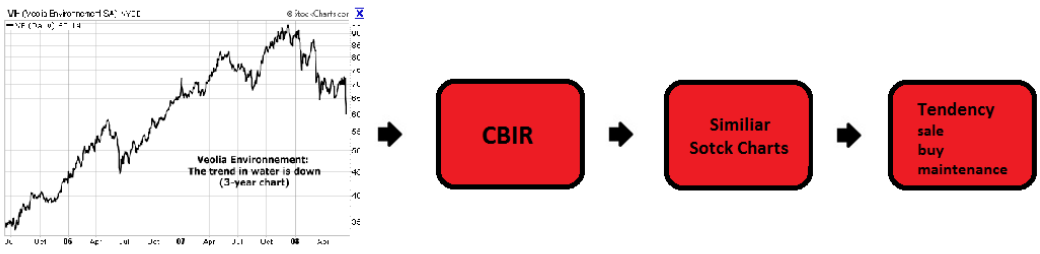


Figure 1.2: CBIR in economy application.

Industry

On the industrial side, a machine or a robot needs to detect certain objects or small variations of parameters on a production line.

Security

In the surveillance and security area, advanced systems try to automatically detect suspect people and unusual events (e.g. fights and assaults) in public spaces or events (e.g. shopping centers, stadiums or casinos).

Crime

In crime prevention, visual image retrieval is used for face recognition and automatic fingerprint matching to identify a criminal person - Figure 1.3.

Army

At the military level, image classification can be used in the analysis of satellite images or aerial mapping.

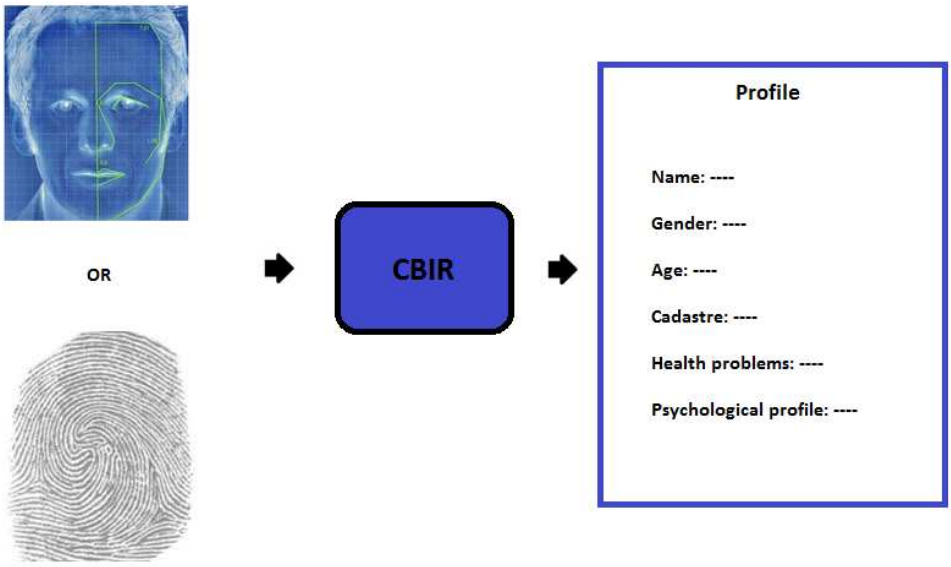


Figure 1.3: CBIR in crime application.

Sports

In professional sports (e.g. NBA or Football World Cup), the statistics, movements and tactics of a team or a game could be analyzed using techniques of image classification by its visual content – Figure 1.4.

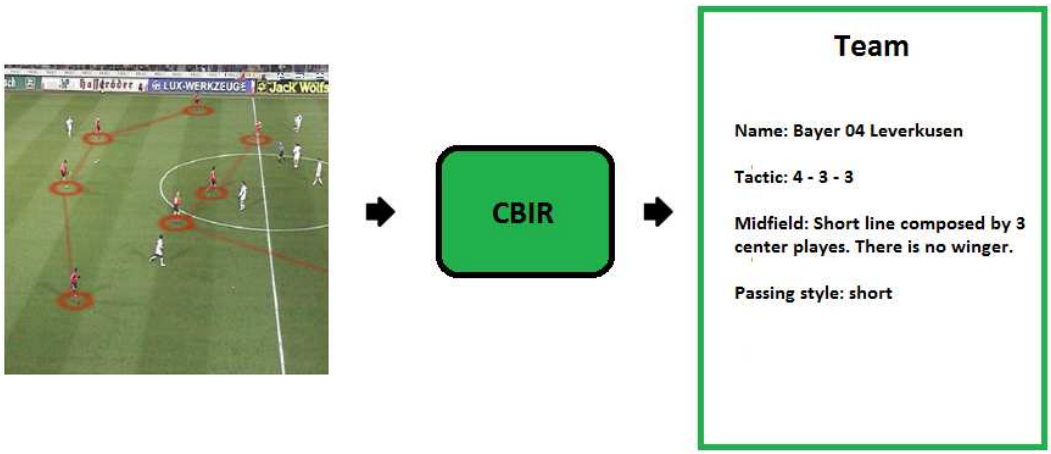


Figure 1.4: CBIR in sport application.

As we have seen, the potential of these CBIR systems is huge in many areas. Instead of using metadata information, it would be interesting to use techniques to automatically access the files by its semantic visual contents. This can be very helpful in organizing and accessing images. However there is not a fully integrated and efficient solution for application in the various areas. This is why image classification has become a great problem and a huge challenge.

1.3 Objectives and Contributions

The objective of this thesis is implementing a CBIR that classifies images by their object categories, through efficient approaches in image classification.

Given a data set of images, like the examples shown in Figure 1.5, the goal is to classify them according to their object category (e.g. leopards, airplanes, sunflowers, faces, pizza). For this purpose, it is necessary to have image databases, which have become popular in computer vision, such as Caltech-101 [2].

Leopards

Airplanes

Sunflowers

Faces

Pizzas

Figure 1.5: Images containing different object categories.

The scheme of image classification system consists of three modules, as Figure 1.6 demonstrates.

In the first module, the features of a group of images from the database and the features of the query image are extracted. This stage uses a set of descriptors to take out the features into vectors. Thus, two groups of feature vectors are created in the database and in the query. In this step, the system developed will test the classification system image with only a singular local descriptor (e.g. SIFT [3]).

The second phase of the system has the purpose of comparing the query image features with the set of features of the images on database. This module applies classifiers and algorithms for image classification, such as Support Vector Machine (SVM) [4] and *k*-Nearest Neighbor (KNN) [5], Decision Trees (DT) [6], Artificial Neural Network (ANN) [7], Naive Bayes (NB) [8], and so on. Note that these algorithms work in the space of the local image descriptors and not in the space of images.

The methods of classification can be divided into two families: parametric classifiers (learning-based classifiers) and non-parametric classifiers.

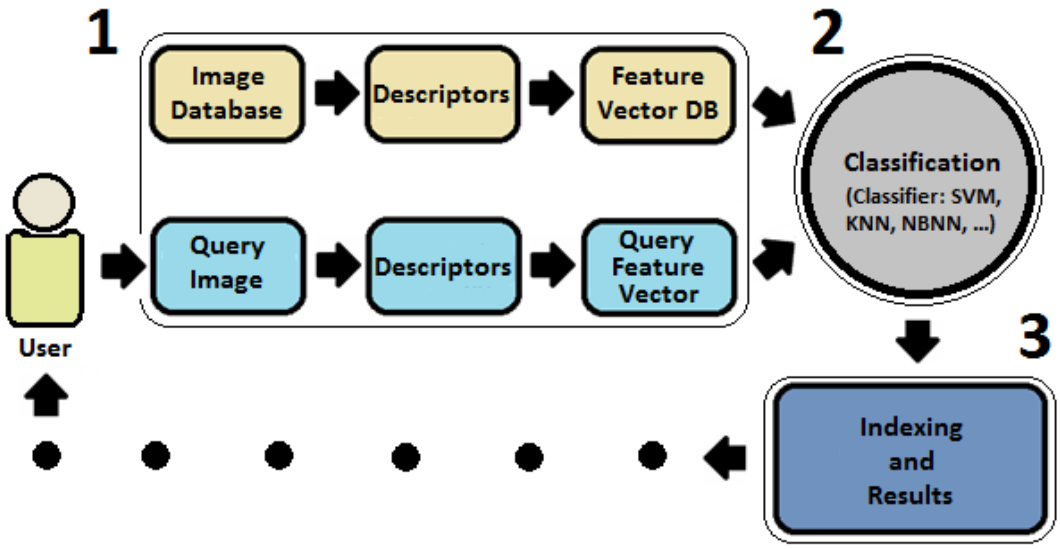


Figure 1.6: Image classification system scheme.

Parametric methods require an intensive learning/training phase of the classifier (e.g. SVM, DT, ANN). The most usual image classifiers are supported on learning, especially SVM-based methods.

On the other hand, more common non-parametric methods are based on Nearest Neighbor (NN) distance estimation and classify an image by the class of its similar image on the database. Non-parametric classifiers use measures (e.g. Euclidean distance) to compute the similarity between the query image and an image on database – distance image-to-image.

This work will focus in the recent approaches supported in non-parametric classifiers, that obtained interesting results, such as Naive Bayes Nearest Neighbor (NBNN) [9]. The idea is to compute direct image-to-class distances without descriptor quantization, under the Naive-Bayes assumption. The advantages of NN-based classifiers are simplicity, efficiency and not requiring a learning phase. This thesis will test the NBNN algorithm using the image-to-image and image-to-class distances.

Finally, the third stage of the system is to index and display the results to the user, through the graphical interface. Figure 1.7 shows the results of an application in CBIR.

The CBIR platform is developed on MATLAB and MATLAB GUI, which provide a graphical interface.

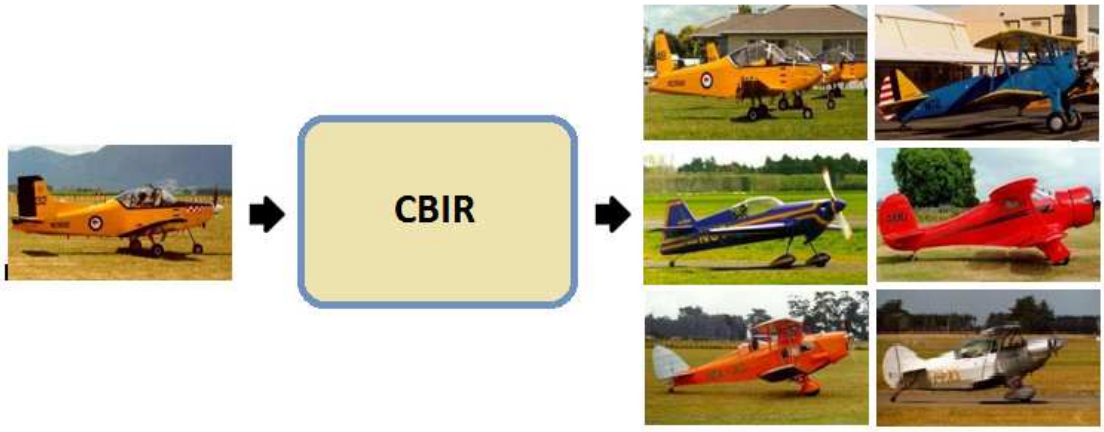


Figure 1.7: Results of an application in CBIR.

Consequently, the contributions of this thesis are:

- Develop a CBIR system using SIFT combined with NBNN.

  We use the SIFT descriptor in dense way over a regular grid to select and extract the feature vectors of images. The distances between feature vectors of database and query image are computed using the classifier NBNN and the approximation of the nearest neighbor through the *kd*-tree algorithm. The *kd*-tree technique improves the computational performance.

- Comparing the image-to-image and image-to-class distances in CBIR with NBNN algorithm.

- Implementing a graphic interface in MATLAB for this CBIR system which classifies images into categories;

1.4 Thesis Organization

The organization of the remainder of this thesis is as follows:

- Chapter 2 focuses on the basic concepts, the problems and the challenges behind image retrieval;
- Chapter 3 presents image representation models and one of most well-known descriptor (SIFT) for extraction image features. Moreover, this chapter gives a description of quantization techniques used in image representation such as Bag-of-Words (BoW);
- Chapter 4 describes the Caltech-101 dataset used in this thesis and presents Caltech-256;
- Chapter 5 details the classification image classifiers, namely the NB, KNN, DT, ANN, SVM, NBNN and SVM-KNN;
- Chapter 6 specifies the CBIR implementation and analyses the results of the image-to class distance vs. image-to-image distance in NBNN classifier;
- Finally, Chapter 7 describes the conclusions of this thesis, and discusses future work.

Moreover, this theses exhibit some appendices:

- Annex A present the low-level features of image such as color, shape and texture;
- Annex B introduce the *k*-means technique used in BoW;
- Annex C exemplify the Naive Bayes classification;
- Annex D describes the *kd*-tree algortihm;
- Annex E focuses the CBIR interace;
- Annex F shows a example of MATLAB processing.

Chapter 2

2. Image Retrieval

This chapter begins with a brief history of evolution of image retrieval (IR). Then, the concept of general image retrieval is presented. Current problems and challenges of image retrieval are introduced at the end of this section.

2.1 Evolution

The idea of image retrieval was created by Database Management community in a conference organized at Florence in 1979 [10]. The early schemes consisted of annotating the images by text to consequently use database management systems. However, these kinds of approaches have problems in generating descriptive texts for large collections of images. Automatic generation is not yet feasible, thus a lot of labor is required to manually annotate the images, which is an expensive task. Moreover, manually image annotations (e.g. metadata) are affected by subjectivity of human perception and different people might perceive images in different ways.

Faced with these limitations, the computer vision (CV) community introduces content-based image retrieval (CBIR) in 1992 [11]. CBIR is based on visual features of images and put textual representations in second plan. After that and until today, a lot of new techniques were developed around the concept of CBIR.

2.2 Concept

The main objective of CBIR interface is finding an image or a set of images similar to those that the user wants to query. In Chapter 1, figure 1.2 showed the processes of modules in CBIR system. Figure 2.1 represents the CBIR concept, which consists of the following philosophy and components [12, 13]:

- Image Database;

  Offline process. The features of each image on database are extracted, through the visual content descriptors, for a multidimensional vector. Then, the feature vectors are stored on a new database.

- Information Need;

  This stage processes the image query. The system uses the query for example (QBE) paradigm. This means that the user employs one or more sample images as the starting point for a search of visual information. The descriptors and vector format are the same as used in Image Database.

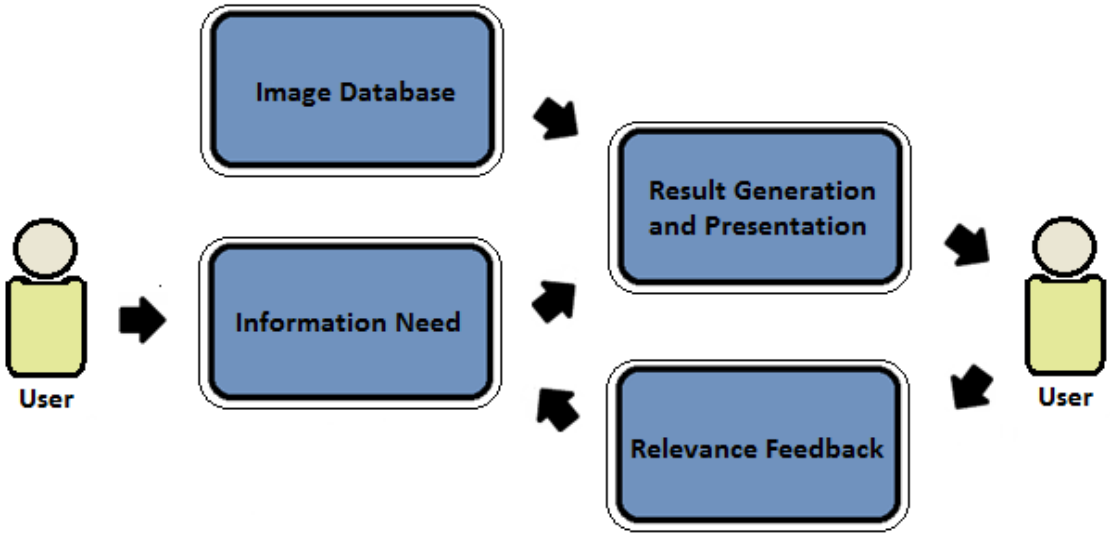


Figure 2.1: CBIR concept.

- Result Generation and Presentation;

This comparison module computes the similarities between the features vectors of database and the query feature vector. After that, this component ranks the images of database. The results of the classification are, then, displayed to the user

- Relevance Feedback.

The user can affine the results by providing interactive information. In other words, the user indicates if a resulting image is a positive example (relevant) or a negative example (irrelevant). Then, the process of information needed is refined and started over again. Relevance Feedback is an optional tool in a CBIR system.

2.3 Levels

In an image, objects or scenes contain elements which could be used in retrieval systems, including [14]:

- The presence of a particular combination of color, texture or shape information (e.g. red tennis racquet);
- The arrangement of specific types of objects (e.g. racquet hitting a ball);
- The presence of named individuals, locations, or events (e.g. Roger Federer at Wimbledon);
- Subjective emotions one might associate with the image (e.g. Roger Federer happiness with a trophy).

These image query types represent a stage of abstraction, and a CBIR system is classified into three levels of increasing complexity [14, 15]:

- **Level 1;**

  Images are compared by low-level features based on color, texture, shape features and spatial location of image elements.

- **Level 2;**

  Intermediate level brings semantic meanings into the search, identifying image classes (e.g. a horse, tree, human face) or a person. This level can involves retrieval techniques of level 1, and is an active research area.

- **Level 3.**

  Image high-level cover retrieval with abstract and subjective attributes. This level includes search requests for types of events (e.g. particular birthday celebration) and for pictures with symbolic significance (e.g. happy beautiful woman). Stage 3 requires techniques of intermediate level and a very complex logic.

2.4 CBIR Systems

This section present several CBIR systems [16] developed over the years. Some of these systems are commercial and most are academic. Table 2.1 shows the characteristics of these systems.

Table 2.1: Developed CBIR systems.

CBIR Systems	Characteristic	Category
VisualSEEk	Image comparison by matching salient color regions for their colors, sizes, absolute and relative spatial locations	Academic
Photobook	iIcludes retrieval mechanisms for two-dimensional shapes, texture images and face recognition	Academic
Multimedia Analysis and Retrieval Systems (MARS)	Incorporate relevance feedback from the user for subsequent result refinements	Academic
NeTra	Uses image segmentation. First, an image is divided into regions of homogenous color, and then color, texture, shape and spatial location are extracted from those regions.	Academic
QBIC	Depending on a number of features that can be selected by the user - extracts color features for individual objects or the entire image in several color spaces. It also includes texture and shape features.	Commercial
Virage	Apart from simple features such as global and local color, texture and shapes, various domain specific primitives (i.e. feature types, computation, indexing and corresponding similarity measures) can be defined when developing an application.	Commercial

2.5 Problems

There are certain constraints in image retrieval process. The next subsections present the problems and challenges in this research area.

2.5.1 Gaps

An image before entering a retrieval process has to be captured by a device (e.g. camera). When an image is produced, some information that was present in the real world is automatically lost. This information loss may be due to several factors, such as low resolution, bad illumination, viewing angles or a deficiency in the camera. This difference between real scene and image information is known as sensory gap (Figure 2.2) [13, 17].

In retrieval procedure the major lack is the difference between the visual features of an image and the objects, meanings or feelings that this image as perceived by a human. This handicap is named semantic gap and it is defined as the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation [13].

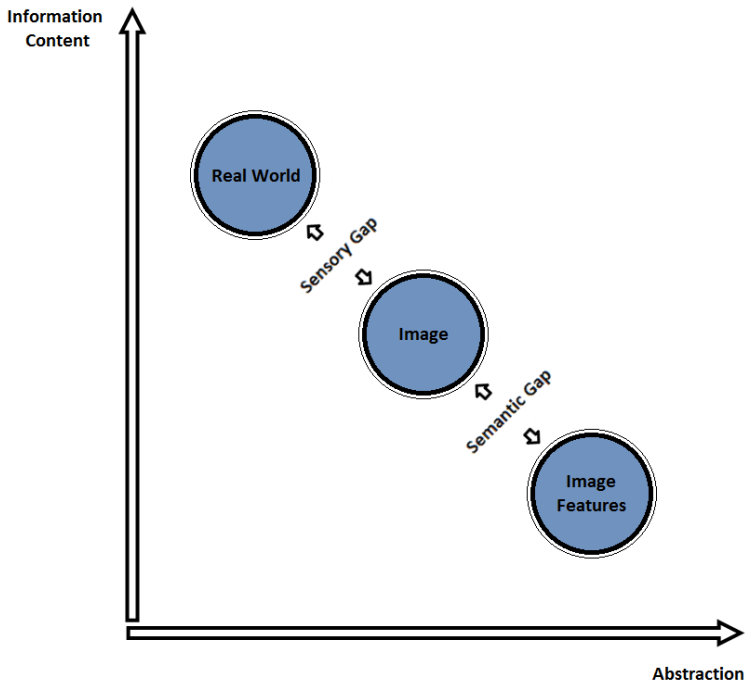


Figure 2.2: Sensory gap and semantic gap [13].

2.5.2 Object Retrieval Problems

When the aim is image retrieval, the challenge is to describe in fine a way the visual content. However there are many kinds of variations in an image that affect the classification, such as:

- **Illumination** - Figure 2.3;

  Lighting causes important variations in the value of the intensity of the pixels. Illumination changes in image have a key influence on its appearance. Illumination and the occurrence of shadows sometimes change the appearance of objects which makes the recognition of an image difficult.

- **Scale and Size** - Figure 2.4;

  An image can contain an object in front or far away. An object may appear alternative at different scales in the image. The scale and size of objects can considerably manipulate the similarity to other object of other classes.

- **Background Clutter** - Figure 2.5;

  A complex background may result in confusion between objects in the foreground and background image. This problem Increases false-positive results in object retrieval.

- **Viewpoint and Pose** - Figure 2.6;

  The position of the camera in relation to the object can change the appearance of an object in an image, which may lead to different results in the classification of the object.

- **Occlusion, Truncation and Articulation** - Figure 2.7;

  The visibility of some part of the object may be damaged because of the proximity or overlapping of another object in the image or position of the same object. This causes large variations between samples of the same class and increases the intra-class variation.

- **Intra-Class Variability** - Figure 2.8;

  Variation among instances between the objects belonging to the same class.

- **Inter-Class Variability** - Figure 2.9.

  Confuses scenes of various categories that are quite similar.

Figure 2.3: Illumination variation.

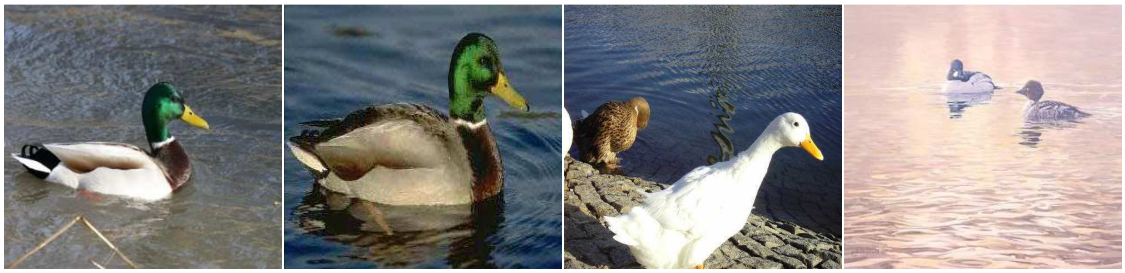


Figure 2.4: Scale and size variation.

Figure 2.5: Background Clutter.

Figure 2.6: Viewpoint and pose variation.

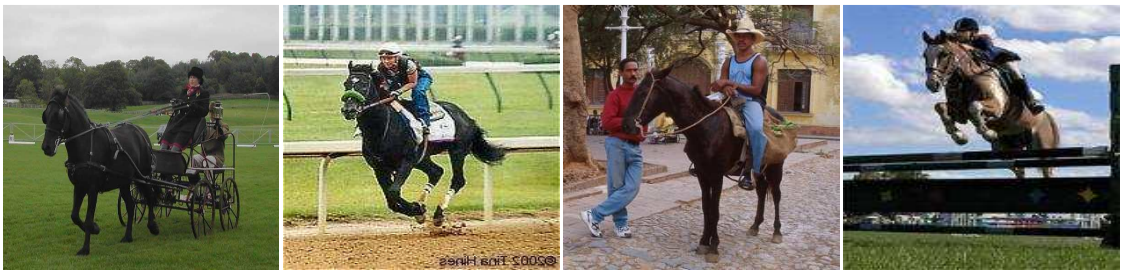


Figure 2.7: Occlusion, Truncation and Articulation.

Figure 2.8: Intra-Class Variability.

Figure 2.9: Inter-Class Variability.

Chapter 3

3. Feature Extraction

Chapter 3 introduces some common tools used in the feature extraction process. The first section of this chapter deals with the concepts of global and local image representation. Then, one of the most well-known feature descriptors are presented, such as Scale Invariant Feature Transform (SIFT). The end of the chapter includes quantization technique for image representation - Bag-of-Word (BoW) – and pyramid representation.

3.1 Image Representation

In the field of image retrieval all content-based image systems require an appropriate representation of the input data – image. An image is formed by pixels, which may or may not represent features. A feature is defined as an interesting part of an image and is used as a starting point for computer vision algorithms.

In fact, it does not matter what the image features signify. The features do not even have to be localized precisely, since the goal is not to match them on an individual basis, but rather to analyze their statistics [18].

Actually, an image can be represented globally or locally [19]. Global approach uses whole image to describe it - Figure 3.1 (a). While in local models, the selection of several regions or blocks of the image is utilized to characterize it - Figure 3.1 (b). In this case, there are sparse and dense representations.

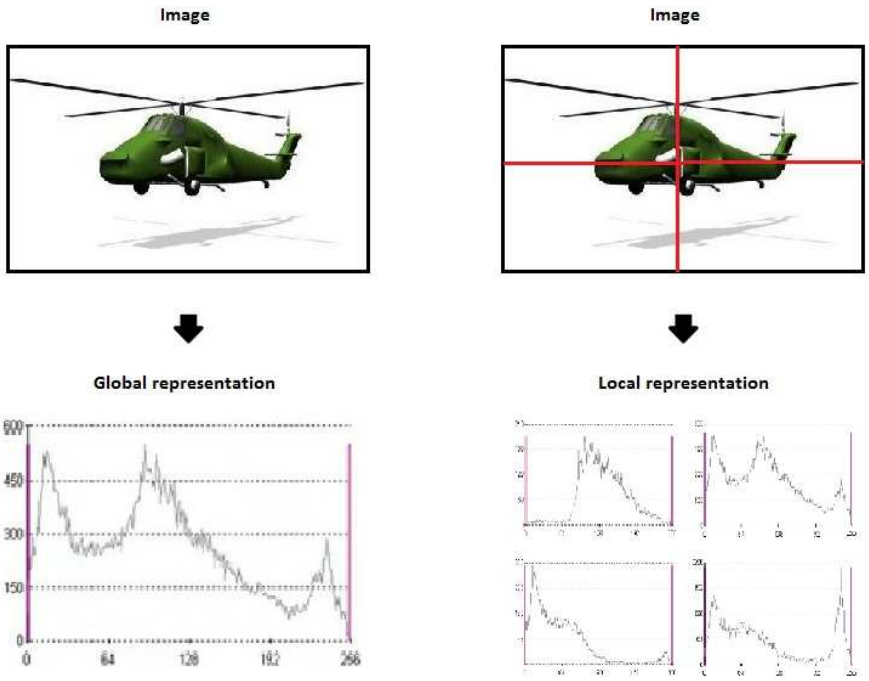


Figure 3.1: Example of global and local representation. (a) Global image representation. Entire image is classified as a block; (b) Local image representation. Each sub-block is independently classified obtaining a category for each one. Theses classifications are combined to obtain an image category.

Beside this, there are other methods to complement an image representation, namely bags-of-words model, which compress a range of values to a single quantum value – quantization.

The next subsections will give a detailed overview about these representations and the techniques to describe images.

3.1.1 Global Representation

In CV many global features have been proposed to describe the image content, through color histograms and techniques of shape and texture [18]. These approaches obtain a good performance, when classifying images into a small number of categories. However, global features cannot distinguish foreground from background of an image, and mix information from both parts together. Indeed, global features do not work for databases with many categories, due to high intra-class variability on images [19].

Even so global features can be combined in the decision of the classification with other representations, through weights. This can slightly improve the classification results.

3.1.2 Local Representation

Not all the pixels of an image provide relevant information. Thus, it is necessary to obtain the interest features of an image.

The interest of local features is that they offer a limited set of well localized and individually identifiable anchor points. Moreover, local information has a specific semantic interpretation in the limited context. Basically, a set of local features can be used as a robust image representation that allows recognizing objects without the need for segmentation. An ideal local feature would be a point as defined in geometry: having a location in space but no spatial extent. The proprieties of a good local feature are [18]:

- Must be highly distinctive - a good feature should allow for correct object identification with low probability of mismatch;
- Should be easy to extract;
- Invariance – a feature should be tolerant to image noise, changes in illumination, uniform scaling, rotation, and minor changes in viewing direction;
- Should be easy to match against a large database of local features.

Local representation, introduced in the late 90s, became a powerful tool that has been applied successfully in a wide range of retrieval systems and applications. To localize features in images, a detector needs to be applied. Local representations have two kinds of approach [20]:

- **Sparse**

  Decompose image into localized image patch descriptors around interest points or keypoints.

- **Dense**

  Divide image into localized image patch descriptors on a regular grid.

3.1.2.1 Sparse Representation

Firstly, sparse representation detects interest points or regions in the image. Then, this representation is extracted by a feature descriptor from each region.

As previously stated, this representation requires an interest point detector to select the best points, edge segments or regions which characterize the image. Even if the original image is rescaled or modified by illumination and viewpoint changes, the detector must localize points that can be repeated. The most common interest point detectors used in image recognition are:

- **Harris corner detector;**

It is the classical point detector [21]. This technique is based on detecting corners as areas with low self similarity – changes of light intensity (Figure 3.2). Harris corner detector used auto-correlation to determinate the key points.

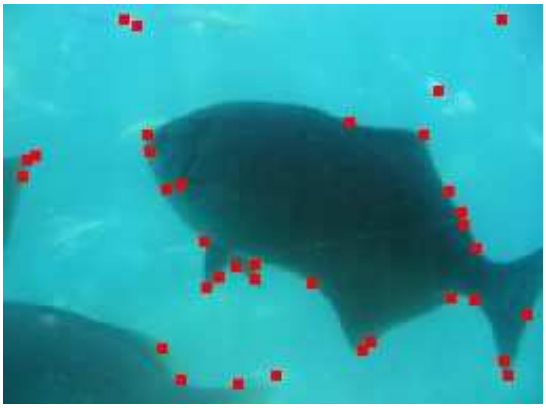


Figure 3.2: Example of a Harris corner detector [21].

- **Hessian-Affine;**

Hessian-Affine [22] can cope with affine[1] changes and belong to affine-covariant detectors. This detector, also invariant to scale and rotation, fit an elliptical region around the interest points (Figure 3.3). The ellipse is covariant and its shape is determined with the second moment matrix of the intensity gradient[2].

Figure 3.3: Example of Harris-Affine [22].

- **Difference-of-Gaussians (DoG);**

DoG [3, 23] is an approximation of the Laplacian, and involves convolving[3] the grayscale image with a Gaussian at several scales, creating a scale space pyramid of convolved images. The key points are detected by selecting positions in the image, which are stable across scales. Stable points are searched in these DoG images by determining local maxima, which appear at the same pixel across scales (Figure 3.4).

[1] transformations that move lines into lines, while preserving their intersection properties.

[2] vector field which points in the direction of the greatest rate of increase of the scalar field, and whose magnitude is the greatest rate of change.

[3] convolution is a mathematical operation on two functions, producing a third function that is typically viewed as a modified version of one of the original functions.

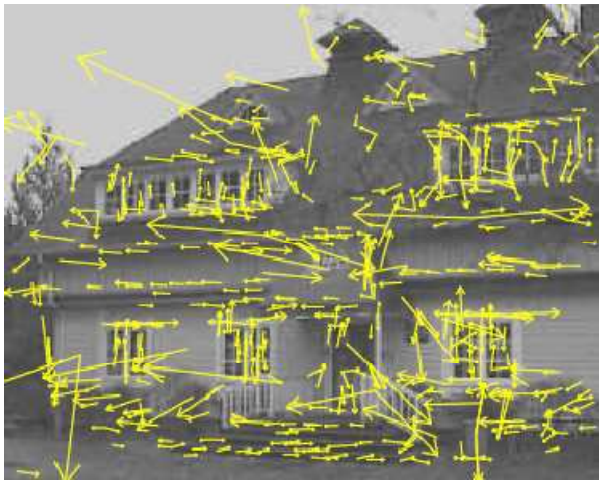


Figure 3.4: Example of Difference-of-Gaussian [3].

- **Maximally Stable Extrema Regions (MSER).**

MSER [24] also belongs to the class of affine-covariant detectors. They are based on connected components of an appropriately thresholded image. Just as with the Hessian-Affine detectors, an ellipse can be fitted to the output regions of the detector (Figure 3.5). The word extremal refers to the property that all pixels inside the MSER have either higher (bright extremal regions) or lower (dark extremal regions) intensity than all the pixels on its outer boundary [10]. Maximally stable is defined as the local minimum of the relative area change as a function of relative change of threshold.

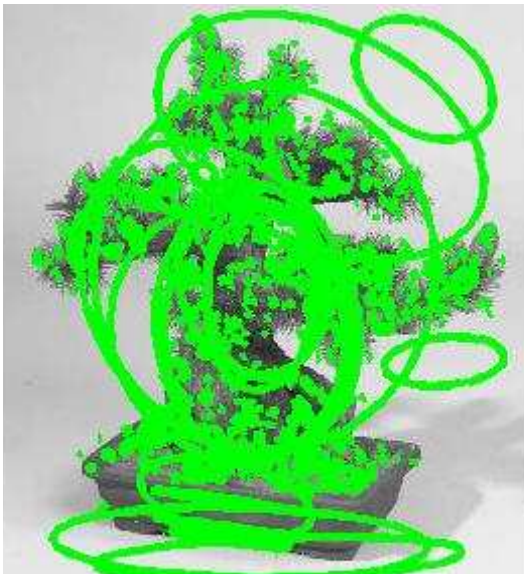


Figure 3.5: Example of MSER.

3.1.2.2 Dense Representation

Dense representation means that the features are not extracted at the key points, but the sense that each pixel contributes to the features description of the image on a dense grid (Figure 3.6).

The combat sparse vs. dense was tested by Jurie et al. [25]. The decision to use a dense regular grid instead of interest points was based on the comparative evaluation in, that have shown that dense features work better for scene classification. They concluded that dense feature has better performance in object categorization task. One reason for this is the fact that regions with uniform texture are well characterized, unlike sparse representation where texture information is not returned by detectors [26].

However, due to computational constraints, a combination of sparse representation with dense model can be useful. The idea is to use a sparse representation to detect interest points and, then, refine the initial object detection by further sampling of dense features around these key points. Dense sampling in the whole image is avoided and must only apply to the potential candidate regions [26].

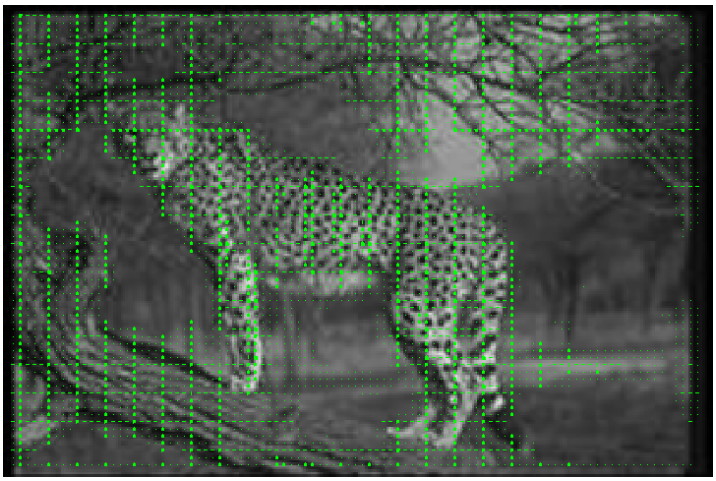


Figure 3.6: Dense grid on an image.

3.2 Features Descriptors

Once features have been detected, the second step of the feature extraction process is characterizing the region around each interest point. For that, feature descriptors or feature vectors are used to compute these regions.

In CV hundreds of descriptors have been introduced. There are descriptors just for color features (e.g. color histogram, color moments, color correlogram), shape information (e.g. moments invariants, Fourier descriptors), and texture attributes (e.g. Tamura features, fractal model). These features are described in Annex A.

However, for a good performance in object recognizing task you need descriptors which characterize features invariant to scale, orientation, affine distortion and partially invariant to illumination changes [3]. Thus, in 1999, David Lowe created an algorithm to detect and describe features with these attributes. This descriptor was designed by Scale Invariant Feature Transform (SIFT) [3].

Most of the recently works in image classification used SIFT descriptors. So, this section will detail the SIFT descriptor.

3.2.1 Scale Invariant Feature Transform

SIFT [3, 23] is decomposed in two stages. The first stage of the SIFT is finding the keypoint localization - Figure 3.7. For that, this descriptor uses DoG detector. The second step is keypoint orientation assignment and the keypoint descriptor computation (Figure 3.8).

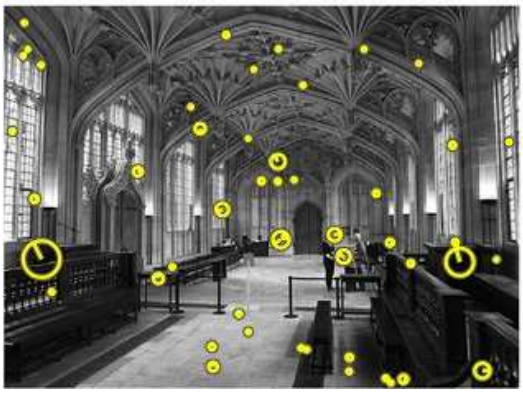
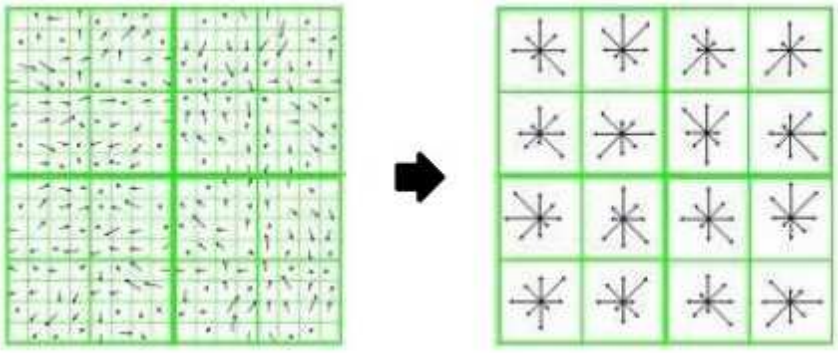


Figure 3.7: Interest points.

So for each interest point in an image there is a descriptor. A region around each keypoint is created and divided into orientation histograms on pixel neighborhoods (4 x 4). Each histogram contains 8 bins and each descriptor contains a *4 x 4* array of 16 histograms around the keypoint. This leads to a SIFT feature vector with 128 elements (4 x 4 x 8). Each image contains *n* keypoints, so an image is *n x 128* elements. The SIFT descriptor process on an image is presented in Figure 3.9.

This feature vector is normalized to enhance invariance to changes in illumination. The gradient histograms seem to contribute significantly to this performance by representing local shape. One disadvantage of SIFT is its high dimensionality [3].



(a) (b)

Figure 3.8: SIFT descriptor computation. (a) The gradients of an image patch around a keypoint. These gradients are then accumulated over *4 x 4* sub-regions, as shown on the (b). The length of the arrow corresponds to the sum of the gradient magnitudes in that direction.

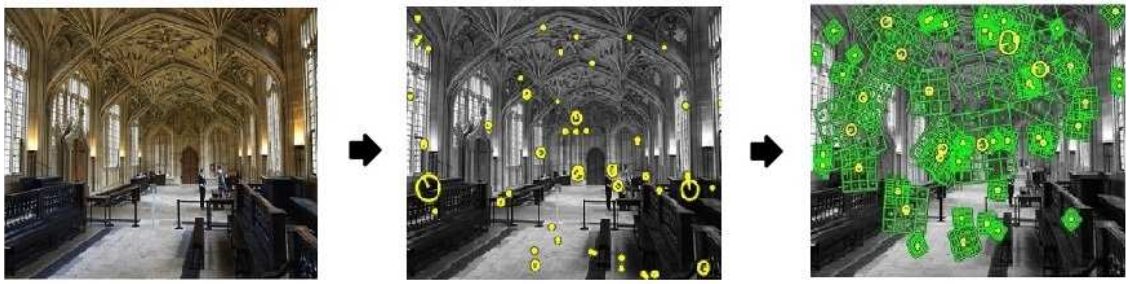


Figure 3.9: SIFT descriptor process.

Recently, some extensions of the original SIFT have been developed, such as PCA-SIFT [34], Gradient Location and Orientation Histogram (GLOH) [35] and Speeded Up Robust Features (SURF) [36]. The PCA-SIFT use Principal Component Analysis for dimensionality reduction. GLOH changes the location grid. SURF is based on Hessian detector.

3.3 Bags-of-Word Model

Over the last years new proposals have emerged in image classification research that uses the BoW model for image representation.

Initially this model was used in text retrieval [37]. However, Sivic and Zisserman [38] introduced this approach into computer vision community. BoW is based on regions and points of interest – local patches – and corresponding features descriptions. BoW uses a clustering method to quantize the features descriptors. The bag-of-words, also know by bag-of-features, is a histogram of words which is applied to images by using a visual analogue of a word formed by vector quantization of visual features. Each interest point is indexed into a visual codebook or vocabulary. This vocabulary is formed by clustering the feature descriptors - Figure 3.10. So, the dataset of images is clustered into *k* representative clusters, where each cluster stands for a visual word. The resulting cluster can be more or less compact, thus representing the variability of similarity for individual features matches. The value of *k* depends on the application, ranging from a few hundred or thousand entities for object class recognition applications up to one million for retrieval of specific objects from large databases. For clustering, most often *k*-means is used. This technique is exposed in Annex B. After vocabulary building an image is then modeled as a bag of those so called visual-words. It can thus be described by a vector or histogram that stores the distribution of all assigned codebook IDs or visual words - Figure 3.11.

Therefore, the BoW process involves the following steps:

- Local patches detection;
- Local descriptors computation over these regions;
- Quantization of the descriptors into words so as to create a visual vocabulary;
- The demand for occurrences in the image of each specific word in vocabulary in order to build a bag-of-words.

Recent works have exposed that local features represented by BoW are suitable for image classification since it showed impressive levels of performance [42, 43]. This model works very well, when it is used to classify images into a big number of categories.

An advantage of this method is the simplicity and the relatively small amount of supervision required, when used by SVM classifier [43].

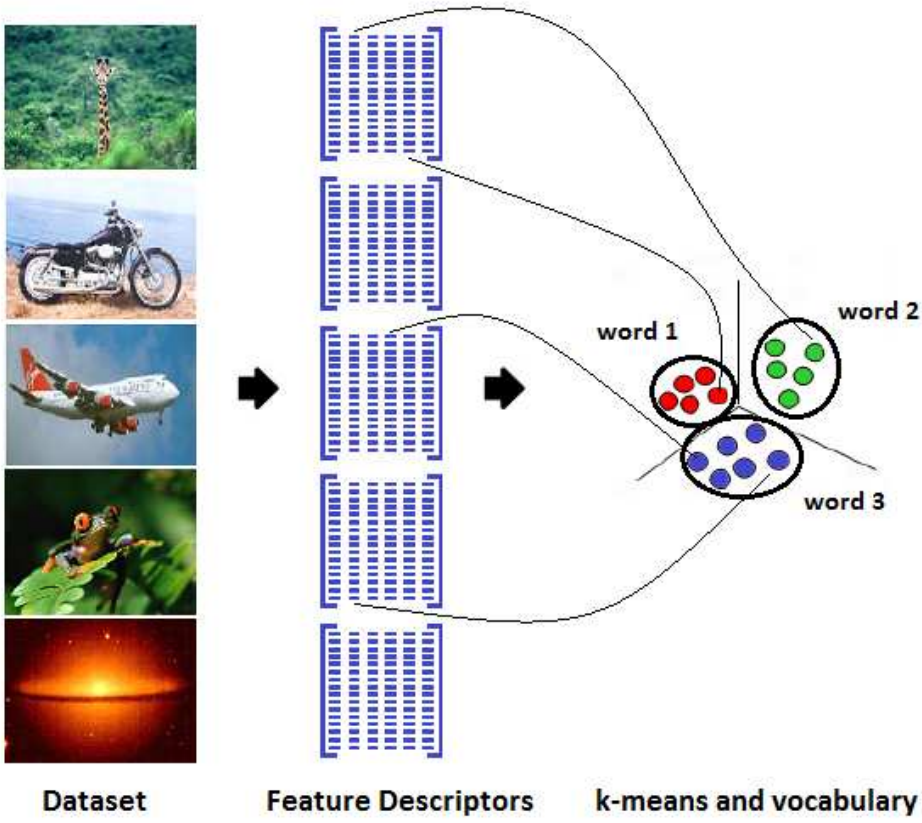


Figure 3.10: Vocabulary construction.

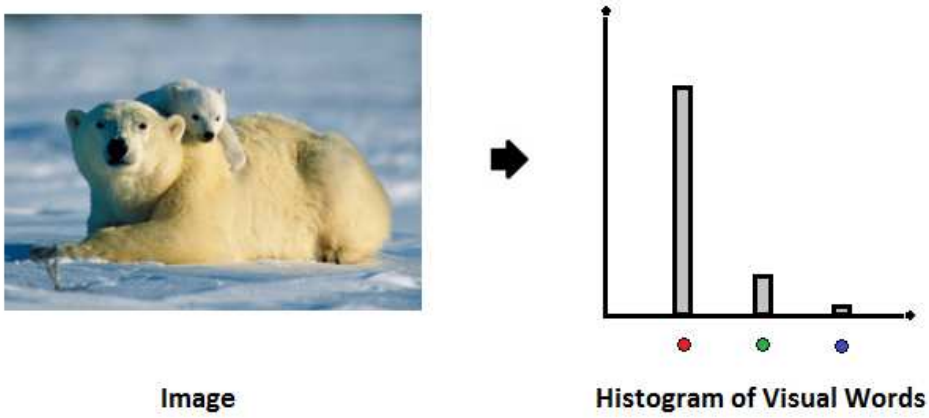


Figure 3.11: BoW representation.

Originally the BoW technique, described above, does not take the spatial information of the image features. Therefore the next sub-section will look for spatial information, which improves the image classification systems.

3.3.1 Pyramid for Representation

Spatial pyramid matching is an extension of the BoW image representation. A histogram of visual words is computed for each image sub-region at each resolution level as illustrated in Figure 3.12. A spatial pyramid is a collection of orderless feature histograms computed over cells defined by a multi-level recursive image decomposition. At level 0, the decomposition consists of just a single cell, and the representation is equivalent to a standard BoW. At level 1, the image is subdivided into four quadrants, yielding four histograms, and so on [44].

Pyramid is frequently used for expressing computationally efficient approximations to scale-space representation. Lazebnik et al. [44] proposed a pyramid representation based on pyramid match kernel planed by Grauman et al. [45], in which an image is tiled into regions at multiple resolutions.

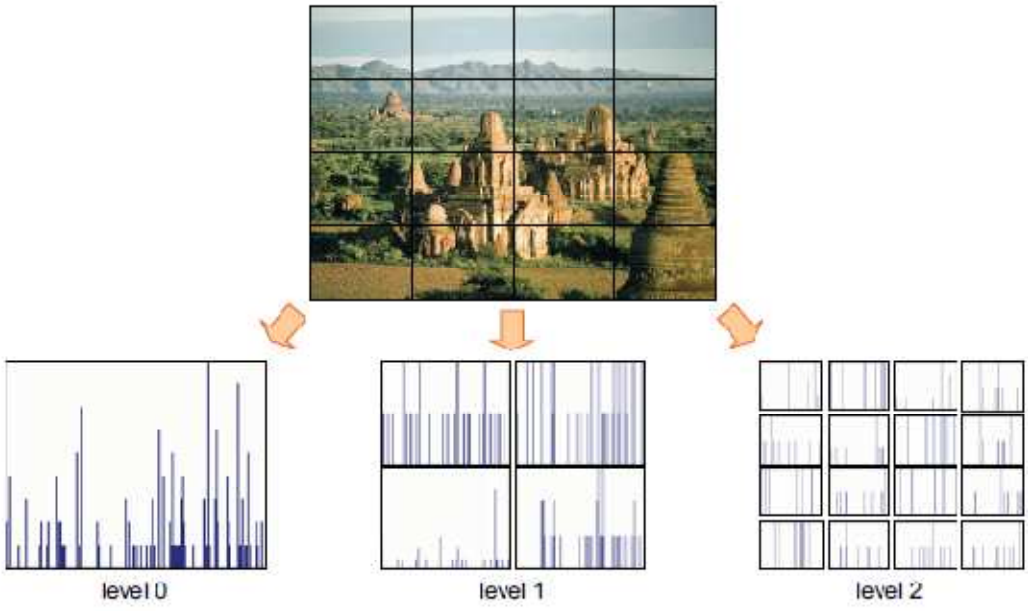


Figure 3.12: Spatial pyramid representation [44].

The pyramid matching works by inserting a sequence of increasing grids over the feature space of the image and taking a weighted sum of the number matches that occur at each level of resolutions (*L*). At any fixed resolution, two points are said to match if they fall into the same bin of the grid. Matches found at finer resolutions are weighted more highly than matches found at coarser resolution, reflecting the fact that higher levels localize the features more precisely. For strong features, the optimal level is *L = 2* [44], because the highest level of *L = 3* pyramid is too finely subdivided. Figure 3.13 shows a toy example of construction a pyramid.

Finally the representation of an appearance descriptor is a concatenation of the histograms of different levels into a single vector which are referred to as Pyramid Histogram of Visual Words (PHOW).

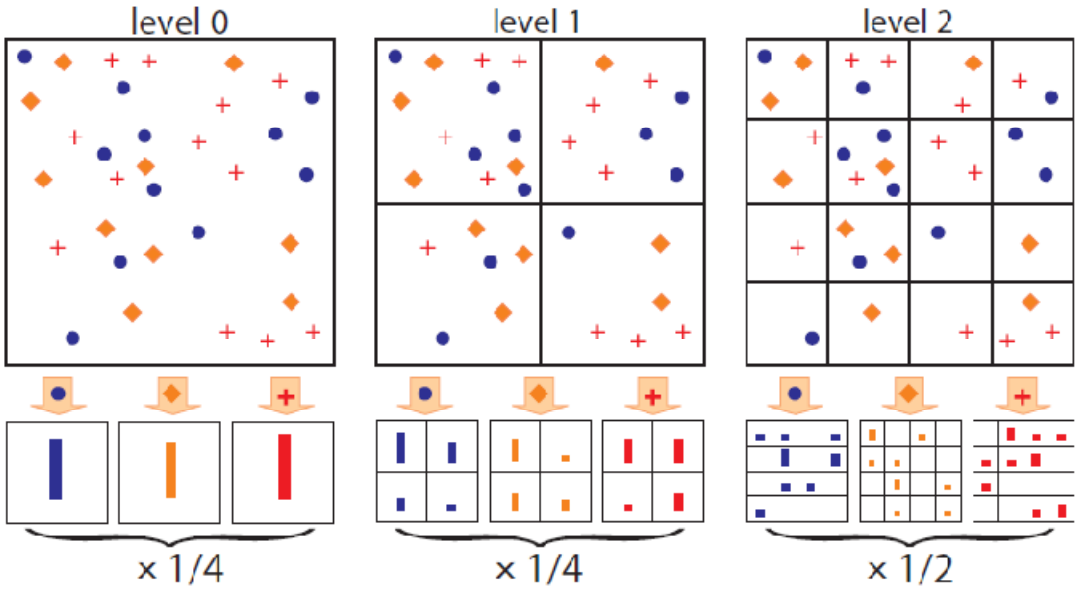


Figure 3.13: Example of constructing a pyramid for *L = 2*. The image has three feature types – circles, diamonds and crosses. At the top, the image is subdivided at three different levels of resolution. For each level of resolution and each channel, the features which fall in each spatial bin are counted. Finally, each spatial histogram is weighed [44].

Chapter 4

4. Datasets

In this chapter, the datasets used in object classification are presented. The datasets can be divided in three groups: natural scene images, object images and video images. Since the goal of this work is to classify images into object categories/classes, the dataset employed are the Caltech-101. Although not used in this work, the Caltech-256 is other object database. This chapter describes in detail these datasets.

4.1 Object Classification Datasets

Image classification literature utilizes two datasets to classify objects: Caltech-101 and Caltech-256.

4.1.1 Caltech-101

The Caltech-101 [2] dataset is formed from 102 object categories and contains 9145 images. Each classes includes between 40 and 800 images. Most images are medium resolution, about 300 x 300 pixels.

This dataset presents large inter-class variability and most images have little or no clutter. Objects are well aligned within each class and centered in each image. Most objects are presented in a stereotypical pose.

An image from each category is shown in Figure 4.1.

4.1.2 Caltech-256

Caltech-256 is an extension of Caltech-101. This dataset is formed from 256 object classes and contain 30608 images. Each category includes between 80 and 827 images per category.

This dataset has a large inter-class variability, as well as a larger intra-class variability than in Caltech-101. Moreover, the objects are not aligned within each class.

Figure 4.2 shows an image from each class of Caltech-256.

Figure 4.1: Caltech-101 categories.

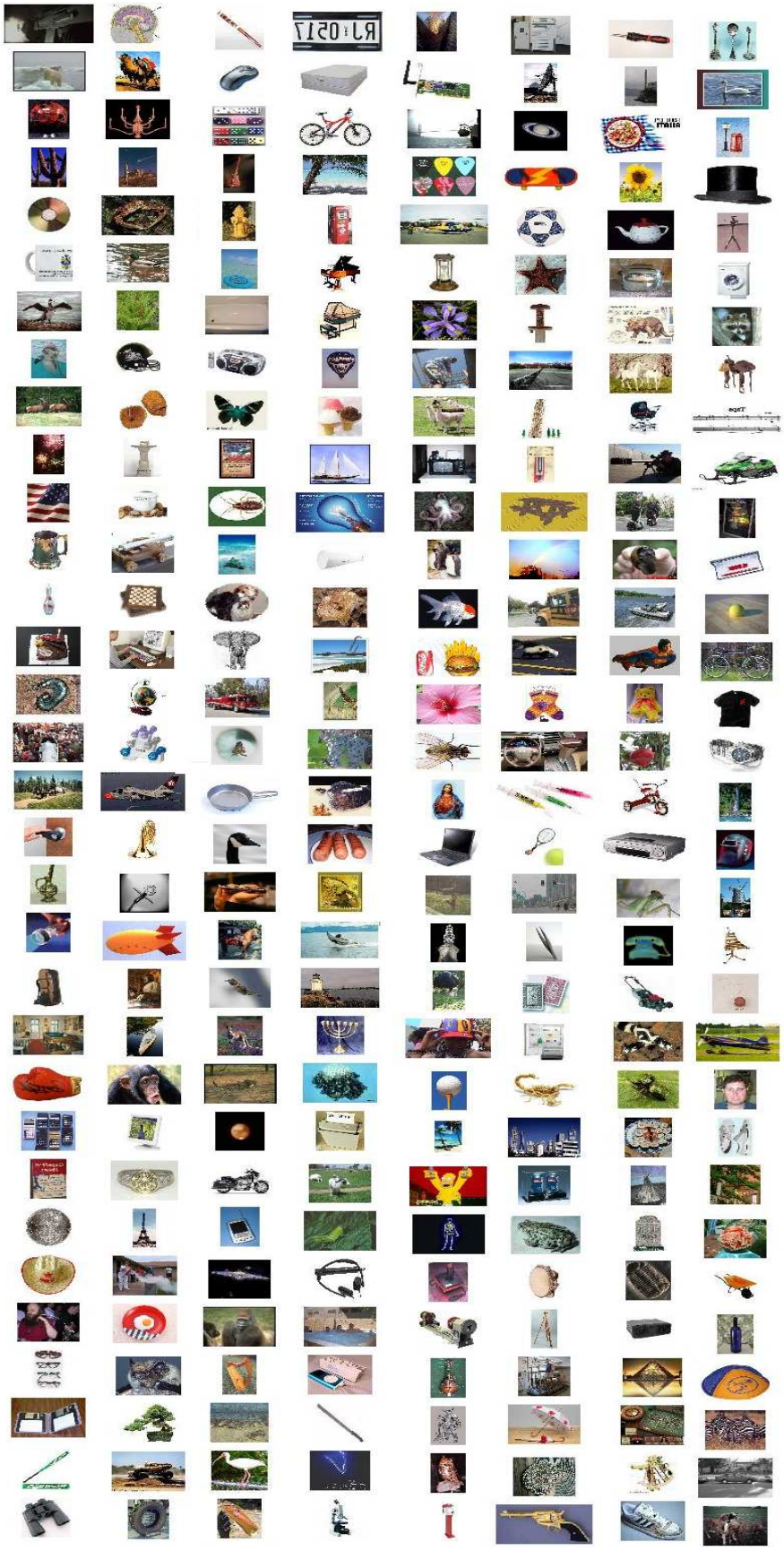


Figure 4.2: Caltech-256 categories.

Chapter 5

5. Classifiers

Initially, this chapter presents a brief architecture and concept of a general classification. Subsequently it presents the families of classifiers used in the literature: Unsupervised and Supervised Learning. First sections strut briefly the Unsupervised Learning (UL) and bring to light the Supervised Learning (SL) by dividing the classifiers into two categories: parametric and non-parametric. In the following sections, the first part characterizes the common supervised learning classifiers, such as Naive Bayes (NB), k-Nearest Neighbor (KNN), Decision Trees (DT), Artificial Neural Network (ANN) and Support Vector Machine (SVM). At the end of each classifier explanation, the applications on image classification where the classifiers become successful are presented. The second part presents the recent algorithms which improve the performance of image classification, such as the hybrid SVM-KNN and the Naive Bayes Nearest Neighbor (NBNN).

5.1 Architecture

Image classification is also an active area in the field of machine learning, in which it uses algorithms that map sets of input, attributes or variables – a feature space X - to set of labeled classes Y. These algorithms are known as classifiers. Basically what a classifier does is assign a pre-defined class label to a sample

Figure 5.1 shows a simple architecture of a classification system.

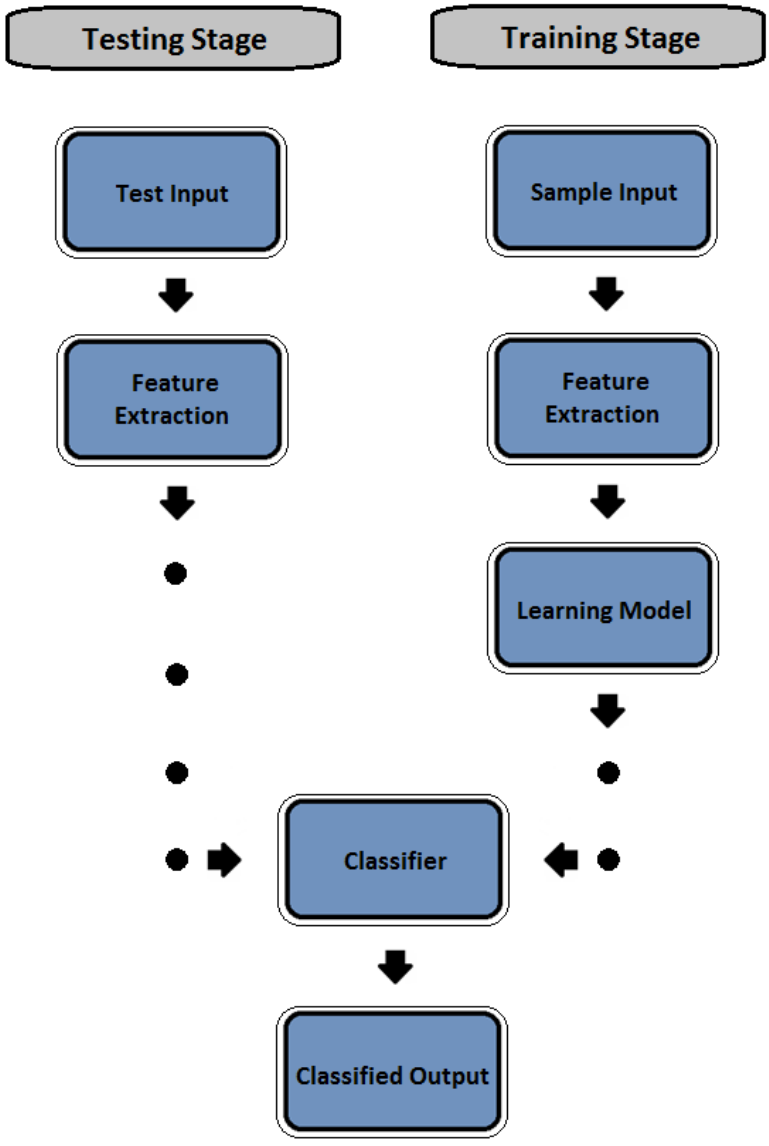


Figure 5.1: Design of classification system.

There are two main stages in a classification system:

- Training stage;
- Testing stage.

Training is the process of defining criteria by which features are recognized. In this process the classifier learns its own classification rules from a training set. In the training process, images are captured and stored in a database. Then there is the process of feature extraction. As previously stated in Chapter 3, an image is represented by a set of descriptors that structure the feature vectors. These feature vectors are considered input variables and are introduced in a learning component. The outputs are labels associated with classes (e.g. airplanes, faces, flowers).

In the learning component you have the discriminative and the generative models. The first model maps input variables directly to output variables in order to perform classification. The generative field models the distribution of features and learning is based on the likelihood of the data [46].

In the testing stage, the feature vectors of a query/test image works as input. A classifier decides on the bases of learning model, with its own classification rules, as to which class that feature vector belongs.

In literature, there are several approaches for classifiers, which can be characterized by two types of families [47]:

- Unsupervised learning (UL);
- Supervised learning (SL).

5.2 Unsupervised Learning

For UL, the system extracts information from unlabelled samples. In this case, the feature space of the entire dataset is clustered on the basis of some similarity criteria, forming a set of clusters. Each cluster belongs to a specify class.

The main problem of classification with unsupervised learning is how to take a decision between the feature vectors provided. Another problem is the selection of an algorithm that will cluster the vectors, since different clustering algorithms lead to different clusters.

5.3 Supervised Learning

SL involves a set of training data and category labels. The classifier is projected by utilizing this prior known information. In this case, the knowledge of the number of classes and their location in the feature space is the prior information.

The problem of this learning is that it takes some time to develop a classifier. However, this classifier is applied before the system is launched for the final users [13].

There are many techniques to design a classifier using supervised learning, which are based on two different categories:

- **Parametric methods;**

  These methods based on statistical parameters that assume a normal distribution (mean, covariance matrix, etc) and require an exhaustive learning or training phase of the classifier parameters. Examples of parametric classifiers are: SVM, DT and ANN.

- **Non-parametric methods.**

  These methods base their classification decision directly on the data, and do not require an intensive learning or training phase of the classifier. Examples of non-parametric classifiers are: NB, KNN and NBNN.

Next subsections will explain most standard supervised learning classifiers, like NB, KNN, DT, ANN and SVM. At the end, this chapter presents recently developed classifiers that improve the performance of image classification: SVM-KNN and NBNN.

5.3.1 Naive Bayes Classifier

Naive Bayes [8, 13, 48] is one of the simplest density estimation methods from which a classification process can be constructed. A Naive Bates classifier categorize patterns to the class C to which it is most likely to belong based on prior knowledge.

Naive Bayes classifiers can handle an arbitrary number of independent variables whether continuous or categorical.

Given a set of feature vectors, $= \{X_1, X_2, \dots, X_{n,}\}$, the objective is to construct the posterior probability for the class $\mathrm{C_j}$ among a set of possible outcomes set of classes $\mathrm{C} = \{\mathrm{C_1}, \mathrm{C_2}, \dots, \mathrm{C_m}\}$.

Using the Bayes theorem, the posterior probability of class $\mathrm{C_j}$ being X can be written as follows:

$$p(C_j|X_1, \dots, X_n) = \frac{p(C_j)\, p(X_1, \dots, X_n|C_j)}{p(X_1, \dots, X_n)} \quad (1)$$

where $p(C_j)$ is the prior probability of class C_j, $p(X_1, \dots, X_n|C_j)$ is the likelihood of X given class C_j and $p(X_1, \dots, X_n)$ is the evidence.

In practice in the equation 1 only the numerator of that fraction matters, since the denominator does not depend on C and the values of the features X are given. So that the denominator is effectively constant. The numerator is equivalent to the joint probability model, which can be rewritten as follows, using repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_j, X_1, \dots, X_n) &= p(C_j)\, p(X_1, \dots, X_n|C_j) = p(C_j)\, p(X_1|C_j)\, p(X_2, \dots, X_n|C_j, X_1) = \\ &= \cdots = p(C_j)\, p(X_1|C_j)\, p(X_2|C_j, X_1) \dots p(X_n|C_j, X_1, X_2, \dots, X_{n-1}) \end{aligned} \quad (2)$$

The nave conditional independence assumptions guarantees that each feature vector X_i is conditionally independent of every other feature vector X_k for $k \neq i$. This means that:

$$p(X_i|C_j, X_k) = p(X_i|C) \quad (3)$$

and, thus, the joint model can be expressed as:

$$p(C_j, X_1, \dots, X_n) = p(C_j)\, p(X_1|C_j)\, p(X_2|C_j) \dots p(X_n|C_j) = p(C_j) \prod_{i=1}^{n} p(X_i|C_j) \quad (4)$$

The Naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable. This is known as the *maximum a posterior* (MAP) decision rule. The corresponding classifier is the equation defined as follows:

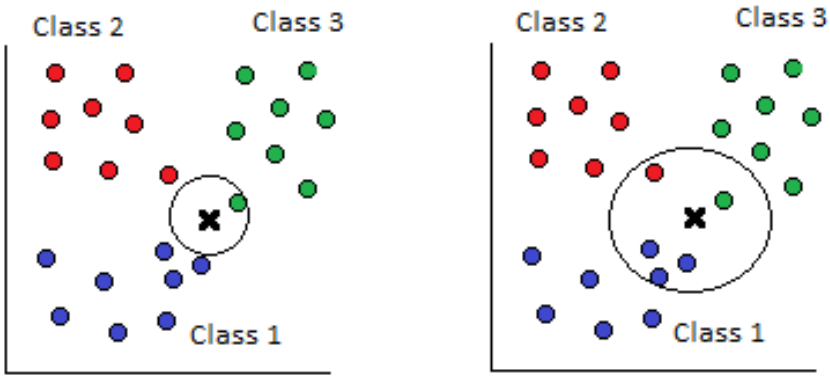
$$\hat{C}(X_1, \dots, X_n) = arg \max_C p(C_j) \prod_{i=1}^{n} p(X_i|C_j) \quad (5)$$

To exhibit the concept of Naive Bayes classification, consider the toy example demonstrated in Annex C.

In image classification, many complex applications based on NB classifier have successfully been implemented [13]. For example, Vailaya et al. [50] used three Bayes classifiers for his three stage classification. They introduced a hierarchical classification scheme in which they first classified indoor or outdoor images. Then, outdoor images were further classified into city or landscape images. Finally, landscape images were classified as sunset, forest and mountain images.

5.3.2 *k*-Nearest Neighbor

KNN [5, 47] classification method is a simplest technique conceptually and computationally, but it still provides good classification accuracy. The KNN classification is based on a majority vote of *k*-nearest neighbor classes. To understand how the KNN works, first define a point which represents feature vectors of an image in a feature space. Then, determine the distance between the point and the points in training data set. Finally, KNN classifier takes only *k*-nearest neighbor classes. So that majority vote is then taken to predict the best-fit class for a point. That means, in a $k = 5$ nearest neighbor classifier, the algorithm will take majority vote of its 5 nearest neighbors. For example, consider Figure 5.2 (a) where $k = 1$, the feature vectors of query image, point X, belongs to class 3 (green circles). If $k = 5$ as in Figure 5.2 (b) the point X best-fit in class 1 (blue circles) according to majority vote of the five nearest points.



(a) (b)

Figure 5.2: *k*-Nearest Neighbor classification. (a) 1-NN Classifier; (b) 5-NN Classifier.

To make predictions with KNN, it is necessary to define a metric for measuring the distance between the test point and cases from the example classes. One of the most popular choices to measure this distance is known as Euclidean.

The Euclidean distance between two points, r and s, is the length of the line segment $\overline{\mathrm{rs}}$. In Cartesian coordinates if $\mathrm{r} = (\mathrm{r}_1, \mathrm{r}_2, \ldots, \mathrm{r}_\mathrm{n})$ and $\mathrm{s} = (\mathrm{s}_1, \mathrm{s}_2, \ldots, \mathrm{s}_\mathrm{n})$ are two points in Euclidean *n*-space, then the distance from r to s is given by:

$$\mathrm{d}(\mathrm{r}, \mathrm{s}) = \sqrt{(\mathrm{r}_1 - \mathrm{s}_1)^2 + (\mathrm{r}_2 - \mathrm{s}_2)^2 + \cdots + (\mathrm{r}_\mathrm{n} - \mathrm{s}_\mathrm{n})^2} = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{r}_\mathrm{i} - \mathrm{q}_\mathrm{i})^2} \tag{6}$$

In image classification, Cheng et al. [51] used KNN classifier for 20 categories: firstly, images are segmented based on color and texture features. Yanai [52] classified color features into 50 conceptual categories using KNN classifier. Vogel et al. [53] used a KNN classifier to assign concept keywords to the different area of an image.

5.3.3 Decision Trees

DT [6, 13] uses a strategy of divide and conquer. This algorithm works by recursive partition of the input space. The tests are internal nodes and the leaf nodes represent the classification.

The classifications of the new test points start from the root node. The various links from the root node represents different possible results. The next step is to make the decision in the subsequent node and this is continued until one in the leaf node is reached. Each leaf node is a class label.

The simple DT is illustrated in Figure 5.3. For example, test T_2 has two outcomes. The left outcome is assigned to the class label 1 and the right one is sent to test T_4. Then T_4 has two outcomes which are class label 2 and class label 3.

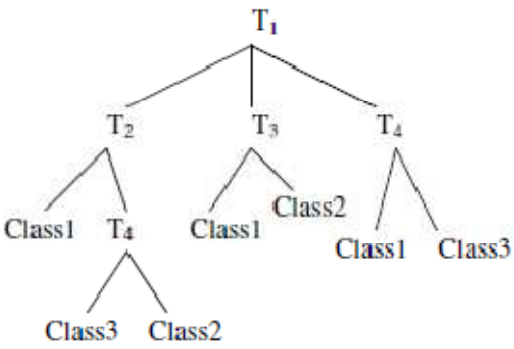


Figure 5.3: Decision tree [13].

The capacity of discrimination of this algorithm comes from:

- Space division defined by attributes $a_1, a_2, \ldots, a_n$ in a set of sub-spaces -
- Figure 5.4;
- Each sub-space is associated by a class.

The construction of a tree is demonstrated in Figure 5.5.

The decision tree is a hierarchical tree structure that is used to classify classes based on a series of rules about the parameters of the attributes of the class. In short, given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to recognize the class.

There are classifiers based on decision trees, like Random Forest (RF). The concept of RF is defined by a collection of decision trees, named forest. This method improves the classification rate. The classification works as follows: the random trees classifier takes the input feature vector of the test image, classifies it with every tree in the forest. The feature vector is passed down each random tree until it reaches a leaf node. The outputs of the classifier are the class label defined by the leaf node that received the majority of votes [20]. All the trees are trained with the same parameters - random variables[4], which make the decision of the output.

In image classification vision, MacArthur et al. [54] used decision trees on high-resolution computed tomography images of the human lung as the basis of their relevance feedback framework.

[4] when the value of a variable is the outcome of a statistical experiment (mean, medians, standard deviation, variance, entropy, etc)

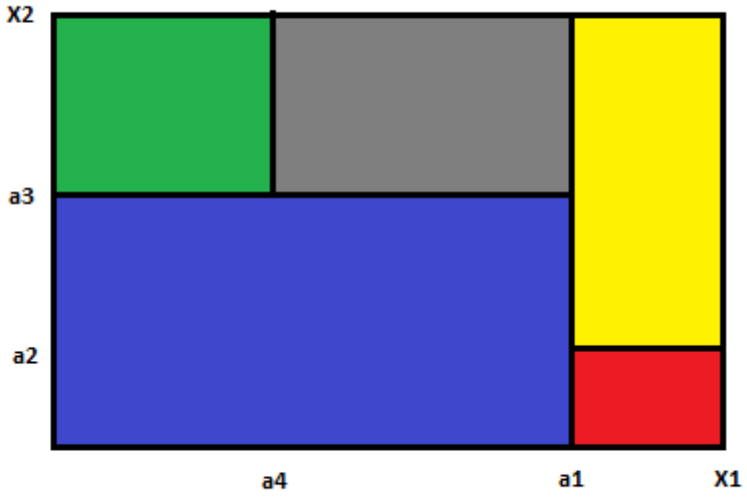


Figure 5.4: Space division [74].

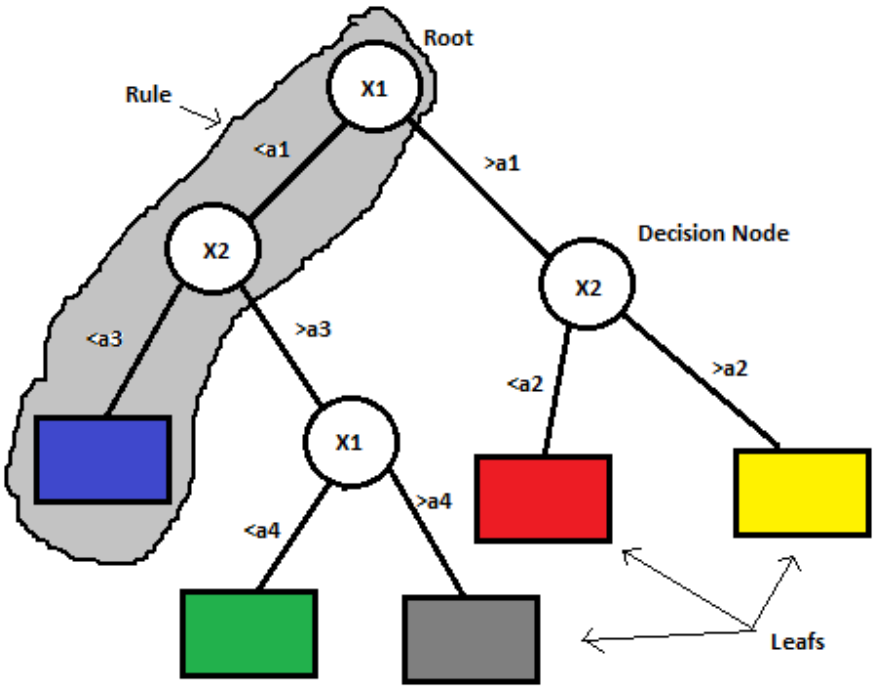


Figure 5.5: Tree Construction. Each node of decision has a test, corresponding to a possible value of this attribute. Each leaf corresponds to a region (hyper-rectangle) of space in classes. Each path of the tree (the root to a leaf) corresponds to a classification rule [74].

5.3.4 Artificial Neural Networks

ANN was designed to function as a human brain. This means that these networks process data, through artificial neurons as humans process information through neurons.

Due to its ability to learn that, the ANN has been very successfully. The first neural networks were created by McCulloch and Pitt [55] and the first classifier based on learning networks was developed by Rosenblatt [56].

The most popular supervised learning network model is multilayer *perceptron* (MLP), which is an extension of Rosenblatt model. Figure 5.6 illustrates the design of a MLP which is formed by an input layer, output layer and a hidden layer.

The inputs create input nodes of the network. In the input layer, the output from each neuron is fed to all the neurons in the hidden layer. There can be one or more hidden layers. Connections between the layers are typically formed by connecting each of the nodes from a given layer to all neurons in the next layer.

Each interconnection is linked by a string of connection scalar weights which are adjusted during the training phase. The outputs are taken from the output nodes.

In image classification, the feature vectors are submitted to an input layer, and the output represents a discriminator between its class and all of the other classes.

In the training phase, inputs of each training example are fed into the input layer of the network and the predicted outputs are computed. The next step is to compare predicted output and the target output and the difference is then measured.

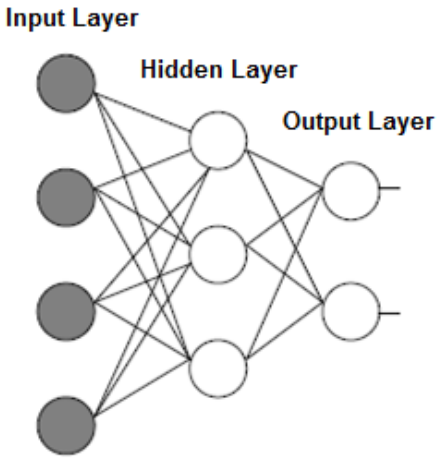


Figure 5.6: MLP Structure.

The error between the desired output and predicted output is propagated back though the network and the weights are adjusted. Thus, the network learns the mapping function by back propagating the error.

Neural network models have been successfully applied in a number of image classification research works. Cappellini et al. [57] applied a neural network for the classification of multi-spectral satellite images. A special neural network has been used by Wu [58] to create categories of human face and trademark images. Another example of human face recognition is by Kouzani [59], whereby in his work he used a multilayer neural network to learn the face and non-face patterns from face and non-face images. Breen et al. [60] developed a system to identify objects from images using neural network and these identified objects are fed into domain dependent ontology for classification. In the field of Bioinformatics, Wilson et al. [61] used neural network for fingerprint image classification. Kim et al. [62] used neural network for object and non-object classification based on color features.

5.3.5 Support Vector Machines

SVM [4, 13, 47, 63] introduced by Vladimir Vapnik is an algorithm which has shown better performance in many domains over other standard machine learning techniques.

The SVM is popular in image classification as this approach tries to find the optimal by separating a hyperplane between classes based on the training cases. Understanding on how SVM classifies data can be illustrated by a simple situation in which there are two linearly separable classes in d-dimensional space.

Using training feature vectors which belong to two different classes $\{x_i, y_i\}$ where $x_i \in R^d$ denotes vectors in a *d*-dimensional feature space and $y_i \in \{-1, +1\}$ is a class label. The goal is to design a hyperplane which is expected to generalize data accurately. This optimal hyperplane would be the one that leaves the maximum margin from both classes. Therefore, the basic idea is to find the hyperplane that has the maximum margin towards the sample object, that is, the greater the margin, the less the possibility that any feature vector will be misclassified.

A hyperplane can be defined by the equation:

$$(w \cdot x) + b = 0 \tag{7}$$

where x is a feature vector lying on the hyperplane, w is normal to hyperplane, b is the bias, and $\frac{|b|}{\|w\|}$ is a perpendicular distance from the origin to the middle hyperplane as depicted in Figure 5.7 which shows the basics of SVM.

The aim is to separate two classes (the white circles represents label -1 and the black circle represents +1). The circles which lie on the two planes P1 and P2 are the support vectors. The optimal hyerplane lies between the two planes P1 and P2 and they are parallel to each other. The margin between these planes is $\frac{2}{\|w\|}$. In order to obtain good generalization, the SVM maximizes the margin of the hyperplane. The hyperplane for the two classes can be described by $(w \cdot x) + b = +1$ (for class $y = +1$) and $(w \cdot x) + b = -1$ (for class $y = -1$). Normally, the classes are not linearly separable. In this case, input space has to be mapped into higher dimensional feature space. In other words, the input vectors, i.e. the feature vectors are mapped into high dimensional feature space H, through a nonlinear transformation, $\Phi: R^d \rightarrow H$.

Then, an optimal hyperplane is constructed in this high dimensional feature space using kernel function: $K(x_i, x_j)$

which products between input vectors x_i and x_j, where:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{8}$$

Kernel functions are necessary to map the data into a different space where a hyperplane can be used to do the separation.

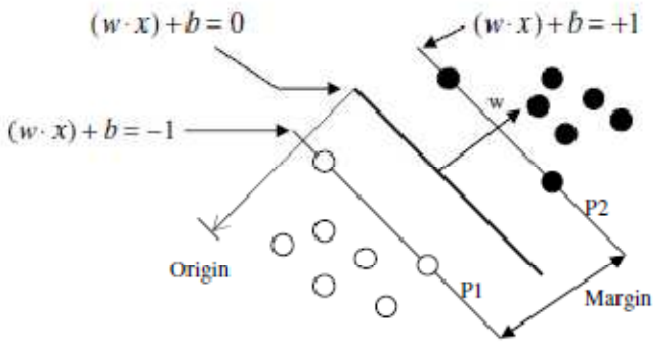


Figure 5.7: The basics of classification by SVM [13].

The most common mappings are polynomials kernel:

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^q \tag{9}$$

where q is the degree of polynomial;

and Radial Basis Function (RBF) kernel:

$$K(x_i, x_j) = e^{\frac{\|x_i - x_j\|^2}{2\sigma}} \tag{10}$$

where σ is a Gaussian sigma.

From the description above SVM is designed for binary classification problems, where there are only two possible class labels, $\{-1, +1\}$. This can be extended to multi-class problems, where each feature vector can be mapped to a label from a set of K labels $\{1, \ldots\ldots, K\}$. There are two general approaches for multi-class classification in SVM:

- One-Agaist-All;
- One-Against-One.

In the one-against-all method SVMs are constructed between each class versus all the other classes. In other words, a given class is compared with all other classes put together. For example, let there be four classes C_1, C_2, C_3 and C_4. Then, four SVMs are constructed. To classify C_1, SVM classifies C_1 and non-C_1 (i.e. C_2, C_3, and C_4) and the same goes for C_2, C_3 and C_4.

In the one-against-one method SVMs are constructed between all pairs of classes. For example, there are four classes, C_1, C_2, C_3, and C_4. Then, six SVMs are constructed i.e. the six SVMs classify C_1 or C_2, C_1 or C_3, C_1 or C_4, C_2 or C_3, C_2 or C_4, and C_3 or C_4 respectively.

The support vector machine is currently quite an attractive technique among the content based image retrieval research community. For example, Chapelle et al. [64] applied SVMs to annotate images on the bases of global RGB and HSV color histograms. In Tsai et al. [65] each image was divided into five parts and color and texture features are extracted from each part. For training SVM, manually clustered labels were used. Then, these trained SVMs were used to classify images into 25 categories. Cusano et al. [66] first divided the image into a fixed number of partially overlapping regions. Then, they applied multi-class SVM to classify the image regions into 7 predefined categories which are, sky, skin, vegetation, water, snow, ground, and building.

5.3.6 SVM-KNN

Recently, a hybrid version classifier based on KNN and SVM (SVM-KNN) was implemented by Zhang et al. [5]. The basic idea is to find close neighbors to a query sample and train a local SVM that preserves the distance function on the collection of neighbors. The steps of this technique are:

1. Compute distances of the query to all training examples.
2. If the *k* neighbors have all the same labels, the query is labeled and exit; else, compute the par-wise distances between the *k* neighbors;
3. Convert the distance matrix to a kernel matrix and apply multiclass SVM;
4. Use the resulting classifier to label the query.

5.3.7 Naive Bayes Nearest Neighbor

Recently, in 2008, Oren Boiman et al. [9] proposed a trivial classifier based on KNN and on Naive Bayes assumption. This algorithm was designed by Naive Bayes Nearest Neighbor (NBNN) and regained the status of non-parametric classifiers due to its good performance in datasets with large intra-class variances (e.g. Caltech-101 and Caltech-256).

This classifier technique employs NN distances in the space of the local image descriptors and introduces the concept of image-to-class distance instead of image-to-image distance.

.The algorithm of this classifier consists of the following [9]:

1. Compute descriptors $d_1, \dots d_n$ of the query/test image Q.
2. $\forall d_i \forall C$ compute the NN of d_i in C: $NN_c(d_i)$.
3. $\hat{C} = arg\min_C \sum_{i=1}^{n} \|d_i - NN_C(d_i)\|^2$.

NBNN is extremely simple, efficient and requires no learning or training phase. It uses the term ‘labeled images’ instead of ‘training images’, i.e. the classifier is fixed for all database image sets.

Chapter 6

6. Implementation and Results

Initially, Chapter 6 presents two approaches that was used in classifier Naive Bayes Nearest Neighbor (NBNN): image-to-class and image-to-image distances. The measure used is Euclidean distance. This chapter describes the implementations of CBIR systems using image-to-class distance and using image-to-image distance. It explains how this work makes the image representation and the feature process, as well how integrate the classifier NBNN in the system. Finally, the results of the CBIR systems are presented. The results decide which distance leads to better performance in the CBIR system.

6.1 NBNN Distances

As already stated, The NBNN classifier can assume two different ways regarding the Neighbor Nearest distance:

- **Image-to-class;**
- **Image-to-image.**

In this context, this work tests the two distances in order to evaluate which one obtains the best results within the CBIR implementation. The measure used in computation is Euclidean distance.

Next two subsections will explain the algorithm NBNN with the two distances.

6.1.1 Distance Image-to-Class

In the training phase all training images I ($I \in class\ C$) of the database compute and add descriptors $d_1, d_2, \dots, d_n$ to a *kd*-tree T_C. A *kd*-tree is an algorithm that improves the runtime and the computational complexity of classification. An explanation of *kd*-tree can be found in Annex D.

In the test stage, the algorithm computes descriptors $\mathrm{d}_1, \mathrm{d}_2, \dots, \mathrm{d}_n$ of the query image. Then, $\forall \mathrm{d_i}\ \forall \mathrm{T_C}$ compute $\mathrm{NN_C(d_i)}$ - the nearest neighbor of $\mathrm{d_i}$ to class C.

Finally, the NBNN classifies the class of the query image by the step 3 of the algorithm, that can be remembered:

$$\hat{\mathrm{C}} = \arg\min_{\mathrm{C}} \sum_{\mathrm{i=1}}^{\mathrm{n}} \|\mathrm{d_i} - \mathrm{NN_C(d_i)}\|^2 \tag{11}$$

6.1.2 Distance Image-to-Image

In the labeled step all training images I of the database compute and add descriptors $d_1, d_2, \dots, d_n$ to a *kd*-tree T_I.

In the test level, the algorithm computes descriptors $\mathrm{d}_1, \mathrm{d}_2, \dots, \mathrm{d}_n$ of the query image. Then, $\forall \mathrm{d_i}\ \forall \mathrm{T_I}$ compute $\mathrm{NN_I(d_i)}$ - the nearest neighbor of $\mathrm{d_i}$ to image I.

Finally, the NBNN classifies the class of the query image by the class of the nearest neighbor image of database. For that, it uses:

$$\hat{C} = \arg\min_{I} \sum_{i=1}^{n} \|d_i - NN_I(d_i)\|^2 \quad (12)$$

6.2 Implementation

The CBIR system implementation consists, as previously cited, of three modules:

- Feature extraction of database and query image;
- Compute the distances between features;
- Results classification.

Regarding the first module, feature extraction, the system randomly selects from Caltech-101 database 15 images per class. The features are extracted in these images, through the dense SIFT descriptor. The dense representation is used instead of sparse representation due to its best performance. The SIFT descriptors are computed over the gray scale image and on a regular grid with spacing *p* pixels (*p=3*). At each grid point the descriptors are computed over. A spatial bin with size *s* (*s=16*) covers these p pixels. This bin size is related to the SIFT keypoint scale.

This implementation does not use BoW representation, because it is a quantization technique which damages non-parametric classifier [9]. Logically, the pyramid representation described in Chapter 2 does not apply due to dependency with BoW.

To evaluate the image-to-class and image-to-distance in NBNN algorithm it was necessary implement two different CBIR systems.

6.2.1 Implementation Image-to-Class

The set of the descriptors extracted from the 15 images per class composes the data which represent the class (Figure 6.1). This means that all image descriptors, each one with a different dimension ($128 \times n_i$), are merged in a single matrix, with a dimension $128 \times (n_1 + n_2 + \cdots n_{15})$.

The next step is to index the classes and to create a *kd*-tree per class in order to improve runtime and computational complexity.

Figure 6.2 illustrates the implementation of CBIR system using image-to-class distance.

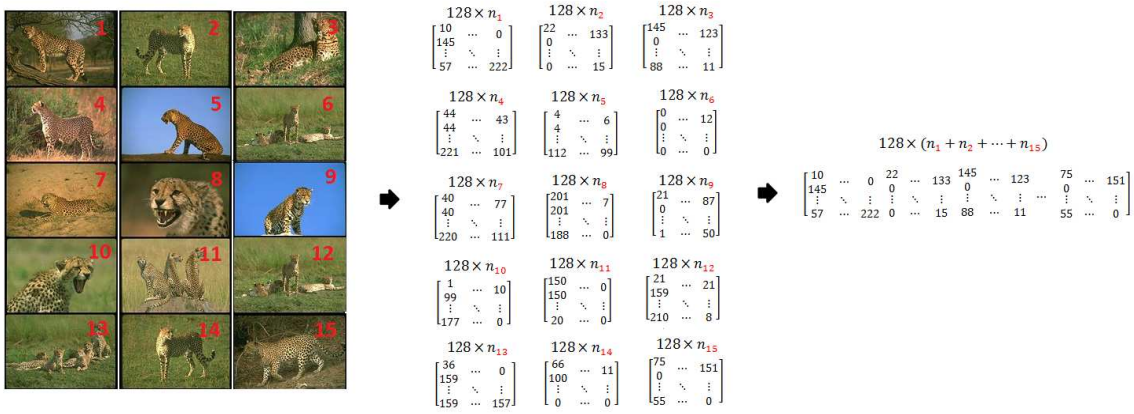


Figure 6.1: Class representation using image-to-class distance. (a) Random selection of 15 images for Leopard class on database; (b) Descriptors of the images. Each set of descriptors of image *i*, has a dimension $128 \times n_i$; (c) Descriptor set which represent Leopard class, with dimension $128 \times (n_1 + n_2 + \cdots n_{15})$.

In the interactive part of these systems, the user introduces a query image. This image suffers the transformation of the dense SIFT descriptor with the same parameters applied in image database, in order to obtain the query image features.

Therefore, the distances between features are compute using the approximation of the nearest neighbor (*kd*-tree) and the equation 11 – image-to-class distance.

The results of the classification are indexed by class approaching to the query image. Then, the images of the nearest class to the query image are displayed in an interface.

The interface of the CBIR system is presented and demonstrated in Annex E. Annex F shows the indexing of distance image-to-class calculated between the features of query image and the database class features.

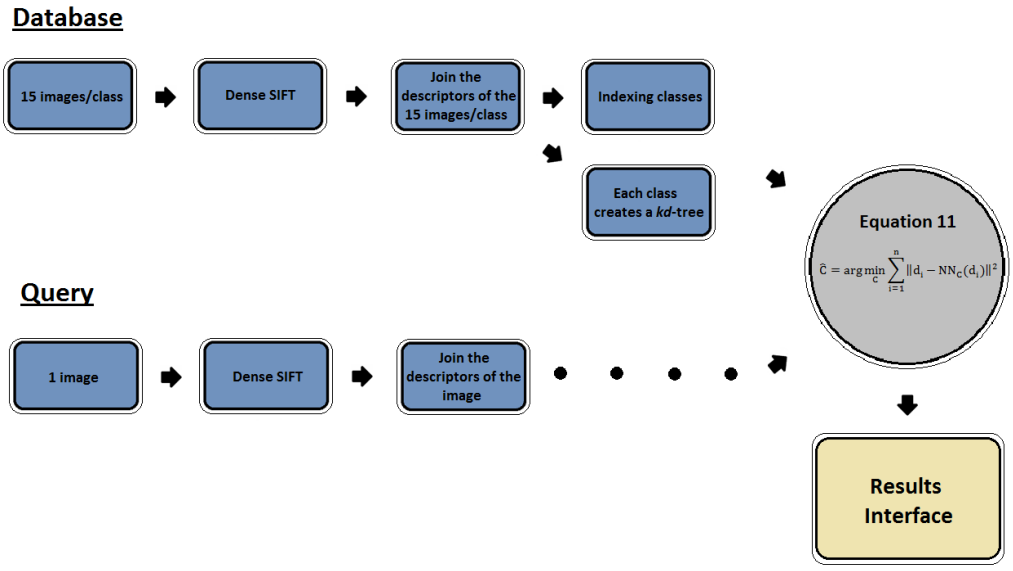
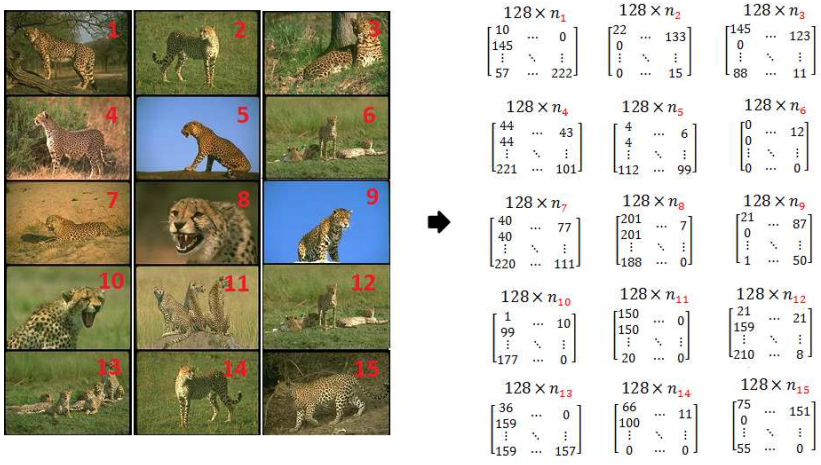


Figure 6.2: CBIR system implementation using image-to-class distance.

6.2.2 Implementation Image-to-Image

In the implementation image-to-image the classes have 15 representations based on the descriptors extracted of each image (Figure 6.3). This means that each class is represented by 15 matrixes with different dimensions ($128 \times n_i$).

The next step is to index the image classes and to create a *kd*-tree per each image descriptors. Figure 6.4 shows the implementation of CBIR system using image-to-image distance.



(a) (b)

Figure 6.3: Class representation using image-to-image distance. (a) Random selection of 15 images for Leopard class on database; (b) 15 individual set of descriptors which represent the Leopard class.

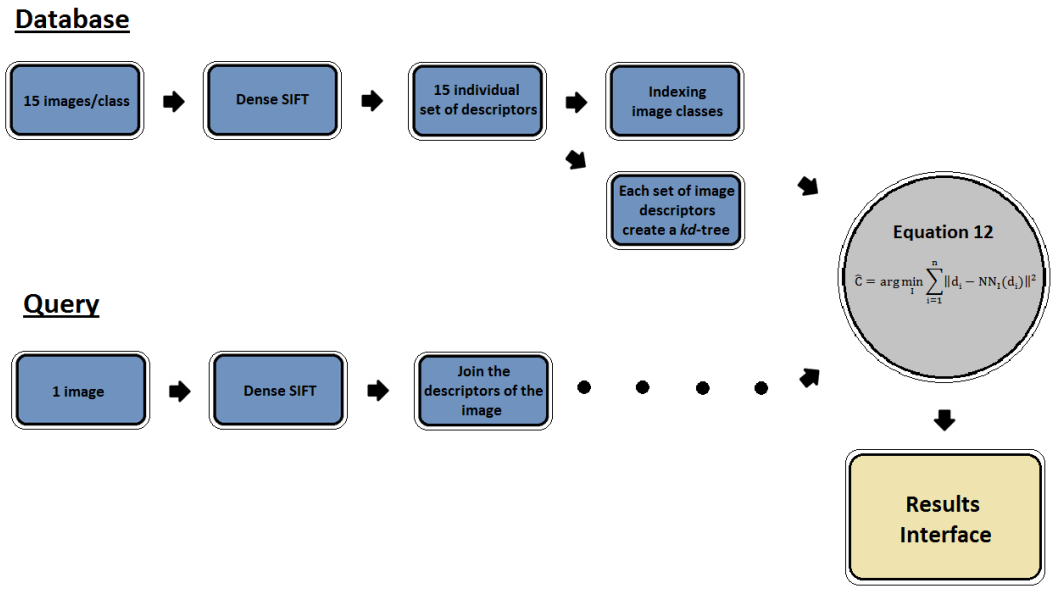


Figure 6.4: CBIR system implementation using image-to-image distance.

In this system, the query image suffers the same process previously described. The distances between features are compute using the approximation of the nearest neighbor (*kd*-tree) and the equation 12 – image-to-image distance.

Then, the results of the classification are indexed by approaching of database image class.

6.3 Results

The SIFT descriptor and the *kd*-tree used in this experiment was developed by VLFeat [69] open source library that implements popular computer vision algorithms. The simulations are realized in MATLAB [70].

To test the accuracy of these systems, it was used 20 query images per class – total of 2040 images tested. If in these images per category, x images - $\{x \in \mathbb{N} \mid 0 < x \leq 20\}$ - match to the true class, the accuracy of the class is x/20.

For the CBIR using the image-to-class distance, Figure 6.5 shows the distribution of number of classes for different levels of accuracy. Although many of the classes present a good performance (17 classes very well and 21 well classified), certain classes had a low classification (21 unsatisfactorily and 17 poorly categorized), which did reduce the overall performance.

Table 6.1 shows each individual class performance. The average time spent to classify a query image is 2.2 seconds per class.

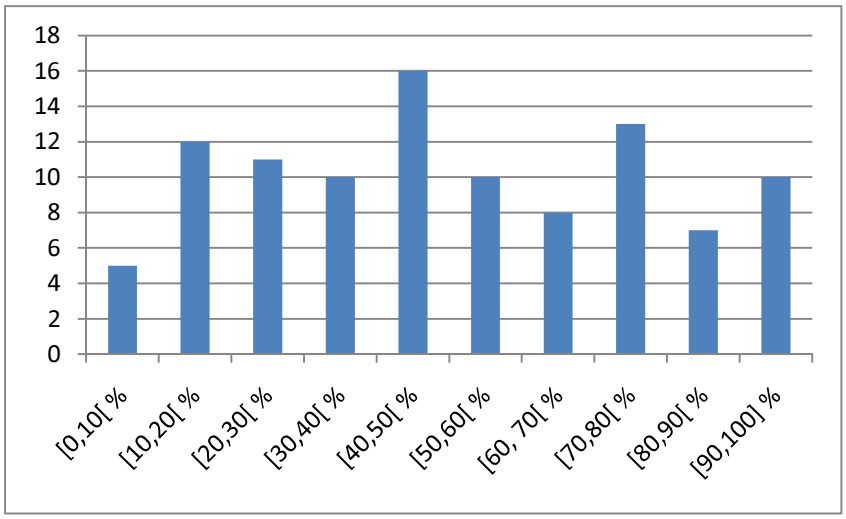


Figure 6.5: Number of classes for different levels of accuracy for image-to-class distance.

For the CBIR using the image-to-image distance, Figure 6.6 shows the distribution of number of classes for different levels of accuracy. In this case, only 14 classes very well classified, and 12 well categorized. Certain classes had a low classification - 27 unsatisfactorily and 27 poorly categorized. Table 6.2 shows each individual class performance.

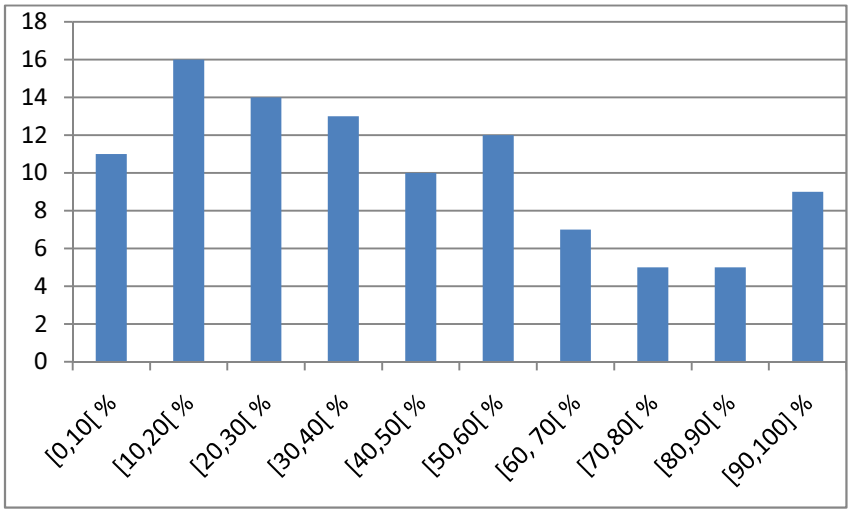


Figure 6.6: Number of classes for different levels of accuracy for image-to-image distance.

Table 6.1: Classification results by each class using image-to-class distance.

Class	Indexing	Accuracy	Class	Indexing	Accuracy
BACKGROUND_Google	1	0,05	helicopter	52	0,05
Faces	2	0,90	ibis	53	0,20
faces_easy	3	0,95	inline_skate	54	0,94
Leopards	4	1,00	joshua_tree	55	0,65
Motorbikes	5	0,85	kangaroo	56	0,15
accordion	6	0,70	ketch	57	0,40
airplanes	7	0,75	lamp	58	0,40
anchor	8	0,05	laptop	59	0,75
ant	9	0,15	llama	60	0,35
barrel	10	0,25	lobster	61	0,10
bass - peixe	11	0,20	lotus	62	0,05
beaver	12	0,15	mandolin	63	0,40
binocular	13	0,33	mayfly	64	0,15
bonsai	14	0,50	menorah	65	1,00
brain	15	0,70	metronome	66	0,88
brontosaurus	16	0,30	minaret	67	0,85
buddha	17	0,45	nautilus	68	0,20
butterfly	18	0,10	octopus	69	0,30
camera	19	0,45	okapi	70	0,70
cannon	20	0,10	pagoda	71	1,00
car_side	21	1,00	panda	72	0,40
ceiling_fan	22	0,30	pigeon	73	0,35
cellphone	23	0,75	pizza	74	0,75
chair	24	0,40	platypus	75	0,42
chandelier	25	0,50	pyramid	76	0,15
cougar_body	26	0,10	revolver	77	0,70
cougar_face	27	0,45	rhino	78	0,25
crab_image	28	0,15	rooster	79	0,70
crayfish	29	0,25	saxophone	80	0,55
crocodile	30	0,05	schooner	81	0,50
crocodile_head	31	0,45	scissors	82	0,70
cup	32	0,40	scorpion	83	0,15
dalmatian	33	0,45	sea_horse	84	0,20
dollar_bill	34	0,75	snoopy	85	0,65
dolphin	35	0,20	soccer_ball	86	0,60
dragonfly	36	0,50	stapler	87	0,50
electric_guitar	37	0,25	starfish	88	0,45
elephant	38	0,20	stegosaurus	89	0,45
emu	39	0,60	stop_sign	90	0,90
euphonium	40	0,60	strawberry	91	0,50
ewer	41	0,55	sunflower	92	0,75
ferry	42	0,50	tick	93	0,85
flamingo	43	0,15	trilobite	94	0,90
flamingo_head	44	0,60	umbrella	95	0,35
garfield	45	0,63	watch	96	0,65
gerenuk	46	0,37	water_lilly	97	0,25
gramophone	47	0,45	wheelchair	98	0,35
grand_piano	48	0,80	wild_cat	99	0,32
hawksbill	49	0,40	windsor_chair	100	0,85
headphone	50	0,75	wrench	101	0,55
hedgehog	51	0,85	yin_yang	102	0,90

Table 6.2: Classification results by each class using image-to-image distance.

Class	Indexing	Accuracy	Class	Indexing	Accuracy
BACKGROUND_Google	1	0,00	helicopter	52	0,05
Faces	2	0,65	ibis	53	0,50
faces_easy	3	0,95	inline_skate	54	0,63
Leopards	4	0,95	joshua_tree	55	0,35
Motorbikes	5	0,90	kangaroo	56	0,05
accordion	6	0,70	ketch	57	0,40
airplanes	7	0,50	lamp	58	0,35
anchor	8	0,05	laptop	59	0,75
ant	9	0,15	llama	60	0,20
barrel	10	0,10	lobster	61	0,10
bass-peixe	11	0,05	lotus	62	0,00
beaver	12	0,05	mandolin	63	0,40
binocular	13	0,17	mayfly	64	0,15
bonsai	14	0,35	menorah	65	0,90
brain	15	0,80	metronome	66	0,88
brontosaurus	16	0,15	minaret	67	0,90
buddha	17	0,40	nautilus	68	0,25
butterfly	18	0,10	octopus	69	0,25
camera	19	0,40	okapi	70	0,55
cannon	20	0,05	pagoda	71	0,80
car_side	21	0,90	panda	72	0,20
ceiling_fan	22	0,30	pigeon	73	0,30
cellphone	23	0,60	pizza	74	0,40
chair	24	0,15	platypus	75	0,42
chandelier	25	0,40	pyramid	76	0,35
cougar_body	26	0,10	revolver	77	0,50
cougar_face	27	0,30	rhino	78	0,15
crab_image	28	0,00	rooster	79	0,35
crayfish	29	0,25	saxophone	80	0,50
crocodile	30	0,10	schooner	81	0,30
crocodile_head	31	0,20	scissors	82	0,70
cup	32	0,30	scorpion	83	0,20
dalmatian	33	0,15	sea_horse	84	0,05
dollar_bill	34	0,75	snoopy	85	0,60
dolphin	35	0,10	soccer_ball	86	0,55
dragonfly	36	0,45	stapler	87	0,40
electric_guitar	37	0,15	starfish	88	0,50
elephant	38	0,10	stegosaurus	89	0,35
emu	39	0,25	stop_sign	90	0,80
euphonium	40	0,65	strawberry	91	0,50
ewer	41	0,35	sunflower	92	0,60
ferry	42	0,00	tick	93	0,75
flamingo	43	0,25	trilobite	94	0,95
flamingo_head	44	0,25	umbrella	95	0,25
garfield	45	0,58	watch	96	0,60
gerenuk	46	0,11	water_lilly	97	0,20
gramophone	47	0,50	wheelchair	98	0,25
grand_piano	48	0,80	wild_cat	99	0,21
hawksbill	49	0,50	windsor_chair	100	1,00
headphone	50	0,55	wrench	101	0,30
hedgehog	51	0,40	yin_yang	102	0,90

The global accuracy of each system is shown in Table 6.3, where the CBIR system using the distance image-to-class has 48,5% of accuracy and the CBIR system using the image-to-image distance presents 39,7% of accuracy.

Table 6.3: Accuracy of CBIR using the image-to-class and image-to-image distances.

Distance	**Accuracy**
Image-to-class	48,5%
Image-to-image	39,7%

The results of this testing showed a difference of 8.8% (different accuracy of distances of the Table 6.3) favorable to distance-to-class.

Oren Boiman et al. [9] showed a 17% difference between the distances, favorable to distance image-to-class. That research uses the database Caltech-101 with 30 images labeled per class and the descriptor SIFT in five different scales with image. The distance used was the Kullback Leibler[5] distance.

In relation to the experience in [9], this test used 15 labeled images per class and only applied the SIFT descriptor on one scale. The measure used was Euclidean distance.

Phil Huynh [72] showed a difference of 0.2% favorable image to the distance-to-class in the Central Park[6] database, using the Euclidean distance.

The difference of this result to the result in [72] must be the database used. While the Central Park database has no inter-class variability, the Caltech-101 presents this variability since it contains objects of different classes.

The results decide that image-to-class distance leads to better performance in the CBIR system.

[5] is a non-symmetric measure of the difference between two probability distributions
[6] stock leaf images in 143 species

Chapter 7

7. Conclusions and Future Work

This chapter summarizes the implementation developed as well as some conclusions. To finalize this thesis, it describes the future work associated to the system developed.

7.1 Conclusions

In this work, a CBIR system was implemented, combining the Scale Invariant Feature Transform (SIFT) descriptor and the Naive Bayes Nearest Neighbor (NBNN) algorithm in order to classify images according to their content. Due to the complexity and "subjectivity" of the problem, a simple classifier was applied. The advantage of this method is related to the absence of a learning phase. It just needs to assemble a group of features that label a class.

In the NBNN classifier, this work tested two different ways on the nearest neighbor measures: image-to-class and image-to-image distances. By comparing the two distances, the conclusion is that the image-to-class distance showed a better performance with an 8,8% difference to the image-to-image distance. Although, it is applied in systems with parametric classifiers, the image-to-image distance limits the ability of classification of non-parametric methods.

Overall, the obtained classification accuracy (2040 images tested – 20 per category) was 48.5%. This performance is satisfactory as this work only applied one single descriptor-type to extract the image features. Another reason is that the database used, Caltech-101, contains a large number of categories – 102 classes. This fact caused some inter-class variability in feature comparison, which causes a poor classification in some classes. On the other hand, the problems such as object pose, viewpoint or articulation of the image content cause difficulties to the classifier.

With the elaboration of this system, it was noticed that the image representation is a qualitative factor that affects the performance of classification. The absence of other descriptors, such as the features of level 1 (color, texture, shape), meant a loss in classification performance. Such low-level features were not implemented as a result of the high dimensionality of the SIFT descriptors. This high dimensionality of features leads to a large computational resources and a considerably large processing time. Another reason for the poor runtime process is associated to the programming code which MATLAB employs on the classification system - bottleneck computations (e.g. loops) must be avoided in MATLAB.

In the future, the computer processing capacity will be significantly better (according Moore's law), as it will improve the classification system performance. More features will be possible to group to labels of the classes and improve the performance of the classifier. The NBNN algorithm has great potential.

7.2 Future Work

The solution developed still leaves room for improvement. Regarding future work directly connected with implementation details, it is suggested:

- **Features**

  Combining different types of descriptors can significantly improve the classification performance. Several types of features can be mixed with weights in the decision rule of the classification.

- **Database**

  Integrating this system with other classification types (e.g. scenes) in appropriate databases.

- **Unknown categories**

  The system proposed is unable to recognize categories which have not been considered in the database. For example, if the user introduces a query image of a rhinoceros it would be wrongly classified as an elephant or a hippopotamus. The ideal solution would be a system able to say that it does not know the category in the example image.

- **Hierarchical model**

  Extending the system to automatically learn a category hierarchy. The highest levels will be able to classify amongst the easiest separable categories, grouping the related ones as the same category. Then lower levels can try to find more specific features to distinguish amongst these more related categories.

- **Feedback relevance**

  The user can affine the classification results by providing interactive information. In other words, the user indicates if a resulting image is a relevant example or a negative example. Then, the process of information needed is refined and started over again.

- **Programming**

  Use a programming more adapted to today's computers. Nowadays almost every computer has several processors (multiple cores), so the CBIR system should take advantage of these characteristics by implementing multithreading processes and divide the processing between all the cores, to speed up the decision-making process. Another possible improvement would be to avoid loops in MATLAB. In order to prevent this, it is necessary to

introduce MEX-files, which are dynamical subroutines produced from C or C+ code. These MEX-files run in MATLAB. This approach increases the speed of bottleneck computations.

References

[1] Long, F. H., H. J. Zhang and D. D. Feng, Fundamentals of content-based image retrieval, *in Multimedia Information Retrieval and Management - Technological Fundamentals and Applications*, Springer-Verlag, pp.1-26, New York, 2003.

[2] Caltech-101, available at: http://www.vision.caltech.edu/Image_Datasets/Caltech101

[3] Lowe, D. G., Distinctive image features from scale invariant features, *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91-110, 2004.

[4] Kumar, A. and Sminchisescu, C., Support kernel machines for object recognition, in *ICCV*, 2007.

[5] Zhang, H., Berg, A., Maire, M. and Malik, J., Svm-knn: Discriminative nearest neighbor classification for visual category recognition, in CVPR, 2006.

[6] Yang, C. C., Prasher, S. O. , Enright, P., Madramootoo, C., Burgess, C., Goel, P. K., Callum, I., Application of decision tree technology for image classification using remote sensing data, in *Agricultural Systems* 76, pp. 1101-1117, 2003.

[7] Musoko, V, Kolínová, M., Procházka, A, Image Classification Using Competitive Neural Networks, *12th Scientific Conference MATLAB2004*, Humusoft, 2004.

[8] Rish, I., An empirical study of the naive Bayes classifier, *in IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, 2001.

[9] Boiman, O., Shechtman, E. and Irani, M., In defense of Nearest-Neighbor based image classification, in *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, pp.1-8, 2008.

[10] Blaser, A., IBM Symposium: Data base techniques for pictorial applications, volume 81 of Lecture Notes in Computer Science (LNCS), Springer, Florence, 1979.

[11] Kato, K., Database architecture for content–based Image retrieval, 377 volume 1662 of SPIE Proceedings, pp. 112–123, San Jose, 1992.

[12] Müller, H., User interaction and evaluation in content-based visual Information retrieval, PhD thesis, University of Geneva, Geneva, 2002.

[13] Grubinger, M., Analysis and evaluation of visual information systems performance, 2007.

[14] Eakins, J.P. and Graham, M. E., Content–based image retrieval, JISC Technology Application Program, University of Northumbria, Newcastle upon Tyne, 2000.

[15] Castelli, V., Bergman, L. D., Image databases: search and retrieval of digital imagery, 2002

[16] Thakore, D. G., Trivedi A.I., Content based image retrieval techniques – Issues, analysis and the state of the art, *BVM Engineering College, Gujarat,* IEEE Member.

[17] Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A. and R. C. Jain, Content-based image retrieval at the end of the early years, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12): pp. 1349–1380, 2000.

[18] Tuytelaars, T., Mikolajczyk K., Local Invariant Feature Detectors: A Survey, foundations and trends in computer graphics and vision, 2008.

[19] Bosch, A., Image classificaion for a large number of object categories, PhD thesis, in *University of Girona*, 2007.

[20] Mihir, J.Towards efficient and scalable visual processing in images and videos, in *IIIT*, Hyderabad, 2010.

[21] Harris, C. and Stephens M., A Combined corner and edge detector, The Plessey Company plc., UK, 1988.

[22] Mikolajczyk. K, Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T. and Gool, L, A comparison of affine region detectors, *in IJCV*, 65(1/2): pp.43-72, 2005.

[23] Vedaldi, A., An inplementation of SIFT detector and descriptor, in *University of California*, Los Angeles.

[24] Forssen, P-E. and Lowe, D.G. Shape descriptors for maximally stable extremal regions, in *ICCV*, 2007.

[25] Jurie, F. and Triggs, B, Creating efficient codebooks for visual recognition, in *International Conference on Computer Vision*, 2005.

[26] Phil, D., Semantic image segmentation and web-supervised visual learning, PhD thesis in *University of Oxford*, 2009.

[27] Yong, R, Huanf, T.S and Chang, S-F, Image retrieval: current tecnhiques, promising directions and open issues, in *Journal of Visual Communication and Image Representation 10*, pp.- 39–62, 1999.

[28] Wang, X-Y, Wu, J-F and Yang, H-I, Robust image retrieval based on color histogram of local feature regions, Springer Netherlands, 2009.

[29] Pass, G., Zabih, R., Miller, J., Comparing Images Using Color Coherence Vectors, *Cornell University Ithaca*.

[30] Huang, Kumar, S.R., Mitras, M., Zhu, W-J, Zabih, R., Image indexing using color correlograms, *Cornell University Ithaca*.

[31] Parker, J.R., Algorithms for image processing and computer vision, Wiley Computer Publishing, 1997.

[32] Acharya, T., Ray, A.K., Image processing: principles and applications, John Wiley & Sons, Inc., 2005.

[33] Haralick, M., K. Shanmugam,K. and Dinstein, I., Textural features for image classification, in *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610-621, 1973.

[34] Ke, Y., Sukthankar, R., PCA-SIFT: A more distinctive representation for local image descriptors.

[35] Tola, E., Lepetit, V. and Fua, P., A fast local descriptor for dense matching, in *EPFL*, Lausanne.

[36] Bay, H., Ess, A., Tuytelaars, T and Gool, L, Speed-up robust features (SURF), 2008.

[37] Yates, B. and Neto, B.R., Modern information retrieval, ACM Press, 1999.

[38] Sivic, J. and Zisserman, A., Video google: A text retrieval approach to object matching in videos, in *International Conference on Computer Vision*, 2003.

[39] *k*-means tutorial**,** available at: http://people.revoledu.com/kardi/tutorial/kMean/Algorithm.htm

[40] MacQueen, J. B., Some methods for classification and analysis of multivariate observations, in *1. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press. pp. 281–297, 1967.

[41] Aurenhammer, F., Voronoi diagrams - A survey of a fundamental geometric data structure, ACM Computing Surveys, 23(3), pp. 345-405, 1991.

[42] Fei-Fei, L. and Perona, P., A bayesian hierarchical model for learning natural scene categories, in *IEEE Computer Society Conference on Computer Vission and Pattern Recognition*, Vol. 2, pp. 524-531, Washington, DC, 2005.

[43] Lazebnik, S., Schimid, S. and Ponce, J., Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in *IEEE Computer Society Conference on Computer Visision and Pattern Recognition*, Vol. 2, pp: 2169-2178, New York, 2006.

[44] Lazebnik, S., Schimid, S. and Ponce, J., Spatial pyramid matching.

[45] Grauman K. and Darrell, T., Pyramid match kernels: Discriminative classification with sets of image features, in *Proc. ICCV*, 2005.

[46] Dietterich, T.G., Becker, S. and Ghahramani, Z., On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes, in *Advances in Neural Information Processing Systems14: Proceedings of the 2001 Conference*, MIT Press, Cambridge, 2001.

[47] Duda, R.O., Hart, P.E., and Stork, D.G., *Pattern Classification*, 2nd Edition, John Wiley, New York, 2001.

[48] Zhang, H., The optimality of naive Bayes, *Proceedings of the 17th International FLAIRS Conference*, AAAI Press, 2004.

[49] Naive Bayes example available at: http://www.statsoft.com/textbook/naive-bayes-classifier

[50] Vailaya, A., Figueiredo, M., Jain, A., and Zhang, H.J., A bayesian framework for semantic classification of outdoor vacation images*, in IEEE Trans. Image Processing*, Vol. 10, No. 1, pp. 157-172, 2001.

[51] Cheng, Y.C., and Chen, S.Y., Image classification using color, texture and regions, in *Image and Vision Computing*, vol. 21(9), pp. 759-776, 2003.

[52] Yanai, K., Generic image classification using visual knowledge on the Web, in *Proceedings of the ACM International Conference on Multimedia*, pp. 167-176, Berkeley, 2003.

[53] Vogel, J. and Schiele, B., Natural scene retrieval based on a semantic modeling step, in *Proceedings of the International Conference on Image and Video Retrieval*, pp. 207-215, Dublin, 2004.

[54] MacArthur, S., Brodley, C., and Shyu, C.R., Relevance feedback decision trees in content-based image retrieval, in *Proceedings of the IEEE Workshop on Content-based, Access of Image and Video Libraries,* pp. 68-72, 2000.

[55] McCulloch, W.S., and Pitts, W., A logical calculus of the idea immanent in nervous activity. *Bulletin* of *Mathematical Biophysics,* vol. 5, pp.115-133, 1943.

[56] Rosenblatt, F., The perceptron: A probabilistic model for information storage and organization in the brain, *Psychologica Review,* vol. 65, pp.386-408, 1958.

[57] Cappellini, V., Chiuderi, A., and Fini, S., Neural networks in remote sensing multisensor data processing, in *Proceedings of the 14th EARSeL Symposium'94,* J. Askne, Rotterdam, pp. 457-462, 1995.

[58] Wu, J.K., Content-based indexing of multimedia databases, in *IEEE Transaction on Knowledge and Data Engineering, 9*(6), pp. 978-989, 1997.

[59] Kouzani, A.Z., Locating human faces within images, in *Computer Vision and Image Understanding*, 91(3), pp. 247–279, 2003.

[60] Breen, C., Khan, L., and Ponnusany, A, Image classification using neural network and ontologies in *DEXA workshop,* pp. 98-102, 2002.

[61] Wilson, C.L., Candela, G.T., and Watson, C.I., Neural network fingerprint classification, in *J. Artificial Neural Networks*, vol. 1, no. 2, pp. 203-228, 1993.

[62] Kim, S., Park, S., and Kim, M., Image classification into object/nonobject classes, in *Proceedings of the International Conference on Image and Video Retrieval*, pp. 393-400, Dublin, 2004.

[63] Cortes, C. and Vapnik, V., Support-vector Networks, Machine Learning, 20, 1995.

[64] Chapelle, O., Haffner, P., and Vapnik, V., Support vector machines for histogram-based image classification, in *IEEE Transactions on Neural Networks*, vol.10(5), pp. 1055-1064, 1999.

[65] Tsai, C.-F., McGarry, K., and Tait, J., Image classification using hybrid neural networks, in *Proceedings of the ACMSIGIR Conference on Research and Development in Information Retrieval*, pp. 431-432, Toronto, 2003.

[66] Cusano,C., Ciocca, G., and Schettini, R. Image annotation using SVM, in *Proceedings of Internet Imaging IV,* Vol. SPIE 530, pp. 330-338., 2004.

[67] Deng, K., Omega: On-line memory-based general purpose system classifier, Chapter 5: Efficient Memory Information Retrieval, 1998.

[68] Moore, A.W., An introductory tutorial on *kd*-trees, *Carnegie Mellon University*.

[69] VLFeat open source library, available at: http://www.vlfeat.org/

[70] Image Processing Toolbox for use with MATLAB User's Guide, The MathWorks, 2008.

[71] MATLAB-GUI tutorial, available at: http://www.mathworks.com/help/techdoc/creating_guis/bqz79mu.html

[72] Huynh, P., An evalution of nearest neighbor images-to-classes versus nearest neighbor images-to-images.

[73] Lindeberg, T., Scale-Space Theory in Computer Vision, Kluwer Academic Print on Demand, 1994.

[74] Construction of a Decision Tree available at: http://www.ccog.up.pt/~jgama/Aulas_ECD/arv.pdf

[75] Kato, T., Database architecture for content–based image retrieval, in *A. A. Jamberdino and W. Niblack, editors, Image Storage and Retrieval Systems*, 1992.

[76] Linda G. and Stockman, G. C, Computer Vision, Prentice Hall, 2003

Annex A

Low-level Features

Color

The color feature is one of the most simple and widely used visual features in content-based image retrieval. It is relatively robust to background complication and independent of image size and orientation [27]. There are different color spaces, the most commonly - RGB (Red, Green, Blue) - and the more effective to measure the color similarity between images – HSV (Hue, Saturation, Value) and HLS (Hue, Lightness, Saturation).

To represent color features in CBIR system there are many techniques, such as:

- Color Histogram [28];
- Color Coherence Vector [29];
- Color Correlogram [30]

The most universal method is color histogram. That technique describes the proportion of pixels of each color in an image. The color histogram is obtained by quantizing image colors into discrete levels and then counting the number of times each discrete color occurs in the image.

In a CBIR system, a query image is compared with the histograms of all the images in database.

Texture

The major characteristic of texture is repetition of patterns over a region in an image [31]. Texture is a natural property of all surfaces, such as woods, clouds, bricks, skin, etc [32].

Textures are extracted from groups of pixels or regions in contrast to color which is extracted from each pixel. Texture analysis methods can be divided into two categories:

- Spectral approaches;
- Statistical approaches.

In spectral approach, texture description is done by generating a Fourier transform of an image and then groups the transform data in a way that it gives you some set of measurements.

Statistical methods describe textures according to their underlying statistical property. There are a number of techniques available to measure the similarity between two textures. The most well-known method is designed by Co-occurrence matrix [33]. This technique constructs a co-occurrence matrix on the basis of orientation and the distance between the pixels. Then meaningful statistics are extracted from a co-occurrence matrix to represent textures.

The CBIR systems which apply statistical methods have good reputation because of their capability in describing very weak real-world textures.

Shape

Shape features can represent spatial information that is not presented in color and texture histograms. Shape contains all the geometrical information of an object in the image which does not generally change even when the location, scale and orientation of the object are changed [13].

To describe shape features it is essential to segment the image to obtain regions or objects. In general the shape representation can be divided into two categories which are:

- Boundary-based;
- Region-based.

The first category uses only the boundary information of an object also commonly known as edge detection. The second category is region-based, using the whole shape region [27].

Annex B

k-means

k-means [39] is an algorithm to group objects based on attributes/features into *k* positive integer number of clusters. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. *k*-means clustering method was developed by J. MacQueen [40].

The basic procedure of this algorithm is to determine:

- The number of clusters – value of *k*;
- The centroid coordinate;
- The distance of each object to the centroids.

Then, the final procedure is to group the objects based on minimum distance.

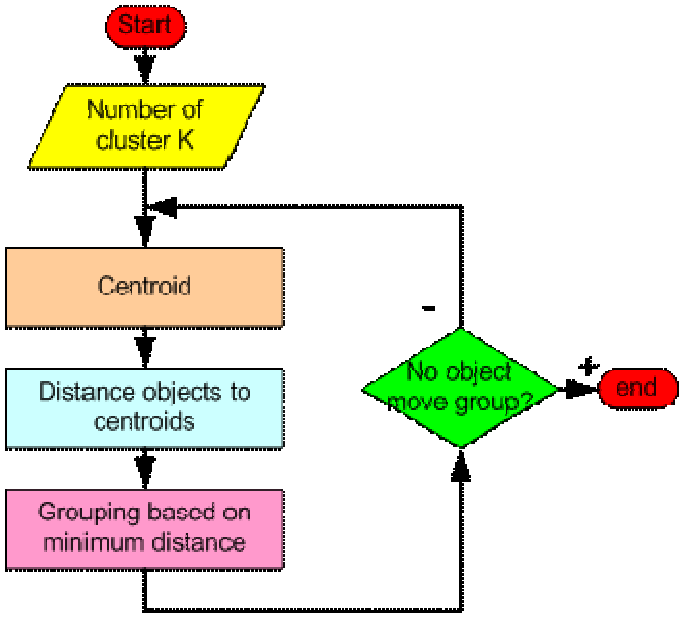


Figure B.1: *k*-means scheme [39].

For that the steps of the algorithm are (illustration in Figure B.1) [39].

1. Take a decision on the value of *k*.
2. Put any initial partition (e.g. Voronoi diagram[7] [41]) that group the data into *k* clusters. In this phase is necessary to assign the training samples randomly or systematically.
3. Take each sample in sequence and compute its distance (e.g. Euclidean distance) from the centroid of each of the clusters. If the sample is not currently in the cluster with the closest centroid, switch this sample to that cluster. Then, update the centroid of the clusters that added the sample and update the centroid of the cluster that subtracts the same sample.
4. Repeat step 3 until convergence is achieved.

A simple *k*-means construction with a Voronoi diagram is exemplified in Figure B.2.

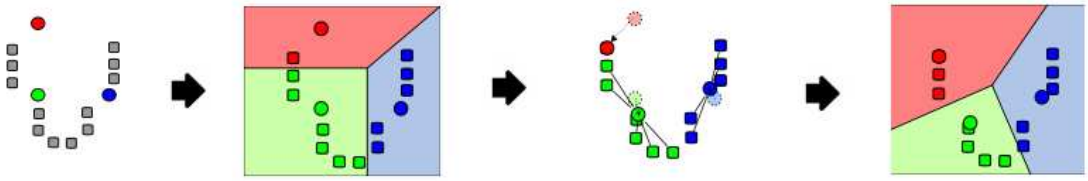


Figure B.2: Simple procedure of *k*-means with Voronoi diagram.

[7] is a special kind of decomposition of a metric space

Annex C

Naive Bayes Example

To demonstrate the concept of Naive Bayes classification, consider the toy example displayed in Figure C.1 [49]. As indicated, the objects can be classified as either green or red. The task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects.

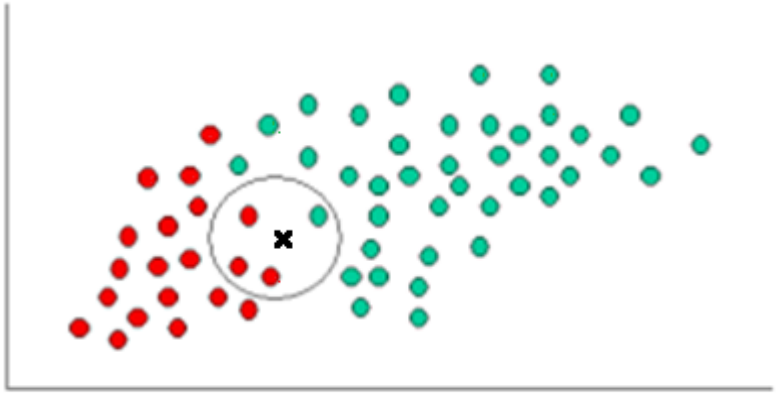


Figure C.1: Simple Naive Bayes classification [49].

Since there are twice as many green objects as red, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership green rather than red. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of green and red objects, and often used to predict outcomes before they actually happen.

Thus, the prior probability of the green and red objects can be written in following way:

$$Prior\ probability\ for\ green \propto \frac{Numbeer\ of\ green\ objects}{Total\ number\ of\ objects}$$

$$Prior\ probability\ for\ red\ \propto \frac{Numbeer\ of\ red\ objects}{Total\ number\ of\ objects}$$

Since there is a total of 60 objects, 40 are green and 20 are red. The prior probabilities for class membership are:

$$Prior\ pobability\ for\ green\ \propto \frac{40}{60}$$

$$Prior\ pobability\ for\ red\ \propto \frac{20}{60}$$

Having formulated the prior probability, the system is now ready to classify a new object (X). Since the objects are well clustered, it is reasonable to assume that the more green or red objects in the vicinity of X, the more likely that the new cases belong to that particular color. To measure this likelihood it is necessary to draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then is calculated the number of points in the circle belonging to each class label. From this the likelihood is calculated:

$$Likelihood\ of\ X\ given\ green\ \propto \frac{Number\ of\ green\ in\ the\ vcinity\ of\ X}{Total\ number\ of\ green\ cases}$$

$$Likelihood\ of\ X\ given\ green\ \propto \frac{Number\ of\ redin\ the\ vcinity\ of\ X}{Total\ number\ of\ red\ cases}$$

From the illustration above, it is clear that the likelihood of X given green is smaller than the likelihood of X given red, since the circle encompasses 1 green object and 3 red ones. Thus:

$$Probability\ of\ X\ given\ green\ \propto \frac{1}{40}$$

$$Probability\ of\ X\ given\ red\ \propto \frac{3}{20}$$

Although the prior probabilities indicate that X may

belong to green (given that there are twice as many green compared to red) the likelihood indicates otherwise; that the class membership of X is red (given that there are more red objects in the vicinity of X than green). In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes rule.

$$Posterior\ probability\ \ of\ X\ being\ green\ \propto Prior\ prob.of\ green\ \times Likelihood\ of\ X\ given\ green$$

$$= \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$

$$Posterior\ probability\ \ of\ X\ being\ red\ \propto Prior\ prob.of\ red\ \times Likelihood\ of\ X\ given\ red$$

$$= \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

Finally, the classifier classifies X as red since its class membership achieves the largest posterior probability.

Annex D

kd-tree

The *kd*-tree [67, 68] is a technique to improve the efficient information retrieval. Given a query point, a task of information retrieval is to find this query's neighboring data points. The brute force approach is to measure the distances (e.g. Euclidean distance) from this query to each of the data points [67]. The drawback of the brute force method is obvious: since its computational cost is $O(N \times d)$, where *N* is the memory size and *d* is the dimensionality of the input space. When the memory size *N* becomes very large, its costs will increase, too.

To improve the efficiency of finding the neighbors, we can partition the input space of the data points into many cells by means of a grid. When a query arrives, we can consult the cell where the query locates and its neighboring cells, instead of visiting all the data points individually.

The *kd*-tree technique is similar to the grid method in the sense that it also partitions the input space into many cells. However, the partition is flexible with respect to the density of the data points in the input space. Wherever, the density is high in the input space, the resolution of the *kd*-tree's partition at that region is also high, so that the cells tend to be small. Otherwise, for those regions where there are only a limited number of data points, the partition resolutions are low, and the cells are large [67].

kd-tree Construction

A *kd*-tree is a binary tree that recursively splits the whole input space into partitions, in a manner similar to a decision tree acting on real-valued inputs. Each node in the *kd*-tree represents a certain hyper-rectangular partition of the input space; the children of this node denote subsets of the partition. Hence, the root of the *kd*-tree is the whole input space, while the leaves are the smallest possible partitions this *kd*-tree offers. And each leaf explicitly records the data points that reside in the leaf. The tree is built in a manner that adapts to the local density of input points and so the sizes of partitions at the same level are not necessarily equal to each other [67].

To construct a tree from a batch of training data points in memory, it uses a top-down recursive procedure. This work uses the common variation of splitting a hypercube in the center of the widest dimension instead of at the median point. This method of splitting does not guarantee a

balanced tree, but leads to generally more cubic hyper-rectangles. The cost of making a tree from *N* data points is $O\left(N \times d \times log(N)\right)$

A sample *kd*-tree is shown in Figure D.1.

To implement the idea, it is necessary to use hyper-rectangles with *kd*-tree. To find the neighborhood of a certain query (triangle in Figure D.1), it is necessary a recursively search of the tree from the root towards the leaves where the query resides. For a different query (reversed triangle), it uses the same kd-tree but chooses different nodes.

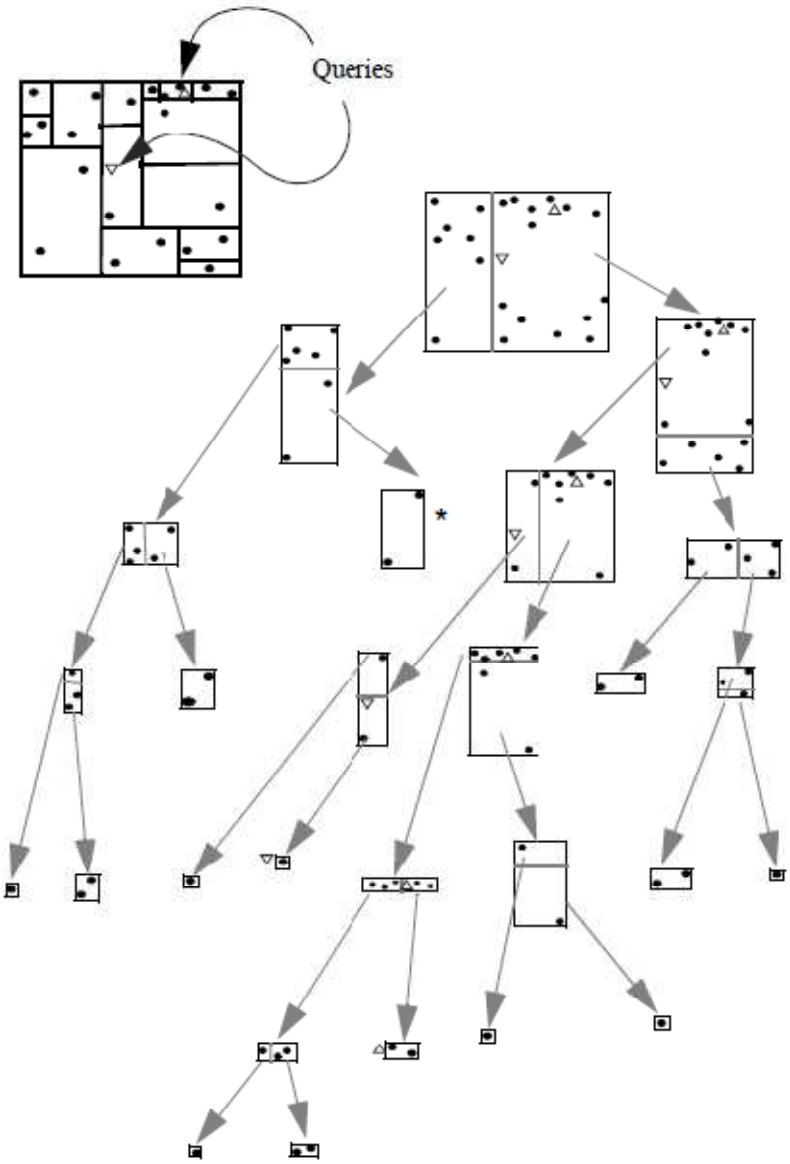


Figure D.1: A sample *kd*-tree [67].

Annex E

CBIR Interface

This section will explain how the user interacts with the CBIR system.

The CBIR system was developed in MATLAB [70] and the menus were created in MATLAB-GUI [71]. Firstly, the interface (Figure E.1) prompts the user to introduce a query image in a popup menu to get the data of the image - Figure E.2.

Then the user pushes the button ‘Procurar’ to initialize the classification – Figure E.3.

After some time (the system takes on average 2.2 seconds per class to classify the query image), a new popup menu appears to show pictures (in this case 12 images) of the similar class of the query example – Figure E.4.

Figure E.1: CBIR graphical interface.

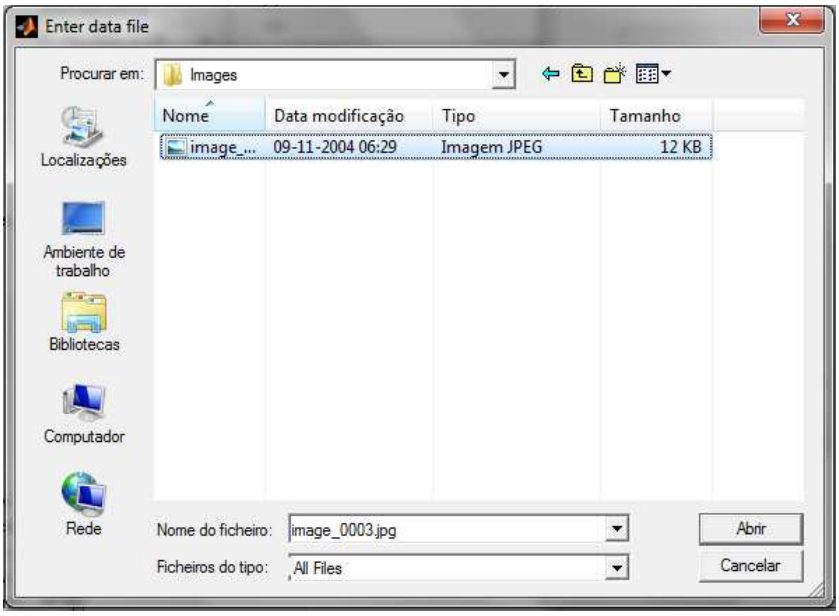


Figure E.2: Menu to introduce the image.

Figure E.3: CBIR prepared to classify the query image.

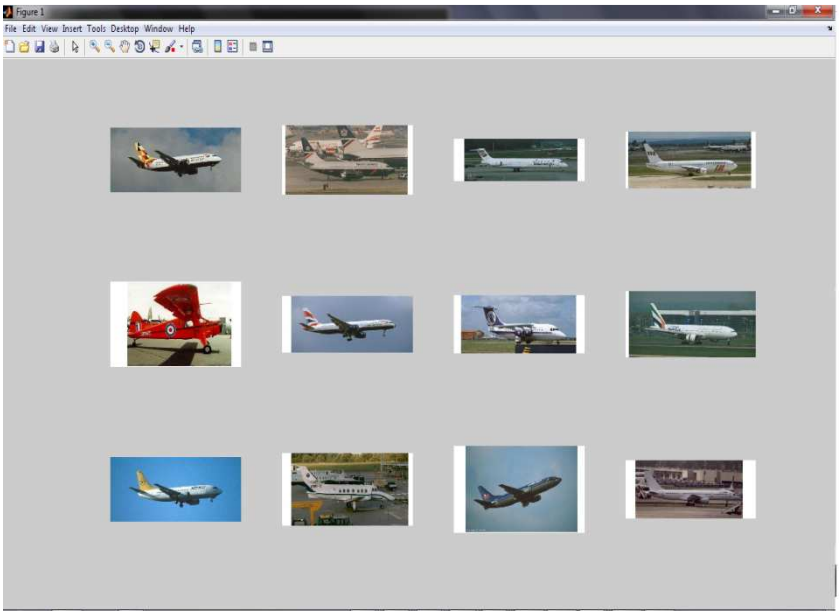


Figure E.4: Images of the similar class of query sample.

Annex F

Example of MATLAB processing

In the case of Table F.1, the system classifies a test image whose content represents a Leopard. The system ranked the closest class of the query, through smallest Euclidean distance. The nearest class in database corresponds to the class index 4, which in database index is associated to the Leopard class.

Table F.1: Example of a result processing.

Index	Class Index	Distance
1	4	409178944
2	49	418233440
3	31	426989408
4	94	427923360
5	74	430095552
6	39	431309216
7	51	431951776
8	55	436180128
9	99	439451200
10	56	440046016
11	46	442046880
12	14	442749248
13	83	443105920
14	12	443155264
15	29	444604992
16	25	445711328
17	70	445970058
18	53	446161536
19	60	446199168
...	...	...
101	102	570429888
102	7	598903360