FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Exploring Visual Content in Social Networks

Mariana Fernandez Afonso

MESTRADO INTEGRADO EM ENGENHARIA ELETROTECNICA E DE COMPUTADORES

Supervisor: Prof. Dr. Luís Filipe Pinto de Almeida Teixeira

July 23, 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"Exploring Visual Content in Social Networks"

foi aprovada em provas realizadas em 14-07-2015

o júri

Presidente Professora Doutora Maria Teresa Magalhães da Silva Pinto de Andrade
Professora Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor António José Ribeiro Neves
Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Professor Doutor Luís Filipe Pinto de Almeida Teixeira
Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Mariana Fernandez Afonso

Faculdade de Engenharia da Universidade do Porto

Abstract

The goal of this project was to develop and implement strategies to extract meaningful clusters from images shared in social media websites, more specifically on Twitter, using only features extracted from the images' content and not any associated tag or text. This represents a big challenge given that there is still no effective way to analyze and process visual content similar to the way humans process it.

Usually, features extracted from images are low-level, which can be, for instance, gradient histograms or color signatures. The challenge is how to use these features to find similar images in the social media. Many algorithms have been developed to try to solve this problem, but by far the most popular is the Bag-of-Features which takes the low-level features and builds a frequency histogram of visual words. However, there are some important limitations concerning this method, for instance, the fact that all the spacial information is lost and that color features are ignored. For this reason, apart from extensively evaluating this method for the purpose of image clustering, other approaches were also implemented and evaluated, namely, the Fisher Vectors, Spatial Pyramid Matching, the Bag-of-Colors and a combination of the Bag-of-Features and the Bag-of-Colors.

At first, before considering images extracted from the social media, three public labeled image datasets were used. The results of all the algorithms applied to the datasets were then compared and discussed. Finally, for the ultimate tests, 1000 images were obtained from Twitter. However, these images did not contain any labels and, for this reason, a user evaluation was conducted in order to discover a subset of pairs of similar and dissimilar images to assess the performance of the algorithms.

The results for the public datasets indicate that the Bag-of-Features model works well for simple object and scene datasets. Nonetheless, the Fisher Vectors and the Bag-of-Feature+Colors were able to outperform it. Moreover, for the Twitter dataset, these three methods achieved relatively acceptable results. This suggests that these methods are able to create a higher level representation which allows the clustering of images. However, there is still a long way to go until automatic algorithms are able to successfully analyze and summarize the content of the huge amount of images shared in the social networks.

Resumo

O objectivo deste projeto foi desenvolver e implementar estratégias para obter *clusters* relevantes a partir de images partilhadas nas redes sociais, mais espcificamente do Twitter, usando apenas características extraídas do conteúdo das imagens e não com os textos associados. Esta tarefa representa um desafio na medida em que ainda não foi desenvolvido um algoritmo para analisar e processar conteúdo visual de uma forma similar à realizada pelo Homem.

Normalmente, as características extraídas a partir das imagens são de baixo nível, como por exemplo, histogramas de gradiente ou padrões de cor. O desafio é então, como usar estas características para obter imagens similares nas redes sociais. Uma grande quantidade de algoritmos foi desenvolvido para tentar resolver esse problema, no entanto, o mais popular é o denomiado Bag-of-Feature em que cada imagem é representada por histogramas de frequência de palavras visuais. Contudo, este método possui algumas limitações, por exemplo, o facto de se estar a perder informação de posicionamento das características e o facto de se ignorar a cor. Por essa razão, além de explorar em detalhe este método, outros métodos foram também implementados e avaliados, nomeadamente, os Fisher Vectors, Spatial Pyramid Matching, Bag-of-Colors, e uma combinação entre o Bag-of-Features e o Bag-of-Colors.

Primeiramente, antes de considerar as imagens extraídas das redes sociais, três conjuntos de imagens públicos foram usados. Os resultados dos algoritmos aplicados a esses conjuntos foram então comparados e discutidos. Finalmente, para o teste final, 1000 imagens foram recolhidas do Twitter. Contudo, essas imagens não possuíam nenhuma anotação e desta forma, uma avaliação por utilizadores foi realizado para se poder avaliar a performance dos algoritmos.

Os resultados para os datasets públicos indicam que o método Bag-of-Features funciona bem para conjuntos de imagens de objects e cenas simples. Mesmo assim, os Fisher Vectors e o Bag-of-Features+Colors apresentaram resultados superiores. Additionalmente, para o cunjunto de imagens do Twitter, esses três métodos atingiram resultados relativamente aceitáveis. Desta forma, os resultados sugerem que esses métodos são capázes de criar representações de mais alto nível que possibilita a tarefa de clustering de imagens. No entanto, ainda é necessário o desenvolvimento de estratégias mais robustas e automáticas para analisar e sumarizar o conteúdo da enorme quantidade de imagens partilhadas nas redes sociais.

Acknowledgments

First, I would like to thank my supervisor, Professor Luís Teixeira for the guidance, discussions and notes on this thesis. His input was always very valuable and even with a substantial amount of work on this hands, he always tried to find time for helping me in anything that I needed.

This thesis would not have been possible without the support from my family, specially my father, Francisco Afonso, who even though had few knowledge about the topic, always provided encouragement and relevant suggestions. Also, my mother, Helena Fernandez, for reading my thesis, and my brother, Carlos Afonso, for helping me with the website for public evaluation.

I would also like to thank everyone that contributed for the public evaluation including my friends Edgar Costa, Tiago Cunha, Diogo Serra, Eduardo Fernandes, Peter Cebola, Catarina Acciaioli, Álvaro Ferreira, Rui Graça and Luciano Ferraz.

Finally, a special thanks to my boyfriend Filipe Silva for providing me with emotional support throughout this period.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

ARI	Adjusted Rand Index
BIRCH	Balanced iterative reducing and clustering using hierarchies
BRIEF	Binary Robust Independent Elementary Features
BoC	Bag-of-Colors
BoF	Bag-of-Features
BoF+C	Bag-of-Features+Colors
CBIR	Content-Based Image Retrieval
DBSCAN	Density-based spatial clustering of applications with noise
FAST	Features from accelerated segment test
FV	Fisher Vectors
NMI	Normalized Mutual Information
ORB	Oriented BRIEF
SIFT	Scale-invariant feature transform
SPM	Spatial Pyramid Matching
SURF	Speeded Up Robust Features

Chapter 1

Introduction

1.1 Problem Statement and Motivation

With the emergence of social media websites like *Twitter*, *Facebook*, *Flickr* and *YouTube*, there has a been a huge amount of information produced everyday by millions of users. *Facebook* alone reports 6 billion photo uploads per month and *Youtube* sees 72 hours of video uploaded every minute [1]. The truth is that we have experienced an explosion of visual data available online, mainly shared by users through the social media websites.

The information shared in the social media websites can have many formats such as text, images or video. Until recent years, there has been a research focus on the analysis and extraction of relevant information from text content. This is because text is easier to analyze and categorize than the other two formats. However, ignoring the visual content shared in the social media could be seen as a waste of important information for research.

Therefore, there is a need to develop more robust and more powerful algorithms for the analysis of images from large image collections. The big issue with visual content is that the features which can be extracted directly from the image, called low-level, do not give information about the content or high-level concepts present in an image. For that reason, it is extremely difficult to compare images in a way that is understandable and acceptable to humans. Consequently, this is an area of great potential for research.

The possibilities for applications of those types of systems are endless and could include safety, marketing and behavioral studies. This could also provide a different user experience when searching for information on a given topic. For example, if a Twitter user would like to search for images from a given event, he/she could receive image results which does not necessarily contain the keywords he specified because the search would be based on the content of the images instead of the metadata associated with them.

1.2 Objectives

The goal of this thesis is to find patterns in images shared via the social network Twitter. These patterns will be found using a technique called clustering, where given a set of unlabeled data, groups are formed based on the similarity of the content. This means that the each cluster will represent a pattern or concept, which could depict, for example, a location, an activity or a photographic trend (e.g. "selfies").

More specifically, the first objective is to study the different strategies and algorithms for image description and image clustering. A similar and more widespread field is Content-Based Image Retrieval (CBIR) which will be explained in detail in section 2.3. It is relevant to understand the most recent trends and approaches in these fields in order to develop something scientifically up to date.

After the theoretical study, several algorithms will be implemented. Following this first implementation, there will be a rigorous evaluation phase with the use of publicly available image collection databases. This evaluation will provide useful information concerning the suitability of those algorithms for the purpose of image clustering. Then, the algorithms implemented will be tested using images extracted from Twitter.

Next, once the clusters are obtained, there needs to be a visual representation of the clusters. In order to do so, algorithms for the visualization of image collections, will be studied and presented in subsection 2.6.

Finally, there will be an attempt at a partial or full integration with TweeProfiles [2].

In sum, the goal of this thesis is to be able to distinguish patterns or groups of interest in the huge image database provided from the user's *tweets*.

1.3 Project Overview

The task of clustering images from Twitter involves a number of different steps from the extraction of the data to the presentation of the clusters. These steps can be summarized into three modules, which were all implemented during this work.

Figure 1.1 presents a diagram that illustrates the different modules involved in an ideal process of image clustering applied to social media content. First, there is the image acquisition module, which is responsible for extracting, downloading and filtering visual content from Twitter. Next, the images pass to the image representation module which transforms the image data into useful and discriminating features. Each image is represented as a D-dimensional feature vector. Lastly, these features are fed to the clustering and visualization module that use the features to obtain the final clusters and present them. In this work, module 1 is separated from modules 2 and 3. Therefore, the images are first extracted and then processed offline.

The biggest challenge is the development of the image representation module, because if the images are transformed into feature vectors which are not discriminative, the clustering algorithm will not be able to obtain meaningful clusters. However, before applying different strategies to

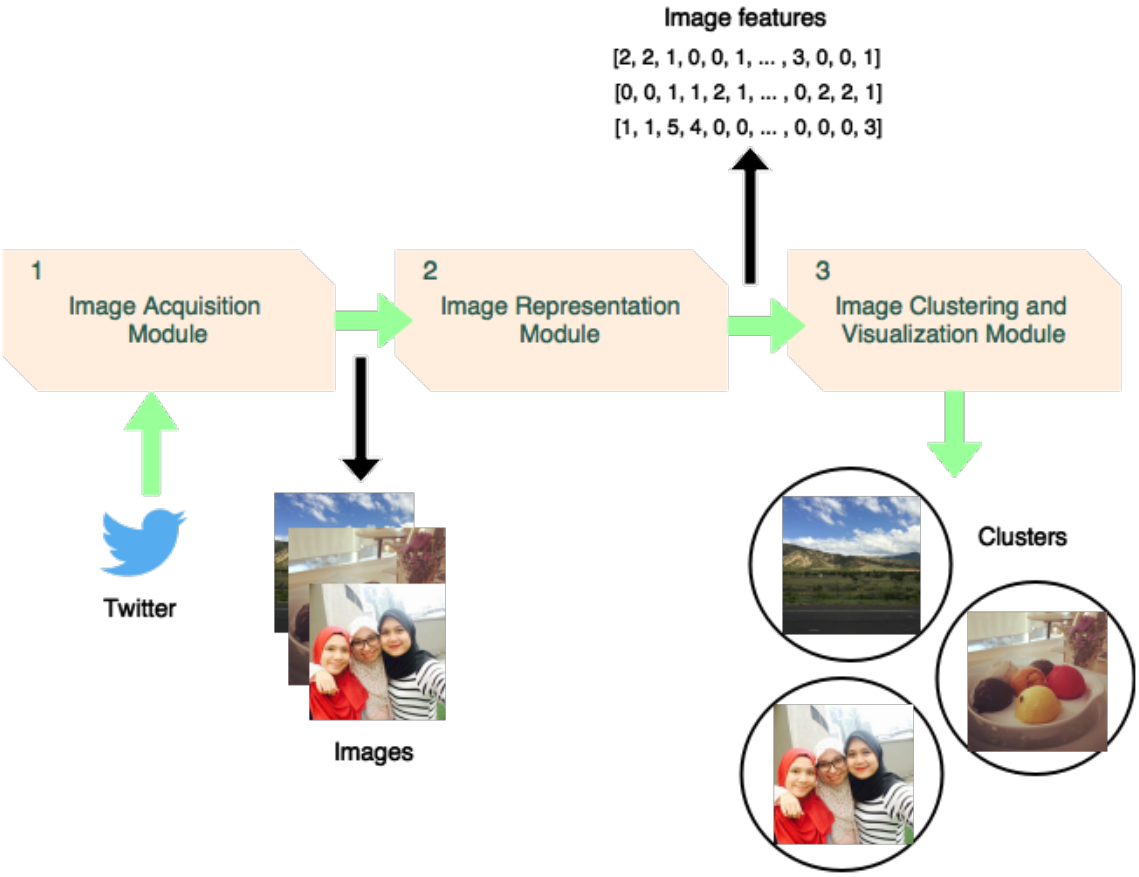


Figure 1.1: Diagram showing the different modules involved in the process of obtaining clusters from the image collected from Twitter.

the images extracted from Twitter, public image datasets were used. These datasets are already labeled, which means that each image has a class or group associated with it.

1.4 Contributions

The first major contribution of this thesis was the development and implementation of Python scripts, which are available open-source, of five different methods for image clustering. These methods can be easily applied to a dataset by calling a single Python script from the command line specifying the algorithm to use and all of its parameters. It is important to note that this enables researchers to test these methods on their own datasets using any parameters.

Additionally, apart from the classical clustering techniques like K-Means or Hierarchical Clustering, the program also contains the implementation of another clustering algorithm, based on graph theory and community detection. This approach is introduced in Section 3.2 and is very useful, since there is no need to choose any parameter such as the number of clusters or a maximum distance. Also, after applying this algorithm, each cluster can be represented by the image with the highest centrality-based measure.

Also, this thesis presents detailed evaluations of all the algorithms implemented on three public datasets and on a dataset obtained from Twitter. This is relevant since, to the best of our knowledge, most of these methods were not previously tested for the purpose of image clustering.

Finally, a paper was submitted for review to the British Machine Vision Conference (BMVC) in May with the results of the first study, which is presented in Section 3. The author notification is scheduled for July.

1.5 Document Structure

In relation to background and similar work, Chapter 2 will describe many important concepts and areas related to this subject. First, in Section 2.1, a small background on social network mining will be given. Also some popular research areas in that field will be presented in order to give some idea of the applications and research trends related to social networks. After that, some concepts of data mining will be presented in Section 2.2. The main focus will be in the area of clustering algorithms and clustering evaluation. Next, in Section 2.3, a very important area of research called Content-Based Image Retrieval (CBIR), which is related to the topic of this thesis, will be presented. Following that, section 2.4 will describe algorithms for image description, which is a way of extracting features from visual content. After that, in Section 2.5, some ideas for performance evaluation will be presented, based on the evaluation methodologies used in similar works. Next, Section 2.6 presents some papers in the topic of visualization of image collections. Lastly, in Section 2.7, some recent and similar work will be presented. These works will give more insight on the best approach to follow in this thesis.

Concerning the development of this work, Chapter 3 will analyze in detail the Bag-of-Features model due to its popularity, on public image datasets. Then, Chapter 4 will explore other algorithms developed thought the years as alternatives or extensions to the Bag-of-Features model, also using public datasets. Next, Chapter 5 will introduce the image acquisition module and present the final results of the application of the algorithms to images obtained from Twitter.

Finally, Section 6 will present the conclusions obtained from all the studies and concerning the overall achievements of this work in relation to the set of objectives designed in the beginning of the project and Section 6.2 will describe suggested future work for anyone who wishes to continue to study and develop further on this topic.

Chapter 2

Concepts, Background and Related Work

2.1 Social Network Mining

Social networks have become very popular over the last couple of decades. During this technological breakthrough, many social networks have appeared such as Facebook, Twitter or Flickr. In general, a social network is defined as [3]:

Definition 1. *A network of interactions or relationships, where the nodes consist of actors, and the edges consist of the relationships or interactions between these actors.*

According to the statistics in [4], Facebook has the largest number of users, 1.39 Billion monthly active users, from which 890 million are daily active users as of the present date. On the other hand, Twitter has 288 million active users and 500 million *tweets* are sent every day [5]. Many such social networks are extremely rich in content, whether it has the form of text, images or other multimedia data. The amount of content that is being shared in these social networks provides huge opportunities from the perspective of knowledge discovery and data mining.

According to [3], there are two types of primary kinds of data which are often analyzed in the context of social networks:

- **Linkage-based and structural analysis:** analysis of the linkage behavior of the network in order to determine important nodes, communities or links, and to study relationships between the different elements involved. For example, the verification of the *small world phenomenon*, *preferential attachment*, and other general structural dynamics has been a topic of great interest in recent years.

- **Content-based analysis:** describing, analyzing, linking and classifying the content shared in social networks. An example is text mining, which is used to discover useful patterns and information from the text shared by users in social networks, which can have wide variety of application.

The next sections will introduce the social networks Twitter and Instagram, present some interesting research areas within the topic of social network mining, and finally introduce a system called TweeProfiles [2]. As mentioned before, this work intends to be a contribution for an extension of TweeProfiles in order to allow the mining of visual content.

2.1.1 Twitter

Twitter is an online social networking used primarily as a microblogging service. Unlike on most online social networking sites, such as Facebook or MySpace, the relationship of following and being followed requires no reciprocation [6]. A user can follow any other user, and the user being followed does not need to follow back. Being a follower on Twitter means that the user receives all the messages from those the user follows. Users share messages called *tweets*, which have a maximum length of 140 characters. Additionally, the *retweet* mechanism empowers users to spread information of their choice beyond the reach of the original *tweet*'s followers.

Twitter users can also share other type of content such as pictures. In the *tweet* box, users can add up to 4 images, which will be converted to a *pic.twitter.com* link in the actual *tweet* sent (although it displays the image instead of the link).

Twitter has been the data source for many interesting studies for various types of applications. For instance, the authors of [6] studied the topological characteristics of Twitter and its power as a new medium of information sharing. For this work, they collected 41.7 million user profiles, 1.47 billion social relations, 4,262 trending topics and 106 million *tweets*. Other example is [7], where a hashtags retrieval system was built, where hashtags were obtained based on users interest expressed in the user's *tweets*. Content analysis was also performed in many researches (refer to section 2.1.6).

2.1.2 Instagram

Instagram is a social network oriented to mobile photo-sharing and video-sharing. It enables users to take pictures, make videos and share them in a number of other different social network platforms including Facebook, Flickr and Twitter. One appealing aspect of Instagram is that users can apply digital filters to the photographs taken. Also, all the images shared are converted to a square shape. As of December 2014, Instagram reported to have 300 million users accessing the site per month [8].

In recent years, several studies have focused on analyzing visual information from Instagram. The authors of [9] investigated the way a culture manifests itself through its visual production on Instagram. On the other hand, [10] attempted to characterize the different categories of images shared to Instagram and the major types of users.

2.1.3 Research Directions and Trends

Next, three fields of research in the area of social network mining are presented. The first is community detection, where the relationships in the social network form a graph that can be partitioned

in order to form groups or communities. The second topic is information cascade and influence maximization, where the objective is to discover influential users in the social network. Finally the last one is opinion mining and sentiment analysis which studies the subjective information present in the text content shared by users.

2.1.4 Community Detection

Relationships in social networks can be described by interconnected nodes that form complex networks or graphs. In order to obtain information on the structure of those networks, a strategy used is to decompose the network into sub-networks or communities, which are groups of highly connected nodes. This is referred as the problem of community detection [11]. Community detection is also applied in other types of networks including computer networks, information networks and biological networks

Community detection requires an algorithm to partition the network in a way that the nodes inside each community are highly interconnected and the nodes belonging to different communities are sparsely connected.

Many algorithms have been proposed for this problem, but generally they can be divided into three categories: divisive (starts with the entire network and then remove weak links), agglomerative (merges similar nodes and communities) and minimization of an objective function.

The quality of the detected partition is usually measured in terms of the modularity (Q), which is a measure that varies from -1 to 1 and can be computed as follows:

$$Q = \frac{1}{2m}\sum_{i,j}[A_{i,j} - \frac{k_i k_j}{2m}]\delta(c_i, c_j) \tag{2.1}$$

Where $A_{i,j}$ is the weight of the edge between nodes i and j (in case of a weighted graph), k_i is the sum of the weights from the node i, c_i is the community of the node i, m is half of the sum of all weights in the network and function $\delta(u,v)$ is 1 if $u = v$ or 0 otherwise.

Next an efficient algorithm for community detection will be presented. It is called the Louvain Community detection algorithm and was introduced in [11]. This algorithm finds high modularity partitions in a short time. It is composed by two phases that are repeated iteratively that are presented next:

1. First assign a different community to each node of the network. For each node i of the network, compute the improvement in terms of modularity by adding node i to the community of each of its neighbors j and adds the node to the community with the maximum gain. One of the reasons for the efficiency of this algorithm is the fact that the gain in modularity is easy to compute.

2. Next, the algorithm builds a new network whose nodes are the communities found during the first stage. The weights between the communities are computed as the sum of the weights of the links between the nodes of the communities. Also, the sum of the weights of the edges from the same community are originate self-loops.

These steps are then iterated until there are no more changes and a maximum of modularity has been achieved. Figure 2.1 illustrates the different steps of the algorithm. In the first iteration, the network is reduced to 4 communities, then in the final iteration it becomes only 2.

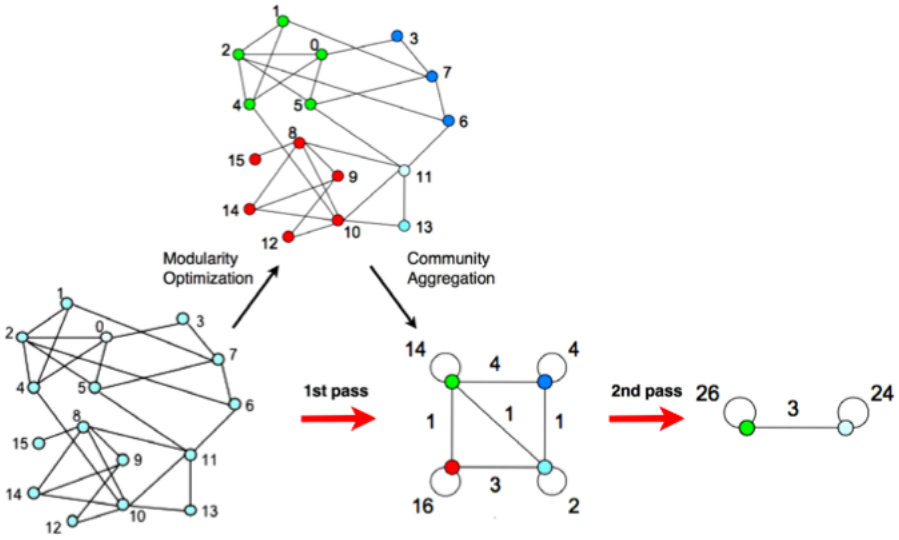


Figure 2.1: Illustration of the steps of the Louvain algorithm for community detection. Extracted from [11].

This algorithm was validated in [11] and compared to a number of different community detection algorithms which proved its efficiency and performance.

2.1.5 Cascade and Influence Maximization

In marketing, companies attempt to promote ideas or behaviors, within a population of individuals. One way to do this could be to target key individuals by offering them free samples of the product or explaining an idea. That individual would then share this idea, engaging others. The question in this situation is whom the companies should target. The answer should be to target the most influential individuals. Therefore, this approach is based on the premise that targeting a few key individuals may lead to strong cascade of recommendations. This is known as the *influence maximization problem* [12].

Online social networks provide good opportunities to address this problem, because they are connecting a huge number of people and they collect a huge amount of information about the social network structures and communication dynamics. However, they also present challenges due to the large scale, complex connection structures, and also variability [13].

A simple way of obtaining the most influential nodes is to compute *centrality*-based measures, which identifies the most important nodes in a graph [14]. There are several centrality-based measures divided into different categories. The most popular ones are: *degree centrality*, *closeness centrality*, *betweenness centrality* and *eigenvector centrality*. The *degree centrality* of a node is the simplest and is measured by the sum of the number of edges that this node has. The more edges, the

more important a node is considered in terms of this measurement. Next, the *closeness centrality* of a node is defined as the inverse of the sum of the distances to all other nodes. That means that the most central node is the one with the smallest distances to all other nodes. *Betweenness centrality* measures the number of times a node acts as a bridge along the shortest path between two other nodes. Finally, the *eigenvector centrality* assumes that each node's centrality is the sum of the centrality values of the nodes that it is connected to. It computes the eigenvectors for the largest eigenvalue of the adjacency matrix of the graph.

2.1.6 Opinion Mining and Sentiment Analysis

Opinions have always been a fundamental part of human relationships. Nowadays, opinion content is present almost everywhere and it is of easy access to almost anyone due to the popularity of the Internet and other media such as TV or music. This availability in data creates an opportunity for information technologies to produce systems that can automatically detect and evaluate opinions.

Opinions and its related concepts such as sentiments, evaluations, attitudes, and emotions are the subjects of study of sentiment analysis and opinion mining [15]. In particular, opinion mining can be defined as a sub-discipline of computational linguistics that focuses on extracting people's opinion from the Web. Sentiment analysis, on the other hand, is about determining the subjectivity, polarity (positive or negative) and polarity strength (e.g. weakly positive, mildly positive, strongly positive) of a text message [15]. The concepts of opinion mining and sentiment analysis usually work together in determining the nature of an opinion.

An interesting project in this area is REACTION (Retrieval, Extraction and Aggregation Computing Technology for Integrating and Organizing News) which is being developed by several Portuguese institutions. One of the objectives of the project is the automatic analysis of content from social networks. The system named Twitómetro attempts to collect and infer polarity on *tweets* related to politics personalities during elections.

2.1.7 TweeProfiles

TweeProfiles [2] is a project started in 2012 that aims to identify *tweet* profiles or trends from user collected data. The profiles integrate multiple types of data, namely, spacial, temporal, social and content. Therefore, the goal is to find interesting patterns and clusters using a combination of different types of information extracted from a Twitter community, called dimensions. Another goal of this project is to develop a visualization tool that would allow the effective displaying of the different patterns.

The data collected and used by TweeProfiles was obtained using SocialBus [16]. SocialBus is a Twitter crawler that collects data from a particular user community for research purposes. It an open-source project with the objective to promote research in the area of social network analysis. Researches can, therefore, collect their own data focusing on different communities of users, choosing different factors such as geographic, demographic, linguistic or even content-based characteristics.

In terms of the implementation of TweeProfiles, it uses data mining methods in order to accomplish the goals mentioned earlier. First, it prepares the data by filtering the tweets that have spacial information (latitude and longitude) and extracts the social relationships between the users that sent the *tweets* processed. After all the data is collected and prepared, dissimilarity matrices are computed for each dimension. For each dimension, a different dissimilarity measure is used. The spacial distance is computed using the Haversine function [17]. For the temporal distance, a timestamp difference is chosen. As for content, the cosine dissimilarity is used (please refer to section 2.2.1.1). Finally, the social distance is calculated using the graph geodesic [18] (the distance between two vertices in a graph, which is the number of edges in a shortest path) between the users in the social graph constructed earlier. A min-max normalization function was applied to all matrices. For combining the matrices of each dimension, weights can be set by the user. This allows to perform and evaluate different combinations of dimensions and obtain different results based on these weights.

After the dissimilarity matrix is computed, clustering is performed. Among all the possible clustering algorithms, the one chosen is DBSCAN (please refer to section 3.4.4).

In the visualization tool each cluster is represented in the map by a circular shape that roughly represented the cluster locations, a time interval (time of the first and last tweet of that cluster), the most frequent words used (the content dimension) and a mini social graph.

An example of the visualization of different clusters obtained using multiple values of the weights is presented in Figure 2.6.

Figure 2.2: Clusters in Portugal; Content proportion 100%.

Figure 2.3: Clusters in Portugal; Content 50% + Spacial 50%.

Figure 2.4: Clusters in Portugal; Content 50% + Temporal 50%.

Figure 2.5: Clusters in Portugal; Content 50% + Social 50%

Figure 2.6: Visualization of the clusters obtained by TweeProfiles using different weights for the different dimensions. Extracted from [2].

Due to lack of time and complexity of the task, there was little attempt to perform evaluation on the clusters obtained. Also, the only content used to find patterns was text, which can be restrictive given that, as mentioned before, Twitter users can also share other types of content. For this and other reasons, many other works have been developed since, having the initial version of TweeProfiles as the reference. This thesis will be one more contribution towards improving TweeProfiles.

2.2 Data Mining Techniques and Algorithms

Generally, data mining is the process of analyzing data from different perspectives and summarizing it into useful information that can be used, for example, in the business field to increase revenue, cuts costs, or both. Algorithms for data mining have a close relationship to methods of pattern recognition and machine learning.

Data mining techniques and algorithms can be separated into a number of different categories, each one with a different purpose [19]:

- **Anomaly detection (or outlier detection):** the identification of unusual data records. Typically the anomalous items will translate to some kind of issue such as bank fraud, medical problems or errors in text [20];
- **Association rule learning:** the discovery of relevant relationships between variables. Product placement inside a commercial establishment is an example of application of association rules. In that case, the goal is to discover products bought usually together in order to make decision about where to presents those products [21];
- **Classification:** a supervised learning method used to predict the label or class of an unseen dataset (test set) based on the information collected from a known dataset (training set). Supervised in this context means that the examples contained in the datasets are labeled, and therefore, there is a reference for training and evaluation [22];
- **Regression:** the method of finding the best function to model a given dataset. It is similar to classification apart from the fact that there is no concept of classes but of target variables that can take any real or numerical value [22];
- **Clustering:** an unsupervised method which consists of finding groups and similarities in the examples of the dataset. In a clustering problem, there are no labels nor classes already present in the data, so the goal is to be discover possible organizations or patterns [22];
- **dimensionality reduction:** where the goal is to reduce the number of dimensions of the feature vectors of a dataset in order to improve the performance of the algorithms for regression, classification or clustering. There are two types of dimensionality reduction techniques: feature selection and feature extraction. In feature selection, some of the variables (features) are considered irrelevant, and therefore are excluded, while in feature extraction, features are combined to create more powerful features for the selected purpose [22].
- **Evaluation:** the task of accessing the performance of a model, which could be a clustering results, a classification result, among others [22].
- **Summarization:** which goal is to extract information a source and present the most important content to the user in a condensed form and in a manner sensitive to the user's application's needs [23].

The next section will cover some algorithms used for clustering, since it is the most important concept used for this thesis. The description will cover simple and classic algorithms as well as some more complex and recent approaches.

2.2.1 Clustering

Clustering is one of the human's primitive mental activities and is part of their learning process [22]. It is used to handle the huge amount of information received everyday by defining basic categories (e.g. animal, kitchen, school). In a similar way, computers can apply clustering algorithms to collected data to reveal hidden information and structures.

Before applying a clustering technique, the data needs to be prepared by defining the features which will be used. The term features denotes the characteristics or variables that describe each example in the dataset. For example, if the dataset is formed by animals in a Zoo, the features collected could be the size of the animal, the color of the fur, the life expectancy, etc. If the features are not carefully selected, in order to have a minimum redundancy and a maximum information encoded, the clustering task will be less efficient [22].

2.2.1.1 Proximity Measures

The proximity measures define how two sets of points or sets in a dataset are similar or dissimilar to each other. A similarity measure quantifies how close two elements of the dataset are in relation to its features. In contrast, the dissimilarity measures evaluate how far apart they are.

Next some examples of popular similarity and dissimilarity measures will be presented.

Starting by the dissimilarity measures, some of the most popular measures can be generalized using the weighted Minkowski distance of order m. Using this measure, the dissimilarity between two points $\mathbf{x} = (x_1, \ldots, x_l)$ and $\mathbf{y} = (y_1, \ldots, y_l)$ in the dataset are computed as follows:

$$d_p(\mathbf{x}, \mathbf{y}) = (\sum_{i=1}^{l} w_i |x_i - y_i|^p)^{\frac{1}{p}} \tag{2.2}$$

Where the following dissimilarity measures can be obtained by changing the values of p and w_i:

- Euclidean distance ($p = 2$ and $w_i = 0$):

$$d_e(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{l} (x_i - y_i)^2} \tag{2.3}$$

  The Euclidean distance is the most common measure of distance between two points. It can be extended to any number of dimensions and represents the smallest distance between two points in an Euclidean space [24].

- Weighted Euclidean distance ($p = 2$ and $w_i \neq 0$):

$$d_{we}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{l} w_i (x_i - y_i)^2} \tag{2.4}$$

- City-block or Manhattan distance ($p = 1$ and $w_i = 0$):

$$d_1(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{l} |x_i - y_i| \tag{2.5}$$

Other dissimilarity measures include:

- Chebyshev distance:

$$d_\infty(\mathbf{x},\mathbf{y}) = \max_{1 \leq i \leq l} |x_i - y_i| \tag{2.6}$$

- Mahalanobis distance, where B is the inverse of the within-group covariance matrix.

$$d_m(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T B (\mathbf{x}-\mathbf{y})} \tag{2.7}$$

If the covariance matrix, B, is the identity matrix (uncorrelated variables with unit variance), the Mahalanobis distance reduces to the Euclidean distance. If the covariance matrix is diagonal, then the resulting distance measure is called a normalized Euclidean distance.

$$d_{ne}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{l} \frac{(x_i - y_i)^2}{s_i^2}} \tag{2.8}$$

The Mahalanobis distance accounts for the variance of each variable and the covariance between variables. Geometrically, it does this by transforming the data into standardized uncorrelated data and computing the ordinary Euclidean distance for the transformed data [25]. Thus, it provides a way to measure distances that takes into account the scale of the data.

In relation to similarity measures, some of the commonly used include:

- Inner product:

$$s_{inner}(\mathbf{x},\mathbf{y}) = \mathbf{x}^T \mathbf{y} \tag{2.9}$$

The inner product is a basic similarity function between two points. If x tends to be high where y is also high, and low where y is low, the inner product will be high, and therefore the vectors are more similar.

- Cosine similarity measure:

$$s_{cosine}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{||\mathbf{x}||\ ||\mathbf{y}||} \tag{2.10}$$

Where $||\mathbf{x}|| = \sqrt{\sum_{i=1}^{l} x_i^2}$ and $||\mathbf{y}|| = \sqrt{\sum_{i=1}^{l} y_i^2}$. Since the inner product is unbounded, one way to make it bounded between -1 and 1 is to divide by the vectors' norms, giving the cosine similarity. If x and y are non-negative, it is actually bounded between 0 and 1. The cosine similarity is a measure of the cosine of the angle between the two points or feature vectors. It is thus a judgment of orientation and not magnitude. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors, as only the non-zero dimensions need to be considered [26]. It is widely used in retrieval systems where each data is represented by a vector of frequencies (for example, of words in text mining), so that the cosine similarity measure will give a good estimate of how similar the examples are without being too expensive computationally [27].

- Pearson's correlation coefficient:

$$s_{person}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}_d^T \mathbf{y}_d}{||\mathbf{x}_d||\ ||\mathbf{y}_d||} \tag{2.11}$$

Where $\mathbf{x}_d = [x_1 - mean(x), ..., x_l - mean(x)]^T$ and $\mathbf{y}_d = [y_1 - mean(x), ..., y_l - mean(y)]^T$

The cosine similarity is not invariant to shifts. If x was shifted to $x+1$, the cosine similarity would change. This problem is solved by using Person's correlation, which is the cosine similarity between centered versions of x and y, again bounded between -1 and 1 [26].

Additionally, there are also distance measures between vectors having discrete or binomial values, but this will not be covered in this document.

To conclude, there are many proximity measures that could potentially be used for clustering analysis. Overall, the choice of the best one depends largely on the application.

2.2.1.2 Clustering Algorithms

In the previous subsection, some popular proximity measures where presented. Each of these measures gives a different interpretation of the terms dissimilarity and similarity associated with the type of clusters one wants to obtain. Next, a number of popular clustering algorithms will be explained.

2.2.1.3 Centroid-based Clustering: K-Means

Also called partition-based algorithms, these types of algorithms construct a partition of a database D of n objects into a set of k clusters. Some domain knowledge is required since an input parameter k is required for these algorithms, which unfortunately is not available for many applications. The partitioning algorithm typically starts with an initial partition of D and then uses an iterative process to optimize an objective function.

K-Means [28] is one of the simplest unsupervised learning algorithms for clustering. It minimizes the within-cluster variance. The number of clusters k is fixed beforehand. The main idea is to define k *centroids*, one for each cluster. The *centroids* are initialized randomly in the feature

space or can be chosen to be coincident with k data points. The next step is to take each data point and associate it to the nearest *centroid* based on the Euclidean distance 2.2.1.1). When all the points are assigned to the nearest *centroid*, the k *centroids* are re-calculated as the mean of the feature values of the data points assigned to it. The procedure continues until there are no changes in the clusters obtained in each iteration or a stopping criteria is reached.

Using a different notation, given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a d-dimensional real vector, K-Means clustering aims to partition the n observations into $k(\leq n)$ sets $S = S_1, S_2, \ldots, S_k$ so as to minimize the within-cluster sum of squares (WCSS). This, its objective is to find:

$$arg_{max_s} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{2.12}$$

Where μ_i is the mean of points in S_i.

An example of the K-Means clustering can be found in Figure 2.7. In this 2-D dataset, the value used for k was 2 and the algorithm converged to the 2 centroids represented by the red X's. The clusters were then colored with green or blue. In this example, the results were perfect due to the nature of the data (two generated Gaussian sets with a well separated mean). When working with real data, the results are usually not expected to be ideal.

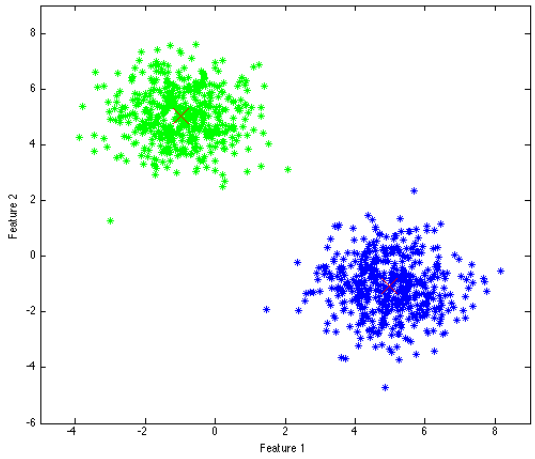


Figure 2.7: Example of K-Means clustering algorithm in a 2-D dataset.

One of the biggest problems with the K-Means clustering algorithm for unsupervised learning is the fact that the value of k (number of clusters) needs to be provided by the user, when in most of the situations this values is not known. One method generally applied is to do the K-Means algorithm using different values for k and comparing the results objectively or subjectively. Some attempts to solve this problem include the called X-means algorithm [29].

Other problems with the K-Means algorithm is that it scales poorly computationally and the search is prone to local minima depending on the initial seeds.

Finally, the shape of all clusters found by any centroid-based algorithm is convex which is very restrictive.

In spite of the problems and disadvantages of this method, and even though K-Means was proposed over 50 years ago when thousands of clustering algorithms have been published since then, K-Means is still widely used [30].

2.2.1.4 Connectivity-based Clustering: Hierarchical Clustering

Hierarchical clustering is a clustering method which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types agglomerative and divisive [22]. In the first type, each observation starts as its own cluster and then clusters are merged based on the similarity or dissimilarity measure chosen until there is only one cluster. Alternatively, the algorithm starts with only one cluster containing all the observation and then are splited until each observation becomes a different cluster. The results of a hierarchical clustering process are usually represented by a *dendrograms*, which is a type of tree diagram. Figure 2.8 represents two examples of *dendrograms* using the two types of hierarchical clustering algorithms.

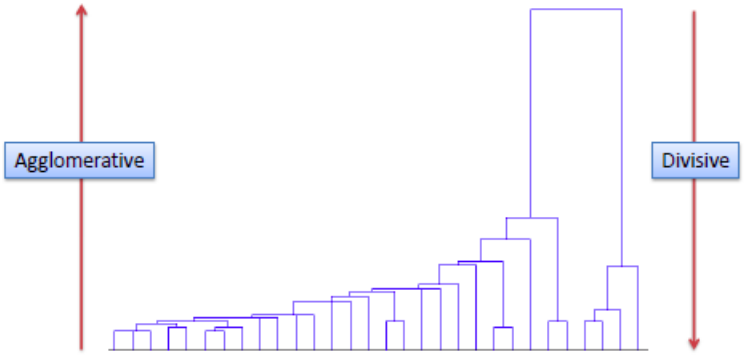


Figure 2.8: Example of a dendrograms for hierarchical clustering, using either agglomerative or divisive methods. Extracted from [31].

In order to decide which clusters should be combined (for agglomerative), or where a cluster should be split (for divisive), a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, this is achieved by use of an appropriate metric (proximity measures mentioned in section 2.2.1.1), and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets. That is, the criterion used to access distances between clusters. For example, if a cluster is composed by n observations and other is composed by m, there should be a way to compute the distance between these clusters using only a single distance measure between two observations.

The most popular linkage criterion are the following:

- single linkage clustering:

$$\min\{d(a,b) : a \in A, b \in B\} \tag{2.13}$$

In single-linkage clustering, the distance between two clusters is computed by a single element pair, namely those two elements (one in each cluster) that are closest to each other.

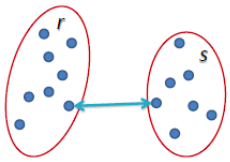


Figure 2.9: Illustration of single-link clustering. Extracted from [31].

- complete linkage clustering:

$$\max\{d(a,b) : a \in A, b \in B\} \tag{2.14}$$

In complete linkage, the distance between two clusters is computed as the distance between those two elements (one in each cluster) that are farthest away from each other. Complete linkage clustering avoids a drawback of the alternative single linkage method - the so-called chaining phenomenon, where clusters formed via single linkage clustering may be forced together due to single elements being close to each other, even though many of the elements in each cluster may be very distant to each other. Complete linkage tends to find compact clusters of approximately equal diameters.

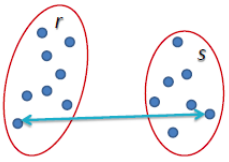


Figure 2.10: Illustration of complete-link clustering. Extracted from [31].

- average linkage clustering:

$$\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b) \tag{2.15}$$

Average linkage is a combination of the two previous methods. The distance between two clusters is the average of the distances between every pair of elements of each cluster.

To conclude, the type of linkage method is important and needs to be chosen considering the nature of the clusters which are to be obtained.

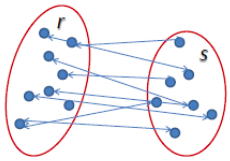


Figure 2.11: Illustration of average-link clustering. Extracted from [31].

The main problem with hierarchical clustering algorithms has been the difficulty of deriving appropriate parameters for the termination (or stopping) condition. The termination condition should find the best place to cut the dendrogram in order to separate all the important clusters and, at the same time, avoid splitting them in more than one cluster. Usually the stopping condition used is a threshold indicating that the clusters are too far apart to be merged (distance criterion) or is related to the number of clusters, when there is a sufficiently small number of clusters (number criterion).

2.2.1.5 Density-based clustering: DBSCAN

For density-based clustering algorithms, clusters are considered as regions in the l-dimensional space that are "dense" in data points [32]. Most of these algorithms do not impose restrictions on the shape of the clusters. In addition, they are able to hand efficiently outliers. Moreover, the time complexity of these algorithms is lower than $O(N^2)$, which makes them eligible for processing large datasets.

DBSCAN (Density-based Spatial Clustering of Applications with Noise) is an algorithm, proposed in [32] to overcome some of the issues concerning the clustering of large spacial databases. This algorithm requires only one input parameter and supports the user in determining an appropriate value for it. It can also discover clusters with arbitrary shape (e.g. spherical, linear, elongated).

The DBSCAN estimates the density around a data point p as the number of points in the dataset that fall inside a certain region in the l-dimentional space surrounding p. That is, the key idea of DBSCAN is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points, i.e. the density in the neighborhood has to exceed some threshold. The shape of a neighborhood is determined by the choice of a distance function for two points p and q, denoted by $dist(p,q)$.

According to DBSCAM, an EPS-neighborhood of a point p, denoted by $NEps(p)$, is the number of points belonging to the neighborhood of distance Eps to p. This is mathematically defined by:

$$NEps(p) = \{q \in D | dist(p,q) \leq Eps\} \tag{2.16}$$

Another important definition for this algorithm is the term directly density-reachable. A point p is directly density-reachable from a point q given the values of EPS and $MinPts$ if:

1. $p \in NEps(q)$ and
2. $|NEps(q)| \geq MinPts$ (core point condition)

Density-reachability is another relevant concept and is a canonical extension of direct density-reachability. A point p is density-reachable from a point q with respect to Eps and $MinPts$ if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$ and $p_n = p$ such that $p_i + 1$ is directly density-reachable from p_i.

Finally, the last concept is density-connection. A point p is density-connected to a point q with respect to Eps and $MinPts$ if there is a point w such that both, p and q are density-reachable from w using Eps and $MinPts$.

After those definitions, density-based notion of a cluster can be presented. Intuitively, a cluster is defined to be a set of density-connected points which has maximal density-reachability. Additionally, noise is the set of points in D not belonging to any of its clusters. More specifically, a cluster is defined as:

Let D be a database of points. A cluster C with respect to Eps and $MinPts$ is a non-empty subset of D satisfying the following conditions:

1. $\forall p,q : if\, p \in C$ and q is density-reachable from p wrt. Eps and $MinPts$, then $q \in C$. (Maximality)
2. $\forall p,q \in C$: p is density-connected to q wrt. EPS and $MinPts$. (Connectivity)

In order to use these definitions to find the clusters, the values of EPS and $MinPts$ are needed for every cluster. The problem is that there is no easy way to get this information in advance for all clusters of the database. However, there is a simple and effective heuristic presented in [32] to determine the parameters Eps and $MinPts$ of the smallest, i.e. least dense cluster in the database. The reason behind choosing the least dense cluster is that the clustering algorithm will find all the other clusters except for the ones with lower parameters, which will should correspond to the noise.

Figure 2.12: Clusters found by DBSCAN in 3 databases. Extracted from [32].

Figure 2.12 shows an example the results of the DBSCAN algorithm applied to some datasets with a variety of different shapes. The database 1 has rounded clusters that could be obtained using almost classic algorithm (e.g. K-Means). Database 2 has various shapes of clusters, which would

be very hard for a simple algorithm to detect. Nonetheless, DBSCAN can effectively obtain the "correct" clusters. Finally, database 3 has noise, which are ignored by DBSCAN.

A Pseudocode for the DBSCAN algorithm can be found in [32] for further insight.

2.2.1.6 Fuzzy Clustering: Fuzzy C-Means

Unlike the algorithms mentioned before, where the final clusters obtained are disjoint (or hard clusters), fuzzy clustering algorithms partition the dataset into non-disjoint clusters (or soft/fuzzy clusters) [33]. For this reason, a data element can belong to more than one cluster, and usually a membership level is associated to it. This family of clustering algorithms is based on the fuzzy logic theory [34].

Fuzzy clustering has been around for a long time [33] and many algorithms where designed. The most popular is the Fuzzy C-Means (FCM) algorithm, which will be explained next.

The goal of the FCM algorithm is to partition a set of n elements $X = \{x_1, \ldots, x_n\}$ into c fuzzy clusters, $C = \{c_1, \ldots, c_c\}$. In the Fuzzy C-Means algorithm, each data point has a degree of belonging to a certain cluster (named membership). This value can range between 0 and 1 and the sum of all the memberships in a given data example needs to be 1. They are represented by $w_{i,j}$ where i is the datapoint and j is the cluster. The FCM algorithm aims to minimize the following objective function [35]:

$$\arg\min_{C} \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^{m} \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2, \tag{2.17}$$

where:

$$w_{ij}^{m} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\left\| \mathbf{x}_i - \mathbf{c}_j \right\|}{\left\| \mathbf{x}_i - \mathbf{c}_k \right\|} \right)^{\frac{2}{m-1}}}. \tag{2.18}$$

which is a similar objective function to the K-Means clustering algorithm. The m variable is called the *fuzzifier* and determines the level of fuzziness of the clusters obtained. The minimal value of m is 1, for which the clusters are no longer fuzzy, but hard.

Given these membership degrees, $w_k(x)$ for each datapoint x in the dataset, the centroid of the kth cluster is obtained by computing the following expression:

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m}. \tag{2.19}$$

which is just the mean of all points weighted by the degree of belonging to the given cluster, excluding a parameter m.

As for the algorithm, it is very similar to the popular K-Means algorithm (sec. 2.2.1.3) [35]:

1. Select the number of clusters
2. Assign the coefficients of belong to each cluster randomly
3. Calculate the centroid of each cluster using equation 2.19

4. Compute, for each point, the new coefficients using equation 2.18

5. Repeat steps 3 and 4 until a stopping criteria is achieved

Fuzzy clustering has been successfully applied to image segmentation, text categorization, among others [36]. For a more a detailed description of other fuzzy clustering algorithms, please refer to [36].

2.2.2 Data Stream Clustering

Sometimes, the data obtained from a given source is generated and captured continuously and with high variability in data rate. This is the case for most of the data extracted from the Web (for example, through Twitter crawlers). The storage, querying and mining of data of this nature is challenging and complex. In order to obtain a fixed dataset, data has to be stored. This is considered to be "offline" processing. Recently, researchers have been focused on designing algorithms for data stream processing [37]. One of these types of tasks is clustering analysis. Here, instead of obtaining a single clustering result, the clusters are constantly changing as new data is added and old data is removed.

Next two algorithms for data stream clustering are presented, BIRCH and Mini Batch K-Means.

2.2.2.0.1 BIRCH The BIRCH clustering algorithm is one of the most popular algorithms for incremental clustering. BIRCH was introduced in [38] and is a type of hierarchical clustering algorithm that is efficient in the presence of large datasets. Its efficiency is based on many characteristics: each clustering decision is made without scanning all data points and currently existing clusters, it considers the importance of different data points to the overall clustering result and it is an incremental method which does not require the whole dataset in advance.

The BIRCH algorithm works by building a clustering feature tree (CF tree). Each element of the CF tree is a cluster and has three features: N (number of points), LS (linear sum of the elements) and SS (square sum of the elements).

The clustering results obtained depend on three parameters: the branching factor, the leaf threshold and the Threshold. The tree size depends on the value of T chosen since the larger this value, the smaller the tree obtained.

Overall, the BIRCH algorithm has several advantages including the fact that it is faster than algorithms such as K-Means for large datasets, it only scans the whole dataset once, it handles outliers better and has a high stability and scalability. However, it cannot handle non spherical clusters since it uses the notion of radius and diameter.

2.2.2.0.2 Mini Batch K-Means The Mini Batch K-Means algorithm was developed with the goal of reducing the computational complexity of the traditional K-Means algorithm for large scale applications. Nonetheless, it still aims to optimize the same objective function [39].

Unlike the K-Means algorithm, it uses mini-batches which are smaller subjects randomly sampled from input data. For this reason, the results of this algorithm are slightly worse than the classical K-Means.

The algorithm has two main steps: the assignment of the samples to the nearest centroid (at first the centroids are picked randomly from the samples) and update the centroids. One important difference from K-Means is that the centroids are updated with each sample considering all the previous samples that have already been assigned to it. These steps are performed until convergence or a predetermined number of iterations is reached.

For a better understanding, the algorithm for the Mini Batch K-Means algorithm is presented in Algorithm 1.

Algorithm 1 Mini Batch K-Means algorithm

Given: k, mini-batch size b, iterations t, data set X
Initialize each $c \in C$ with an x picked randomly from X
$v \leftarrow 0$
for $i = 1$ to t **do**
    $M \leftarrow b$ examples picked randomly from X
    for $x \in M$ **do**
        $d[x] \leftarrow f(C,x)$ ▷ Store the nearest center to x
    end for
    for $x \in M$ **do**
        $c \leftarrow d[x]$ ▷ Get the nearest center to x
        $v[c] \leftarrow v[c]+1$ ▷ Update per-center counts
        $\eta \leftarrow \frac{1}{v[c]}$ ▷ Get per-center learning rate
        $c \leftarrow (1-\eta)c+\eta x$ ▷ Update center
    end for
end for

2.2.3 Cluster Validity

As seen in the last section, clustering is a technique for unsupervised learning. This means that there are no true classes represented in the dataset. Therefore, the question now is how to evaluate the quality of the clusters found using the clustering algorithms. To answer this question, researchers have created techniques for what is known as clustering validation or clustering validity, which is the process of estimating how well a partition fits the structure underlying the data [40].

The objectives of cluster validity methods include determining whether non-random structure exists in the data, evaluating how well the results of a cluster analysis fit the data, comparing the results of two different sets of cluster results from two different algorithms and determining the best number of clusters present in the data.

The last objective mentioned refers to the fact that most clustering algorithms require a tuning of the input parameters to achieve optimal results. For example, the K-means algorithm, as seen before, requires the number of clusters as an input. A way of determining this parameter would

be to test with many different values of k and compute, for each a validity index, which would measure the quality of the partition. Therefore the clustering validity methods are a good way of determining the best input parameters to use for a certain clustering analysis.

In order to numerically measure different aspects for clustering validity, there are three main types of indexes or criteria:

- External Index: used to measure how well the clusters obtained match externally supplied class labels (which could not be available).
- Internal Index: used to measure the quality of the clusters obtained without respect to external information.
- Relative Index: used to compare two different clusters or clustering results.

In this section, the focus will be on the external and internal indexes, which are the ones generally used in unsupervised learning. The external indexes are applied when a benchmark dataset is available, otherwise, the internal indexes are used.

2.2.3.1 External Indexes

The external criteria measures the similarity of the clusters obtained against the class labels present in the dataset. Although there is a variety of external indexes available, but only 4 will be presented in the following. All of these indexes were described in [41].

The first external index is *purity*, which is computed by assigning each cluster to the majority class of the elements of that cluster, and then the accuracy is measured by counting the number of correctly assigned points and dividing by the total number of points N. Mathematically, purity is defined as:

$$purity(W,C) = \frac{1}{N}\sum_{k} max_j |w_k \cap c_j| \tag{2.20}$$

where $W = \{w_1, \ldots, w_k\{$ is the set of clusters and $C = \{c_1, \ldots, c_J\}$ is the set of labels.

Since purity is 1 if each data point has its own cluster, which means it is 1 if the number of clusters is N, purity cannot be a quality measure for defining the optimal number of clusters.

The next widely used external index, is *Normalized Mutual Information* (NMI), which can have many forms depending on the upper-bound chosen [42]. One of these measures is computed by:

$$NMI(W,C) = \frac{I(W,C)}{\sqrt{H(W)H(C)}} \tag{2.21}$$

where $I(W,C)$ is called the mutual information between W and C and can be obtained as follows:

$$I(W;C) = \sum_{k}\sum_{j} P(w_k \cap c_j) log \frac{P(w_k \cap c_j)}{P(w_k)P(c_j)} \tag{2.22}$$

and H is the entropy:

$$H(W) = -\sum_k P(w_k) log P(w_k) \tag{2.23}$$

The value of the mutual information is expected to be 0 if the clusters are obtained randomly and 1 if the clusters are perfect with the labels. Nonetheless, mutual information has the same problem as purity, which is that it is maximum for the number of clusters N. What fixes this problem is the normalization since entropy tends to increase with the number of clusters. The NMI is always a number between 0 and 1.

Another alternative is to use the *Rand index* (RI) measure. The idea is to view a clustering as a series of decisions, which are taken for pairs of data points. The different types of decisions are: true positives (TP) - two similar data points are assigned to the same cluster, true negatives (TN) - two dissimilar data points are assigned to different clusters, false positive (FP) - two dissimilar data points are assigned to the same cluster and false negatives (FN) - two similar data points are assigned to different clusters. Then, the index is computed as:

$$RI = \frac{TP + TN}{TP + FP + EN + TN} \tag{2.24}$$

which is simply the accuracy.

Rand index gives the same weights for false positives and false negatives, which can be inappropriate for some situations. In this case, the *F measure* can be used. This measure penalizes the false negatives more than the false positives with a chosen value $\beta > 1$. This measure can be computed as follows:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{2.25}$$

where $P = \frac{TP}{TP+FP}$ is the precision, and $R = \frac{TP}{TP+FN}$ is the recall.

Finally, although the Rand Index can measure the validity of a clustering result, it has some strong disadvantages: it is not 0 for random clustering and also increases with the number of clusters obtained. Therefore, in order to overcome these issues, one of the most used validation indexes for clustering is the Adjusted Rand Index (ARI) [43], which is ensured to have a value close to 0 for random labeling independently of the number of clusters and samples and exactly 1 when the clusterings are identical. This measure can also have a negative in some cases. The ARI can be computed as follows, where n is the number of images in the dataset:

$$ARI = \frac{\binom{n}{2}(TP + TN) - [(TP - FP)(TP - FN) + (TN - FP)(TN - FN)]}{\binom{n}{2}^2 - [(TP - FP)(TP - FN) + (TN - FP)(TN - FN)]} \tag{2.26}$$

2.2.3.2 Internal Indexes

A very simple way to measure the quality of a clustering result is by computing the correlation between the points of each cluster. High correlation indicates that points that belong to the same cluster are close to each other, with respect to some similarity measure.

For this calculation, two matrices need to be computed using all data points: the proximity matrix and the incidence matrix. Both the proximity matrix and the incidence matrix have n rows and n columns, with n being the total number of data points in the dataset. The proximity matrix has the similarity measures between all the data points, normalized to be in the interval from 0 to 1. On the other hand, the incidence matrix is, at each entry, 1 if the associated pair of points belong to the same cluster (according to the results obtained after the clustering analysis), and 0 if they belong to different clusters. Then, the correlation is computed as the multiplication of the two matrices. Figure 2.13 shows two examples of clustering results on two datasets. The first has a high correlation (corr = -0.9235) and the second has a smaller correlation (corr = -0.5810). This measure has many limitations, especially when the clusters have an arbitrary shape.

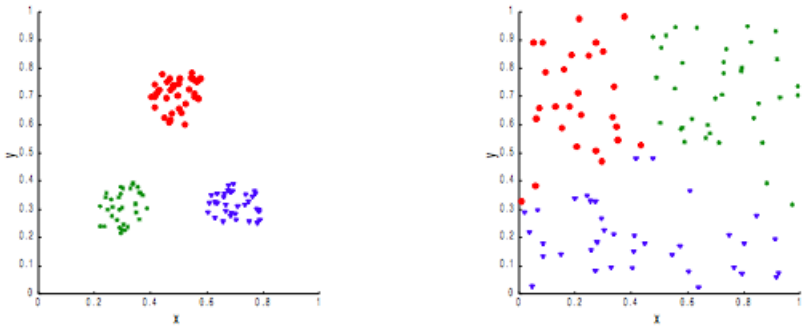


Figure 2.13: Example of two clustering results (obtained by K-Means algorithm) in two different datasets.

Other than the correlation, most clustering validity indexes are usually defined by combining the following pair of evaluation criteria:

- Compactness or cohesion: Measures how closely related objects are in a cluster.
- Separability: Measure how distinct or well-separated a cluster is from other clusters.

There are many articles that study and compare different clustering validity indexes. One of the most recent and extensive one is [44] in which the authors compared as many as 30 clustering validity indexes in a well defined experimental design using both synthetic and real datasets.

The best performing index in almost all the experiments was the silhouette index [45], which is a normalized summation-type index. The cohesion is measured based on the distance between all the points in the same cluster and the separability is based on the nearest neighbor distance. To define this index, the dataset X is denoted as the set of N objects represented as vectors in an F-dimensional space: $X = \{x_1, x_2, \ldots, x_N\} \subseteq \Re^F$. The k partitions C are represented as $C = \{c_1, c_2, \ldots, c_k\}$, where c_k is the mean vector of the cluster.

Finally, the silhouette index is defined as:

$$Sil(C) = \sum_{c_k \in C} \sum_{x_i \in c_k} \frac{b(x_i, c_k) - a(x_i, c_k)}{max\{a(x_i, c_k), b(x_i, c_k)\}} \tag{2.27}$$

where

$a(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \in c_k} d_e(x_i, x_j)$,

$b(x_i, c_k) = \{\frac{1}{|c_k|} \sum_{x_j \in c_k} d_e(x_i, x_j)\}$

Other indexes that achieved got scores in the experiment performed by [44] where the Calinski-Harabasz index [46] and a variation of the Davies-Bouldin index presented in [47].

The Calinski-Harabasz index is a ratio-type index where the cohesion is estimated based on the distances from the points in a cluster to its centroid. The separation is based on the distance from the centroids to the global centroid. It can be defined as:

$$CH(C) = \frac{N-K}{K-1} \frac{\sum_{c_k \in C} |c_k| d_e(\overline{c_k}, \overline{X})}{\sum_{c_k \in C} \sum_{x_i \in c_k} d_e(\overline{x_i}, \overline{c_k})} \tag{2.28}$$

where

$\overline{c_k} = \frac{1}{|c_k|} \sum_{x_i \in c_k} x_i$,

$\overline{X} = \frac{1}{N} \sum_{x_i \in X} x_i$

and

$d_e(x_i, x_j)$ is the euclidean distance between vector x_i and x_j.

To conclude, the clustering analysis is one of the most important techniques in the area of pattern discovery and data mining and its applications are endless. Nevertheless, evaluating the quality of a clustering analysis is an essential but difficult task, especially when there is no reference to the underlying structure of the dataset. This evaluation is done using clustering validity indexes.

2.3 Content-based Image Retrieval

With the widespread of Internet and of image capturing devices, the size of digital image collections is increasing rapidly. For this reason, there is a great need for developing efficient image search systems. Until now, two classes of systems have been developed: text-based and content-based. The text-based is an older approach, that dates back to the 1970s [48]. In such systems, images are searched by a given text query that relies on the text present in the image annotation. One disadvantage with this approach is that many images are not annotated, and therefore, cannot be found using this method. Other disadvantage is the fact that many images are inaccurately annotated, so the results will probably not correspond to the user's expectations. To overcome the above disadvantages, content-based image retrieval (CBIR) was introduced in the early 1980s [49]. In CBIR, images are indexed based on visual features such as color, shape or texture. Since then, there has been a great amount of research focus on the area of CBIR.

CBIR is the most developed application filed, which applies image representation techniques. The difference between CBIR and image clustering is that CBIR attempts to find search items similar to the query image and clustering deals with finding groups in all the images from a dataset. Nonetheless, the concepts are very similar, and therefore, some important background about CBIR is presented in this section.

The initial step for a CBIR system is to analyze the images from its database. For this purpose, the system needs to describe the content of the image. First, the images are processed using image processing techniques in order to enhance relevant aspects. Next, features are extracted from the images. Usually, low-level features are extracted and combined into creating higher level features. There are several types of low-level features that can be extracted from the images, including color, shape, texture, and global features. After a feature vector for each image is obtained, these feature vectors are stored in a feature database. In the case of CBIR by example, when a query image is given, the process of image processing and feature extraction is also performed in the query image. The following step is to compute the similarity between the query image and the images in the database. This is accomplished by computing similarity measures between the feature vectors. Figure 2.14 In relation to a semantic CBIR system, there are more steps to take since the query will be text-based, which either has to be converted to a feature-based query or the feature database needs to have high-level concepts associated with the images.

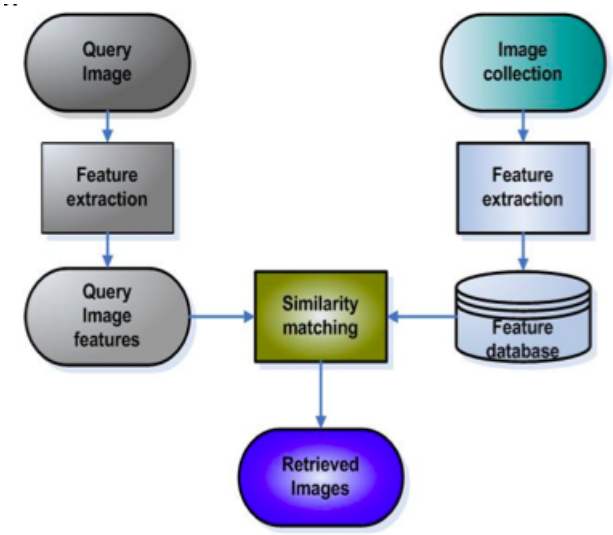


Figure 2.14: A general model of CBIR systems. Extracted from [50].

2.3.1 The Semantic Gap

An important difference between text-based and content-based image retrieval systems is the fact that humans tend to use text and keywords to express high-level concepts in order to interpret images and measure their similarity, whereas visual features automatically extracted from images can usually only represent low-level features. In general there is no link between the low-level

features and the high-level features [48]. Therefore, the performance of CBIR is still far from the user's expectations. According to [51], there are three levels of queries in CBIR:

- Level 1: Retrieval by primitive (low) features such as color, texture and shape. A typical query of this kind is a query by example, i.e. "find a picture like this".
- Level 2: Retrieval of objects of a given type. For example, "find a picture of a flower".
- Level 3: Retrieval by abstract attributes, involving a significant amount of high-level reasoning about the purpose of the objects or scene depicted. An example could be "find pictures of a joyful crowd".

Levels 2 and 3 together are referred as semantic image retrieval, and the gap between Levels 1 and 2 as the semantic gap. A more detailed definition of the semantic gap is presented in [49]:

Definition 2. *The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation the same data have for a user in a given situation.*

Users in Level 1 retrieval are usually required to submit an example image, which can not be available. Therefore, semantic image retrieval is a more desirable approach.

2.3.2 Reducing the Sematic Gap

There has been many attempts of creating CBIR algorithms for reducing the semantic gap explained previously. According to [48], these approaches can be classified into five different categories: using object ontology to define high-level concepts, using machine learning methods to associate low-level features with query concepts, using relevance feedback to learn user's intentions, generating semantic template to support high-level image retrieval and fusing the evidences from HTML text and the visual content of images for WWW image retrieval.

A description of the first three approaches is presented in the following sections.

2.3.2.1 Object-ontology

Object-ontology is used to derive simple semantics. First, different intervals are defined for low-level image features, with each interval corresponding to an intermediate-level descriptor of images, for example, "light blue, large, middle". These descriptors form a simple vocabulary, called object-ontology, which provides a qualitative definition of high-level query concepts [48].

An example of a CBIR system which used object-ontology is the one presented in [52]. First, the proposed approach employs a segmentation algorithm to divide images into regions. After that, each region of the image is described by color, position, shape and size.

One key components in a ontology-based system is the quantization of the color and texture features. A widely used method for quantization of color is *color naming*. This method relates a numerical color space with semantic color names used in natural language. A well-known color naming system is CNS (Color Naming System) presented in [53].

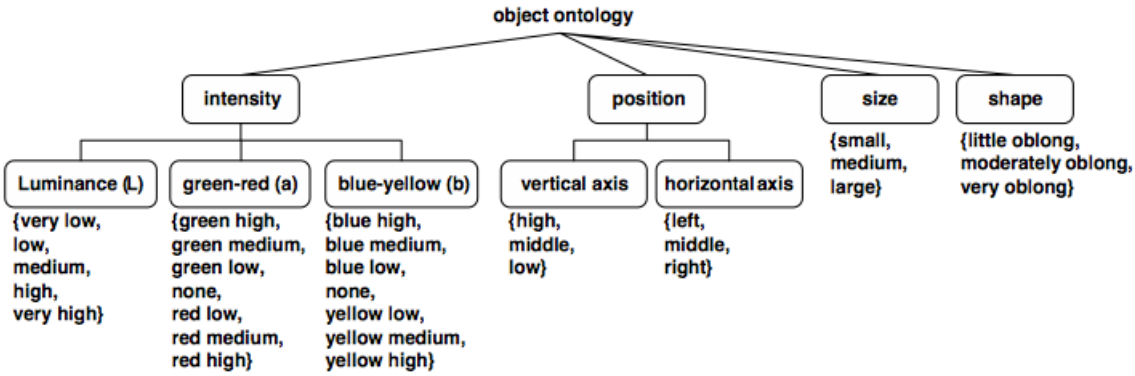


Figure 2.15: Example of object-ontology model used in [52]. Extracted from [52]

The object-ontology approach for reducing the semantic gap may be a possibility but when considering large image databases with a variety of contents, more powerful tools need to be derived.

2.3.2.2 Machine Learning

Another way to learn higher level concepts from low-level features extracted from images is to use machine learning tools such as supervised or unsupervised learning, which were mentioned in Section 2.2.

In terms of the algorithms, many supervised learning techniques have been applied to CBIR. SVM (Support Vector Machines), Bayesian classifier and neural networks are often used for this purpose [48]. For example, in [54], SVM is employed for image annotation. In the training stage, a binary SVM model is trained for the 23 classes (concepts). After that, in the testing stage, the input are unlabeled regions and the resulting label of the model trained is associated with the image or region.

Unsupervised learning techniques such as clustering, can also be used to reduce the semantic gap in CBIR systems. In [55], a method called CLUE is presented. Unlike other CBIR systems, which display the top matched target images, this system attempts to retrieve semantically coherent image clusters.

2.3.2.3 Relevance Feedback

Relevance Feedback (RF) is a powerful tool originally used in text-based information retrieval systems. It is an online processing task which tries to reduce the semantic gap by attempting to learn the users' preferences as it searches for information [48].

During an image search, the first K images in the similarity ranking are presented to the user, who has the opportunity of marking them as relevant or non-relevant if not satisfied with the result [56].

A typical scenario for RF in CBIR is described by the steps bellow [48]:

1. The system provides initial retrieval results through query-by-example, sketch, etc.
2. User judges the above results as to whether and to what degree, they are relevant (positive examples)/irrelevant (negative examples) to the query.
3. Machine learning algorithms is applied to learn the users' feedback. Then go back to (2).

Steps (2) and (3) are repeated until the user is satisfied with the result. A representative diagram of the RF process can be seen in Figure 2.16.

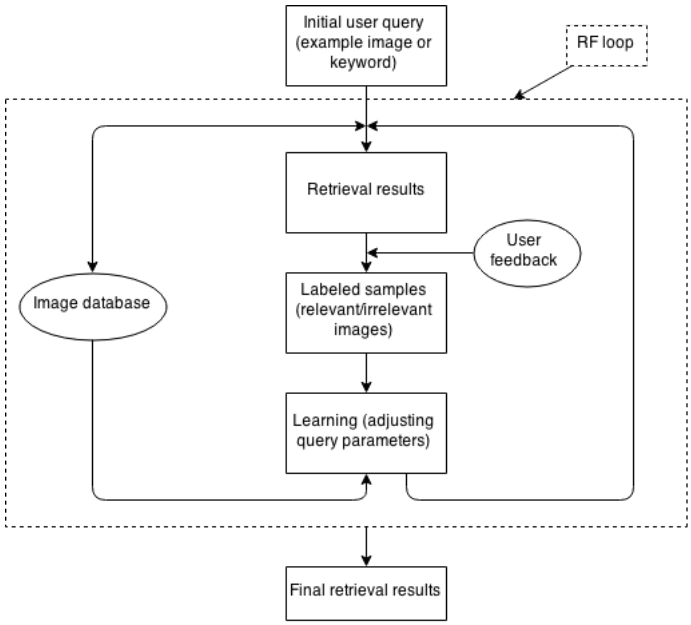


Figure 2.16: CBIR with RF. Extracted from [48].

In order to take advantage of the user interaction, an effective learning method should be adopted to improve the retrieval results. One of the most simple methods for learning from relevance feedback is to automatically adjust the weights of each of the low-level features extracted from the images.

The authors of [57] proposed a CBIR system with RF which is based on the *random walker* algorithm introduced in the context of interactive image segmentation. The idea is to treat the relevant and non-relevant images labeled by the user at every feedback round as "seed" nodes for the *random walker* problem [58]. The ranking score for each unlabeled image is computed as the probability that a random walker starting from that image will reach a relevant seed before encountering a non-relevant one.

2.4 Image Descriptors

An image descriptor is an information or characteristic that can be extracted from an image. There are generally three types of image descriptors: low-level descriptors, mid-level descriptors, and

high-level descriptors. Figure 2.17 shows a schematic representation of the three levels of image descriptors.

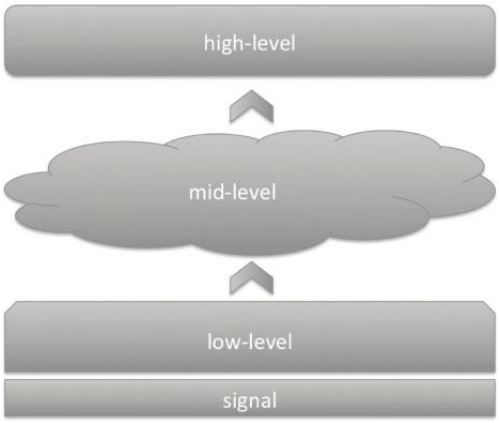


Figure 2.17: Schematic representation of the different types of image descriptors. Extracted from [59].

From this figure, it can be seen that the low-level image descriptors are the ones extracted directly using the raw signal (in this case, a image pixels), whereas the mid-level descriptors use information gathered by the low-level descriptors, and so does the high-level descriptors, in which the input is generally given by the mid-level descriptors.

The high-level descriptors represent a piece of human-interpretable semantic information describing an image. They represent the goal of describing and annotating images [59].

In this section, low-level image features and techniques for building mid-level image descriptors will be described and presented.

2.4.1 Low-level Image Features

A low-level descriptor is a continuous, discrete or symbolic measurement, which is computed directly from the signal (e.g. image pixels) [59]. It is designed to capture a certain visual property of an image, either globally for the entire image or locally for a small group of pixels. The most commonly used features include those reflecting color, texture, shape and salient/interest points in an image.

In this section, some important and widely used low-level descriptors are presented. These descriptors belong to a number of different groups and have different characteristics and applications.

2.4.1.1 Color Features

Color features are one of the most widely used features in describing images or segmented regions in an image. They are the most intuitive and most obvious image features. Compared to other image features such as texture and shape, color features are very stable and robust. It is not sensitive to rotation, translation and scale changes as it is usually very simple to compute [60].

2.4.1.1.1 Color Spaces A color space is the dimensional space where the colors are defined. A variety of color spaces exist and each of them has its own applications. A brief description of the RGB, YCbCr, HSV and L*a*b color spaces will be presented next. A more detailed presentation of color spaces can be found in [61].

RGB The most widely used color space is called RGB (Red, Green and Blue), which is based on the three primary colors and a white reference point. Using an appropriate scale along each primary axis, the space can be normalized, so that the maximum value is 1. Therefore, as can be seen in figure 2.18, the RGB color solid is a cube with vertices at (0,0,0), which corresponds to the color black and (1,1,1) which represents white.

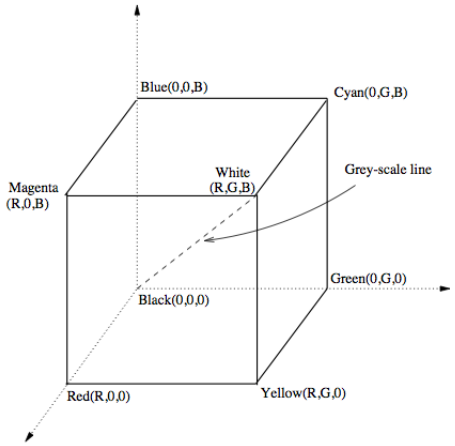


Figure 2.18: The RGB color model. Extracted from [61].

YCbCr For storage and transmission reasons, the human's visual system needs to be taken into account to allow for compression. According to [61], it is believed that the human's visual system forms an achromatic channel and two chromatic color-difference channels in the retina. Therefore, a color image can be represented as a wide band component corresponding to brightness, and two narrow band color components.

YCbCr or Y'CbCr is a color space linearly related to the RGB space and is used for digital video (MPEG). Y'CbCr is calculated using nonlinear gamma corrected values of R, G, B denoted R',G',B'. The relationship between Y'CbCr is presented bellow:

$$\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ 37.797 & -74.203 & 112.0 \\ 112.0 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{2.29}$$

The Y'CbCr color space is a scaled and offset version of the YUV color space.

HSV The HSV (Hue Saturation Value) color model belongs to the group of hue-oriented color coordinate systems (which also includes HSI). It was originally proposed in [62], and is a cylindrical coordinate system. It can be represented by a hexcone model, which is shown in figure 2.19.

The advantages of the hue-oriented color spaces are, among others [61]:

- Good compatibility with human intuition.
- Separability of chromatic values and achromatic values.
- The possibility of using only one component, hue, for segmentation purposes.

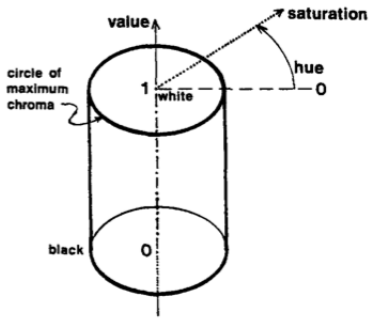


Figure 2.19: HSV color representation. Extracted from [63]

.

L*a*b Due to the fact that R,G and B components are highly correlated, chromatic information is not directly fit for use. The L*a*b is designed to approximate human vision. Therefore, it is a good choice for representing two color points. L* stands for lightness and a* and b* for the color-opponent dimensions.

The nonlinear relations for L*, a*, and b* are intended to mimic the nonlinear response of the eye. Furthermore, uniform changes of components in the L*a*b* color space aim to correspond to uniform changes in perceived color, so the relative perceptual differences between any two colors in L*a*b* can be approximated by treating each color as a point in a three-dimensional space and taking the Euclidean distance between them.

To convert between RGB and L*a*b, a number of different computations need to be done and can be found in [61].

2.4.1.1.2 Color Histograms A color histogram is the most used method to extract color features [60]. A color histogram is a frequency statistic for different colors in a certain color space (such as the ones described previously). It does not require any form of segmentation of the image since it describes the color content of the global image. However, one of its drawback is that it cannot describe the local distribution of the image in color space and the spatial position of each

color. This means that the color histogram cannot describe specific objects or regions in an image. Another disadvantage of using color histograms is that it is not unique and robust to noise [64].

In order to compute a color histogram, the color space needs to be divided into several small ranges, which are denoted as bins. Thus, the color is quantized. Each bin represents the number of pixels that fall into that color space interval. Figure 2.20 shows an example of a color histogram of an image. In total, there are three histograms, one for each color component red, green and blue.

Figure 2.20: Example of color histograms for an example image.

Color features include global color histogram and block color histogram.

In [60], the authors used a global color histogram for Content-based image retrieval as one of the features. First, the images where converted to the HSV color space in order to meet visual requirements of humans. Next, the image colors where quantified using 8 levels for H, 3 levels for S and V. After the histogram of each image was computed, the similarity between images was computed as the euclidian distance between the histograms of the images. These features extracted are invariant to rotation and translation. The drawback is that two completely different images can get the same global color histogram, which is a problem for the purpose of the article (image retrieval).

Color histograms have proven effective for small databases, but their limitations become rapidly apparent with larger databases [65].

2.4.1.1.3 Color Moments Color moments are another way of characterizing a color distribution of an image. It is based on the idea of that a color distribution of an image can be interpreted as a probability distribution, which can be characterized by its central moments. The authors of [66], who were the first to propose such features, suggested storing of the first, the second and the third moments of each color channel. The first moment is the average color whereas the second and the third moments are the variance and the skewness of each color channel. If the value of the

i-th color channel at the *j*-th image pixel is *pij*, then the first, second and third color moments can be calculated as follows:

$$E_i = \frac{1}{N}\sum_{j=1}^{N} p_{ij} \tag{2.30}$$

$$\sigma_i = (\frac{1}{N}\sum_{j=1}^{N} (p_{ij} - E_i)^2)^{\frac{1}{2}} \tag{2.31}$$

$$\sigma_i = (\frac{1}{N}\sum_{j=1}^{N} (p_{ij} - E_i)^3)^{\frac{1}{3}} \tag{2.32}$$

The difference between two color images can be thus calculated using a (weighted) distance measure between the moments of all the color channels or the images.

The greatest advantage of using color moments comes from the fact that there is no need to store the complete color distribution, and therefore, it is an efficient method.

2.4.1.1.4 Color Coherence Vectors A color coherence vector (CCV) stores the number of coherent versus incoherent pixels with each color [67]. By separating coherent pixels from incoherent pixels, CCV provide finer distinctions than color histograms.

A CCV can be defined, intuitively, as the degree to which pixels of that color are members of large similarly-colored regions. These regions are referred as coherent regions, and they have a significant importance in characterizing images.

For example, the images shown in figure 2.21 have similar color histograms, despite being taken in completely different contexts. The color red appears in both images in approximately the same quantities, but in the left image the red pixels (from the flowers) are widely scattered, while in the right image the red pixels (from the golfer's shirt) form a single coherent region.

Figure 2.21: Two images with similar color histograms. Extracted from [67].

In order to compute the coherence vectors, the pixels of a given color regions need to be classified as either coherent or not, given that a coherent pixel is part of a large group of pixels of the same color, based on an 8-neighborhood. When this process is done, each pixel will belong to exactly one connected component. The classification between coherent and incoherent is based on the size of its connected component.

To calculate the difference between two images based on the color coherence vectors, one needs to compute the difference between the number of coherent pixels and the difference between the incoherent pixels for each color bin in the two images.

CCV creates a finer distinction than color histograms since an image can have the same histogram but different CCVs based on the coherent color regions.

2.4.1.2 Texture Features

Texture gives information about the spatial arrangement of the colors or intensities in an image [68]. Suppose that the histogram of a region shows that it has 50% white pixels and 50% black pixels. Figure 2.22 shows three different images of regions with this intensity distribution that has three very different textures. As illustrated by the figure, texture provides important information in image description.

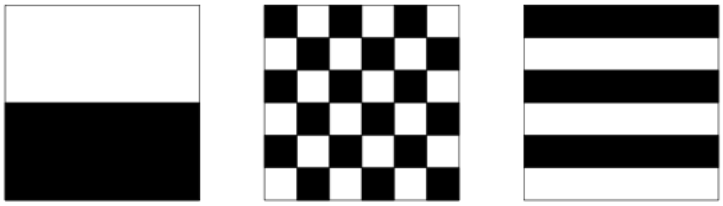


Figure 2.22: Three different textures with the same distribution of black and white. Extracted from [68].

There are many features that reveal information about the texture of an image. Some of the most used are: Gabor filtering, wavelet transform, Tamura texture features and Edge Histogram Descriptor (EDH). In the following sections, a brief description of Gabor filtering and the Wavelet transform will be presented.

2.4.1.2.1 Gabor Filtering One of the most popular signal processing based approaches for texture feature extraction is the use of Gabor filters. It has been proposed that Gabor filters can be used to model the responses of the human visual system. Frequency and orientation information can be extracted by using a bank of filters at different scales and orientations which allows multichannel filtering of an image. These can then be used as texture features [69].

A two dimensional Gabor function $g(x,y)$ and its Fourier transform $G(u,v)$ can be written as [70]:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} exp[-\frac{1}{2}(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}) + 2\pi jWx] \tag{2.33}$$

$$G(u,v) = exp\{-\frac{1}{2}[\frac{(u-W)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}]\} \tag{2.34}$$

where $\sigma_u = 1/2\pi\sigma_x$ and $\sigma_v = 1/2\pi\sigma_y$. Gabor functions form a complete but nonorthogonal basis set. They provide a localized frequency description.

After this, a bank of Gabor filters can be generated by dilating and rotating the function from equation 2.33:

$$h_{i,j}(x,y) = a^{-i}h(x',y'), i,j = integer \tag{2.35}$$

where $x' = a^{-i}(x\cos\theta + y\sin\theta)$, $y' = a^{-i}(-x\sin\theta + y\cos\theta)$, $\theta = j\pi/K$ and K is the total number of orientations. The scale factor a^{-i} is meant to ensure equal energy among different filters. The statistics of the detected features can be used to characterize the underlying texture information. Given an image $I(x,y)$, filtering the image with Gabor filters $h_{i,j}$ results in:

$$O_{i,j}(x,y) = \int I(x,y)h^*_{i,j}(x-x_1, y-y_1)dx_1dy_1 \tag{2.36}$$

where * indicates the complex conjugate. Therefore, a simple texture feature can be constructed using the mean and the standard deviation of the amplitude information [70].

2.4.1.2.2 Wavelet Transform A wavelet transform decomposes a 1-D signal $f(x)$ onto a basis of wavelet functions [71]:

$$(W_a f)(b) = \int f(x)\psi^*_{a,b}(x)dx \tag{2.37}$$

Which is obtained translating and dilating a single mother wavelet ψ:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi(\frac{x-b}{a}) \tag{2.38}$$

The wavelet decomposition of an 2-D image can be obtained by performing the filtering consecutively along the horizontal and the vertical directions. The decomposition steps are depicted schematically in figure 2.23. This leads to a representation with an equal amount of pixels as the original image.

The wavelet image decomposition provides a representation that is easy to interpret. Each subimage contains information of a specific scale and orientation, and therefore, the spacial information is not lost.

2.4.1.3 Shape Features

A Shape feature attempts to quantify shape of the relevant objects in an image in ways that agree with human intuition [72]. There are many shape features available, used in image processing and computer vision applications such as aspect ratio, circularity, moments and chain codes. However, shape features are more difficult to apply than color and texture features due to the fact that it usually requires a segmentation procedure to be implemented previously, which can be highly inaccurate. Nonetheless, shape features have shown to be useful in many domain specific images such as man-made objects [48].

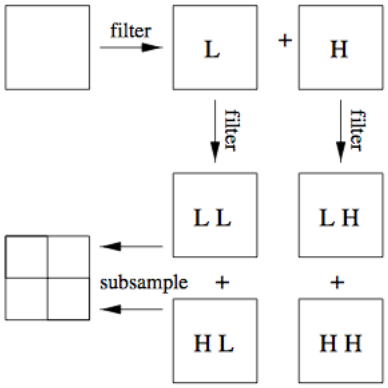


Figure 2.23: First level of a wavelet decomposition in 3 steps: Low and High pass filtering in horizontal direction, the same in the vertical direction, subsampling. Extracted from [71].

Among the most popular shape features are the Hough transform and Moments. Therefore, both methods will be described in the following sections.

2.4.1.3.1 Hough Transform The Hough transform is a feature extraction technique used in many different application in image processing and computer vision. The goal of this technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform [73].

The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes.

The Hough transform is usually applied to a binary image. In order to convert a image to a binary image, significant image feature points need to be determined. One of the alternatives is to apply a thresholding to the output of a mask-based, edge detection operators such as *Sobel*, or *Prewitt*. A more sophisticated edge/contour detector like Canny can also be used.

After obtaining the significant points, the key ideas of the Hough transform can be illustrated by considering identifying sets of colinear points in an image (a line). A set of image points *(x, y)* which lie on a straight line can be defined by a relation, f, such that:

$$f((\hat{m},\hat{c}),(x,y)) = y - \hat{m}x - \hat{c} = 0 \tag{2.39}$$

where m is the slope parameter and c is the intercept parameter, which characterize the line. In equation 2.39, the hat symbol is used to indicate that the pair of parameters *(m,c)* are constant, and therefore, the pair *(x,y)*can take many values. An alternative perspective is to consider the opposite, which is that the pair *(m,y)* can vary and the the pair *(x,y)* is constant (a point in the xy

space). This function, called g, is defined in equation 2.40:

$$f((\hat{x},\hat{y}),(m,x)) = \hat{y} - \hat{x}m - c = 0 \tag{2.40}$$

Therefore a point in the xy space corresponds to a line in the mc space and vice-versa. Therefore, a group of points that form a straight line are converted to lines in the mc space, which have a interception point that corresponds to the slope (m) and the intercept (c) of the line in the xy space. An example of this behavior can be found in figure 2.24.

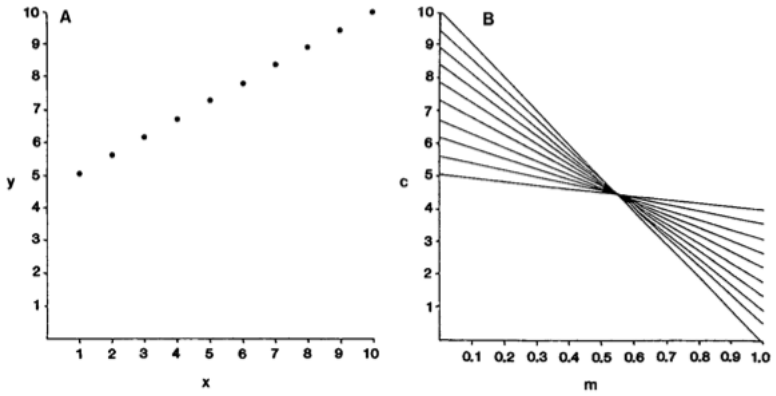


Figure 2.24: The basis of Hough transform line detection; (A) *(x,y)* point image space; (B) *(m,c)* parameter space. Extracted from [73].

The determination of the point of intersection in parameter space is a simple local operation.

In order to detect parametrically defined curves other than straight lines, instead of defining the function with parameters m and c, the parameters will be $\{a1,\ldots,an\}$:

$$f((\hat{a_1},\ldots,\hat{a_n}),(x,y)) = 0 \tag{2.41}$$

And by swapping the roles of the parameters and the variables comes:

$$g((\hat{x},\hat{y}),(a_1,\ldots,a_n)) = 0 \tag{2.42}$$

This equation maps a point into a hypersurface in the n-dimensional space defined by the parameters. Similarly to the line case, the most probable curve in the image is detected by the point with the most intersections of the several hyperserfaces.

The Hough transform can be used to detect lines and other shapes, and that can be used as a feature to describe images [73].

2.4.1.3.2 Moments An image moment is a certain particular weighted average (moment) of the image pixels' intensities, or a function of such moments, usually chosen to have some attractive property or interpretation [74].

Image moments can be used to obtain simple properties of a image region which includes area, centroid, and information about its orientation.

Among the region-based descriptors, moments are very popular. These include invariant moments, Zernike moments and Radial Chebyshev moments [72].

The general form of a moment function m_{pq} of order $(p+q)$ of a shape region can be given as:

$$m_{pq} = \sum_x \sum_y \Psi_{pq}(x,y) f(x,y) \quad p,q = 0,1,2,\ldots \tag{2.43}$$

where Ψ_{pq} is know as the moment weighting kernel or the basis set and $f(x,y)$ is the shape region values.

Invariant moments (IM) are also called geometric moment invariants. Geometric moments are the simplest of the moment functions, with basis $\Psi_{pq} = x^p y^q$. Despite being complete, they are also not orthogonal [72]. Geometric moments function m_{pq} are given as:

$$m_{pq} = \sum_x \sum_y x^p y^q (x,y) f(x,y) \quad p,q = 0,1,2,\ldots \tag{2.44}$$

The so called geometric central moments, which are invariant to translation are defined by:

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q (x,y) f(x,y) \quad p,q = 0,1,2,\ldots \tag{2.45}$$

where $\bar{x} = m_{10}/m_{00}$ and $\bar{x} = m_{01}/m_{00}$.

Finally, a set of 7 invariant moments (IM) are given by:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{2.46}$$
$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{2.47}$$
$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{2.48}$$
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{2.49}$$
$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \tag{2.50}$$
$$(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{2.51}$$
$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + \tag{2.52}$$
$$4\eta_{11}^2 (\eta_{30} + \eta_{12})(\eta 21 + \eta_{03}) \tag{2.53}$$
$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \tag{2.54}$$
$$(3\eta_{12} - \eta_{03})(\eta_{21} + \eta_{03}[3(\eta_{30} + \eta_{12})^2 - \eta_{21}\eta_{03})^2] \tag{2.55}$$

where $\eta_{pq} = \mu_{pq}/\mu_{00}^{\gamma}$ and $\gamma = 1 + (p+q)/2$ for $p+q = 2,3,\ldots$

IM are simple to compute, invariant to rotation, scaling and translation. However they suffer from information redundancy and noise sensitivity (in higher order moments).

Zernike Moments (ZM) are other type of moments. Unlike the invariant moments, they are orthogonal. The complex Zernike moments are derived from orthogonal Zernike polynomials

[72]:

$$V_{nm}(x,y) = V_{nm}(r\cos\theta, \sin\theta) = R_{nm}(r)exp(jm\theta) \tag{2.56}$$

where $R_{nm}(r)$ is the orthogonal radial polynomial:

$$R_{nm}(r) = \sum_{s=0}^{(n-|m|)/2} (-1)^2 \frac{(n-s)!}{s! \times (\frac{n-2s+|m|}{2})!(\frac{n-2s-|m|}{2})!} r^{n-2s} \tag{2.57}$$

where $n = 0,1,2,\ldots$; $0 \leq |m| \leq n$; and $n-|m|$ is even.

The Zernike moments of order n with repetition m of the region $f(x,y)$ is given by:

$$Z_{nm} = \frac{n+1}{n} \sum_r \sum_\theta f(r\cos\theta, r\sin\theta) \cdot R_{nm}(r) \cdot exp(jm\theta) \quad r \leq 1 \tag{2.58}$$

The Zernike moments are rotation invariant, robust to noise, and since the basis is orthogonal, they have minimum information redundancy [72]. However, it is more complex computationally than the other moments.

2.4.1.4 Local Descriptors

Global features have the ability to generalize an entire object with a single vector. Consequently, their use in standard classification or retrieval techniques is straightforward. Local features, on the other hand, are computed at multiple points in the image and are consequently more robust to occlusion and clutter. They also do not require a segmentation of the object from the background, unlike many texture features, or representations of the object's boundary (shape features) [75]. One of the disadvantages of dealing with local features is that there may be differing numbers of feature points in each image, making comparing images more complicated.

A general framework for computing local descriptors is [76]:

1. Find the interest points.
2. Consider the region around each keypoint.
3. Compute a local descriptor from the region and normalize the feature.
4. Match local descriptors in multiple images.

The first stage of these types of systems, is the search for interesting points (keypoints). These points should be chosen so that they can be matched in different images with different types of conditions. Some available algorithms to find keypoints are: Hessian/Harris corner detection, Laplacian of Gaussian (LOG) detector and Difference of Gaussian (DOG) detector. Usually corners are good candidates for keypoints since in the region around a corner, image gradient has two or more dominant directions, and therefore, they are repeatable and distinctive.

Next, the SIFT, PCA-SIFT, SURF and ORB algorithms will be introduced.

2.4.1.4.1 SIFT SIFT (Scale Invariant Feature Transform) features, presented by [77], are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. It is widely used nowadays due to its robustness.

To compute the SIFT features, four major stages are required [77]:

1. **Scale-space peak selection:** the first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian (DoG) function to identify potential interest points that are invariant to scale and orientation.

2. **keypoint localization:** at each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

3. **orientation assignment:** one or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4. **keypoint descriptor:** the local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

An important aspect of this approach is that it generates large numbers of features that densely cover the image over the full range of scales and locations.

Next, a more detailed explanation will be made of the steps required for computing the SIFT local descriptors.

Scale-space Peak Selection

This step has the goal of identifying candidate locations for the local descriptors. In order for the descriptors to be scale invariant, the locations need to be invariant to scale change. That can be accomplished by searching for stable features across all possible scales, using a continuous function of scale known as *scale space*. For that, the Gaussian function is chosen as the scale-space kernel. The scale-space of an image is defined as a function $L(x,y,\sigma)$, produced by the convolution of the variable-scale Gaussian $G(x,y,\sigma)$ and the image $I(x,y)$.

$$L(x,y,\sigma) = G(x,y,\sigma) * \mathbf{I}(x,y) \tag{2.59}$$

where G is the gaussian function:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.60}$$

The difference-of-Gaussians function convolved with the image, $D(x,y,\sigma)$ can be computed from the difference of two nearby scales separated by a constant multiplicative factor k:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * \mathbf{I}(x,y) = L(x,y,k\sigma) - L(x,y,\sigma) \tag{2.61}$$

An efficient approach to construction of $D(x,y,\sigma)$ is shown in Figure 2.25. The initial image is incrementally convolved with Gaussians (eq. 2.60) to produce images separated by a constant factor k in scale space, shown in the left column. Each octave of scale space (i.e., doubling of σ) has s intervals, thus $k = 2^{1/s}$. Each stack of blurred images must have $s+3$ images for each octave. The adjacent image scales are subtracted to produce the difference-of-Gaussian images shown on the right using eq. 2.61. Once a complete octave has been processed, a resampling is implemented in the Gaussian image that has twice the initial value of σ by taking every second pixel in each row and column.

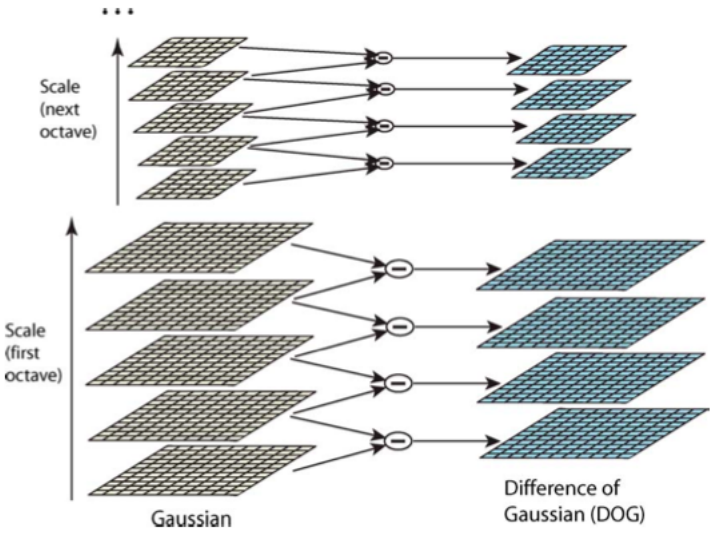


Figure 2.25: For each octave, adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images. After each octave, the images are downsampled with a factor of 2, and the process is repeated. Extracted from [77].

In order to detect the local maxima and minima of $D(x,y,\sigma)$, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below. This is illustrated in figure 2.26. The point is selected only if it is larger than all of these neighbors or smaller than all of them.

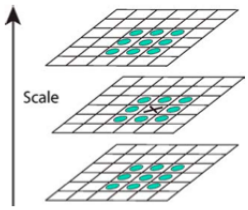


Figure 2.26: Detection of the maxima and minima of the difference-of-Gaussian images by comparing the neighbors. Extracted from [77].

Keypoint Localization

Once the set of keypoints has been found, the next step is to perform a detailed fit to the nearby data for location, scale, and ratio of principal curves. This process is required since the first keypoint step detects a huge amount of points that are not good candidates.

First, for each candidate keypoint, interpolation of nearby data is used to accurately determine its position. For that, the interpolated location of the extremum is calculated. The interpolation is done using the quadratic Taylor expansion of the Difference-of-Gaussian scale-space function, $D(x,y,\sigma)$ with the candidate keypoint as the origin. This Taylor expansion is given by:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{2.62}$$

The location of the extremum, $\hat{\mathbf{x}}$, is determined by taking the derivative of this function with respect to $\mathbf{x}$ and setting it to zero. If the offset $\hat{\mathbf{x}}$ is larger than 0.5 in any dimension, then that is an indication that the extremum lies closer to another candidate keypoint. Otherwise the offset is added to its candidate keypoint to get the interpolated estimate for the location of the extremum.

To discard the keypoints with low contrast, the value of the second-order Taylor expansion $D(\mathbf{x})$ is computed at the offset $\hat{\mathbf{x}}$. If this value is less than 0.03, the candidate keypoint is discarded. Otherwise it is kept.

The difference-of-Gaussian function will also have a strong response along edges, which is not desirable. They can be characterized as having a poorly defined peak in the difference-of-Gaussian function which will indicate a large principal curvature across the edge but a small one in the perpendicular direction. The principal curvatures can be obtained from a 2×2 Hessian matrix, H, computed at the location and scale of the keypoint:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.63}$$

The derivatives are estimated by taking differences of neighboring sample points. The eigenvalues of H are proportional to the principal curvatures of D. If the ratio $\mathrm{R} = \mathrm{Tr}(\mathbf{H})^2 / \mathrm{Det}(\mathbf{H})$, for a candidate keypoint is larger than a given threshold, that keypoint is poorly localized and hence rejected. This process suppresses the response at the edges. Figure 2.27 shows an example of the SIFT descriptor steps until the identification of the final keypoints.

Orientation Assignment

The next step has the objective to make this method invariant to rotation.

For an Gaussian-smoothed image $L(x,y)$ at scale σ, the gradient magnitude, $m(x,y)$, and orientation, $\theta(x,y)$, are computed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{2.64}$$

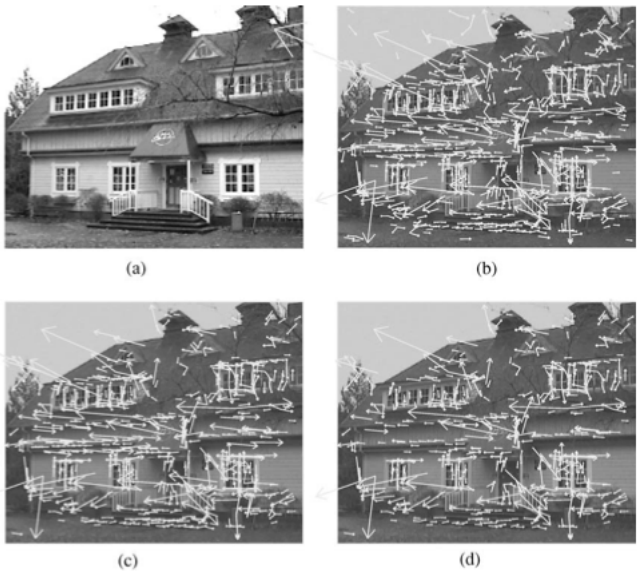


Figure 2.27: Stages of keypoint selection. (a) The 233×189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures. Extracted from [77].

$$\theta(x,y) = \text{atan2}\left(L(x,y+1) - L(x,y-1), L(x+1,y) - L(x-1,y)\right) \tag{2.65}$$

The magnitude and direction calculations for the gradient are done for every pixel in a neighboring region around the keypoint. An orientation histogram with 36 bins is formed, and each bin covers 10 degrees. Each sample in the neighboring window adds to a histogram bin. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint.

Keypoint Descriptor The goal of this final step is to create a descriptor vector for each keypoint such that the descriptor is highly distinctive and partially invariant to the remaining variations.

First a set of orientation histograms is created on 4×4 pixel neighborhoods with 8 bins each. The magnitudes are then weighted by a Gaussian function with σ equal to one half the width of the descriptor window. The descriptor then becomes a vector of all the values of these histograms. In the end, each vector has 128 elements. This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination. Furthermore, to reduce the effects of non-linear illumination a threshold of 0.2 is applied and the vector is again normalized.

Matching Images After the description stage, two images can be matched by comparing the feature vectors obtained by using the SIFT descriptors. It searches in the feature vectors for matches

in the descriptors in a nearest neighbor methodology. The nearest neighbors are defined as the keypoints with minimum Euclidean distance from the given descriptor vector. Figure 2.28 presents two images of the same sight with some differences in illumination and partial occlusion.

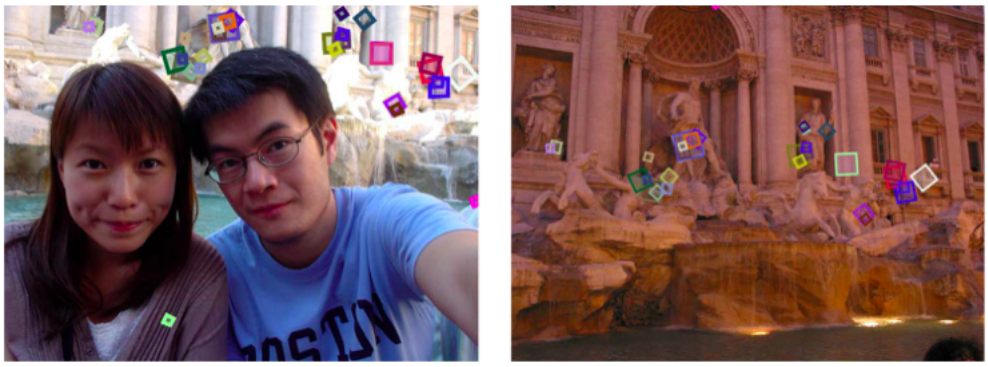


Figure 2.28: Example of a SIFT descriptor matching found in two different images of the same sight. Extracted from [78].

To conclude, the SIFT descriptor is a very robust and distinctive image descriptor, and is well suited for applications such as object recognition. Nonetheless it has be applied to many task that requires identification or matching.

2.4.1.4.2 PCA-SIFT PCA-SIFT is a variation of the SIFT descriptors, introduced in [79]. The essential difference between the algorithms lies in the description stage, where to encodes the salient aspects of the image gradient in the feature point's neighborhood, instead of using SIFT's smoothed weighted histograms, it applies Principal Components Analysis (PCA) to the normalized gradient patch. This creates significantly smaller feature vectors than the standard SIFT feature vector, and can be used with the same matching algorithms. The experiments held in [79] demonstrate that the PCA-based local descriptors are more distinctive, more robust to image deformations, and more compact than the standard SIFT representation.

Next, a short background information about PCA is given.

PCA Principal Component Analysis (PCA) [80] is a standard technique for dimensionality reduction and has been applied to a broad class of computer vision problems. It is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (uncorrelated with) the preceding components.

Back to PCA-SIFT, after the initial steps of keypoint determination and orientation, a 41×4 patch (regions of pixels) at the given scale, centered over the keypoint, and rotated to align

its dominant orientation to a canonical direction is extracted. The rest of the algorithm can be summarized in the following steps [79] :

1. Pre-compute an eigenspace to express the gradient images of local patches.

   To build the eigenspace, the authors of [80] ran the first three stages of the SIFT algorithm on a diverse collection of images and collected 21,000 patches. Each was processed as described in the next step to create a 3042-element vector, and PCA was applied to the covariance matrix of these vectors. The matrix consisting of the top n eigenvectors was stored on disk and used as the projection matrix for PCA-SIFT.

2. Given a patch, compute its local image gradient.

   The input vector is created by concatenating the horizontal and vertical gradient maps for the 41×41 patch centered at the keypoint. Thus, the input vector has 3042 elements, which are then normalized to minimize the impact of illumination variations.

3. Project the gradient image vector using the eigenspace to derive a compact feature vector. A feature space of $n = 20$ of principal components were used in [80].

This method is effective because projecting the gradient patch onto the low-dimensional space appears to retain the identity-related variation while discarding the distortions induced by other effects [80].

2.4.1.4.3 SURF SURF (Speeded Up Robust Features) is a robust local feature detector, first presented in [81]. It is similar to the SIFT descriptor, although being several times faster. It is based on sums of 2D Haar wavelet responses and makes an efficient use of integral images.

The SURF algorithm is defined by a similar three-step procedure as the SIFT features (keypoint detection, orientation assignment and description).

A detailed explanation about the steps of the SURF algorithm will be done next.

Fast-Hessian Detector In SURF, squares are used instead of Gaussians. The idea is to rely on integral images for a much faster processing time. Integral images allow for the fast implementation of box type convolution filters. The entry of an integral image $I_\Sigma(\mathbf{x})$ at a location $\mathbf{x} = (x, y)$ represents the sum of all pixels in the input image I of a rectangular region formed by the point x and the origin, as given by the following equation:

$$I_\Sigma(\mathbf{x}) = \sum_{i=0}^{i<=x} \sum_{j=0}^{j<=y} I(i, j) \tag{2.66}$$

With the integral image calculated, it only takes four additions to calculate the sum of the intensities over any upright, rectangular area, independent of its size.

As for the keypoint detector, SURF uses a blob detector based on the Hessian matrix. Given a point (x, y) in an image I, the Hessian matrix $H(\mathbf{x}, \sigma)$ at scale σ is calculated as follows:

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{2.67}$$

where $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second order derivative with the image I in the point $\mathbf{x}$, and similarly for $L_{xy}(\mathbf{x}, \sigma)$ and $L_{yy}(\mathbf{x}, \sigma)$. However, instead of Gaussian filters SURF approximates by using box filters, which can be computed significantly faster, given that the image is an integral image. The filter approximation can be found in figure 2.29.

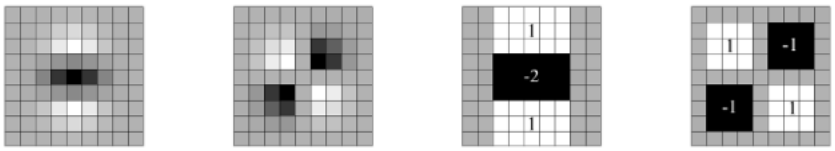


Figure 2.29: Left to right: The (discretised and cropped) Gaussian second order partial derivatives in y-direction and xy-direction, and our approximations thereof using box filters. The grey regions are equal to zero. Extracted from [81].

The approximations are denoted D_{xx}, D_{xy} and D_{yy}. Then the determinant of the approximate matrix is computed:

$$det(H_{aprox}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.68}$$

The approximation of the determinant of the Hessian matrix represents the response blob image on location x. These responses are stored in the blob response map on different scales. After that, the local maxima are searched.

Orientation Assignment In order to be invariant to rotation, a reproducible orientation for each interest points needs to be identified. The Haar wavelet responses in both x and y directions within a circular neighborhood of radius $6s$ around the point of interest are computed, where s is the scale at which the point of interest was detected. The obtained responses are weighed by a Gaussian function centered at the point of interest, then plotted as points in a two-dimensional space, with the horizontal response in the abscissa and the vertical response in the ordinate. After that, the dominant orientation is estimated by computing the sum of all responses within a sliding orientation window of size $\pi/3$. The two summed responses (horizontal and vertical) yield a local orientation vector. Finally, the longest such vector overall defines the orientation of the point of interest.

Descriptor Components The first step of the description phase consists of constructing a square region (of size $20s$) centered around the interest point, and oriented along the orientation obtained in the previous section.

The interest region is split up into smaller 4×4 square sub-regions, and for each one, a Haar wavelet responses at 5×5 regularly spaced sample points is computed and weighted with a Gaussian ($\sigma = 3.3s$) centered at the interest point.

Then, the wavelet responses dx and dy are summed up over each sub-region and form a first set of entries to the feature vector. Hence, each sub-region has a four-dimensional descriptor vector v for its underlying intensity structure:

$$v = (\sum dx, \sum dy, \sum |dx|, \sum |dy|) \tag{2.69}$$

The wavelet responses are invariant to a bias in illumination (offset). Also, invariance to contrast (a scale factor) is achieved by turning the descriptor into a unit vector.

To illustrate, Figure 2.30 shows the properties of the descriptor for three distinctively different image intensity patterns within a sub-region.

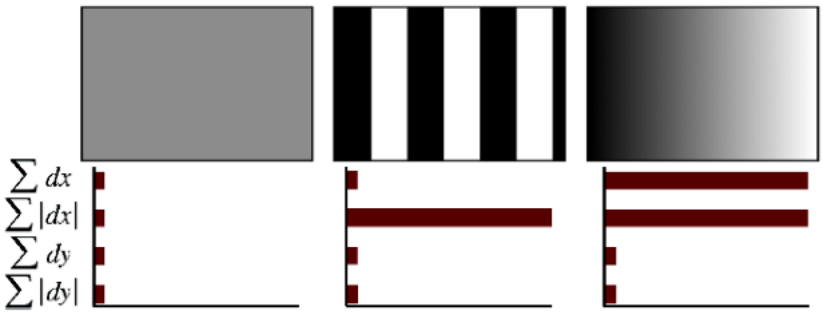


Figure 2.30: The descriptor entries of a sub-region represent the nature of the underlying intensity pattern. Left: In case of a homogeneous region, all values are relatively low. Middle: In presence of frequencies in x direction, the value of $|dx|$ is high, but all others remain low. If the intensity is gradually increasing in x direction, both values $|dx|$ and $|dx|$ are high. Extracted from [81].

To conclude, the authors stated in [81] that the SURF descriptors outperforms the other algorithms until that date, both in speed and accuracy. The fact that is is a it is less complex are more efficient than the SIFT descriptors makes it a good choice for applications that require real-time computation.

2.4.1.4.4 ORB ORB (Oriented FAST and Rotated BRIEF) [82] is a recent algorithm for detection and description of local image features that uses improved versions of the FAST (Features from accelerated segment test) keypoint detector [83] and the BRIEF (Binary Robust Independent Elementary Features) descriptor [84]. According to [82], it is an efficient alternative to SIFT and SURF.

Next, before describing the modifications introduced by ORB, the FAST detector and the BRIEF descriptor are presented.

FAST FAST is a method for corner detection that can be used to extract keypoints for low-level feature extraction from images. It has a very high computational efficiency and it is faster than

the Difference of Gaussians (DoG) and Harris detector. For this reason, it is suitable for real-time applications.

As shown in Figure 2.31, the algorithm works by first creating a circle of 16 pixels around every pixel. Then it considers that pixel a corner if there exists a set of n contiguous pixels in the circle which are all brighter than the intensity of the pixel plus a margin, or darker than intensity of the pixel minus a margin. Also, in order to improve the speed of the detector, a procedure is done to exclude a large number of non-corners.

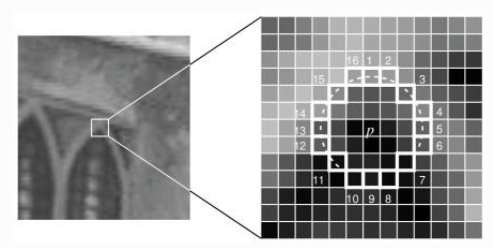


Figure 2.31: Illustration of the circle considered by the FAST detector. Extracted from [83].

In order to avoid the detection of multiple features adjacent to one another, non-maximal suppression is performed. Therefore, for every pixel considered a corner in the first step, the sum of the differences in intensity of all the circle of 16 pixels is computed. Then, for adjacent pixels, only the one with the highest value of the sum can be a corner.

BRIEF BRIEF is a binary descriptor that intends to reduce the amount of memory and time to compute the features and to match them. First it take the smoothened image and selects, following some kind of algorithm, pairs of pixel inside the patches obtained previously by a detector. It then compares these pairs of pixels. If the value of the intensity of the first pixel is larger than the second pixel, then the resulting binary feature is 1. Therefore, it obtains n_d binary features for each patch. Usually the values of n_d can be either 128, 256 or 512.

ORB uses the FAST detector and then applies Harris corner measure to find the top N points. Additionally, it uses pyramid to produce multiscale features. However, the FAST detector does not compute the orientation of the keypoint. For this reason, ORB computes a weighted centroid that helps to give the orientation of the interest point detected. Also, moments are computed.

In relation to the descriptors, ORB uses the BRIEF descriptor with a few modifications that allow it to perform better under rotation transformations. It does this by rotating the BRIEF descriptor according to the orientation of the keypoint obtained previously.

An additional improvement introduced by ORB is that it runs a greedy search among all possible binary tests of BRIEF to find the ones that have both high variance and means close to 0.5, as well as being uncorrelated. The result is called rBRIEF.

2.4.1.4.5 CenSurE or STAR Detector The CenSurE or STAR detector was first introduced in [85] and was developed as a feature detector for the specific problem of visual odometry, which is the process of determining the position and orientation of a robot by analyzing camera images.

The detector approach is similar to the SIFT detector but uses bi-level filters to approximate the Laplacian instead of the Difference of Gaussians (DoG). These filters can only multiply the image pixels by -1 or 1. The most used shape of the bi-filters are octagons.

Therefore, first computes a simplified center-surround filter at all locations and all scales, and finds the extrema in a local neighborhood. Then, these extrema are filtered by computing the Harris measure and eliminating those with a weak corner response.

One of the key characteristics of the CenSurE detector is the efficiency in which it computes the bi-level filters at all sizes, which is done using integral images, which are also used in the SURF detector. This detector is said to suitable for real-time applications, unlike the SIFT detector.

The results presented in [85] show that the CenSurE detector performs better than the SIFT and SURF detector for image matching when there is small viewpoint changes.

2.4.1.5 Global Descriptors

As mentioned before, global descriptors obtain features shared by all the pixels of an image. Therefore, there is no need to perform any kind of segmentation or keypoint detection. The only global descriptor that will be introduced in this section is the recent GIST descriptor.

2.4.1.5.1 GIST The GIST descriptor was initially proposed in [86]. It was designed for the application of scene recognition. The idea was to develop a low-dimensional representation of the scene, called the special envelop, which does not require any form of segmentation. The name GIST is given due to the fact that it is a global descriptor, that represents the *general idea* of the image/scene. The authors proposed a set of perceptual dimensions: naturalness, openness, roughness, expansion and ruggedness. These dimensions represent the dominant spacial structure of a scene.

As defined by the authors of [86], a scene is when there is a larger space between the observer and a fixed point. Therefore, a scene is not an object, but can be considered to be a place where people can move.

The features extracted from the images are the discrete Fourier transform (DFT) and the windowed Fourier transform (WFT).

The discrete Fourier transform can be computed as follows:

$$I(f_x, f_y) = \sum_{x,y=0}^{N-1} i(x,y)h(x,y)^{-j2\pi(f_x x + f_y y)} = A(f_x, f_y)e^{j\Phi(f_x, f_y} \tag{2.70}$$

where $i(x,y)$ is the intensity of the image I along the spacial variables, f_x and f_y are the spacial frequency variables and $h(x,y)$ is a circular Hanning window to reduce boundary effects. Also, $I(f_x, f_y)$ is a periodic function and the central period is between $(f_x, f_y) = [-0.5, 0.5] \times [-0.5, 0.5]$. The amplitude spectrum of the image is $A(f_x, f_y) = |I(f_x, f_y)|$ and the phase function of the Fourier transform is $\Phi(f_x, f_y)$.

The amplitude spectrum $A(f_x, f_y)$ gives unlocalized information about the image structure, orientation, smoothness, length and width of the contours that compose the scene image. In contrast,

the phase function $\Phi(f_x, f_y)$ contains information relative to the form and the position of the image components.

Additionally, the energy spectrum, $A(f_x, f_y)^2$ is computed as the squared magnitude of the Fourier transform. This feature gives information about the distribution of energy among the different spatial frequencies.

Finally, in order to obtain information about the spatial distribution of the spectral information, a windowed Fourier transform is computed, which is given by:

$$I(x, y, f_x, f_y) = \sum_{x',y'=0}^{N-1} i(x', y')h(x' - x, y' - y)^{-j2\pi(f_x x' + f_y y')} \tag{2.71}$$

where $h_r(x', y')$ is a circular hamming window of radius r. The energy spectrum (spectrogram) of this function provides localized structural information and can be used for a detailed analysis of the scene.

After extracting these features from the image, a Principal Component Analysis (PCA) is implemented in order to do dimensionality reduction of the set of features extracted.

2.4.2 Mid-Level Image Descriptors

A mid-level image descriptor is a continuous or discrete numeric or symbolic measurement obtained after a global analysis of the low-level descriptors, possibly by applying supervised or unsupervised learning methods on a subset of images in a collection or involving the use of external knowledge [59].

Mid-level image descriptors represent an intermediate step between low and high-level descriptors dedicated to a specific task or application.

Next, a review of different approaches to the construction of mid-level image descriptors is done.

2.4.2.1 Bag-of-Features (BoF)

Bag-of-Features (BoF) approaches are characterized by the use of an orderless collection of image features, lacking any structure or spatial information. Due to its simplicity and performance, the Bag-of-Features approach has become well-established in the field of computer vision [87].

The name comes from an analogy with the Bag-of-Words representation used in textual information retrieval (text mining). The idea is to consider that an image is composed by *visual words* with a given topology. A *visual word* is a local segment in an image, defined either by a region (image patch or blob) or by a reference point with its neighborhood. Then, the analysis of the visual word occurrences and configurations allows the development of a frequency histogram of *visual words* for each image.

The general steps for building a Bag-of-Features representation of a collection are [88]:

1. Compute low-level image descriptors: usually, local image descriptors are extracted from the image, such as the SIFT descriptors presented in section 2.4.1.4.1, preceded by a salient/interest point detection step (such as the Hessian affine region detector).

2. Quantize the descriptors into clusters: once the descriptors for each image are computed (a feature vector), an unsupervised method is used (clustering) to reduce the number of visual words (quantize) to only the centroids of each cluster (that represent the cluster). These will be the *visual words* used to form the vocabulary.

3. Represent each image by a vector of frequencies of visual words in order to do matching, retrieval or similarity analysis.

This process is illustrated in figure 2.32 for the application of content-based image retrieval.

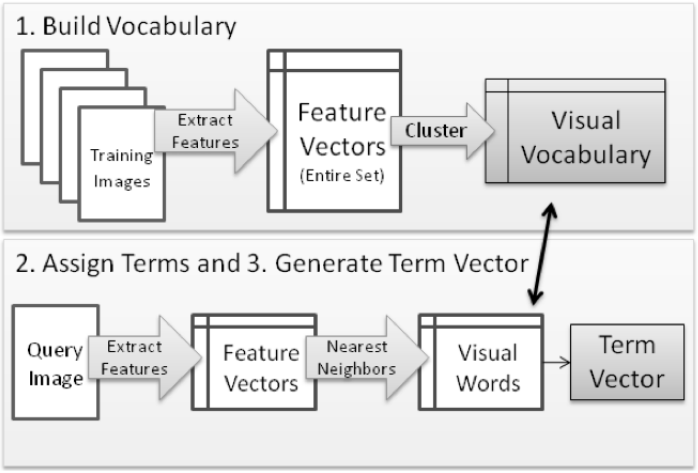


Figure 2.32: Process for Bag-of-Features Image Representation for CBIR. Extracted from [87].

Due to the fact that, usually, a large vocabulary is selected, and most images have only a few of those visual words, most of the feature vector obtained by the BoF approach is zero. The strong sparsity of these vectors allows for efficient indexing schemes and other performance improvements, as discussed in later sections [87].

To improve the results of the Bag-of-Features model, and in analogy to text retrieval, it is usual to apply a weighting to the components of the feature vector (*visual words*), rather than using the feature vector directly. The standard weighting used is known as 'term frequency-inverse document frequency', *tf-idf*, as is computed as follows. Suppose there is a vocabulary of k words, then each document is represented by a k-vector $V_d = (t_1, \ldots, t_k)^T$, of weighted word frequencies with components [88]:

$$t_i = \frac{n_{id}}{n_d} log \frac{N}{n_i} \tag{2.72}$$

where n_{id} is the number of occurrences of word i in document d, n_d is the total number of words in the document d, n_i is the number of occurrences of term i in the whole database and

N is the number of documents in the database. With the strategy of using term weights, one can penalize terms found to be too common to be discriminative and emphasize those that are more unique.

An early work defining the Bag-of-Features image retrieval approach is [88], where the authors presented a project called Video Google. Video Google uses both MSER [89] and Harris-Affine keypoint detectors to extract features, which are represented by SIFT descriptors. The vocabulary is built using K-Means clustering. Nearest Neighbor term assignment and the Euclidean distance on *tf-idf* weighted term vectors are used for similarity scoring.

2.4.2.2 Fisher Vectors

The Fisher Vector (FV), which was introduced in [90] is a method for image representation based on the Fisher Kernel [91]. In sum, this method first extracts local descriptors from the images, then it tries to fit a GMM (Gaussian Mixture Model) to a sampled portion of the data, and finally, it computes statistics regarding each descriptor in relation to the GMM fitted. This gives a great amount of information about the structure of the descriptors of the images. It can be seen as a generalization of the BoF model, since it can be reduced to it under several restrictions.

It has several advantages over the BoF model, w.r.t. image classification, namely:

1. It was shown to provide more accurate results.
2. It can be computed using much smaller vocabularies reducing the computational complexity.
3. It performs well even with simple linear classifiers.

A GMM is a linear combination of Gaussians. A GMM of K-components is represented by the following equation:

$$u_\lambda(x) = \sum_{k=1}^{K} w_k u_k(x) \tag{2.73}$$

where each u_k is a different Gaussian, which can be written as follows:

$$u_k(x) = \frac{1}{(2\pi)^{\frac{D}{2}}(|\Sigma_k|)^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x-u_k)'\Sigma_k^{-1}(x-u_k)\} \tag{2.74}$$

Usually, $\forall_k : w_k \geq 0$:

$$\sum_{k=1}^{k} w_k = 1 \tag{2.75}$$

The data extracted from the images using the descriptors is first sampled, and then a GMM is fitted to that data. To accomplish this task, Expectation Maximization (EM) to optimize a Maximum Likelihood (ML) criterion [92] is used. Just like the BoF model, the GMM is considered to be the dictionary and its size is given by k.

Now, the features for each image are computed as the log-likelihood of the descriptors considering the GMM obtained. Therefore, let $X = \{x_t, t = 1, \ldots, T\}$ be the group of $D-$dimensional descriptors extracted from an image, assuming that they are independent. The FV of this image can be written as:

$$\Phi_\lambda^X = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t) \tag{2.76}$$

One of the statistics used is the posterior probability, which can be computed as follows, where x_t is a single descriptor:

$$\gamma_t(k) = \frac{w_k u_k(x_t)}{\sum_{j=1}^{K} w_j u_j(x_t)} \tag{2.77}$$

After the FV is obtained, it is normalized in order to improve the representation. In [90] two normalization procedures were used: the $l2-$normalization and the power normalization. The $l2-$normalization is applied to remove the fact that different images contain different amounts of background information, whereas the power normalization makes the FV representation less sparce and therefore makes it more suitable for comparison with the dot-product.

Additionally, in order to obtain better results, PCA is usually applied to the local descriptors obtained from the images. A number of components often used is 64, reducing the dimensionality of the features from 128 (in the case of SIFT) in half. The reason behind using PCA is that it obtains more independent features, which is one of the restrictions of the algorithm.

As stated in [90], compared to the BoF model, the Fisher Vector offers a more complete representation of the dataset, as it encodes not only the count of occurrences but also higher order statistics related to its distribution. The better use of the information provided by the model translates also into a more efficient representation, since much smaller vocabularies are required in order to achieve a given performance. The experiments also show an improved performance compared to the BoF model in terms of classification accuracy.

Ultimately, the algorithm for computing the FVs is presented in Algorithm 2, as presented in [90].

2.4.2.3 Spatial Pyramid Matching (SPM)

Spatial Pyramid Matching (SPM) [93] is a technique that works by partitioning the image into increasingly fine sub-regions and computing histograms of local features found inside each sub-region. It is an extension of the Bag-of-Features representation which attempts to add information concerning spacial location. The authors of the method demonstrated significant improved performance on challenging scene categorization tasks.

SPM is a kernel-based recognition method that works by computing rough geometric correspondence on a global scale using an efficient approximation technique adapted from the pyramid matching scheme of [94].

Algorithm 2 Fisher Vectors Algorithm

Input:
- Local image descriptors $X = \{x_t \in \Re^D, t = 1,\ldots,T\}$
- Gaussian mixture model parameters $\lambda = \{w_k, \mu_k, \sigma_k, k = 1,\ldots,K\}$

Output:
- Normalized Fisher Vector representation $\Phi_k^X \in R^{K(2D+1)}$

1: **Compute Statistics**

- For $k = 1,\ldots,K$ initialize accumulators
  - $S_k^0 \leftarrow 0$, $S_k^1 \leftarrow 0$, $S_k^2 \leftarrow 0$
- For $t = 1,\ldots,T$
  - Compute $\gamma_t(k)$ using equation 2.77
  - For $k = 1,\ldots,K$:
    - $S_k^0 \leftarrow S_k^0 + \gamma_t(k)$
    - $S_k^1 \leftarrow S_k^1 + \gamma_t(k)x_t$
    - $S_k^2 \leftarrow S_k^2 + \gamma_t(k)x_t^2$

2: **Compute the Fisher Vector signature**

- For $k = 1,\ldots,K$:
  - $\Phi_{\alpha_k}^X = (S_k^0 - Tw_k)/\sqrt{w_k}$
  - $\Phi_{\mu_k}^X = (S_k^1 - \mu_k S_k^0)/(\sqrt{w_k}\sigma_k)$
  - $\Phi_{\sigma_k}^X = (S_k^2 - 2\mu_k S_k^1 + (\mu_k^2 - \sigma_k^2)S_k^0)/(\sqrt{2w_k}\sigma_k^2)$
- Concatenate all Fisher vector components into one vector

3: **Apply Normalizations**

- For $i = 1,\ldots,K(2D+1)$ apply power normalization
  - $[\Phi_\lambda^X]_i \leftarrow sign([\Phi_\lambda^X]_i)\sqrt{|[\Phi_\lambda^X]_i|}$
- Apply $l2-$normalization:
  - $\Phi_\lambda^X \leftarrow \Phi_\lambda^X / \sqrt{\Phi_\lambda^{X\top}\Phi_\lambda^X}$

Pyramid match kernels work by placing a sequence of increasingly coarser grids over the feature space and taking a weighted sum of the number of matches that occur at each level of resolution. At any fixed resolution, two points are said to match if they fall into the same cell of the grid. Matches found at finer resolutions are weighted more highly than matches found at coarser resolutions. More specifically, the number of matches $I(H_X^l, H_Y^l)$ (or I^l) for each level l, is computed as:

$$I(H_X^l, H_Y^l) = \sum_{i=1}^{D} min(H_X^l(i), H_Y^l(i)) \tag{2.78}$$

where $H_X^l(i)$ and $H_Y^l(i)$ are the number of points from X and Y that fall into the ith cell of the grid at level l. Also, the grid at level l has 2^l cells. This is called the histogram intersection function.

It can be noted that the number of matches found at level l also includes all the matches found at the finer level $l+1$. Therefore, the number of new matches found at level l is given by $I^l - I^{l+1}$ for $l = 0, \ldots, L-1$. Therefore, in order to penalize matches found in larger cells, due to the fact that they involve increasingly dissimilar features, a weight is associated with the matches found on level l which is given by $\frac{1}{2^{L-l}}$. Therefore the final expression for the pyramid match kernel is:

$$\kappa^L(X,Y) = I^L + \sum_{l=0}^{L-1} \frac{1}{2^{L-l}(I^l - I^{l+1})} = \frac{1}{2^L} I^0 + \sum_{l=1}^{L} \frac{1}{2^{L-l+1}} I^l \tag{2.79}$$

The next step is to quantize all feature vectors into M discrete types. Each channel m gives two sets of two-dimensional vectors, X_m and Y_m, representing the coordinates of features of type m found in the respective images. The final kernel is then the sum of the separate channel kernels:

$$K^L(X,Y) = \sum_{m=1}^{M} \kappa^L(X_m, Y_m) \tag{2.80}$$

This approach is reduced to the Bag-of-Features when $L = 0$.

For L levels and M channels, the resulting vector has a dimensionality of:

$$M \sum_{l=0}^{L} 4^l = M \frac{1}{3}(4^{L+1} - 1) \tag{2.81}$$

The authors of [93] noted no performance increase beyond $M = 200$ and $L = 2$.

A illustration of the SPM can be found in figure 2.33. In this figure, the image has three types of descriptor classes indicated by the circles, the crosses and the diamonds. In level 0, the image is not split and the histogram has 3 different bins (which are the three different classes). Next, in resolution with level 1, the image is subdivided into 4 sub-images and for each of then, the number of examples of the three different classes of features is computed making 4×3 histogram bins. Finally, in level 2, the image is divided in 4×4 sub-regions and the histogram is again computed. Each spacial histogram is also weighted according to equation 2.79.

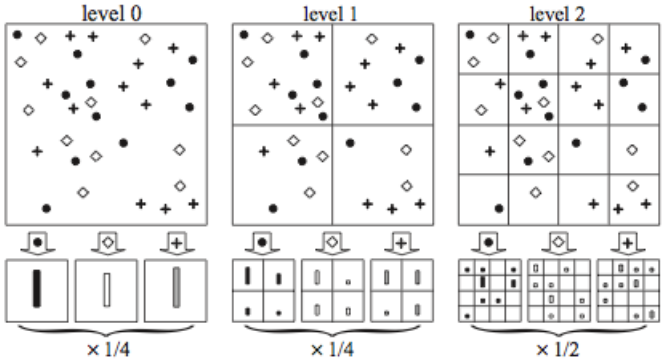


Figure 2.33: Example of a construction of a three-level pyramid. Extracted from [93].

2.4.2.4 Bag-of-Colors (BoC)

In contrast with the Bag-of-Features model that uses local image descriptors like SIFT or SURF to extract information from the images, the Bag-of-Colors (BoC) model, which was introduced in [95], uses only color information. Therefore, it extracts color patches from the images, creates a color dictionary (codebook) and then represents each image as a histogram of frequency of colors from the dictionary.

The procedure, presented in [95] for the BoC model is as follows:

1. Convert images to a more efficient color spaces, in this case, the authors chose CIE Lab due to the more consistency with the euclidean space.
2. Split the images into blocks of 16×16 pixels.
3. For each block, compute the most frequent color (using 3-D a color histogram).
4. Apply K-Means clustering algorithm to produce the color dictionary of K colors.
5. For each image, build a histogram of frequency of the colors from the color dictionary. This is called the color signature.

The essential parameters for this model are the number of color patches extracted per image and the size of the color dictionary. Figure 2.34 shows examples of color dictionaries of different sizes.

It is shown in [95] that the BoC, used as a global image descriptor achieves comparable results when evaluated against various state of the art global descriptors. Nonetheless it is much simpler than others.

2.5 Performance Evaluation

The performance evaluation of a image mining (image clustering, image classification and image retrieval) system can be achieved using many different types of methodologies. One method is

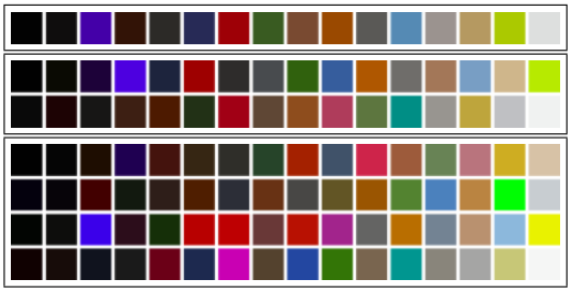


Figure 2.34: Example of color dictionaries learned for different number of colors ($K = 16, 32, 64$). Extracted from [95].

to use labeled data (supervised evaluation), where images were previously annotated. Here, the external indexes for clustering validity (sec. 2.2.3.1) can be used to compare different algorithms and to access its performance. The other type is using unsupervised techniques such as estimating statistical parameters that represent the quality of the clusters obtained (please refer to section 2.2.3). The problem with this last method for evaluating the performance of an image mining algorithm is that there is no concrete relationship between the values of the the descriptors obtained from the image and the true concept represented by the image (as seen by humans). This is related to the concept of the semantic gap, described in section 2.3.1.

There is also other method that has been used to evaluate image clustering, which is to estimate the relevance of the cluster results as perceived by users. Thus, this is a subjective measure. This approach has been used, for instance in [96], where a study involving 20 users was conducted on a subset of the derived image clusters in order to assess the perceived relevance of the produced clusters.

Next, some image databases are presented. All these databases have been used for research in the area of image mining.

2.5.1 Image databases

There are many different image databases available. In CBIR, the most popular image database for evaluation and comparison purposes is Corel image database [97]. Corel image database contains a large amount of images of various contents such as animals, sports and people. These images are divided into 100 different categories. Figure 2.35 presents some example images from Corel dataset. Although this dataset is one of the most widely used databases for CBIR, it has some problems [98]. First, some images with very similar content are divided in different groups. Also, some category labels are very abstract, and the images within the same category can be largely varied in content. Hence, it is usual for researchers to select a subset of the original database (Corel 5-K, Corel 10-K, etc) or to do some changes in the labels of some categories. Additionally, there is the issue of image quality (images have approximately 200 pixels of size), which is particularly important for modern image feature representations. For example, it is not recommended to use

Difference-of-Gaussian based SIFT features because the approach is not suited for small images due to the sparsity of extracted interest regions [99].

Figure 2.35: Examples of images from the Corel dataset.

Another image database available and used for many purposes is the Urban and Natural Scenes dataset [86] which was collected in the MIT labs. It is composed by 2688 color images divided into 8 classes. Another scene dataset is the Event dataset [100] that contains images from 8 sports event categories with roughly 200 images per category.

Columbia Object Image Library (Coil-100) is a database for object-detection [101]. It consists of 7,200 color images of 100 objects (72 images per object). There is also a smaller subset of this dataset with only 20 classes of objects with only grayscale images called Coil-20 dataset [102]. Since the database only has the same exact object in each category, although shifted or rotated, this database is not at all representative of what is shared on Twitter. Nevertheless, it is a very interesting image dataset that could be potentially used for testing different algorithms for image description. Other object datasets are Caltech-101 [103] and ETH-80 [104].

Figure 2.36: The 100 objects (classes) from the Coil-100 image database. Extracted from [102].

In 2008, a new image dataset was released, MIR Flickr dataset [105]. It is composed by 250,000 high quality annotated images taken from the Flickr website, which are freely available for research purposes. The images represent a real community of users both in the image content and image tags. In Flickr, users share and search pictures based on image tags provided by them.

The average number of tags per image collected for the database was 8.94. The majority of tags are in English and were subdivided in various categories to allow for recognition tasks. Figure 2.37 shows some examples from images in the database together with the Creative Common attribution license icons and creators of each images.

Figure 2.37: Examples of images from the MIR Flickr dataset. Also listed are Creative Common attribution license icons and the creators of the images. Extracted from [105].

Another interesting image database is ImageNet, which was released in 2009 [106]. It is organized according to the WorldNet hierarchy [107]. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset". The aims of ImageNet is to provide on average 1000 images to illustrate each synset. Images of each concept are human-annotated. As of the latest update (April 30, 2010), ImageNet had a total of 14,197,122 images covering 21841 synsets. There are 27 high-level concepts that represent the first level of the WorldNet three. These concepts include concepts such as animal, food, plant, sport, vehicle and person. Figure 2.38 shows some examples of images belonging to two different subtree groups (mammals and vehicles). The other concepts (synsets) are more specific than from left to right. For each synsets, 9 images are presented.

Finally, some works in image mining use a combination of different benchmark image databases [108, 109]. These databases do not need to represent an extensive amount of diversity due to the fact that they will be sampled and combined, which would potentially create a more diverse and robust dataset. Examples of datasets that have been used for this purpose is MNIST database [110], USPS database [111], USF HumanID [112] and UMIST face image database [113].

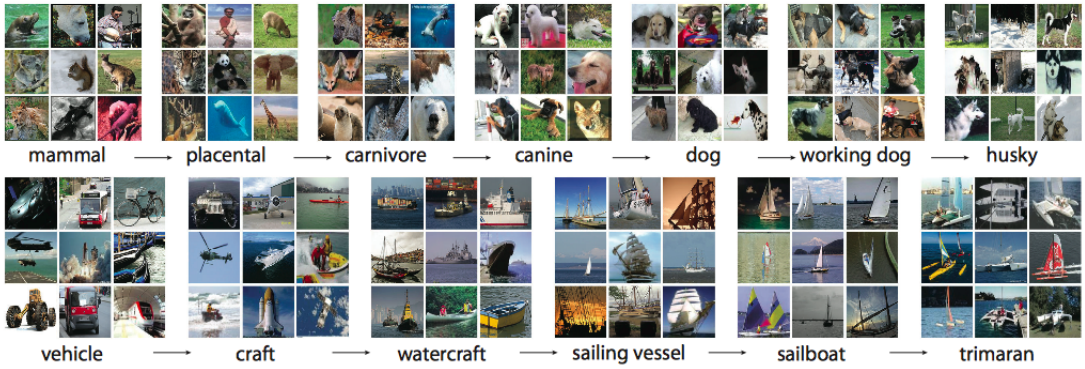


Figure 2.38: A snapshot of two root-to-leaf branches of ImageNet: the top row is from the mammal subtree; the bottom row is from the vehicle subtree. For each synset, 9 randomly sampled images are presented. Extracted from [106].

2.6 Visualization of Image Collections

After the clustering process is performed and the images are grouped relative to their similarity in content, there is still probably a large number of images in each group. For this reason, there is a need for developing a scheme for image visualization that allows the user of the application to browse the clusters obtained.

One alternative would be to present for each cluster a few representative images, as done in [114]. The problem with this method is how to pick these representative images, since they would need to represent well the content and similarity found by the algorithm in that cluster.

Other approach would be to represent each cluster by one image or a set of modes represented in that cluster. These images would be the combination of all the images in the cluster (or a few for the case of the modes). In [1], a technique called weighted image averaging was used. The system proposed, AverageExplorer, is an interactive system that was developed recently for this purpose. The idea of AvergaeExplorer is to summarize the visual data by weighted averages of the image collection, with the weights reflecting the user-indicated image and feature importance. It is an interactive system since the user can "edit" the average image by using various types of brushes and warps provided, similarly to a image edition software. The input of the system is image collection representing the same semantic concept (for instance, "cats", "shoes", "Paris", etc), but with a variety of appearances (images retrieved using a search engine). The output is a set of average images that depict different modes in the data. Figure 2.39 presents the results of the AverageExplorer applied to a query "Kids with Santa" after editing and extraction of informative modes.

Given a set of N images $\{I_1, \ldots, I_N\}$, the simple weighted average (I_{avg}) of the images is given by:

$$I_{avg} = \frac{\sum_{i=1}^{N} s_i \cdot I_i}{\sum_{i=1}^{N} s_i} \tag{2.82}$$

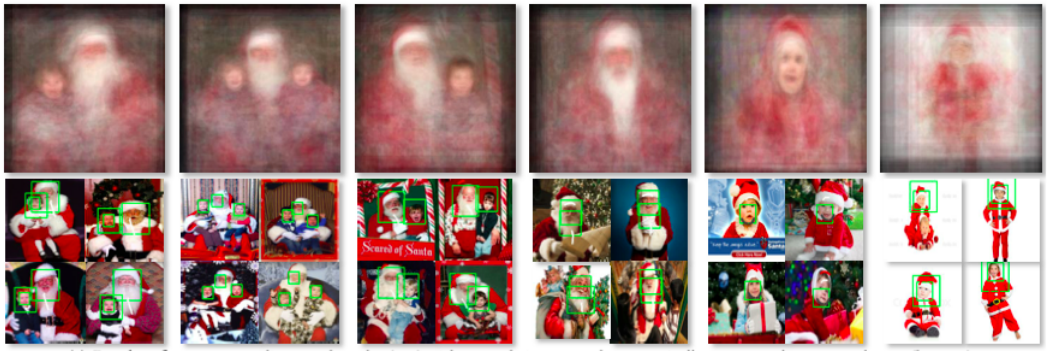


Figure 2.39: Results obtained by AverageExplorer: the many informative modes and examples of images and patches within each mode. Extracted from [1].

where the weights, s_i are initialized as $s_i = 1/N$ for all i. Once the editing begins, the weights are updated to reflect how well that image matches the user's edits:

$$s_i = \sum_{i=1}^{T} match(w^t_{user}, I_i) \tag{2.83}$$

where s_i are the weights updated after T edits, w^t_{user} represents the user edit at time t, and match(.) returns how similar an image I_i is to the user edit w^t_{user}.

Three brush tools are provided [1]:

- Coloring brush tool: The coloring brush allows the user to paint on the average image by adding color strokes. This changes the weights by assigning a higher weight to the images with the selected color in that spacial area.

- Sketching brush tool: The sketching brush tool allows the user to add line strokes to the average image. It is useful for adding fine details (for instance, adding glasses when exploring faces).

- Explorer brush tool: The explorer brush tool is the most important tool of the AverageExplorer system. The goal is to allow the user to find hidden information in the image collection. The main idea is to collect local patches situated in the same spacial position across all database images, and cluster them into a set of visually informative modes.

Assuming that the images are taken from the Web, using a search engine, for example, even if they have the same semantic concept, that does not mean that it will be spatially aligned. Therefore, if no alignment is performed, the resulting averages will be blurry. AverageExplorer provides, during the interactive edition, an image wraping algorithm which transforms the images leading to image stacks with increasing image alignment.

Figure 2.40 shows three image collections of "cats", "bikes" and "horses". In the left, the initial averages are presented. It is a very blurry image and almost no pattern can be distinguished. Next, these averages are edited four times and the resulting images are displayed ("final average"). It

can be seen that these images are very sharp and the objects are clearly represented. To the right, the versions of the final average with no alignment are shown. Compared to the final images with alignment, it is clear why the alignment step is needed in this process, since the quality is much higher.

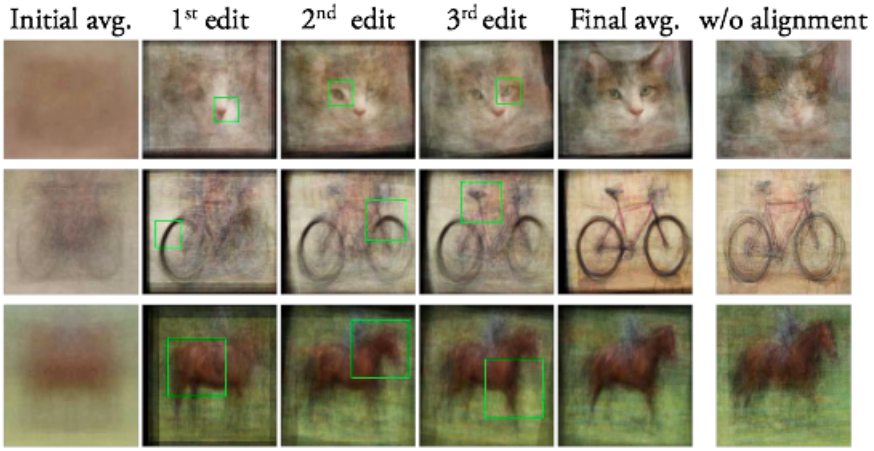


Figure 2.40: Examples of average images edited and aligned compared to the unaligned versions, using the AverageExplorer interactive visualization tool. Extracted from [1].

To conclude, although the AverageExplorer system can successfully provide an effective visualization of an image collection, it requires user interactivity, which means that this is not an automatic process. Still, these ideas could potentially be used to create an automatic version, if needed.

Other image visualization systems are also presented by other authors, for example, [115], which proposes a solution for the problem of scene summarization which examines the distribution of images in the collection to select a set of canonical views to form the scene summary, using clustering techniques on visual features.

First, local image descriptors are obtained for all the images. Then, the features are matches between every pair of images and an incidence matrix is computed using the pairs of images that match in the set.

In order to select views (images) as a summary for the scene, the concept of image likelihood is used, which states that an image should be included in the summary if it is similar to many other images in the input set. The similarity between images is computed as [115]:

$$sim(V_i, V_j) = \frac{|V_i \cap V_j|}{\sqrt{|V_i||V_j|}} \tag{2.84}$$

The equation 2.86 measures the cosine angle between the normalized features of the two images. If the views have the same number of features, it can be defined as:

$$sim(V_i, V_j) = V_i \cdot V_j \tag{2.85}$$

And therefore, the likelihood of a view is simply:

$$lik(V) = \sum_{V_i \in \nu} Vi \cdot V \tag{2.86}$$

where ν is the set of target image views.

The algorithm attempts to maximize the following function:

$$Q(C) = \sum_{V_i \in \nu} (V_i \cdot C_{c(i)}) - \alpha |C| \tag{2.87}$$

where $C_{c(i)}$ is a canonical view. This objective function promotes the orthogonality of canonical views. Then, a greedy algorithm selects the final set of canonical views.

This system attempts to provide a much better image browsing experience than available nowadays, for example, for the image collections of the website Flickr. The authors used a set of 500,000 photos from the city of Rome. Most of the photos contained user tags. However, some tags were missing, some were misleading and others uninformative. Therefore, they presented a browsing system that can work either with no tags or by using an algorithm, it can incorporate the tag information to produce better results.

An example of the results obtained by this system is shown in Figure 2.41. Here, given a set of images tagged as "Vatican", the algorithm was able to select 10 canonical views to server as a summary.

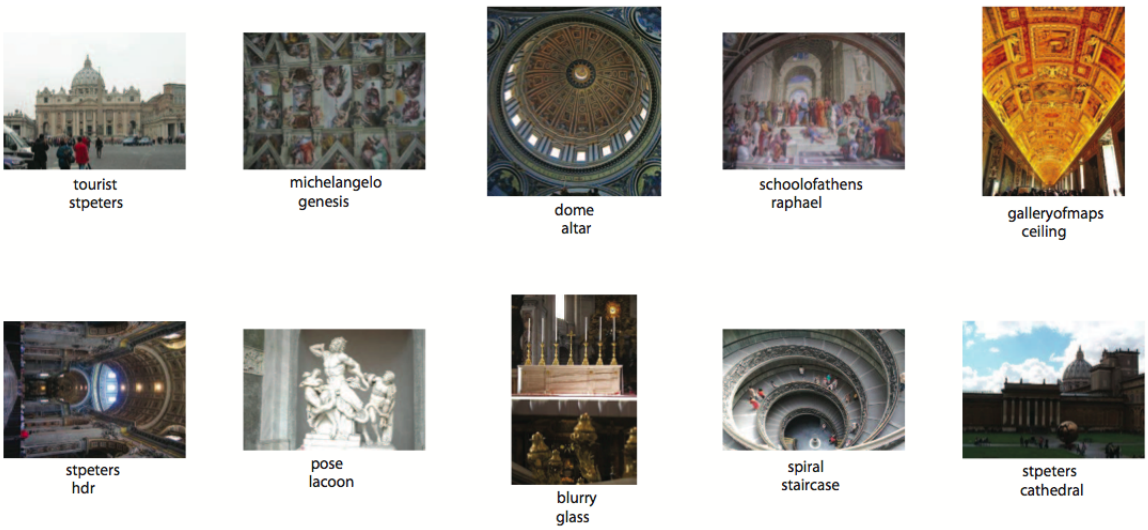


Figure 2.41: A summary of 10 images extracted from a set of 2000 images of the Vatican computed by our algorithm. Extracted from [115].

2.7 Related Work

In this section, some relevant and recent work closely related to the topic of this thesis will be presented and discussed. In general, image clustering is the focus of this work, and therefore, the studies mentioned will be mainly in that area of research.

Clustering has been used extensively for many types of data from documents to images. Image clustering has been used for many applications such as to improve the performance of CBIR systems [116], image annotation and indexing [117] and image summarization [115]. The most used algorithm for image clustering has been K-means due to its simplicity.

In terms of recent works in this area of Web image clustering, it is important to refer the predecessor of this thesis, which is [114]. In this study, the problem was the same (find patterns and clusters in images shared via Twitter). As for the approach, the author used local image descriptors, more specifically the SIFT descriptor (sec. 2.4.1.4.1) to extract features from the images. Then, using a Bag-of-Features technique (sec. 2.4.2.1), a visual vocabulary of 1000 words was created using K-means clustering algorithm (sec. 2.2.1.3) on the descriptors of a random sub-sample of the total number of images in the database. After that, a frequency histogram was created for each image in relation to the occurrence of the visual words. Next, a distance matrix was computed for all the images in the database using the euclidean distance (sec. 2.2.1.1). Finally, the clusters were obtained using DBSCAN clustering algorithm (sec. 2.2.1.5). Each clusters obtained was then, represented in a matrix of 9 images, chosen randomly, in the visualization tool implemented.

Another recent work, [117], also used the BoF approach for image clustering, with the final purpose of image indexing. The difference is that the authors proposed two new algorithms, SAIL (Summation-based Incremental Learning) and V-SAIL (Summation-based Incremental Learning with Variable Neighborhood Search), which are extensions of the Info-Kmeans algorithm [118], for clustering high-dimensional data in an attempt to deal with the high sparsity of image data (the features extracted from the images). The Info-Kmeans clustering belongs to a type of clustering called information-theoretic clustering, in which the loss of mutual information due to the partitioning is the minimized [117]. More specifically, Info-Kmeans uses a K-means clustering algorithm with a KL-divergence [119] distance measure. One of the problems with this algorithm is the *zero-feature dilemma* which is caused by the number of features in the data that have a zero value (e.g. the histogram of the BoF method). Apart from other types of data, the authors also presented an experimental design to test the algorithms to the application of image clustering. For that, an image dataset called *Oxford 5K* was used, which contains 11 classes of images retrieved from the social network Flickr. In terms of feature extraction of the image data, the authors used the BoF approach with 1 million visual words. Other interesting procedure implemented in this work, prior to the image clustering, was the use of noise removal, using an algorithm named Cos-Miner [120]. This noise refers to images that belong to a given class but do not present visual characteristics that could distinguish and assign to the given cluster, and therefore, are seem as noise. The results with this challenging dataset, show that the algorithms proposed improve the performance of the Info-Kmeans algorithm, for highly sparce data.

In [121], the authors proposed a different approach to the usually applied BoF model. The idea was to integrate the processes of image clustering and codebook learning, which are traditionally performed separately. The reason behind this idea is that the two processes are related and therefore, the correlation between them should not be ignored. The relationship is that the codebook representation is the input for the clustering algorithm and thus, influences the clustering results, and, at the same time, the clusters obtained by the clustering algorithm can be helpful to guide the codebook learning by serving as supervised information. For this reason, the authors propose two relevant contributions which are a Double Layer Gaussian Mixture Model (DLGMM) to integrate clustering and codebook learning and a Spatially Coherent Double Layer Gaussian Mixture Model (SC-DLGMM) which uses a Markov Random Field (MRF) [122] to incorporate the idea of spacial coherence between neighboring patches in the process of codebook training. This last contribution attempts to solve the problem of the BoF approach to ignore the spacial distribution of the local patches by building a model in which patches near each other are more likely to be assigned to the same visual word. The experiments conducted in [121] demonstrate that integrating image clustering and codebook learning can improve the codebook and that incorporating spacial coherence may produce better final results.

In [123], the authors attempted to cluster images in order to detect landmarks and events in large image collections. This is very useful since, usually, images shared by users online capture experiences related to some landmarks (e.g. touristic sites) or events (e.g. a concert). The clustering is performed using the notion of community detection in similarity graphs. Both visual and tag information is used to build the similarity graph. Then, the authors classify each cluster as a landmark or event using temporal, spacial, social and tag characteristics of the image cluster. For the process of clustering, the different types of input (visual and tag) generate separate image similarity graphs, which are then combined in order to create a hybrid graph. The visual features extracted from the images are the SIFT descriptors (sec. 2.4.1.4.1), followed by a BoF model (sec. 2.4.2.1) with a vocabulary of 500 words and K-means clustering. After that, similarity measures are computed for the image collection's features and the 20 most similar images of each image are considered their neighbors on the graph, given that they pass on the similarity threshold chosen. On the other hand, for the tag similarity graph, an inverted table of tags and images is obtained. The edges in the graph are weighted by the number of times a pair of images are found together in the tag list. Additionally, tags which have many images associated with it are excluded to avoid suspicious edges and to reduce the computational time required. After that, a hybrid graph is computed and community detection based clustering is performed. For the classification of images into the two large categories (landmarks or events), a supervised learning approach is used based on labeled data acquired (training examples). Finally, spacial information provided by some tagged images assist in predicting the labels of the images classified as landmarks. The evaluation experiments were done using a dataset of 207,750 images collected from Flickr using a geoquery (query on the location) of the city of Barcelona, which demonstrated the efficiency and performance of the proposed techniques.

Due to the fact that the dimensionality of the feature vectors of an image after the image

description procedure is usually very high, there is the problem of the *curse of dimensionality* [124], which states that the performance of a learning algorithm will tend to decrease with the number of dimensionalities, if the number of samples remains unchanged. This is caused by the sparcity of the data in a high-dimensional space. For this reason, many attempts have been made to develop more efficient algorithms for these types of data. One of these approaches is to perform dimensionality reduction, such as Principal Component Analysis, PCA (sec. 2.4.1.4.2) and Linear Discriminant Analysis, LDA [125], on the original feature set in order to project the dataset into a space with less features. LDA is a supervised method which attempts to reduce the dimensionality of the feature set while preserving as much of the class discriminatory information as possible. More specifically, for image clustering applications, a simple solution has been to project the high-dimensional data into a low-dimensional subspace using PCA (as a pre-processing step), and then apply a clustering algorithm like K-means. Recently, LDA has shown to be very useful and to produce better results than PCA. Usually, the K-means algorithm is first used to obtain the clusters, which are labeled. After that, the LDA algorithm is performed to obtain the most discriminative subspace. Examples of works in those directions are [109], [126] and [127].

Other approach to describe an image collection, similar to a similarity graph is presented in [128], where a type of graph named *Image Webs* is introduced. These graphs are different than the similarity graphs discussed until now since each node no longer represents and image but a region of collected pixels in an image. Therefore, what is connected is distinctive regions in the image from the collection. The process of extracting these regions is called *Affine Cosegmentation*, which implements a local-feature matching technique between a pair of images at a time. The result of these processes is a graph which connects components of images and could be potentially useful for exploring an image collection and for doing image clustering. Another interesting work that shares similar ideas is [129]. In this paper, the authors propose an automatic supervised system that seeks to find visual elements (sub-regions of the image), that are at the same time, frequently occurring, within the class and informative (can distinguish between different classes). The specific application for this work was to geographically classify images from the Google Street View. The idea could potentially be useful for this thesis since it could be an alternative to computing local image descriptors that are harder to understand and to evaluate by the human eye.

2.8 Discussion

To sum up, social networks (Section 2.1) have been the source for many research in the field of data mining. In terms of the nature of the data analyzed, research has been focused, mostly, in the textural content shared on the Web. Still, researchers have also been exploring the visual information contained in the huge image collections, mainly in the field of CBIR (Section 2.3). The images are characterized by image descriptors (Section 2.4) obtained directly from the data (low-level) or by further processing (mid-level). However, there is also the issue of capturing the essence or semantic meaning of the images, which has been a constant challenge. The next step is to obtain the same level of maturity found in text mining for the topic of image mining.

In order to access the performance of a image mining system, many image databases (Section 2.5.1) have been created and some of them are widely used as benchmarks to compare algorithms. Also, a number of different measures can be applied if a database is not available.

Additionally, for presenting to the users the final results from the clustering analysis performed, visualization tools (Section 2.6) are needed, and could rely on applying aligned averages of the images from the cluster or to present some representative examples.

In relation to previous similar work (Section 2.7), although image clustering has been studied in image processing and computer vision fields, it remains a topic of constant improvement due to the fact that a clear and obvious solution for this problem has not yet been presented. Nonetheless, there are many works describing different algorithms. However, most of them concentrate on a small application or a small dataset (with low variability). Also, it is clear that the most popular approach is to use local image features, then apply a Bag-of-Features method to create the features for a clustering algorithm such as K-Means for obtaining the final clusters. Still, there are also other approaches using techniques such as graph theory to represent similarity between image in the collection or using dimensionality reduction techniques in order to other try to overcome the problem of high-dimensionality of the feature vectors produced. In the end, the goal is to provide a summarization of the huge amount of information provided by the images shared in the Web.

Chapter 3

Study 1: Experimental Evaluation of the Bag-of-Features Model for Image Clustering

3.1 Introduction

As described in section 2.4.2.1 The Bag-of-Features (BOF) is a model that aims to represent images as an orderless collection of image features without the use of any spatial information. Each image is represented by a frequency histogram of visual words from a codebook. A visual word is a local segment in an image, defined either by a region (image patch or blob) or by a reference point with its neighborhood. The name comes from an analogy with the Bag-of-Words representation used in textual information retrieval (text mining).

Although the model is quite simple in terms of the implementation, there are several steps in which parameters and algorithms need to be chosen. This chapter aims to assess the performance of this model for the application of unsupervised learning for a set of images, also called image clustering. Additionally, it aims to provide valuable insight on the different steps of the model and to compare different algorithms in order to achieve the best performance for a given dataset.

The fundamental difference between supervised learning (e.g. classification) and unsupervised learning (e.g. clustering) is that the data is not annotated and thus there is no previous information about the and categories. For this reason, the methods used for this purpose aim to find an underlying structure of the data and obtain relevant partitions.

The process of the BOF model and the main steps are summarized in Figure 3.1. As shown in this Figure, there are three main parts in the BOF model designed for image clustering. The first one is the image description in which the input images from the dataset are processed by first detecting keypoints or patches and then describing them using a certain strategy (as described in Section 2.4.1.4). The number of keypoints per image is a parameter that can be varied in the implementation for almost all the algorithms tested. The next step is the codebook learning where a portion of the feature vectors from the images are used in order to obtain a codebook of

visual words. Here, the codebook size is a very important parameter that can be specified to obtain different codebooks. The following step is the BOF representation of the images where each image is represented by a histogram of frequencies based on the codebook obtained. The histograms are then weighted and normalized following a chosen methodology. Finally, the images are clustered using a clustering algorithm of choice.

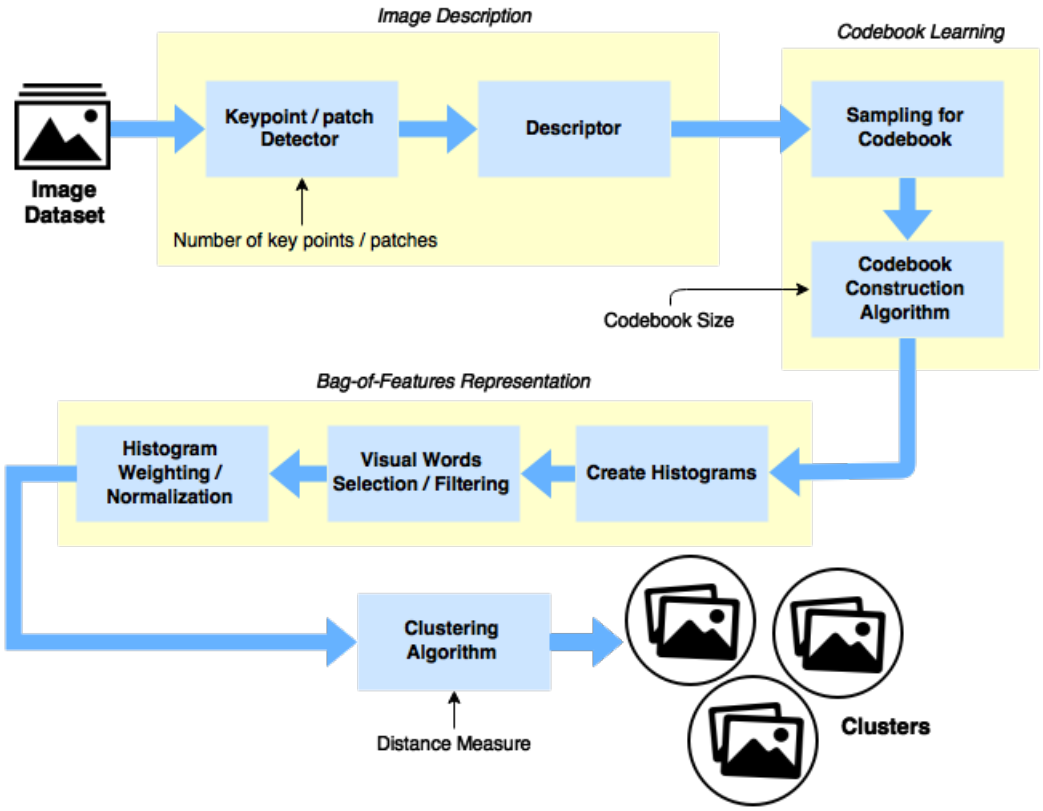


Figure 3.1: Bag-of-Features model for image clustering, presenting the main steps and parameter setting of the system.

The next sections will cover the following material: Section 3.2 will present similar studies about this topic, Section 3.3 will provide useful information and details concerning the test setting and design, Section 3.4 will present the results of the experiments and small discussions and finally Section 3.5 will provide some final remarks and conclusions.

3.2 Similar Studies

Due to the popularity of the BoF model for many applications in the field of Computer Vision, a number of works have been focused on evaluating its performance. Moreover, due to the great number steps needed to apply the model to a given problem, these studies also compare different strategies for each of the steps. For instance, in [130] the authors presented the results of an experimental study concerning the BoF model applied to the problem of image classification. Several key steps of the model were tested using different algorithms and parameters, including

the detection of the interest points, the size of the codebook and the histogram normalization procedure. Their results show the most influential parameter is the number of patches extracted from the images. Additionally, they have also determined that the codebook learning method does not have a significant impact on the performance provided that even randomly sampled codebooks also performed fairly well.

Another empirical study presented in [131] evaluated the impact of applying techniques used in text categorization to the BoF model for the application of scene classification. More specifically, they have tested, among others, the use of term weighting, stop word removal and feature selection. The results indicate that these techniques successfully improve the classification results. Other example of a similar work that propose and evaluate the use of text classification techniques for the BoF model is [132].

The main contributions of this study are: (1) the experimental analysis of the BoF model for image clustering, (2) the addition of a number of steps and algorithms (e.g. sampling the features for codebook learning and visual words selection), (3) the evaluating the techniques proposed originally for text mining (e.g. term frequency weighting) and (4) the proposal of an undersampling technique for the features obtained from the images.

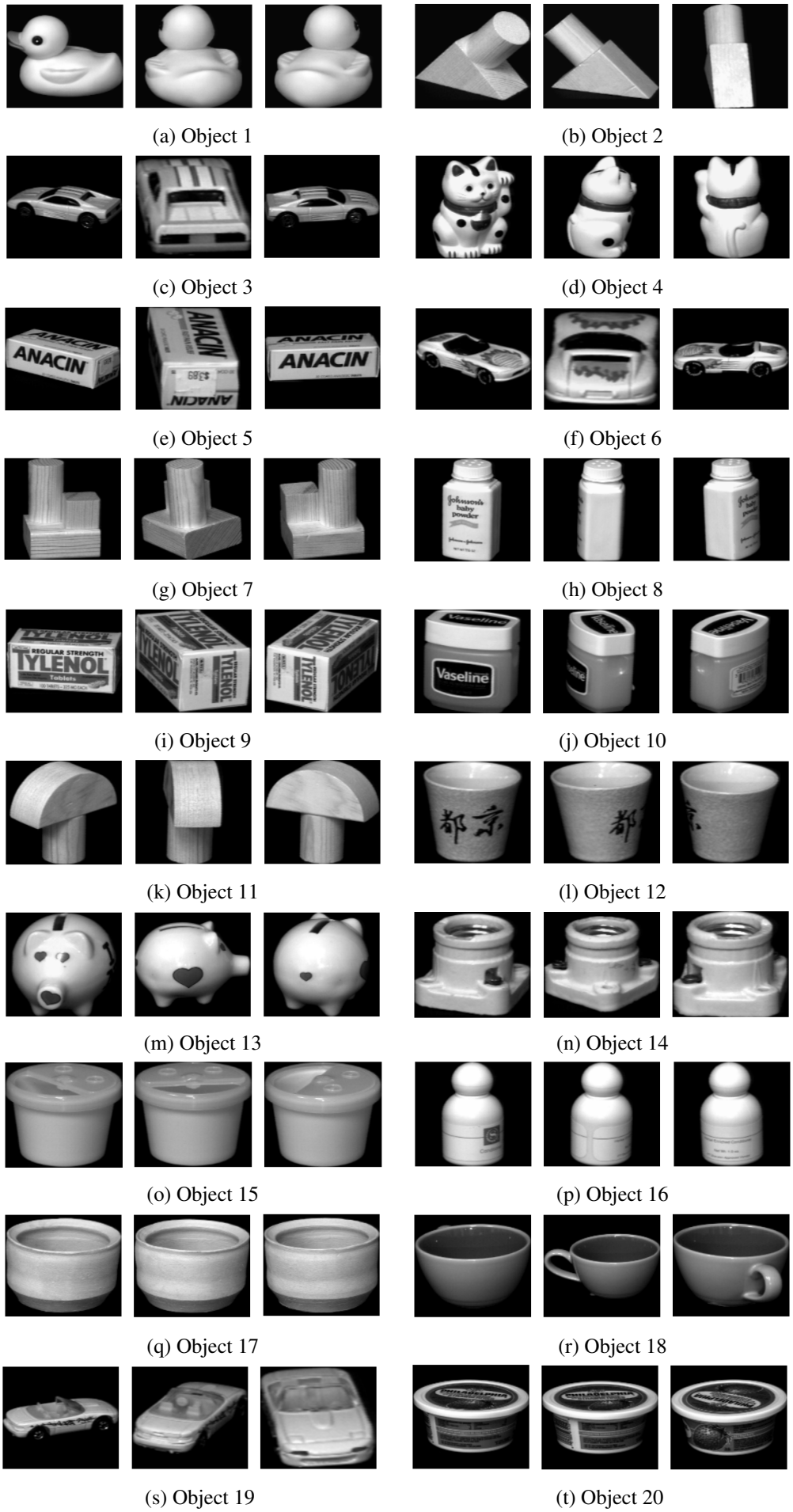
3.3 Experimental Design

3.3.1 Datasets

Three datasets were used in this empirical study. The first one is the popular Coil-20 dataset [102], which is an object-based dataset created for the purpose of object recognition. It is composed by 1440 small pictures of size 26×26 pixels divided into 20 classes of objects taken under different perspectives (lightning, rotation, etc). The second dataset used was the Natural and Urban Scenes dataset [86], which is made from 8 nature and human-made scenes such as coastlines and buildings. It has 2688 images which have a dimension of 264×264 pixels. Finally, the last image dataset used was the Event Dataset [100] which is composed by 1580 images of 8 sports event categories. The images from this datasetwere reduced to 500 pixels in the largest side for simplicity and to reduce the execution time.

These datasets were chosen in order to obtain different levels of difficulty and complexity for the purpose of image clustering. The Coil-20 dataset is the simplest due to the fact that it is an object dataset. The Natural and Urban dataset is a scene dataset containing simple images so it was considered medium difficulty. Finally, the Event dataset is composed by images of complex activities that may be hard for the model to distinguish, and therefore, it was considered as high difficulty.

Figures 3.2 to 3.4 show examples of images from all the classes from the datasets Coil-20, Natural and Urban and Event, respectively.



(a) Object 1 (b) Object 2

(c) Object 3 (d) Object 4

(e) Object 5 (f) Object 6

(g) Object 7 (h) Object 8

(i) Object 9 (j) Object 10

(k) Object 11 (l) Object 12

(m) Object 13 (n) Object 14

(o) Object 15 (p) Object 16

(q) Object 17 (r) Object 18

(s) Object 19 (t) Object 20

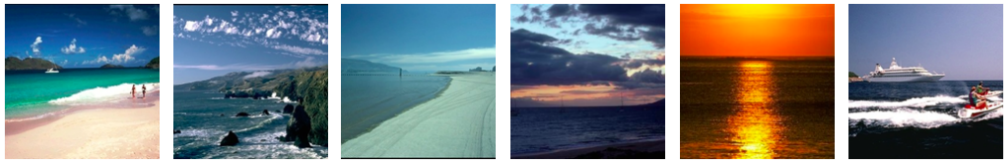
Figure 3.2: Example of images from the 8 classes of the Event dataset.

(a) Mountain

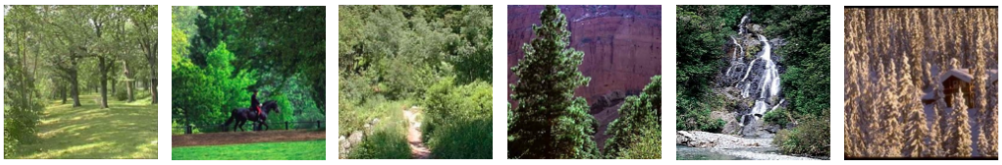
(b) Open Country

(c) Street

(d) Tall Buildings



(e) Coast



(f) Forest

(g) Highway

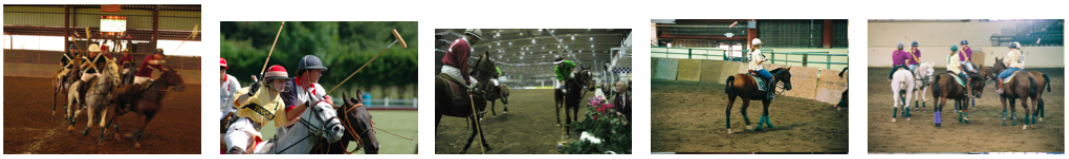
(h) Inside City

Figure 3.3: Example of images from the 8 classes of the Natural and Uban scene dataset.

(a) Bocce

(b) Badminton

(c) Croquet

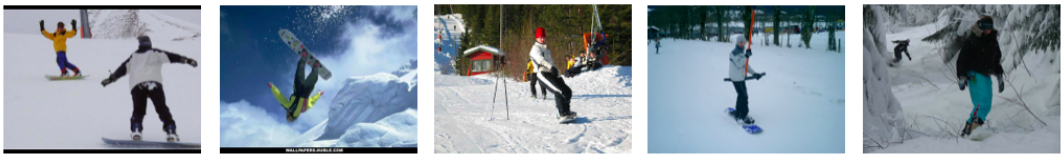


(d) Polo

(e) Rowing

(f) Sailing

(g) Rockclimbing



(h) Snowboarding

Figure 3.4: Example of images from the 8 classes of the Event dataset.

3.3.2 Test Methodology

As mentioned before, the first step of the BoF model consists of obtaining image descriptors for each image in the dataset. Usually, the adopted technique is to extract local image descriptors represented as keypoints in the image so that they are fully or partially invariant to afine transformations (rotation, scale and translation) and to luminosity changes. In this way, these patches can be matched in similar images. Among the different methods, the detectors used in this study were: SIFT (Scale-Invariant Feature Transform) [77], SURF (Speeded Up Robust Features) [81], FAST (Features from Accelerated Segment Test) [133], STAR - derived from CenSurE (Center Surrounded Extrema) [85] and ORB (Oriented FAST and Rotated BRIEF) [82] and the descriptors were: SIFT, SURF, BRIEF (Binary Robust Independent Elementary Features) [84], ORB, and FREAK (Fast Retina Keypoint) [134].

All of these algorithms were described in Section 2.4.1.4. To sum up, the pair of detectors and descriptors SIFT and SURF are the most popular choices although not being fast enough for real-time applications. In contrast, FAST and STAR are among the fastest detectors. However, they tend to produce the most number of irrelevant interest points, which could generate noise. In relation to the descriptors, they can either produce numerical or binary features. Unlike SIFT and SURF, the descriptors BRIEF, ORB and FREAK produce binary feature vectors. Finally, the ORB algorithm is an optimized version of the FAST detector and the BRIEF descriptor.

Similarly to [130], a random generator of patches (RANDOM) was also used. It works by randomly sampling the output of the a DENSE detector [135], which produces a regular grid of interest patches.

In Figure 3.5, the five detectors used are applied to an example image taken from the Event dataset. All the detectors were used with their default parameters. Note that the local descriptors are computed using the grayscale image and therefore, do not use any color information. By analyzing the images, it can be seen that, by default, the SURF and FAST detectors are the ones that obtain the most keypoints. Also, they do not only find keypoints in objects (in this case, the horse and person) but in the background as well. However, they obtain many overlapping keypoints. In contrast, the ORB and STAR detectors are the ones that obtain the fewest number of keypoints and they obtain more keypoints within the objects of the image. The SIFT detector seems to be the one that is able to find the most appropriate keypoints.

After the descriptors for each image are obtained, the codebook learning method is performed. For this purpose two clustering algorithms were selected: K-Means [28] and Mini Batch K-Means [39]. These algorithms were chosen due to the high scalability property which is required for the computation of the codebook, since both the number of data examples and dimensionality are very large. K-Means is by far the most popular algorithm for this application and has been used in almost all the works that applied the BoF for either image classification or image clustering [136, 117, 123]. The Mini-Batch K-Means algorithm is an online version of K-Means which allows the use of small data batches to update the centroids instead of using all the data at the same time. This method is several times faster than the K-Means algorithm but generally produces slightly

(a) Original image (b) SIFT detector: 1371 kp (c) SURF detector: 2995 kp

(d) FAST detector: 4419 kp (e) ORB detector: 500 kp (f) STAR detector: 335 kp

Figure 3.5: Example of application of the five different detectors: SIFT, SURF, FAST, ORB and STAR, using the default parameters. Here, kp stands for keypoints.

worse results. Additionally, with the aim of testing whether the codebook learning algorithm is significantly relevant to the performance of BoF model, the last methods adopted for constructing the codebook was using randomly selected feature vectors from the images (RANDOMV) and also entirely random vectors (RANDOM).

Next, instead of using all the features obtained from the images to produce the codebook, two types of sampling strategies were adopted. The first one is simply considering all the vectors as a unique group and selecting random keypoints from that group. However, given that some images generate more interest points than others, we believe that this could potentially have a negative impact on the codebook and consequently on the performance of the model. The reason behind this argument is that generally the keypoint detectors tend to find more keypoints in regions where there is more contrast, and therefore, if a class of images has, on average, more contrast areas than other, the detectors will find more keypoints on the images of that class. Then, during the codebook learning algorithm, these will have a bigger impact on the decision of the visual words for the codebook, potentially creating a bad representation for the images that have less keypoints. Empirically, we found that scenes like forests or buildings produce a much higher number of keypoints in contrast to scenes such as sea or snow. For this reason, we tested a simple algorithm for adaptative sampling of the images in order to reduce the standard deviation of the keypoints detected per image. The algorithm first selects a random sample of images. Then, for each image that was chosen, it randomly selects a proportion of the keypoints in order to construct the codebook. This proportion of keypoints sampled per image depends on the relation between the number of keypoints of that image and the average number of keypoints per image of the entire dataset. This algorithm attemps to reduce the variations in the number of keypoints per image when selecting the visual words. Figure 3.6 shows the relationship between the proportion of features sampled per image based on the number of features it has. As can be seen by the

function, if an image has a number of features equal to the average number of features per image in the dataset, 50% of its features will be sampled and used for codebook learning. In contrast, if it has much less or much more features than average, less or more features will be selected.

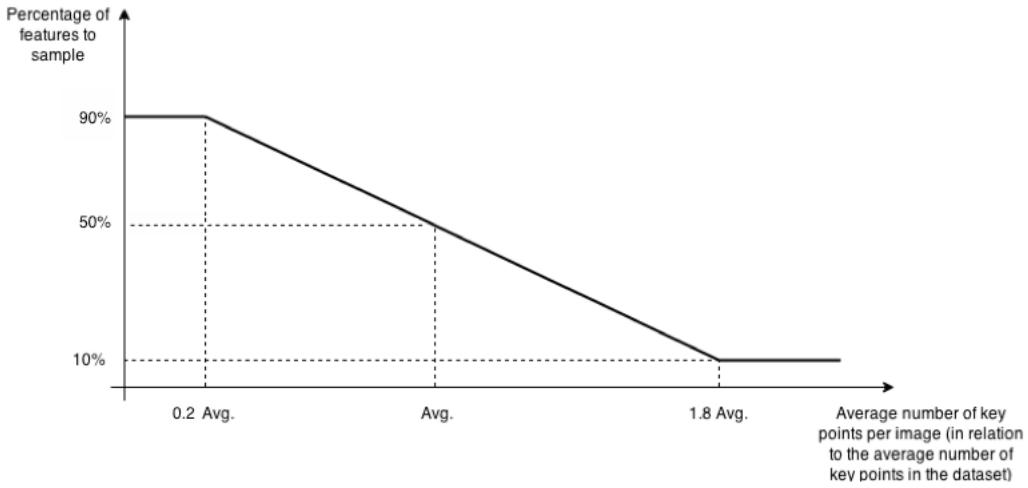


Figure 3.6: Adaptative sampling: function that relates the number of features per image and the proportion of features that is selected for obtaining the codebook.

After obtaining the codebook, each image is represented by a histogram of frequency of visual words from the codebook. These histograms are the features that will allow the comparison between images in order to obtain the final clustering result. However, before the features are ready for clustering, the histograms are weighted and normalized. Although usually applied in the area of text processing, these steps have also been applied to image data. The methods tested for weighting and normalization were simple binarization (reducing the features to a binary value that represents the presence or absence of that visual word) and different forms of the term frequency-inverse document frequency technique (tf-idf) [88].

In the last step of the process, the clusters are obtained using a given clustering algorithm. A number of different approaches were tested including K-Means, DBSCAN, BIRCH and Hierarchical Clustering. Additionally, some of these methods allow the choice of the dissimilarity metric used. That parameter was also varied in order to obtain different clustering results.

Since there is a huge number of possible combinations of algorithms and parameters, it is not possible to test all of them in finite time. For this reason, a test procedure was used. Each step was tested separately, thus for each step, only the algorithms and/or parameters of that step were varied maintaining the rest unchanged. This procedure required the choice of an initial setting.

3.3.3 Performance Measure

In order to assess the performance of the model for the purpose of unsupervised learning of image data, and given that the datasets are annotated (every image has a label), external clustering indexes were used. Therefore, an information-based method, Normalized Mutual Information (NMI) [42]

and a decision-based method, Adjusted Rand index (ARI) [137], were chosen. These measures were described in Section 2.2.3.1.

3.3.4 Implementation Details

The framework for testing the BoF model was developed in Python and it is openly available on GitHub [1]. The implementation required three libraries: OpenCV [135] library for the functions related to image description, Scikit-Learn [138] and Scipy [139] for the implementation of the machine learning algorithms tested.

3.4 Results

3.4.1 Image Description

3.4.1.1 Detectors and Descriptors

First, the different detectors and descriptors for the stage of image description were tested. For these tests, all the other settings of the BoF model for image clustering were fixed. The K-Means algorithm was selected as the codebook learning algorithm and the final clustering algorithm. Also, the size of the codebook and the proportion of images to be used for the process of codebook learning were fixed for each dataset. The values of the parameters used can be found at Table 3.1. Additionally, as the K-Means clustering algorithm does not take into account if the features have different scales, a whitening transformation of the features from the histograms was applied prior to the application of the K-Means clustering algorithm. Finally, the number of keypoints extracted per image vary because it could not be successfully fixed for each of the detectors due to differences in their nature.

Table 3.1: Values of the codebook size and proportion of keypoints for the codebook learning step used in the first tests of the detectors and descriptors.

Datasets	Codebook size	Proportion of keypoints for codebook
Coil-20	110	0.3
Natural and Urban	300	0.05
Event	500	0.05

The results of this analysis for all three datasets can be found in Table 3.2, where the performance of the best and the worst combination of detectors and descriptors are presented. The table contains the following information: average ARI, standard deviation of the ARI, average NMI score and standard deviation of the NMI score, average number of keypoints per image and finally a relative qualitative value for the computational time required. In order to obtain the average and the standard deviation of the indexes, every test was repeated 10 times.

[1] The source code of this project can be found in the link: https://github.com/marianafza/ImageClustering

By analyzing the results for the Coil-20 dataset, it can be seen that the best performing combination was the FAST detector with the FREAK descriptor with an average ARI of 52.2% and an average NMI score of 78.7%. Also, as expected, the SIFT detector with the SIFT descriptor combination also performed fairly well since it usually has a good performance for object recognition tasks due to the level of invariance to several transformations [77]. Overall, the FAST detector obtains good results, which could be related to the higher number of keypoints per image it is able to detect. In contrast the worst combinations of detectors and descriptors for this dataset was found to be the use of the RANDOM detector with the SURF descriptor. In general, the RANDOM detector performed poorly for this dataset. These results are not surprising since the images represent objects with a black background which will most likely generate a great number of keypoints and will be seen as noise for the BoF model.

Table 3.2: Performance of the BoF model for image clustering using different detectors and descriptors in order to extract the features in the images from the three datasets.

Dataset	Detector	Descriptor	Avg ARI	Std ARI	Avg NMI	Std NMI	Avg. # of keypoints / img.	Computational time
Coil-20	**FAST**	**FREAK**	**52,2%**	**3,9%**	**78,7%**	**1,5%**	**88**	**High**
	FAST	**SURF**	**48,8%**	**4,3%**	**76,2%**	**1,5%**	**88**	**Medium**
	SIFT	**SIFT**	**46,4%**	**4,9%**	**75,3%**	**2,1%**	**51**	**High**
	RANDOM	FREAK	32,8%	2,8%	53,7%	2,0%	50	High
	ORB	ORB	19,3%	2,1%	40,6%	1,8%	11	Low
	RANDOM	SURF	12,4%	0,9%	28,4%	1,4%	50	Medium
Natural and Urban	**STAR**	**SIFT**	**34,2%**	**2,2%**	**46,0%**	**1,6%**	**130**	**Low**
	RANDOM	**SIFT**	**31,2%**	**0,8%**	**41,8%**	**1,2%**	**500**	**Medium**
	SURF	**SIFT**	**27,1%**	**1,6%**	**38,7%**	**1,0%**	**332**	**High**
	SIFT	SURF	14,0%	1,1%	25,2%	1,4%	393	Medium
	STAR	BRIEF	13,8%	1,3%	23,4%	1,4%	130	Very Low
	FAST	FREAK	11,8%	0,5%	21,1%	0,4%	851	Low
Events	**RANDOM**	**SURF**	**18,7%**	**0,8%**	**27,1%**	**0,6%**	**1000**	**High**
	STAR	**SIFT**	**16,5%**	**1,0%**	**26,5%**	**0,9%**	**554**	**High**
	SURF	**SIFT**	**15,9%**	**0,8%**	**25,9%**	**0,3%**	**972**	**Very High**
	FAST	BRIEF	5,4%	0,3%	13,0%	0,2%	1038	Low
	FAST	FREAK	5,2%	0,2%	11,1%	0,4%	972	Medium
	ORB	ORB	4,1%	0,3%	8,1%	0,5%	957	Low

In relation to the Natural and Urban dataset, the best performing descriptor is definitely the SIFT descriptor with an average ARI of 32% and an average NMI score of 42%. An interesting result is that the RANDOM detector achieved very good results, by which can be concluded that using specific interest point detectors can yield worse results for scene datasets than randomly selecting patches from the whole image. Additionally, in spite of achieving the best results for the Coil-20 dataset, the combination of the FAST detector with the FREAK descriptor was the worst performer for the Natural and Urban dataset.

Finally, concerning the Event dataset, the descriptors SIFT and SURF achieved the best results (>16% ARI and >26% NMI score) in contrast to the binary descriptors BRIEF, FREAK and ORB (<5% ARI and <13% NMI score). It is clear from the poor results that this is a very challenging dataset with regards to unsupervised learning.

In relation to the computational time, the SURF and SIFT detectors and descriptors are among the slowest algorithms. Therefore, for larger datasets and/or real-time applications a more efficient

combination of detectors and descriptors should be selected, for instance, using the FAST, STAR or RANDOM detectors and the BRIEF or FREAK descriptors.

In conclusion, after analyzing these results, it can be observed that the performance of the BoF model applied to unsupervised learning of image data highly depends on the algorithms for the description of the images. Also, the choice of the algorithms is dependent of the dataset in which one is working with.

For the following steps, only one of the pairs of detectors and descriptors were selected for each dataset. Therefore, the combination of the FAST detector with the FREAK descriptor, the RANDOM detector with the SIFT descriptor and the RANDOM detector with the SURF descriptor were chosen for further testing for the Coil-20, Natural and Urban, and Event datasets, respectively. In relation to the choice for the Natural and Urban datasets, although the STAR detector achieved the best results, we were unable to efficiently adjust its number of keypoints per image, and for this reason, the RANDOM detector was preferred.

3.4.1.2 Number of Keypoints and Codebook Size

In this step, several combinations of the average number of keypoints per image and the size of the codebook were tested. These parameters are correlated since the more features extracted from the images, the more diversity of visual words will exist and therefore, the size of the codebook can increase. Figure 3.7 presents the results for all the datasets. Here, the performance index was chosen as the NMI score since both the NMI score and the Adjusted Rand index followed the same trends.

As shown in the charts of Figure 3.7, regardless of the codebook size used, the performance almost always increases with the average number of keypoints per image. As referred in Section 3.2, this result was also obtained in [130] for the problem of image classification.

Another interesting conclusion, also observed in [130], is that the performance increases with the codebook size until a certain point in which the performance starts to go down. This behavior can probably be attributed to the curse of dimensionality [22]. The curse of dimensionality states that the accuracy of a learning algorithm decreases with very high dimensionalities due to unpredictable effects that comes with it. Additionally, it could be observed that the ideal size of the codebook increases with the complexity of the dataset.

Regarding the average number of keypoints per image and the size of the codebook used for the next steps of the BoF model, for the Coil-20 dataset, the ideal codebook size and number of keypoints per image were selected as 100 and 109, respectively. Next, in relation to the Natural and Urban dataset, the number of keypoints was chosen as 500 and the size of the codebook as 300 visual words. Finally, for the Event dataset, 1500 keypoints and 500 visual words were the values selected.

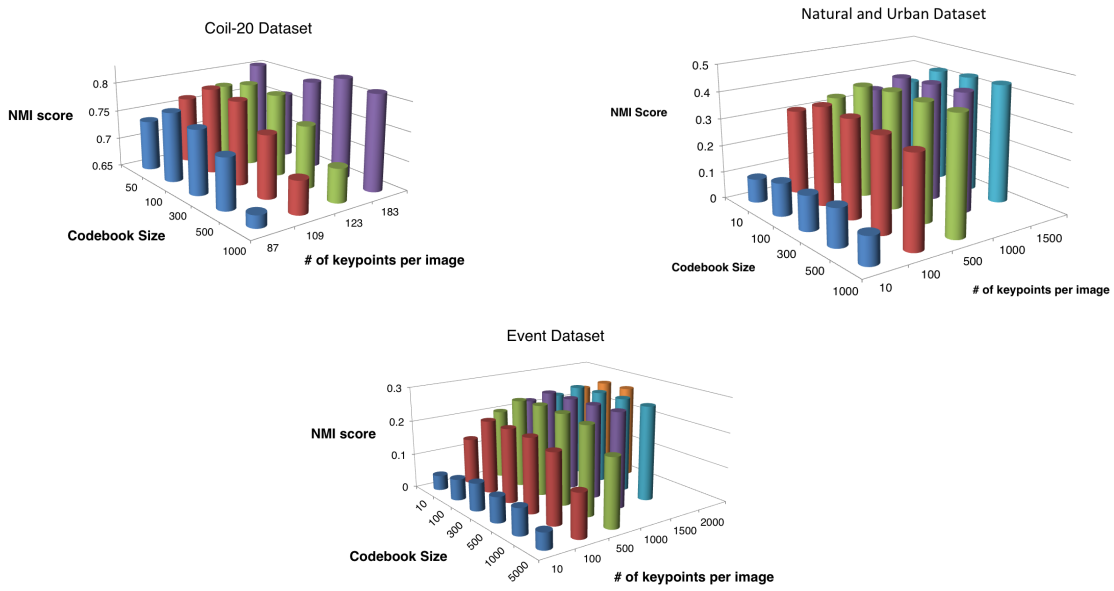


Figure 3.7: Results for the three datasets using different values for the average number of keypoints per image and the codebook size. The performance was evaluated using the NMI score.

3.4.1.3 Sampling for Codebook Learning

Next, the influences of the sampling technique and the proportion of features used for the codebook learning algorithm were tested. As mentioned before, two methods for sampling were evaluated, named in our testing framework SAMPLEP and SAMPLEI. SAMPLEP is the simple sampling technique using a proportion of all the features of all the images, while the SAMPLEI is the adaptative sampling technique described in Section 3.3.2. For both techniques, the number of features used was varied between a very small value and a large one, considering the time constraint for each dataset.

Considering that the RANDOM detector was chosen in the previous steps for the Natural and Urban dataset and the Event dataset and this detector extracts the exact same number of keypoints per image, it is not suited for the comparison of the sampling techniques mentioned. For this reason, the combination of the SURF detector with the SIFT descriptor was selected for these datasets (also due to its good performance).

The NMI scores for the three datasets are presented as charts in Figure 3.8. The chart's error bar represents the standard deviation of the NMI score. It is important to note that for the SAMPLEI method, the x-axis refers to the percentage of images used for codebook learning whereas for the SAMPLEP method it is the percentage of features. This is because SAMPLEI method downsamples the images with above average number of keypoints per image and therefore less features will be sampled.

In terms of the sampling algorithm, it can be seen that for the Coil-20 dataset, the SAMPLEI method performs, in average, slightly better than the SAMPLEP, and therefore, it was used in

the next testing stages. Additionally, in relation to the proportion of features to construct the codebook, the performance increases until approximately 60% and then decreases. Since using 60% of the images would require a great amount of computational time, the value chosen as the percentage for the further testing steps was 30%.

Considering the results for the Natural and Urban dataset and the Event dataset, it can be inferred that there is no clear method that performs the best. This behavior could possibly be related to the fact that these datasets are composed of more complex images which do not have significantly different number of keypoints per image in each category. To evaluate this, the total number of features extracted from each category of the datasets were compared. It was observed that the ratio between the category with the most features and the category with the least features was approximately 11 for the Coil-20 dataset, whereas for the other two datasets it was around 2.

For the reasons mentioned above, the percentage of features selected was 3% and 1%, respectively, for the Natural and Urban dataset and the Event dataset using the SAMPLEP method.

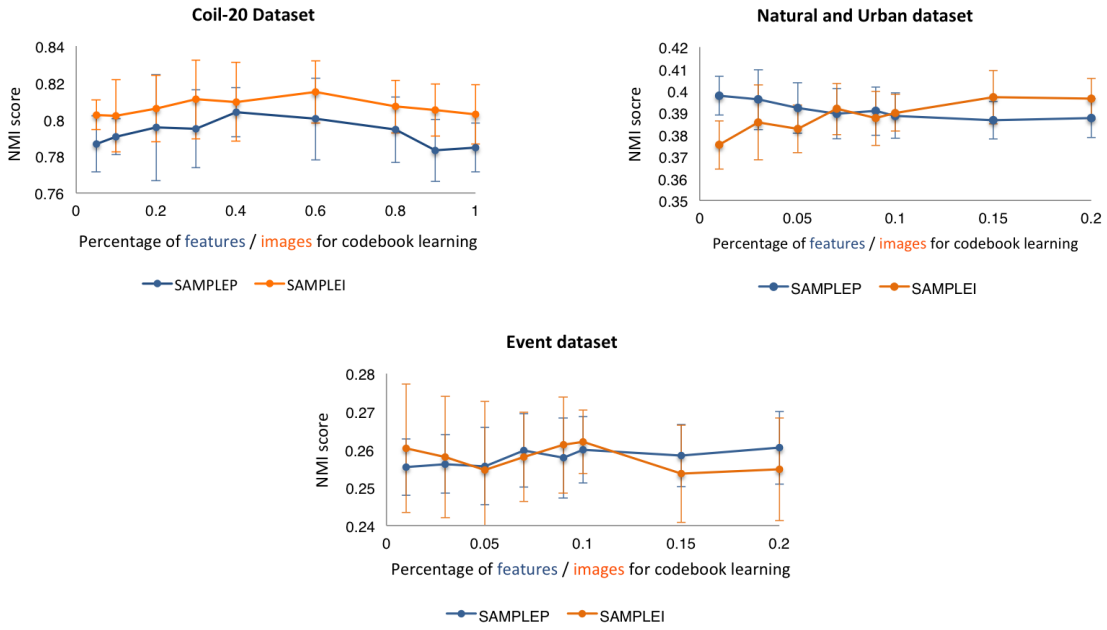


Figure 3.8: Results for the three datasets using different values for the proportion of images and features used for the codebook learning step and using two different methods for selecting or sampling these features SAMPLEP and SAMPLEI.

3.4.2 Codebook Learning Method

Regarding the codebook learning method, two algorithms, K-Means and Mini-batch, were tested due to their scalability property to deal with high dimensionality and high number of samples. The results are presented in Table 3.3.

For all the datasets, the K-Means was, on average, the best performing algorithm. Nonetheless, for the Natural and Urban dataset, the Mini-Batch had almost the same score, and therefore, is

preferred considering the less computational time required. As for the Event dataset, K-Means, Mini-Batch and Random Vectors got the same score, resulting in the choice of the Random Vectors for the same reason mentioned previously. For the Coil-20 dataset, there is a significant difference between K-Means and Mini-Batch and therefore, K-Means will be used in further testing stages. Finally, as expected, the completely random codebook got significantly poorer results in terms of the clustering validity indexes.

One of the reasons for the poor performance of the Random codebook is that, in practice only a small portion of the visual words obtained will be actually assigned to a keypoint. It was found, for example, that for the Event dataset, out of the 500 visual words, only 20 were used, in average. This is because the dimensionality space is so sparse that many visual words will be extremely far away from any keypoint.

By the analysis of the results, given that codebook obtained by randomly selecting feature vectors achieved good results, it can be concluded that the choice of the codebook learning method does not significantly influence the performance for image clustering, as long as it is not defined as all random vectors.

Table 3.3: Performance of the BoF model for image clustering evaluated by the NMI score for different algorithm for codebook learning.

Algorithm	Coil-20 Dataset	Natural and Urban Dataset	Event Dataset
K-Means	81.0%	42.8%	27.2%
Mini-Batch	77.9%	42.4%	27.4%
Random Vectors	75.9%	40.8%	27.7%
Random	53.9%	16.3%	21.7%

3.4.3 Histogram Weighting and Normalization

After obtaining the histograms of frequency of visual words for each image in the dataset, the normalization and weighting of the histograms can be performed.

For this purpose, five types of normalization and weighting procedures usually applied to the text processing were tested. Details of the techniques used are presented in Table 3.4. The variable $f_(t,d)$ refers to the frequency of the word t in the document (or in this case, image) d. The number of images is given by N and the number of images that have the visual word t is given by n_t. In the tf-idf variation, $N*$ is the total number of features (sum of all visual word frequencies) and n_t* is the total number of incidences of that visual word on all images. The results of the application of these normalization procedures for the datasets tested can be found in Figure 3.9.

By analyzing the charts in Figure 3.9, it is clear that no technique outperforms the others in all three datasets used. More specifically, for the Coil-20 dataset, the method that achieved the best performance was the tf-idf and the one that got the worst was the tf-idf variation. In relation to the Natural and Urban dataset, the best was simple binary normalization while the least was the tf-idf normalized. Finally, concerning the Event dataset, almost all methods got similar result apart from the simple binarization, which obtained significantly worse results.

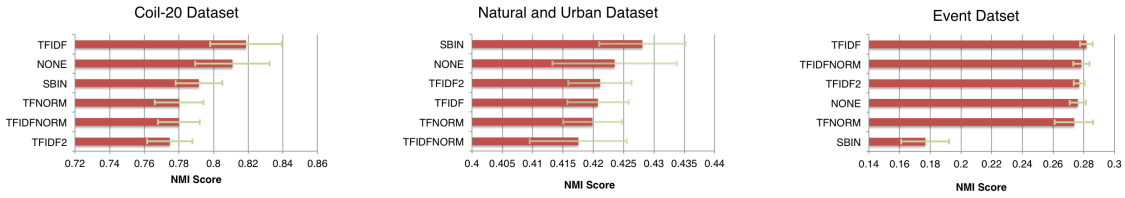


Figure 3.9: Performance of the BoF model for image clustering evaluated by the NMI score for different techniques for weighting and normalization of the histogram representations of the images.

In summary, although the use of normalization and weighting can help to improve the results of the BoF model for image clustering, it does not significantly influence it, and therefore is not a core step.

Table 3.4: Methods tested for histogram normalization and weighting and the mathematical expression for the final value for the histogram.

Method	Mathematical Expression
Simple Binarization (SBIN)	1 if ti is present, 0 if not
tf-idf (TFIDF)	$f_{(t,d)} \cdot \log(1 + \frac{N}{n_t})$
tf-idf variation (TFIDF2)	$f_{(t,d)} \cdot \log(1 + \frac{N*}{n_t*})$
tf normalized (TFNORM)	$\frac{f_{(t,d)}}{\sum_d f_{(t,d)}}$
tf-idf normalized (TFIDFNORM)	$\frac{f_{(t,d)} \cdot \log(1+\frac{N}{n_t})}{\sum_d f_{(t,d)} \cdot \log(1+\frac{N}{n_t})}$

3.4.4 Clustering Algorithm

The last step of the testing procedure is the clustering algorithm. For this, five algorithms were tested: K-Means, BIRCH, DBSCAN and two different implementations of hierarchical clustering, one from the Scikit-learn library (HIERAR1) and another from the Scipy library (HIERAR2). Additionally, for DDBSCAN and Hiearchical clustering, the dissimilarity measure used to compute the distances between images was varied between: euclidean, cosine, correlation, city-block and hamming. The results are shown in Table 3.5.

Most of the algorithms tested require the specification of the number of clusters as an input. In a total unsupervised manner that would be impossible, and therefore, DBSCAN and HIERAR2, which do not require that parameter were tested. Nonetheless, by analyzing the results, only in the Coil-20 dataset, the HIERAR2 achieved comparable results with the other methods. It was verified that regardless of the attempts in changing the parameters of the DBSCAN algorithm, it either found too many data points as noise, or considered a great number of image to be part of the same cluster. Therefore, and since it is developed to find consistent density-based clusters, it fails for the datasets considered here.

In contrast, both BIRCH and K-Means performed very well. Regarding Coil-20 dataset, the best combination was to use simple binarization as the method for histogram normalization,

BIRCH clustering algorithm and hamming distance measure. Nevertheless, for both the Natural and Urban and the Event datasets, K-Means with no normalization still got the best results.

The conclusion of this last step of the BoF method applied for unsupervised learning of images is that, although an algorithm which does not require the number of clusters is desirable, it is not an easy task, since usually those algorithms require other parameters that need to be adjusted and can be very specific to a given set of images and parameter configurations. For this reason, a better alternative might be to compute the clustering algorithm for different number of clusters and then pick the one that maximized a given internal index, such as the silhouette index [44].

Table 3.5: Results for the three datasets using different algorithms for the final clustering step.

Normalization	Clusting Algorithm	Distance measure	Avg. Rand Index	Avg. NMI Index	Number of clusters
Coil-20 dataset					
SBIN	BIRCH	hamming	**67,4%**	**84,8%**	**20**
TFIDF	KMEANS	euclidean	59,6%	81,9%	20
NONE	HIERAR1	correlation	56,9%	82,7%	20
NONE	HIERAR2	cosine	54,5%	81,4%	>150
SBIN	DBSCAN	correlation	18,0%	64,2%	15 avg.
Natural and Urban dataset					
NONE	KMEANS	euclidean	**30,2%**	**40,6%**	**8**
NONE	BIRCH	euclidean	27,4%	37,9%	8
SBIN	HIERAR1	cosine	25,7%	37,6%	8
NONE	HIERAR2	cosine	8,1%	44,6%	>700
NONE	DBSCAN	correlation	5,8%	36,6%	>1200
Event dataset					
NONE	KMEANS	euclidean	**19,4%**	**27,4%**	**8**
TFIDF	BIRCH	euclidean	17,1%	25,6%	8
TFIDF	HIERAR1	correlation	15,8%	23,6%	8
NONE	HIERAR2	correlation	9,5%	48,0%	>860
TFIDF	DBSCAN	cosine	2,3%	35,0%	>700

3.5 Discussion

This study aimed to evaluate the performance of a very popular model for image representation called Bag-of-Features, in which the goal is to represent an image as a collection of visual words. Although this model has been applied with good results to image classification, there has been few works towards solving the problem of image clustering. Therefore, the objective of this study was to develop some insight on this model including the evaluation of different algorithms and parameters for the different steps of the model.

As a result of this experimental evaluation, the steps or parameters that most influenced the performance of the model for image clustering were the algorithm for image description, the average number of keypoints per image, the size of the codebook and the final clustering algorithm.

Another interesting observation was that, although having been proposed several decades ago, the K-Means algorithm continues to be a very fast and robust choice for the codebook learning algorithm and for the clustering algorithm compared to other recent approaches.

Additionally, from all the different experiments developed and presented in this work, it can be concluded that although the Bag-of-Features model can be successfully applied to the problem of unsupervised learning for visual data, it provides a poor representation of the images when the datasets represent complex scenes. This was clearly illustrated by the results for the Event dataset.

For this reason, there needs to be further research towards a better understanding of visual data and the way humans categorize images in order to really be able to achieve comparable automatic results.

Chapter 4

Study 2: Alternatives to the Bag-of-Features Model

4.1 Introduction

The Bag-of-Features (BoF) model is very popular and achieves good results on simple datasets. However, the representations of the images into histograms contains limited information. Additionally, it lacks the use of both color and spacial information. Thus, four methods or extensions to the BoF were implemented: the Fisher Vectors (Section 2.4.2.2), Spatial Pyramid Matching - SPM (Section 2.4.2.3), the Bag-of-Colors - BoC (Section 2.4.2.4) and the Bag-of-Features+Colors - BoF+C. The last method is the combination of the Bag-of-Features and the Bag-of-Colors.

These algorithms were implemented in Python and is program is also publicly available at Github [1]. The first version of the software only implemented the BoF model. However, the alternative methods being tested in this section were added. Therefore, a total of 5 algorithms can be tested. With regards to execution, the program is called from the command line and requires a number of input parameters including the path of the image dataset, the method being tested and the parameter settings. This software represents a valuable contribution of this thesis.

In relation to the strategy for testing these algorithms, the three datasets used in the previous study were also applied. Each method was then compared with the Bag-of-Features model in terms of performance, complexity and computational time.

In the following sections, implementation details and test results will be presented for each algorithm. In the end, a summary and some conclusions will be drawn from the results obtained.

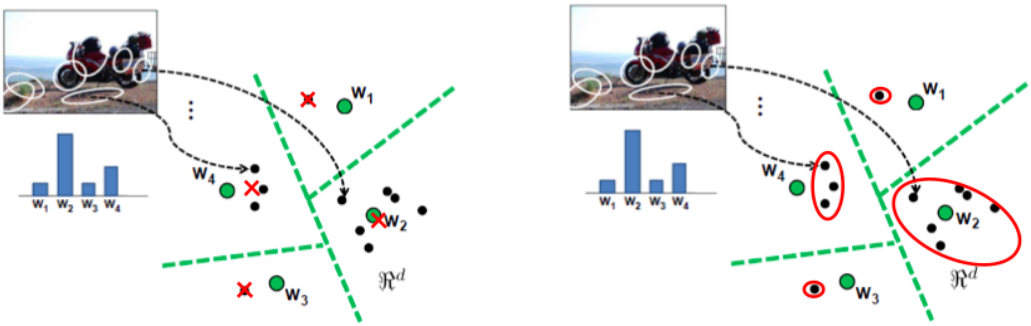
4.2 Fisher Vectors

Fisher Vectors (FV) offer a more complete representation of the local features extracted from the images. The idea behind the FV is fitting a Gaussian Mixture Model (GMM) to a sampled portion of the data and then computing the statistics that relate the log-likelihood of all the data collected

[1] The source code of this project can be found in the link: https://github.com/marianafza/ImageClustering

and the fitted GMM. The statistics are computed for the distribution of keypoints in an image and refer to the weights, the means and the covariance matrixes that would make the GMM fit the data better. Therefore, the main difference between the BoF model and the FV is that the BOV only counts the number of local descriptors assigned to each region with the centers being the visual words, whereas the FV computes higher order statistics.

To illustrate the idea behind the FVs, consider Figure 4.1. The local descriptors extracted from the image are represented in the feature space together with the visual words $w_1, \ldots, w_4$. The BoF model calculates the number of keypoints that fall into the region of the visual words. However, that does not give any information concerning the distribution of the keypoints in each region. Therefore, the FV attempts to add, for example, information concerning the mean of the keypoints (Figure 4.1a) and the covariance matrixes (Figure 4.1b). More information concerning the mathematical equations and the algorithm for computing the fisher vectors can be found in Section 2.4.2.2.



(a) Mean of the keypoints.

(b) Standard deviation of the keypoints.

Figure 4.1: Illustration about the way the BOF works and how the idea behind the FV attemps to add higher level information concerning the distribution of keypoints: the mean of the keypoints (a) and the standard deviation of the keypoints (b). Extracted from [140].

Usually, the dimensionality reduction method PCA (Section 2.4.1.4.2), which attemps to create uncorrelated features, is applied to the local descriptors obtained. This step is important given that the FV assumes that the local descriptors are independent, due to the diagonal approximation of covariance matrices of the GMM. For this reason, PCA was also used in this study. The number of PCA components has to be less than the original number of features. For instance, if the SIFT descriptor is used, the number of components can be any number less than 128.

Compared to the dimensionality of the BoF representation, which is N - the number of visual words, the FV has a dimensionality of $K(2D+1)$, where K is the number of Gaussians in the GMM and D is the dimensionality of the features extracted from the images. This value can be computed as the sum of the features obtained from the FV's components, which are the gradient with respect to the GMM's:

- **weights:** one for each Gaussian in the GMM and represents the importance of that Gaussian in the overall distribution. Therefore, the number of features is K.

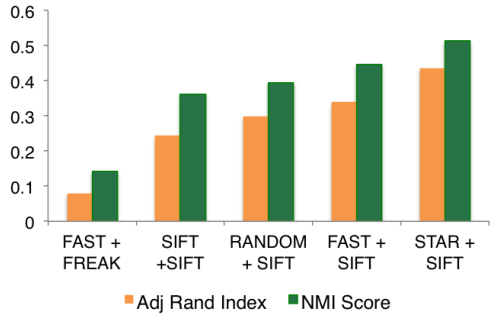
- **means:** each Gaussian has a dimensionality of D, so KD values are required for the means.
- **standard deviations:** as it is assumed that the covariance matrixes are diagonal, the dimensionality is also given by KD.

The following sections present the results of this method on the public datasets used previously varying some key parameters of the algorithm including the keypoint detectors and descriptors, the number of PCA components, the number of Gaussians for the GMM and the influence of the different components of the FVs. Finally, its performance will be compared against the BoF model.

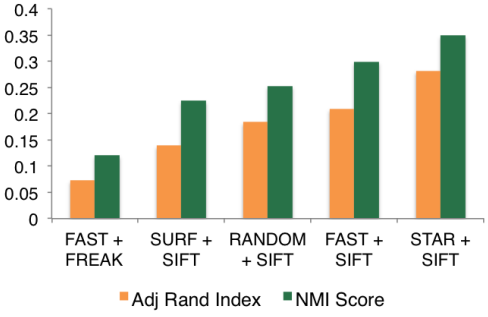
4.2.1 Impact of the Keypoint Detectors and Descriptors

In order to assess the impact of the keypoint detection and description algorithms for the performance of the FV method, a number of tests were applied. For these tests, the other parameters remained constant. The number of Gaussians was chosen to be 1, 3 and 5 for the Coil-20, Natural and Urban and Event datasets, respectively. Moreover, the descriptors were reduced to 64 PCA components, except for the BRIEF and ORB descriptors (which have an original size of 32). The results are presented in Figure 4.2. Both the ARI index and the NMI score are presented in the charts.

(a) Coil-20 dataset



(b) Natural and Urban dataset



(c) Event dataset

Figure 4.2: Results of the FV method using different algorithms for keypoint detection and description applied to the three datasets tested.

For the Coil-20 dataset, five combinations of detectors and descriptors were tested: FAST with FREAK, SIFT with SIFT, SURF with SURF, FAST with BRIEF and ORB with ORB. The performance measures for this dataset indicate that the best detectors and descriptors are FAST with FREAK, SIFT with SIFT and SURF with SURF with approximately 75% ARI and 90% NMI.

In relation to the Natural and Urban dataset, and given the good results obtained by using the SIFT descriptor in the last chapter, all the combinations of detectors were tested with this descriptor. Additionally, the FAST with FREAK combination was also used. The results clearly indicate that the STAR detector with the SIFT descriptor provide the highest performance for this dataset with an ARI of about 45% and a NMI score of over 50%.

Lasly, with regards to the most complex dataset, the Event dataset, which was tested using a number of different combinations of detectors and descriptors. The results show that the best representation is also achieved using the STAR detector with the SIFT descriptor, with approximately 28% ARI and 35% NMI score.

Similarly to what was observed using the BoF model, the keypoint detector and descriptor largely influence the results for the FV. Based on the results presented here and for the BoF model, it can be concluded that the STAR detector combined with the SIFT descriptor produces a representation which is highly effective in terms of clustering performance for both these approaches.

4.2.2 Impact of the Use of PCA and Number of Components

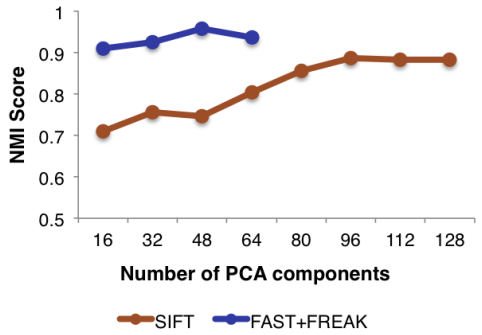
Previously, it was referred that the PCA was a key step for the FV method due to the fact that it intends to obtain independent features. The importance of the step is now tested by comparing the image clustering results with and without PCA. The tests performed here assumed a number of Gaussians of 3 for all the datasets. Additionally, only one of the descriptors and detectors was chosen for each dataset. Therefore, FAST with FREAK was selected for the Coil-20 dataset and STAR with SIFT was selected for both the Natural and Urban and the Events datasets. In this section, only the NMI score is presented, for simplicity.

Without the use of PCA, the NMI score for the Coil-20 dataset was 22,2%. As for the Natural and Urban dataset, the NMI score was 37,4%. Finally, the FV obtained a NMI score of 24,4% for the Event dataset without using PCA. These results will be compared with the ones achieved using PCA.

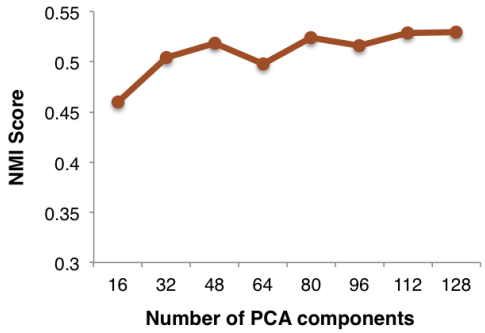
Next, PCA was applied to the image descriptors. The number of PCA components were varied from 16 to 128. The results can be found in Figure 4.6. Also, since the FV obtained by the FREAK descriptor has dimensionality of 64, another descriptor was also tested to achieve the maximum of 128 PCA components.

First, by comparing the results with and without the use of PCA, it can be concluded that, as expected, this transformation is an essential step for the FV model.

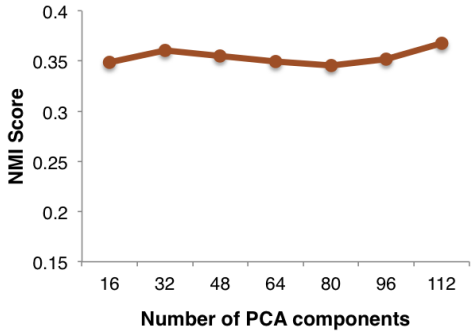
In relation to the number of PCA components, for the Coil-20 dataset and the Natural and Urban dataset, it can be seen that the score tends to increase with the number of PCA components until a certain value, where the improvement becomes less significant (less than 3% change). For



(a) Coil-20 dataset



(b) Natural and Urban dataset



(c) Event dataset

Figure 4.3: Results of the FV method applied to the three datasets, for different numbers of PCA components.

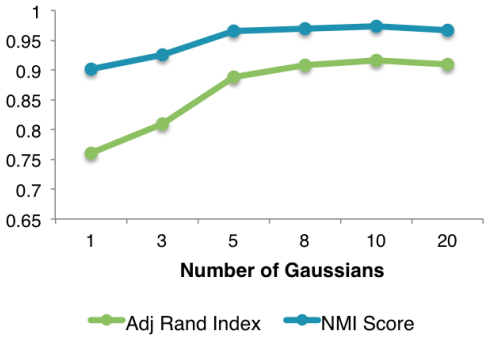
this reason, the ideal number of PCA components should be chosen as the "elbow" of the graph. With regards to the Coil-20 dataset using the SIFT descriptor, this value would 96. Similarly, for the Natural and Urban dataset, the "elbow" would probably be located at 48 PCA components.

However, the performance does not always increase with the number of PCA components. The results for the Event dataset indicate that the number of PCA features does not significantly influence the performance of the FV given that the variations in performance were less than 3% by using 16 to 112 features.

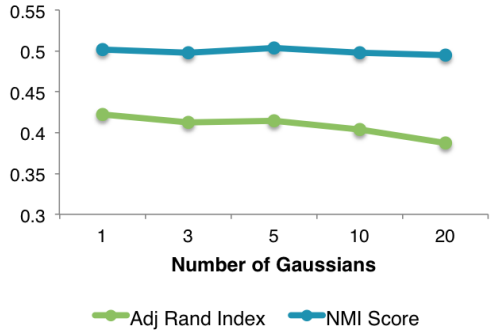
4.2.3 Impact of the Number of Gaussians in the GMM

One of the most important parameters of the FV is the number of Gaussians used in the mixture model K. In the work that introduced the FV [90], the authors tested up to a thousand Gaussians. However, due to the limited computational power available, it was impossible to use more than 20 Gaussians. For the all the datasets, a 64-dim PCA was used. The results are presented in Figure 4.4.

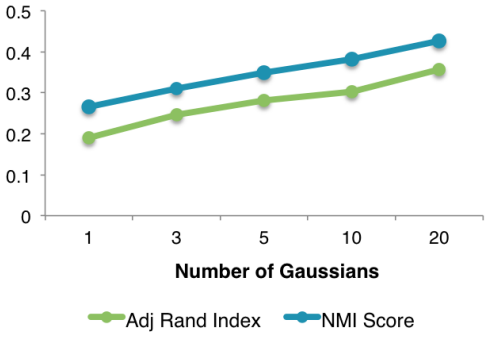
In relation to the Coil-20 dataset, the results indicate that there is significant improvement in performance by using up to 5 Gaussians, which corresponds to a 645-dim FV. After that, the results stabilize.



(a) Coil-20 dataset



(b) Natural and Urban dataset



(c) Event dataset

Figure 4.4: Results of the FV method applied to the three datasets for different numbers of Gaussians.

Surprisingly, for the Natural and Urban dataset, increasing the number of Gaussians tends to decrease the performance. Because of the fact that it is a scene dataset as opposed to an object dataset, the use of more Gaussians might result in an overfitting of the Gaussians on the sampled amount of data used to fit the GMM. Nonetheless, these results were not at all expected considering the results presented in [90].

In contract, for the Event dataset, the performance increased linearly from 1 to 20 Gaussians. Probably this trend would have continued for higher number of Gaussians, however, this was not tested due to computational limitations.

4.2.4 Impact of the Different FV Components

As discussed previously, the FV is composed by three parts, the gradient w.r.t the weights, means and standard deviations of the Gaussians of the GMM. In this section, the influence of each of these components is evaluated separately and combined with other components. The results are presented in Figure 4.5. For simplicity, only the Coil-20 dataset was used.

The results reveal that the least important component is the mixture weights, which does not provided significantly discriminative representation for clustering (30% NMI compared to over 90% for the Coil-20 dataset). It was also shown that individually, the means provide the best

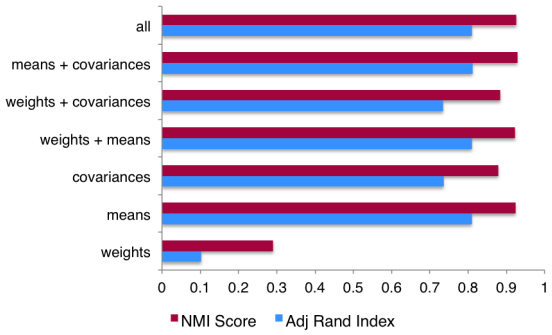


Figure 4.5: Results of the FV method applied to the Coil-20 dataset, for different combinations of the FV components.

representation. Moreover, the addition or not of the weights to the FV does not seem to change the result. Consequently, this component could be excluded from the FV representation to reduce complexity.

4.2.5 Comparison with the Bag-of-Features Model

According to the results of the tests performed in this section, compared to the BoF model, the FV provides significant improvements in terms of performance. Nonetheless, the complexity of the algorithm and the computational time are increased. The best results achieved for the three datasets considered for both methods are presented in Table 4.1.

Table 4.1: Best results for image clustering obtained using FV for each dataset including the parameters and the performance indexes.

Datasets	Detector	Descriptor	K	PCA-dim	Clustering Algorithm	Distance Measure	FV ARI	BoF ARI	FV NMI	BoF NMI
Coil-20	FAST	FREAK	10	64	BIRCH	Correlation	**91.6%**	67.2%	**97,3%**	85.3%
Natural and Urban	STAR	SIFT	3	112	K-Means	Euclidean	**45.1%**	34.2%	**52.9%**	46.0%
Event	STAR	SIFT	20	64	K-Means	Euclidean	**35.7%**	19.2%	**39.1%**	27.1%

The results show a very impressive increase in the performance for all of the datasets tested, from the most simple to the most complex one. The largest improvement was obtained for the Coil-20 dataset with 24.4% increase in percentage difference of the ARI and 12,0% in the NMI score. In relation to the Natural and Urban dataset, a 10.1% difference improvement was obtained in the ARI and 6.9% in the NMI score. Finally, the best ARI and NMI scores for the Event dataset using FV were 35.7% and 39.1% compared to 19.2% and 27.1% obtained with the BoF model. Interestingly, the improvements were higher for the ARI than for the NMI index in all the datasets.

Overall it can be concluded that the FV adds a great amount of useful information to the representation of an image for the purpose of clustering. However, in order to achieve those good results, PCA needs to be applied to the descriptors obtained from the images. This helps create independent features and at the same time reduces the dimensionality of the FV representation. Another important parameter is the number of Gaussians to use in the GMM. In the majority of the datasets, the performance increased with this parameter. However, there is a very significant

amount of computational cost that comes with having to compute an increased number of statistics for each image. Lastly, the most important component that is responsible for the discriminative power of the FV are the gradient w.r.t. the means.

4.3 Spatial Pyramid Matching

One of the characteristics of the BoF model is that is ignores the spacial location of the local descriptors extracted from the images. Although this can be an advantage, in some cases this information could be essential to distinguish scenes. For this reason, SPM was created (Section 2.4.2.3). The idea of SPM is to create different spatial levels (pyramid) from 0 to L with increasing number of regions and for each level. Then, for each region, a histogram of frequency of visual words is computed. For example, if a SPM of 2 levels is selected, a normal BoF histogram is computed, then, the image is divided into four regions and a separate histogram is computed for each region. After that, the histograms are concatenated to create the SPM representation. Thus, the dimensionality of the feature vectors is $M\sum_{l=0}^{L} 4^l = M\frac{1}{3}(4^{L+1}-1)$.

The number of regions is always a power of 4 given that each region is always divided into four regions in the next level of the pyramid. In this implementation, for computational efficiency, first the image is divided into the maximum number of regions according to the number of levels selected and the histograms are computed from the highest to the lowest level. Since the regions need to be concatenated for computing the histograms of the level immediately lower, a transfer 8×8 matrix $T(i, j)$ was constructed. Given a row i and column j of the region, this matrix gives the order to which that region's descriptors should be stored in a list. In this way, once the histograms of that level are computed, the next regions are given by concatenating the four adjacent regions in the vector (1 to 4, 5 to 8, etc). Figure 4.6 shows the transfer matrix. The colors indicate the levels: blue is level 0, yellow is level 1, red is level 2 and finally the cells of the matrix are level 3.

	0	1	2	3	4	5	6	7
0	0	1	4	5	16	17	20	21
1	2	3	6	7	18	19	22	23
2	8	9	12	13	24	25	28	29
3	10	11	14	15	26	27	30	31
4	32	33	36	37	48	49	52	53
5	34	35	38	39	50	51	54	55
6	40	41	44	45	56	57	60	61
7	42	43	46	47	58	59	62	63

Figure 4.6: Transfer matrix that relates the row and column of the regions to the index in which its descriptors should be stored in a list.

4.3.1 Impact of the Number of Pyramid Levels and Comparison with the Bag-of-Features Model

Next, the results of the evaluation of the SPM algorithm using different number of levels will be presented. For these tests, the Coil-20 dataset was not considered because the images have a black background which would result in some regions having only black keypoints. In terms of the local descriptors algorithms, for the Natural and Urban dataset, the RANDOM detector with the SIFT descriptor was used and for the Event dataset, RANDOM with SURF due to the good results obtained using BoF. Additionally, for the Natural and Urban dataset, 128 keypoints were extracted and for the Event dataset, 512 keypoints were extracted.

The most important parameter to be chosen when applying SPM to a dataset is the number of levels of the pyramid. This parameter influences the computational cost of the algorithm as well as the dimensionality of the feature vector obtained for each image. The results of the tests performed can be found in table 4.2. Note that using only 1 level is equivalent to the BoF model.

Table 4.2: Results of the SPM for the Natural and Urban dataset and the Event dataset with different values of the number of levels of pyramid considered.

	Natural and Urban		Event	
# Levels	ARI	NMI	ARI	NMI
1 (BoF)	17,9%	27,5%	**19,0**%	**26,7**%
2	**18,1**%	**28,4**%	17,4%	26,3%
3	14,1%	24,8%	14,2%	21,7%
4	9,3 %	17,2%	6,8%	14,8%

In the work of [93], in which SPM was applied to image classification, the maximum number of levels in which improvements were found was 3. Here, there was only a slight improvement by using 2 levels in the Natural and Urban dataset. After 2 levels, the performance decreased significantly. In relation to the Event dataset, there was no improvement at all by applying the SPM over using the BoF model. Although these results are disappointing, they can possibly be explained by the fact that the increased information obtained by considering the location of the local description is overpowered by the increase in the dimensionality of the feature vectors. Additionally, the results reported in [93] refer to image classification which is a different problem than image clustering.

4.4 Bag-of-Colors

The majority of the works related to image classification, image retrieval and image clustering ignore color information. The Bag-of-Colors (BoC) model attempts to, therefore, explore this information in a similar fashion as the the BoF explores local descriptors.

First, each image is transformed from the RGB color space to the L*a*b color space. Then, it is brokendown into n regions. For each region, the color histograms are computed and the most frequent color vector is stored. This is called the color signature of an image. The next steps are

identical to the BoF model and include the learning of the codebook which is called *color palette* and the histogram computation. In the end, each image is represented by a histogram of *color palette* frequencies. In order to illustrate the process of extracting the color signatures from the images, refer to Figure 4.7 which shows the process of extracting the signature color of one region of the image.

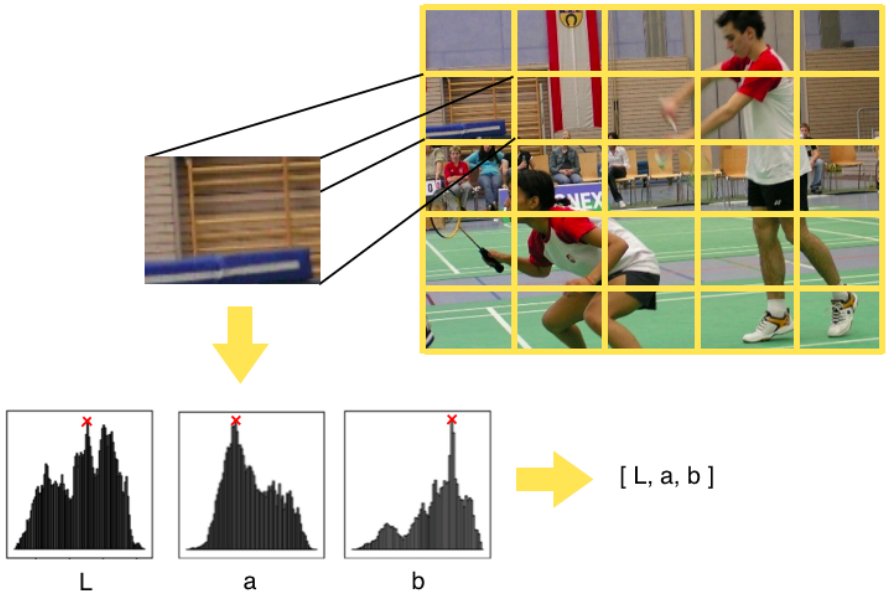


Figure 4.7: Illustration of the method of extracting color features for the BoC representation.

In order to assess the performance of the BoC method, the Natural and Urban dataset and the Event dataset were used. Unfortunately the Coil-20 dataset was not suited since the images are grayscale. In the following sections, the results obtained will be presented and compared to the BoF.

4.4.1 Impact of the Number of Regions and the Color Palette Size

The most important parameters to be chosen for the BoC method are the number of colors extracted from each image and the size of the *color palette*. These parameters are equivalent to the number of patches or keypoints and the size of the codebook in the BoF method.

For both datasets, the Mini-Batch algorithm was used to obtain the color palette from the color signatures obtained from the images and for the final clustering algorithm, the K-Means clustering algorithm was applied. The results are presented in Table 4.3. For simplicity, only the NMI score is presented in the table.

As expected from the results for the BoF model in 3.4.1.2, in general, the performance increases with the number of colors extracted from the images. Also, for the Event dataset, there is an increase in performance with the size of the *color palette* until 64 colors, when the performance starts to decrease. Therefore, for the datasets considered, the best choice for the size of the palette is 256 for the Natural and Urban dataset and 64 for the Event dataset.

Table 4.3: NMI score of the application of the BoC method to the two color datasets varying the number of colors extracted from each image and the size of the color palette.

	Natural				Event			
	Size of the color palette				Size of the color palette			
#colors	36	64	128	256	36	64	128	256
16	15,4%	16,5%	16,7%	16,1%	21,0%	21,2%	20,6%	16,4%
25	16,4%	17,3%	17,5%	18,1%	23,2%	24,1%	22,5%	19,1%
36	17,1%	18,1%	18,3%	17,7%	23,6%	24,1%	22,7%	20,9%
64	17,2%	17,8%	18,2%	18,3%	24,1%	23,9%	23,7%	23,7%
128	18,1%	18,4%	18,5%	**18,6%**	24,4%	**24,5%**	24,0%	23,8%

Surprisingly, by comparing the results from both datasets, it can be seen that the performance of the BoC is significantly higher for the Event dataset than for the Natural and Urban dataset, which never occurred with previous algorithms tested. This exemplifies how the choice of method to use is highly influenced by the dataset.

4.4.2 Comparison with the Bag-of-Features Model

For the Natural and Urban dataset, the BoC is clearly inferior to the BoF model, which achieves a maximum NMI score of 41% compared to the 14.1% obtained using the BoC. In contrast, the results for the Event are similar to those obtained using the BoF model (in the order of 25%).

Overall, it is clear that extracting only color features from the images does not yield a very informative representation for clustering. However, it could be used to complement other feature representation such as the BoF. This approach will be explored in the next section.

4.5 Bag-of-Features+Colors

Finally, the last method implemented was the combination of the BoF with the BoC, which we named Bag-of-Features+Colors (BoF+C), aiming at obtaining information from both local descriptors and colors. A similar approach was presented in [95] using local color descriptors. However, in this work, we chose to use the global color representation as introduced in the last section. Therefore, the BoF+C is implemented by simply concatenating the histograms obtained from the BoF and the BoC into a bigger and more complete histogram, resulting in a larger dimensionality. Figure 4.8 illustrates the process used for this method.

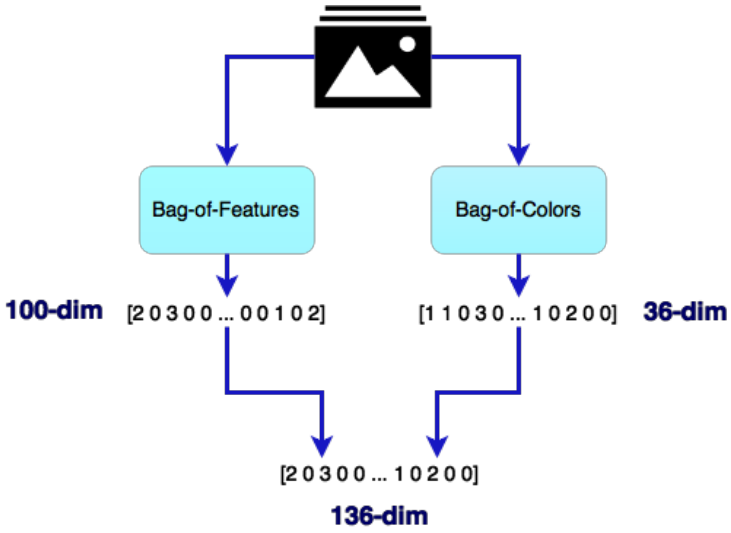


Figure 4.8: Illustration of the BoF+C method, which is the combination of the representations from the BoF and the BoC models. In this example the size of the codebook of the BoF model is 100 and the size of the color palette for the BoC model is 36.

The results obtained for this method are presented 4.4 using two different configurations of parameters for the number of colors extracted per image and the size of the *color palette*. These parameters were selected considering the results of the BoC model in the previous section. Similarly to the case of the BoC model, only the two color datasets were used. Additionally, the equivalent performance measures for the BoF model is presented in the table.

Table 4.4: Results of the BoF+C for the Natural and Urban dataset and the Event dataset with different local feature detection and description algorithms.

Natural and Urban				Event			
number of colors	color palette size	ARI	NMI	number of colors	color palette size	ARI	NMI
36	128	34,0%	46,7%	36	64	**19,5%**	**29,8%**
128	256	**37,3%**	**51,0%**	128	256	19,2%	29,6%
BoF		34,2%	46,0%	BoF		19,2%	27,1%

4.5.1 Comparison with the Bag-of-Features Model

Compared to the performance of the BoF model (last row of Table 4.4), the BoFC presented an increase in performance difference (approximately 3% for the ARI and 5% for the NMI score of the Natural and Urban dataset, and 3% for the NMI of the Event dataset). This is justified by the addition of color information to the BoF representation. Therefore, this proves that color should not be ignored and can successfully be used combined with other methods to achieve more complete representations of the images.

4.6 Discussion

In this section, four alternatives or extensions to the BoF were tested: the Fisher Vectors (FV), Spatial Pyramid Matching (SPM), Bag-of-Colors (BoC) and Bag-of-Features+Colors (BoF+C). Also, for each of the methods, different parameters were varied to evaluate their impact.

For both the FV and the BoF+C, improved performance was achieved. However, the improvements were much higher for the for FV than for the BoF+C. In relation to the FV method, the PCA was shown to be an indispensable step. Additionally, the number of Gaussians in the GMM have shown to highly influence the performance for the majority of the datasets tested. Color is used, in the BoF+C method, to complement the information obtained from the local descriptors. The results indicate that these higher dimensional feature vectors provide better discriminative power.

For one of the datasets, the BoC obtained only slightly worse results than the BoF, which is positive since it only uses color information. The performance of the BoC almost always increased with the number of colors extracted from the images. Unfortunately, for the datasets tested, SPM did not bring significant improvements and as the number of levels increased, the performance decreased. This might indicate that this method is not suited for the problem of image cluster or for the datasets considered.

Overall, for the purpose of image clustering, the BoF, the FV and the BoF+C provide the best results for the public datasets tested. For this reason, they were applied to the mining of images shared in the social networks presented in the following chapter.

Chapter 5

Study 3: Evaluation on Twitter Images

5.1 Introduction

In this chapter, instead of using images from public datasets, the algorithms will be tested with images extracted from Twitter. These images usually contain highly complex content with many objects and backgrounds. This makes the clustering task much more difficult than with the other three datasets tested until this point. Additionally, the images from the public datasets were annotated (there were labels indicating the classes each image belonged to). When dealing with images shared online, more specifically, on the social networks, this is not the case. However, most of the images come with some text, but in relation to the content presented in the image, the text is likely to contain useless or even misleading information. Thus, without a "ground truth" it is impossible to assess the performance of the algorithms. For this reason, user evaluation was performed.

When dealing with unclassified images, another issue is that the number of clusters is not known and is highly subjective. Thus, the only option is to use clustering algorithms that do not require the number of clusters as an input parameter. Consequently, and considering that the Hierarchical Clustering and DBSCAN obtained poor results in the previous studies, another clustering algorithm was implemented. This algorithm has many advantages which are discussed in Section 5.5.

For the first part of this study, the acquisition of the images will be described, including how the images were extracted from Twitter and the filtering steps necessary. Then, the dataset used will be described and justified. Then, the user evaluation procedure will be detailed. Finally, the results will be presented.

5.2 Image Acquisition Module

Several steps need to be applied in order to be able to successfully download images shared by users on Twitter. Figure 5.3 shows the procedure. First, a software called SocialBus was used to connect to the Twitter Streaming API, which provided the online data. This software is responsible

for extracting, filtering and storing the data, in this case, the *tweets*. Then, the images need to be downloaded and filtered before finally storing them.

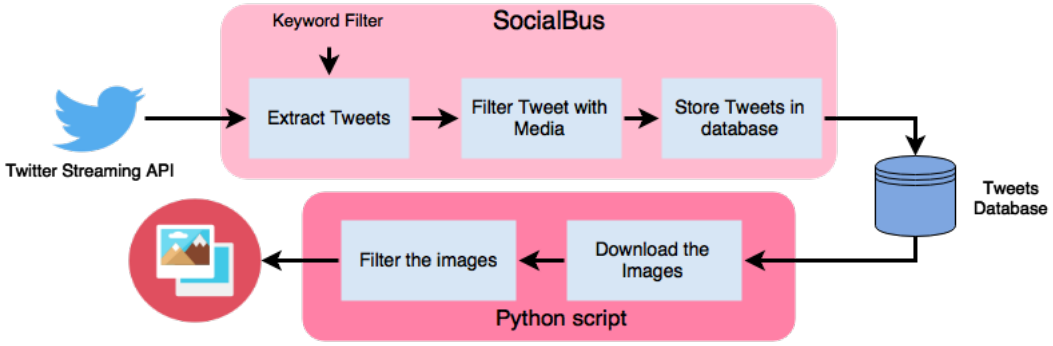


Figure 5.1: Illustration of the steps performed for the image acquisition module.

In the next sections, the SocialBus project will be described, the structure of a *tweet* and how it is filtered will be discussed. Finally, the download and filtering of the images will be covered.

5.2.1 SocialBus

As mentioned before, the data for this study was acquired through a Twitter crawler called SocialBus [16], which is a project being developed at the Faculty of Engineering of the University of Porto. It is developed in Java and uses the Twitter Streaming API.

In order to obtain data from Twitter, an authorized access needs to be setup. For this, a Twitter user needs to register a new application. Next, an access token file needs to be obtained and will serve as an authentication for the application.

In terms of the practical implementation, SocialBus is downloadable as Eclipse projects. The projects used in this thesis were: *socialbus-parent*, *socialbus-core*, *socialbus-twitter-consumer* and *socialbus-twitter-oauth*. To be able to run the socialbus-twitter-consumer project, a number of important parameters (configuration files) need to be specified [141]:

- **config:** main configurations including server connection and storage (database).
- **oauth:** access tokens obtained in the authorization step mentioned before.
- **filter:** filter configurations including type of filter (KEYWORD, GEOLOCATION or USERS) and filter text file with the topics, locations or users to filter.

5.2.2 Filtering the Tweets

The *tweets* extracted from the API have a JSON format. JSON (JavaScript Object Notation) [142] is format that uses human-readable text to transmit data objects consisting of attribute–value pairs. A Tweet has a very complex structure and contains information about the text shared (at most 140 characters), hashtags, geolocation (if enabled by the user), time tweeted, user identification and profile, retweet info, media and others.

Twitter users can share media content, more specifically images directly by uploading an image file together with a Tweet. The image is stored in a *url* which is located in the JSON file under the field *media_url* which is inside *extended_entities* filed. Listing 5.1 presents some important fields of a Tweet with shared media. If the Tweet had no shared image attached to it, there would be no *extended_entities* field.

Listing 5.1: Structure of a Tweet with media

```
{
  "_id": {
    "$oid": "556f0d1c300442adbfde89e4"
  },
  "retweeted": false,
  "lang": "en",
  "extended_entities": {
    "media": [
      {
        "id": 606103089712480257,
        "media_url_https": "https://pbs.twimg.com/media/CGlPjH-UQAEC_rA
            .jpg",
        "media_url": "http://pbs.twimg.com/media/CGlPjH-UQAEC_rA.jpg",
        "expanded_url": "http://twitter.com/jackie90272/status/60610309
            0211586048/photo/1",
      }
    ]
  },
  "timestamp_ms": "1433341212214",
  "text": "Pre-school Francophile and their parents: Paris-Chien is 1
      of 10 picture books set in Paris on http://t.co/2FBZZZdN8u http:
      //t.co/1ifDnavTqL",
  },
  "user": {
    "location": "l.a.",
    "lang": "en",
    "id": 367209090,
    "followers_count": 81,
    "profile_image_url_https": "https://pbs.twimg.com/profile_images/59
        9989924037726210/gRfmV9QE_normal.jpg",
    "friends_count": 236,
  }
}
```

Therefore, in order to filter only Tweets with media content, the program has to check if the *extended_entities* field is present in a Tweet.

There is also another way to share images on Twitter, which is to publish a photograph on Instagram and choose to share it on Twitter. As mentioned before, Instagram is a very popular

media driven social network. For this reason, the *tweets* containing Instagram links were also filtered. Listing 5.2 shows some of the most important fields of a *tweet* that has an Instagram link. The *url* for the image content is present in the *expanded_url* field which is inside the *entities* field. Therefore, to filter those Tweets, it is necessary to check if the *url* field contains the words Instagram.com.

Listing 5.2: Structure of a Tweet with an Instagram link

```
{
  "_id": {
    "$oid": "557218c0ef8696aa38d3a0a3"
  },
  "retweeted": false,
  "lang": "en",
  "text": "Yay it's here!! #Bombay's best times!! Time for a #monsoon #
     ride! #trip #rains #epic https://t.co/LdhfJRODeb",
  "entities": {
    "trends": [],
    "symbols": [],
    "urls": [
      {
        "expanded_url": "https://instagram.com/p/3kCU1jw7xP/",
        "display_url": "instagram.com/p/3kCU1jw7xP/",
        "url": "https://t.co/LdhfJRODeb"
      }
    ],
  },
  "user": {
    "location": "Mumbai",
    "lang": "en",
    "id": 600839170,
    "followers_count": 96,
    "friends_count": 239,
  }
}
```

Additionally, two other types of filtering were performed before storing the *tweets*. The first one was to remove *tweets* that are *retweets* as they represent duplicated data. This is done by analyzing the *retweeted* field (which can be either true or false). The last was to consider only *tweets* in English, and therefore, only *tweets* with the field *lang* set to *en* where considered.

In order to store the Tweets obtained in the previous step, the database MongoDB [143] was used, which is the one used in the SocialBus project. It is a NoSQL database and has the advantage of working with JSON-like formats which is the format the Tweets are obtained.

5.2.3 Downloading and Filtering the Images

After storing the data in the database, the next step is to obtain the image content. This was implemented offline in a Python script. First, a connection to the database needs to be done. Then all the entries in the database are scanned for the media fields, for either the images shared directly on Twitter or the ones via Instagram. Next, the images where downloaded. Finally, some images where removed due to three reasons:

- **Size:** small images, with dimensions less than 200 pixels per side or 90.000 in pixel total were removed due to quality issues.
- **Content:** since the goal was to analyze natural images (photographs), there was an attempt to eliminate unnatural images.
- **Repetitions:** images from links that were already used where removed to avoid duplicated content.

In relation to the filtering of unnatural images, a simple algorithm was developed and implemented. It basically explores the fact that the histograms of the natural images are different in shape compared to the unnatural images. Figure 5.2 shows the grayscale histogram of 2 images. One of them is a natural image and the other is unnatural. It can be seen that the natural image has a much softer histogram shape than the unnatural which is mostly composed by peaks. This phenomenon can be explained by the fact that the unnatural images usually do not have a great number of different colors. For example, it is clear that the example image has only 3 major colors.

Algorithm 3 Unnatural Image Detector

Input: Grayscale histogram of the image *hist*, width w and height h of the image
Output: Boolean $unnaturalImage$

```
1:  thres1 ← 0.01wh
2:  thres2 ← 0.6
3:  empty ← 0
4:  for i = 1 to 255 do
5:      if hist[i] < thres then
6:          empty ← empty + 1
7:      end if
8:  end for
9:  if empty > thres2 then
10:     unnaturalImage ← True
11: else
12:     unnaturalImage ← False
13: end if
```

The presence of the peaks on the gray level distribution of the unnatural images leads to most of the histogram being empty. Hence, it is possible to evaluate the emptiness of the histograms and make a decision based on that. Algorithm 3 presents the solution proposed.

(a) Natural image.

(b) Unnatural image.

(c) Natural image histogram.

(d) Unnatural image histogram.

Figure 5.2: Examples of grayscale histograms of natural and unnatural images.

This algorithm depends on two thresholds that can be varied. The values presented in the algorithm were found as the most appropriate for the images obtained from Twitter. Additionally, although this procedure does not detect successfully all the unnatural images, it still provided good results given its simplicity.

5.3 Twitter Dataset

After many visual inspections of the images shared directly to Twitter, it was seen that a substantial amount of them did not have good quality and were used for marketing purposes. In contrast, the images from Instagram were high quality and most of them were personal photographs. Since these are the images this study was aiming for, it was decided that the dataset should only contain images from Instagram. Nonetheless, we will still refer to them as the Twitter dataset.

The dataset contains 1000 images, which were extracted from 5 to 6 of June of 2015. A keyword filter was applied with the words "trip" and "weekend" (since June 6 was a Saturday). Hence, they contain, for instance, scenes of people traveling, landscapes and food. Additionally, the images are squared and were resized to 400 $\times$ 400 pixels. Figure 5.3 shows examples of

images from this dataset. It is relevant to note that no manual selection was performed on the images.

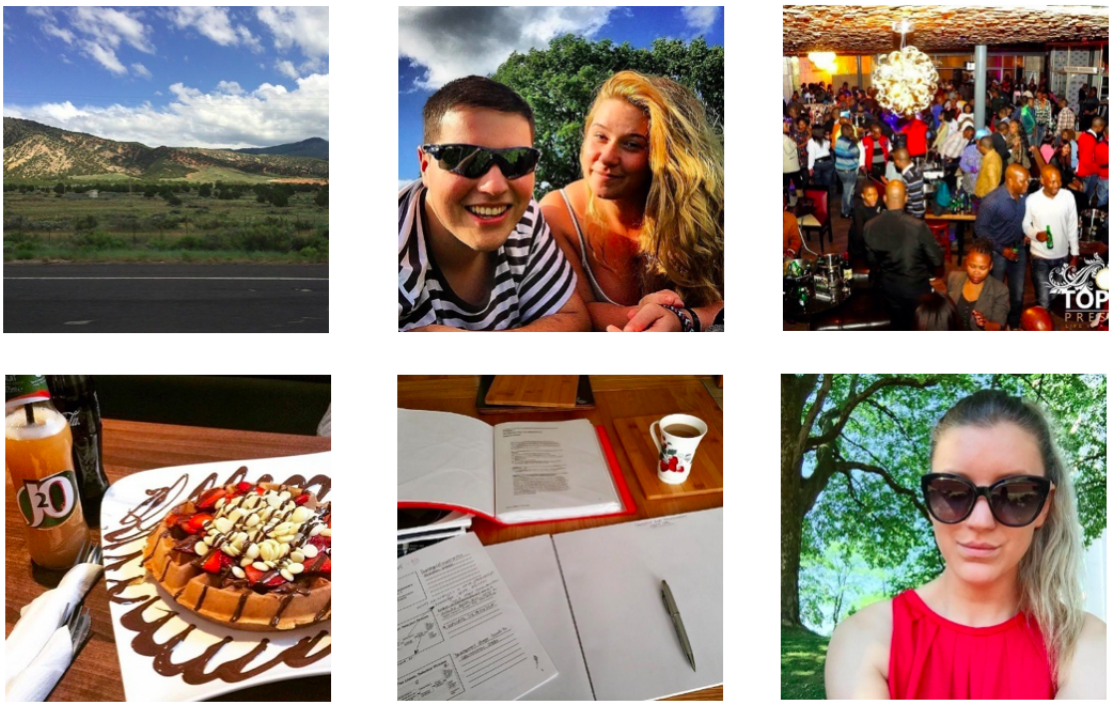


Figure 5.3: Examples of images from the dataset extracted from Twitter/Instagram.

5.4 User Evaluation

As mentioned before, the images do not have a label indicating their class. For this reason, it would be impossible to compare the algorithms implemented. The solution was to create a webpage that would allow the general public to help annotate the dataset by evaluating the content of the images. Therefore, the approach selected was to ask the users to compare pairs of images, in relation to their similarity. The interpretation of similarity here is obviously very subjective but for reference they were told that similar images shared similar concepts, scenes or context.

Hence, in total, there are 500,000 different pairs of images in the dataset. Consequently, it was not realistic to expect to obtain that many answers from the public. However, it is likely that a rough performance estimate of the algorithms can be achieved with only a portion of those pairs of images. Therefore, we aimed at 1% of those combinations, which is equal to 5,000 answers. In a few hours, we were able to achieve a total of 5,674 answers, in which roughly 20% were "yes" and 80% were "no" w.r.t image similarity.

The implementation of the website was done in JavaServer Pages (JSP) and Java servlet [144]. JSP is a tool to create dynamic webpages and it is based on HTML. On the other hand, just like PHP, Java servlet is a Java programming language that extends the capabilities of the webserver to responde to requests. The combination of those technologies allow the creation of fully operational

websites. The servlet container used was Apache TomCat [145], which is an open-source web server that runs Java code.

Figure 5.4 shows a screenshot of the webpage developed for the user evaluation procedure. Each time a user enters the webpage, a pair of images is selected, at random, from the dataset and is shown. The user is told to decide if the images are similar or not and the decision made is stored in a text file in the server together with the identification of the images. After that, the process is repeated, until the user leaves the page. For guidance, at the top left corner of the webpage, the overall number of pairs of images evaluated (by all users) and the goal number are shown. Finally, at the bottom of the page, more information about the instructions and the project are presented.

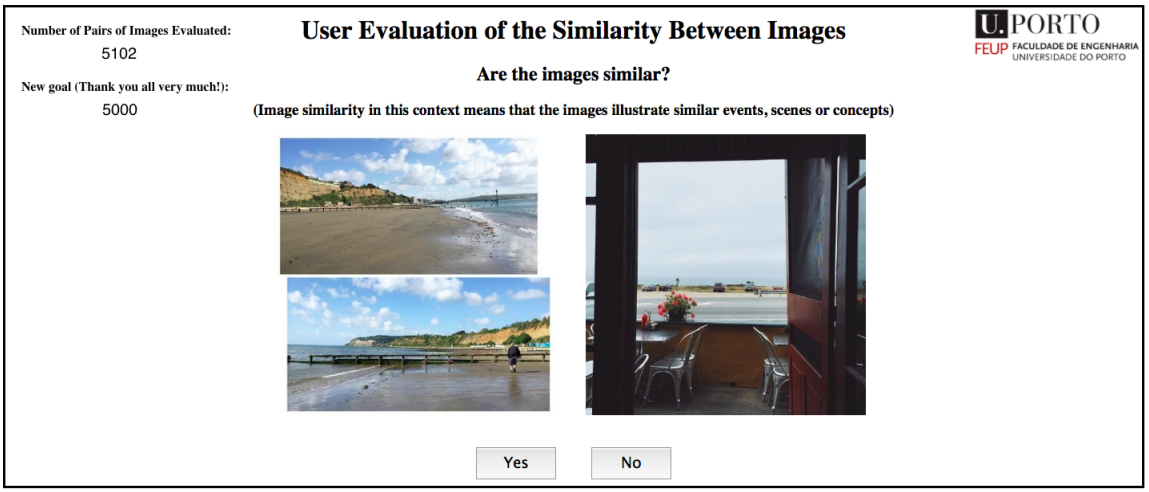


Figure 5.4: User Evaluation website of the dataset obtained from Twitter.

After the answers are obtained, the Adjusted Rand Index (Section 2.2.3.1) was used as the validity index of the algorithms implemented, given that it evaluates the decisions the algorithms makes on the pairs of images which compose the dataset. However, due to the fact that in this case, only 1% of the pairs of images were evaluated, the full ARI is not able to be computed. However, considering the pairs that were evaluated, an estimate of the ARI can be calculated using the same mathematical expression in Equation 2.26. As the values obtained are estimates, there is an error associated with it, other than the usual errors due to the non-deterministic nature of the algorithms. For that reason, and in order to try to quantify it, the ARI was computed for one of the public image datasets, the Event dataset, in the same way, using only 1% of the pairs of images. The absolute average value of the error, in relation to the full ARI was 0.6%. Therefore, for the purpose of this analysis, this error is not very significant.

Additionally, an individual classification was also considered. This classification was performed by a volunteer, which split the whole dataset into groups of related images and outliers. Therefore, there are two different perspectives on the manual classification of images, which can be compared. The public evaluation will be refer to as the group evaluation and the other as the individual evaluation.

During the tests of the algorithms using the dataset collected from Twitter, in order to be able to assess if the algorithms were obtaining efficient representation in terms of obtaining low distances between images that appeared to be similar and large distances to the ones that were not similar, the idea of using a graph representation was created. Given that the approach of using graphs was already explored in related work detailed in Section 2.7, another clustering approach was implemented. This method will be presented next.

5.5 Community Detection for Image Clustering

Similarly to the relationships between users in a social network, the relation between images can also be represented as a graph. The nodes of the graph are the images and the edges' weights are the dissimilarity between the images. The dissimilarity between the images can be obtained from the distance matrix which is a N by N matrix that contains the distance measures between the feature vectors of the images. For the construction of the distance matrix, any distance measure can be used (Section 2.2.1.1), from the Euclidean distance to the inverse correlation ($1-$ correlation).

Given that the distance matrix relates every image to every other image in the dataset, each image would be connected to the others in the graph (which is called a *complete graph*). Therefore, in order to reduce the number of edges of the graph, the maximum number of edges per image is reduced to a 20. Those edges are chosen as the ones with the lowest values which represents the 20 closest images. Additionally, to filter out the edges even more, in order to achieve a more compact representation of the dataset, the edges with weights higher than the median of all the weights are excluded. This step reduces the number of edges to half, keeping only the strongest connections. It is important to note that after this step, some images are likely to be represented as a separate node. In this case, they are said to be outliers. This follows the same strategy used in the work of [123], described in Section 2.7.

Figure 5.5 illustrates a graph of images in which the filtering step has been applied. As can be seen in the figure, some images are connected to each other and others are not. The weights of the edges have illustrative values from 1 to 3 to indicate the level of dissimilarity found between them. Also, one of the images is isolated from the others, becoming an outlier.

Now, after obtaining the final graph that represents the dataset, the goal is to discover clusters or communities of images. In graph theory, this is the problem of community detection. One of the algorithms for this purpose it the Louvain algorithm for community detection, introduced in Section 2.1.4. It is implemented in the *python-louvain* [146] package and works by building an hierarchical structure. This makes it very efficient in terms of computational time. This algorithm was applied to the graph obtained from the previous edge filtering steps and generates the best partition in terms of modularity, which is a measure of the density of the links inside the communities in relation to the links between communities.

Therefore, this whole procedure is equivalent to using a clustering algorithm on the feature representation obtained for each image. However, it has the advantage of not requiring the knowledge of the number of clusters or the requirement of any parameter since it computes the best partition

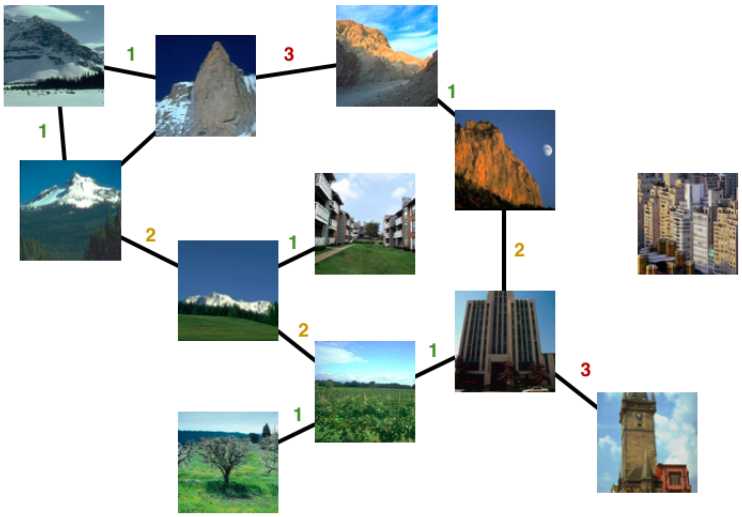


Figure 5.5: Example of a graph representation of images. The edge weights are for illustrative perpose only and represent the dissimilarity between the images.

to achieve maximum of an objective function. Additionally, the process is highly intuitive and relatively easy to visualize.

5.6 Results

In this section, the results of the application of the best performing algorithms for image representation and clustering from the previous studies with public datasets will be presented. The algorithms that were tested were the BoF, the FV and the BoFC. Both the individual and the group evaluation were be considered. As mentioned before, the ARI is the index responsible for evaluating the performance of the algorithms.

For all the models, two final clustering algorithms were considered: hierarchical clustering (HIERAR2) and community detection (COMM). DBSCAN was not considered since its results were very poor compared to the other algorithms. Also, the hierarchical clustering algorithm requires constant update of the threshold parameter, which is responsible for determining the clusters from the hierarchical structure. For this reason, the community detection approach was preferred and tested more extensively. When using the community detection clustering algorithm, the correlation measure was used to compute the dissimilarities between feature vectors. On the other hand, the cosine dissimilarity measure was used for the hierarchical clustering algorithm.

For the BoF model, the image description algorithms tested were the RANDOM detector with the SIFT descriptor, the RANDOM detector with the SURF descriptor and the STAR detector with the SIFT descriptor. Apart from that, the size of the codebook was also varied. In relation to the FV method, the number of Gaussians (K) and the number of PCA components were varied. Finally, for the Bag-of-Features+Colors, the number of colors extracted from each image and the number of colors of the palette were changed.

The results, for both the individual and group evaluations are presented in Table 5.1.

Table 5.1: Results of the application of the best three methods for image representation and clustering to the dataset extracted from Twitter.

Bag-of-Features				
Image Description	Codebook Size	Clustering Algorithm	Individual ARI	Group ARI
RANDOM + SIFT	300	HIERAR2	7,3%	10,2%
RANDOM + SIFT	300	COMM	10,7%	10,1%
RANDOM + SIFT	500	COMM	13,6%	10,5%
RANDOM + SIFT	1000	COMM	**13,9%**	**11,9%**
RANDOM + SIFT	2000	COMM	13,9	9,8%
RANDOM + SURF	300	COMM	2,1%	1,6%
STAR + SIFT	300	COMM	6,4%	7,1%
Fisher Vectors				
Number of Gaussians (K)	PCA components	Clustering Algorithm	Individual ARI	Group ARI
1	48	HIERAR2	9,1%	8,2%
1	48	COMM	**11,2%**	8,7%
1	64	COMM	9,5%	**9,3%**
5	48	COMM	9,1%	6,8%
5	64	COMM	7,9%	6,7%
10	48	COMM	7,5%	6,4%
10	64	COMM	6,8%	6,2%
Bag-of-Feature-Colors				
Number of Colors	Size of Color Palette	Clustering Algorithm	Individual ARI	Group ARI
64	64	COMM	**13,4%**	11,1%
64	256	COMM	9,5%	9,6%
128	64	COMM	13,2%	**12,6%**
128	256	COMM	10,5%	10,6%

Overall, the ARI for the individual evaluation was higher than for the group evaluation. This might come from the fact that different people had different perspectives when considering the similarity of the images, and therefore, there might not have been a very consistent pattern. Also, it is important to consider the possibility of mistakes or users who did not answer according to the instructions.

In relation to the BoF, the best detector and descriptor for this dataset was the RANDOM detector and SIFT descriptor, which achieves a maximum ARI of 13,9% for the individual evaluation and 11,9% for the group evaluation. In general, the performance increased until a certain point with the number of visual words. The community detection algorithm was a very good choice since no tuning of parameters are required and it achieves very good results.

Unlike for the public datasets, the FV performed worse than the BoF. The best result was 11,2% for the individual evaluation and 9,3% for the group evaluation. Interestingly, the highest results for the FV method was achieved using only 1 Gaussian ($K = 1$).

Finally, the BoF+C achieved a best result of 13,4% for the individual evaluation and 12,6% for the group evaluation by extracting 64 colors and using a 64 size color palette for the individual and 128 colors and a 256 size color palette for the group. These results are similar to the BoF model. and therefore, the addition of color information did not successfully improve the results.

The reason behind this is probably due to the nature of the dataset, given that the images had very mixed colors, unlike the public scene datasets, where there were color patterns or trends associated to each class.

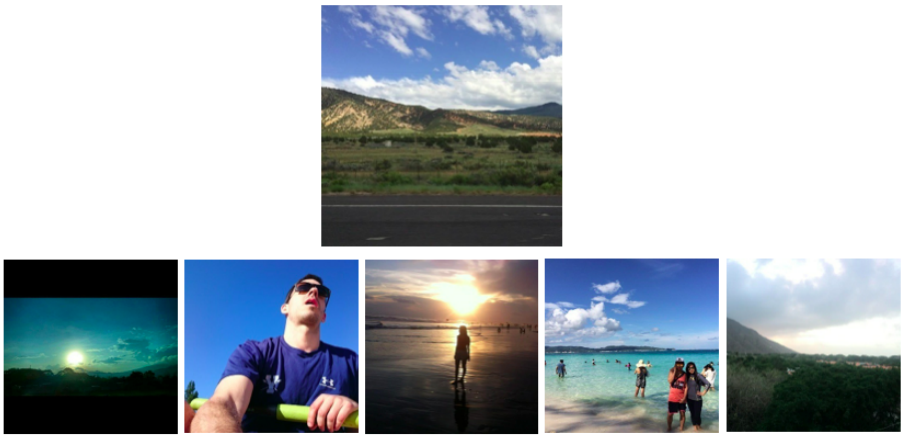
5.7 Presentation of the Clusters

After obtaining the clusters, it is important to consider a strategy to visualize and present the outcome. Some approaches that had been implemented in previous works were discussed in Section 2.6. However, following the idea of using graph theory and community detection as one of the alternative clustering algorithms, a much simpler strategy was implemented. In Section 2.1.5, the problem of finding the most influential nodes in a network was addressed, in which one of the approaches was to use centrality-based methods to score the nodes. One of these measures is closeness centrality. The node with the highest closeness centrality means that it has the smallest distances to all other nodes. For this reason, the approach implemented for visualization of the clusters obtained by the algorithms was to find the most central node (image) for each community (cluster). This methodology follows the three steps below:

1. Obtain the feature vectors of each image using the algorithms implemented (BoF, SPM, Fisher, etc).
2. Obtain the communities using the community detection approach described in Section 5.5.
3. For each community, compute the most central node according to the closeness-centrality score.

In the end, each cluster can be represented by a single image, for summarization and visualization purposes.

Figure 5.6 presents examples of images from three clusters obtained by one of the algorithms and their central images. It can be seen that each cluster presents images with different concepts: cluster 1 shows mostly images from landscapes, cluster 2 shows images of food and drink and cluster 3 of people. However, not all the images in the same cluster present similar content when manually analyzing them, as can be also seen the the figure.



(a) Cluster 1

(b) Cluster 2

(c) Cluster 3

Figure 5.6: Examples of images from three clusters obtained (bottom) and their central image (top).

5.8 Discussion

In this chapter, the knowledge about the algorithms for image representation were put to the test in a real dataset composed by images shared on Twitter. In order to do so, first, the software SocialBus was used to obtain the tweets and store them in a database. After that, the images were downloaded and filtered, to remove unsuited images. After that, the visual representation and clustering modules were applied using different methods evaluated in the previous chapters for public image datasets. However, the Twitter images do not have any labels and therefore, it is impossible to use external clustering validity indexes, such as the ARI and the NMI score. Therefore, to overcome this issue, a user evaluation was conducted. This user evaluation was twofold: a group evaluation, where 1% of the total number of pairs of images were evaluated as similar or not, and a individual evaluation of the whole dataset by slitting it into groups or outliers. With this information, it was possible to assess the performance of the algorithms on the Twitter dataset.

The results were positive, considering the complexity of the dataset, in which no image was manually filtered. The best performance for the individual evaluation was obtained by the BoF with an ARI of 13,9% and the highest ARI for the group evaluation was achieved by using the BoFC with an ARI of 12,6%. These results clearly demonstrate that the algorithms are able to distinguish patterns in the content of the images. As expected, these results were lower than the ones obtained using the public image datasets, which were more simple in their content and in the groups presented.

Chapter 6

Conclusions

6.1 Conclusions

A few years ago, scientists were concerned with providing means of producing, storing and sharing data in a large scale. Today, almost everyone in the world has access to a mobile phone or a computer with Internet connection and is connected through social media. This means that people now have the ability to easily create and share content which accounts for the huge amount of information shared everyday online, whether it is in the form of text, image or video. For these reasons, the concern today is how to process this content in order to obtain useful information. The opportunities and applications of this type of analytics are endless and would definitely benefit society.

This work aimed to study and evaluate the performance of some popular algorithms for image representation for the purpose of clustering images, which is the task of obtaining groups of related images. The final goal was to apply these methods to the photographs shared on the social networks. If this could be successfully accomplished, it would be possible to visualize and summarize the enormous amount of visual data produced everyday by millions of people around the world.

The first step consisted on the study of concepts related to this topic including social network analytics, data mining and image representation. This also involved searching for related work on the subject in order to obtain knowledge on the most recent techniques and methods developed.

After these introductory chapters, the development of this thesis was divided into three studies. In Study 1, the most widely used model for image classification and clustering, the Bag-of-Features, was extensively evaluated using three public image datasets, considering different choices for all the steps involved. Next, Study 2 was concerned about the implementation and evaluation of alternative or extensions to the Bag-of-Features model. Similarly to the previous study, only public image datasets were used. Finally, Study 3 introduced the extraction of content from the social networks (Twitter and Instagram) and the application of the algorithms tested before to these contents.

The contributions of this thesis include the development and implementation of Python scripts for the testing of five different methods for image clustering, which is openly available on Github, the implementation of a community detection-based clustering strategy for the images and the extensive evaluation of all the algorithms and parameter settings involved.

From the first and second study, it can be concluded that the Bag-of-Features, the Fisher Vectors and the Bag-of-Features+Colors are able to obtain good results for simple image datasets like the Coil-20 dataset and the Natural and Urban dataset. However, they are unable to obtain the same level of performance for more complex datasets like the Event dataset. For the public datasets, the best performing algorithm was found to be the Fisher Vectors, which provided a more informative representation of the distribution of keypoints in each image.

In relation to the final study, where the images used where obtained from the social networks without any manual filtering, the results indicate that the algorithms are able to distinguish patterns in the content of the images. However, as expected, these results were worse than the ones obtained using the public image datasets. Nonetheless, considering the low correlation among the images, the results were considered positive.

Overall, we feel that this work has shown that more complex approaches need to be developed in order to be able to successfully process the huge amount of information shared by social media users. These approaches will probably require an extensive learning process, similar to how humans learn from a very young age.

6.2 Future Work

In relation to future work, some suggested tasks are:

- The development of a visualization tool or analysis of how the methods are working, for instance, the representation of the visual words obtained, the histograms and other components of the methods implemented.

- The development of real-time modules for image description for the purpose of extracting clusters in an online fashion, as it receives new images, instead of first extracting the images, and them computing the clusters offline.

- The integration of this work with the project TweeProfiles. This would allow the addition of a visual content analysis to the project which currently only uses text content from the Tweets.

- The use of fuzzy clustering algorithms for clustering the images. This would allow one image to be part of many clusters, given that an image can have more than one concept.

- The application of segmentation techniques to distinguish the object or focus of the image and the background in order to compute separate descriptors and features for each and compare them to other images. The reason for this is that, usually the images contain a central

object (a person, animal, etc) and a background. Therefore, the object could be compared with the objects of different images and the background with the backgrounds of different images to obtain a more faithful representation.

References

[1] Jun-Yan Zhu, Yong Jae Lee, and Alexei A Efros. Averageexplorer: Interactive exploration and alignment of visual data collections. *ACM Transactions on Graphics (TOG)*, 33(4):160, 2014.

[2] Tiago Cunha, Carlos Soares, and Eduarda Mendes Rodrigues. Tweeprofiles: detection of spatio-temporal patterns on twitter. In *Advanced Data Mining and Applications*, pages 123–136. Springer, 2014.

[3] Charu C Aggarwal. *Social network data analytics*. Springer, 2011.

[4] Facebook. Company info. http://newsroom.fb.com/company-info/. Accessed: 2015-02-04.

[5] Twitter. Company info. https://about.twitter.com/company. Accessed: 2015-02-04.

[6] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[7] Miles Efron. Hashtag retrieval in a microblogging environment. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 787–788. ACM, 2010.

[8] Instagram. Press. https://instagram.com/press/. Accessed: 2015-06-27.

[9] Nadav Hochman and Raz Schwartz. Visualizing instagram: Tracing cultural visual rhythms. In *Proceedings of the Workshop on Social Media Visualization (SocMedVis) in conjunction with the Sixth International AAAI Conference on Weblogs and Social Media (ICWSM–12)*, pages 6–9, 2012.

[10] Yuheng Hu, Lydia Manikonda, Subbarao Kambhampati, et al. What we instagram: A first analysis of instagram photo content and user types. *Proceedings of ICWSM. AAAI*, 2014.

[11] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[12] David Kempe, Jon Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. In *Automata, languages and programming*, pages 1127–1138. Springer, 2005.

[13] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[14] Mark Newman. *Networks: an introduction*. Oxford University Press, 2010.

[15] David Osimo and Francesco Mureddu. Research challenge on opinion mining and sentiment analysis. *Universite de Paris-Sud, Laboratoire LIMSI-CNRS, B^atiment*, 508, 2012.

[16] Matko Boanjak, Eduardo Oliveira, José Martins, Eduarda Mendes Rodrigues, and Luís Sarmento. Twitterecho: a distributed focused crawler to support open research with twitter data. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 1233–1240. ACM, 2012.

[17] MT Markou and P Kassomenos. Cluster analysis of five years of back trajectories arriving in athens, greece. *Atmospheric Research*, 98(2):438–457, 2010.

[18] Jérémie Bouttier, Philippe Di Francesco, and Emmanuel Guitter. Geodesic distance in planar graphs. *Nuclear Physics B*, 663(3):535–567, 2003.

[19] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[20] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[21] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. *Knowledge discovery in databases*, pages 229–238, 1991.

[22] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, 2008.

[23] Inderjeet Mani. *Automatic summarization*, volume 3. John Benjamins Publishing, 2001.

[24] Michiel Hazewinkel. *Encyclopaedia of mathematics, supplement III*, volume 13. Springer Science & Business Media, 2001.

[25] Martin Anderson, Shan Chen, James Hacking, Marc R Lieberman, Mark Lundin, Vaida Maleckaite, Allan Martin, Ryan Parham, and Mark Steed. Modern pension fund diversification. *Journal of Asset Management*, 15(3):205–217, 2014.

[26] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al. *Introduction to data mining*, volume 1. Pearson Addison Wesley Boston, 2006.

[27] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[28] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, number 14, pages 281–297. California, USA, 1967.

[29] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

[30] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[31] Saed Sayad. Hierarchical clustering. http://www.saedsayad.com/clustering_hierarchical.htm. Accessed: 2015-01-25.

[32] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[33] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[34] Irina Perfilieva and Jiří Močkoř. *Mathematical principles of fuzzy logic*. Springer Science & Business Media, 1999.

[35] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203, 1984.

[36] J Valente de Oliveira, Witold Pedrycz, et al. *Advances in fuzzy clustering and its applications*. Wiley Online Library, 2007.

[37] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[38] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.

[39] D Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[40] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[41] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[42] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010.

[43] Jorge M Santos and Mark Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *Artificial neural networks–ICANN 2009*, pages 175–184. Springer, 2009.

[44] Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús M Pérez, and Iñigo Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013.

[45] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[46] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[47] Minho Kim and RS Ramakrishna. New indices for cluster validity assessment. *Pattern Recognition Letters*, 26(15):2353–2363, 2005.

[48] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

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

[50] Babita Singh and Waseem Ahmad. Content based image retrieval: A review paper. 2014.

[51] John Eakins, Margaret Graham, and Tom Franklin. Content-based image retrieval. In *Library and Information Briefings*. Citeseer, 1999.

[52] Vasileios Mezaris, Ioannis Kompatsiaris, and Michael G Strintzis. An ontology approach to object-based image retrieval. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 2, pages II–511. IEEE, 2003.

[53] Toby Berk, Lee Brownston, and Arie Kaufman. A new color-naming system for graphics languages. *IEEE Computer Graphics and Applications*, 2(3):37–44, 1982.

[54] Rui Shi, Huamin Feng, Tat-Seng Chua, and Chin-Hui Lee. An adaptive image content representation and segmentation approach to automatic image annotation. In *Image and Video Retrieval*, pages 545–554. Springer, 2004.

[55] Yixin Chen, James Ze Wang, and Robert Krovetz. An unsupervised learning approach to content-based image retrieval. In *Signal Processing and Its Applications, 2003. Proceedings. Seventh International Symposium on*, volume 1, pages 197–200. IEEE, 2003.

[56] Samuel Rota Bulò, Massimo Rabbi, and Marcello Pelillo. Content-based image retrieval with relevance feedback using random walks. *Pattern Recognition*, 44(9):2109–2122, 2011.

[57] Xiang Sean Zhou and Thomas S Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*, 8(6):536–544, 2003.

[58] Karl Pearson. The problem of the random walk. *Nature*, 72(1865):294, 1905.

[59] Jean Martinet and Ismail Elsayad. Mid-level image descriptors. In *Intelligent Multimedia Databases and Information Retrieval*. 2012.

[60] Jun Yue, Zhenbo Li, Lu Liu, and Zetian Fu. Content-based image retrieval using color and texture fused features. *Mathematical and Computer Modelling*, 54(3):1121–1127, 2011.

[61] Konstantinos N Plataniotis and Anastasios N Venetsanopoulos. *Color image processing and applications*. Springer, 2000.

[62] Alvy Ray Smith. Color gamut transform pairs. In *ACM Siggraph Computer Graphics*, volume 12, pages 12–19. ACM, 1978.

[63] George H Joblove and Donald Greenberg. Color spaces for computer graphics. In *ACM siggraph computer graphics*, volume 12, pages 20–25. ACM, 1978.

[64] Juxiang Zhou, Tianwei Xu, and Wei Gao. Content based image retrieval using local directional pattern and color histogram. In *Optimization and Control Techniques and Applications*, pages 197–211. Springer, 2014.

[65] Greg Pass and Ramin Zabih. Comparing images using joint histograms. *Multimedia systems*, 7(3):234–240, 1999.

[66] Markus A Stricker and Markus Orengo. Similarity of color images. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pages 381–392. International Society for Optics and Photonics, 1995.

[67] Greg Pass, Ramin Zabih, and Justin Miller. Comparing images using color coherence vectors. In *Proceedings of the fourth ACM international conference on Multimedia*, pages 65–73. ACM, 1997.

[68] George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[69] Peter Howarth and Stefan Rüger. Evaluation of texture features for content-based image retrieval. In *Image and Video Retrieval*, pages 326–334. Springer, 2004.

[70] Wei-Ying Ma and Bangalore S Manjunath. Netra: A toolbox for navigating large image databases. *Multimedia systems*, 7(3):184–198, 1999.

[71] S Livens, P Scheunders, G Van de Wouwer, and D Van Dyck. Wavelets for texture analysis, an overview. In *Image Processing and Its Applications, 1997., Sixth International Conference on*, volume 2, pages 581–585. IET, 1997.

[72] Mingqiang Yang, Kidiyo Kpalma, Joseph Ronsin, et al. A survey of shape feature extraction techniques. *Pattern recognition*, pages 43–90, 2008.

[73] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988.

[74] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[75] Dimitri A Lisin, Marwan A Mattar, Matthew B Blaschko, Erik G Learned-Miller, and Mark C Benfield. Combining local and global image features for object class recognition. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 47–47. IEEE, 2005.

[76] Kristen Grauman and Bastian Leibe. *Visual object recognition*. Morgan & Claypool Publishers, 2010.

[77] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[78] Noah Snavely, Ian Simon, Michael Goesele, Richard Szeliski, and Steven M Seitz. Scene reconstruction and visualization from community photo collections. *Proceedings of the IEEE*, 98(8):1370–1390, 2010.

[79] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE, 2004.

[80] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1):37–52, 1987.

[81] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[82] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[83] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, 2010.

[84] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, pages 778–792, 2010.

[85] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision–ECCV 2008*, pages 102–115. Springer, 2008.

[86] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[87] Stephen O'Hara and Bruce A Draper. Introduction to the bag of features paradigm for image classification and retrieval. *arXiv preprint arXiv:1101.3354*, 2011.

[88] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.

[89] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[90] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.

[91] Tommi Jaakkola, David Haussler, et al. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999.

[92] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[93] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[94] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1458–1465. IEEE, 2005.

[95] Christian Wengert, Matthijs Douze, and Hervé Jégou. Bag-of-colors for improved image search. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1437–1440. ACM, 2011.

[96] Symeon Papadopoulos, Christos Zigkolis, Giorgos Tolias, Yannis Kalantidis, Phivos Mylonas, Yiannis Kompatsiaris, and Athena Vakali. Image clustering through community detection on hybrid image similarity graphs. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2353–2356. IEEE, 2010.

[97] Correl. Home page. http://www.corel.com. Accessed: 2015-02-05.

[98] Ying Liu, Dengsheng Zhang, and Guojun Lu. Region-based image retrieval with high-level semantics using decision tree learning. *Pattern Recognition*, 41(8):2554–2570, 2008.

[99] Jonathon S Hare and Paul H Lewis. Automatically annotating the mir flickr dataset: Experimental protocols, openly available data and semantic spaces. In *Proceedings of the international conference on Multimedia information retrieval*, pages 547–556. ACM, 2010.

[100] Li-Jia Li and Li Fei-Fei. What, where and who? classifying events by scene and object recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[101] S Nayar, Sammeer A Nene, and Hiroshi Murase. Columbia object image library (coil-100). *Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96*, 1996.

[102] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). Technical report, Technical Report CUCS-005-96, 1996.

[103] Computer Vision at Caltech. Caltech 101. http://www.vision.caltech.edu/Image_Datasets/Caltech101/. Accessed: 2015-02-05.

[104] Max Plank Institut Informatik. Analyzing appearance and contour based methods for object categorization. http://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/research/object-recognition-and-scene-understanding/analyzing-appearance-and-contour-based-methods-for-object-categorization/. Accessed: 2015-02-05.

[105] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

[106] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[107] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[108] Jun Yu, Richang Hong, Meng Wang, and Jane You. Image clustering based on sparse patch alignment framework. *Pattern Recognition*, 47(11):3512–3519, 2014.

[109] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *Image Processing, IEEE Transactions on*, 19(10):2761–2773, 2010.

[110] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits, 1998.

[111] CEDAR. Usps office of advanced technology database. http://www.cedar.buffalo.edu/Databases/CDROM1/. Accessed: 2015-02-06.

[112] Sudeep Sarkar, P Jonathon Phillips, Zongyi Liu, Isidro Robledo Vega, Patrick Grother, and Kevin W Bowyer. The humanid gait challenge problem: Data sets, performance, and analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(2):162–177, 2005.

[113] Image Engineering Laboratory University of Sheffield. Face database. https://www.sheffield.ac.uk/eee/research/iel/research/face. Accessed: 2015-02-06.

[114] Ivo Filipe Valente Mota. Olhó-passarinho: uma extensão do tweeprofiles para fotografias. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2014.

[115] Ian Simon, Noah Snavely, and Steven M Seitz. Scene summarization for online image collections. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[116] Yixin Chen, James Ze Wang, and Robert Krovetz. Clue: cluster-based retrieval of images by unsupervised learning. *Image Processing, IEEE Transactions on*, 14(8):1187–1201, 2005.

[117] Jie Cao, Zhiang Wu, Junjie Wu, and Wenjie Liu. Towards information-theoretic k-means clustering for image indexing. *Signal Processing*, 93(7):2026–2037, 2013.

[118] Laurent Galluccio, Olivier Michel, Pierre Comon, and Alfred O Hero. Graph based k-means clustering. *Signal Processing*, 92(9):1970–1984, 2012.

[119] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.

[120] Junjie Wu, Shiwei Zhu, Hongfu Liu, and Guoping Xia. Cosine interesting pattern discovery. *Information Sciences*, 184(1):176–195, 2012.

[121] Pengtao Xie and Eric P Xing. Integrating image clustering and codebook learning. In *29th AAAI Conference on Artificial Intelligence*, 2015.

[122] Stan Z Li. *Markov random field modeling in image analysis*. Springer Science & Business Media, 2009.

[123] Symeon Papadopoulos, Christos Zigkolis, Yiannis Kompatsiaris, and Athena Vakali. Cluster-based landmark and event detection for tagged photo collections. *IEEE MultiMedia*, 18(1):52–63, 2011.

[124] Eamonn Keogh and Abdullah Mueen. Curse of dimensionality. In *Encyclopedia of Machine Learning*, pages 257–258. Springer, 2010.

[125] Bernhard Scholkopft and Klaus-Robert Mullert. Fisher discriminant analysis with kernels. In *Proceedings of the 1999 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing IX, Madison, WI, USA*, pages 23–25, 1999.

[126] Chris Ding and Tao Li. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *Proceedings of the 24th international conference on Machine learning*, pages 521–528. ACM, 2007.

[127] Jieping Ye, Zheng Zhao, and Mingrui Wu. Discriminative k-means for clustering. In *Advances in neural information processing systems*, pages 1649–1656, 2008.

[128] Kyle Heath, Natasha Gelfand, Maks Ovsjanikov, Mridul Aanjaneya, and Leonidas J Guibas. Image webs: Computing and exploiting connectivity in image collections. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3432–3439. IEEE, 2010.

[129] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 31(4), 2012.

[130] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *Computer Vision–ECCV 2006*, pages 490–503. Springer, 2006.

[131] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 197–206. ACM, 2007.

[132] Christophe Moulin, Cécile Barat, and Christophe Ducottet. Fusion of tf. idf weighted bag of visual features for image classification. In *Content-Based Multimedia Indexing (CBMI), 2010 International Workshop on*, pages 1–6. IEEE, 2010.

[133] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

[134] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEE, 2012.

[135] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[136] Vincent Delaitre, Ivan Laptev, and Josef Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *BMVC 2010-21st British Machine Vision Conference*, 2010.

[137] Ka Yee Yeung, David R. Haynor, and Walter L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–318, 2001.

[138] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[139] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2015-02-26]. URL: http://www.scipy.org/.

[140] Florent Perronnin and Herve Jegou. Large-scale visual recognition novel patch aggregation mechanisms. CVPR tutorial on Large-Scale Visual Recognition, 2012.

[141] SocialBus. Home. http://reaction.fe.up.pt/socialbus/. Accessed: 2015-06-22.

[142] Json (javascript object notation). http://json.org. Accessed: 2015-06-10.

[143] MongoDB. Info. http://www.mongodb.org. Accessed: 2015-06-11.

[144] Bruce Perry. *Java Servlet & JSP Cookbook*. " O'Reilly Media, Inc.", 2004.

[145] Apache Tomcat. Home. http://tomcat.apache.org/index.html. Accessed: 2015-06-22.

[146] Python-Louvain Package. Home. http://perso.crans.org/aynaud/communities/. Accessed: 2015-06-21.