Dynamic machine learning for supervised and unsupervised classification

Adela-Maria Sîrbu

Normandie Université

THÈSE EN CO-TUTELLE INTERNATIONALE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'INSA de Rouen

Dynamic Machine Learning For Supervised and Unsupervised Classification

Présentée et soutenue par

Adela-Maria SÎRBU (RUS)

Thèse soutenue publiquement le 06.06.2016 devant le jury composé de		
Mr. Fawzi NASHASHIBI	Docteur HDR, responsable équipe RITS, INRIA Paris	Rapporteur
Mrs. Daniela ZAHARIE	Professeur, Université de West, Timișoara	Rapporteur
Mr. Horia Florin POP	Professeur, Université Babes-Bolyai, Cluj-Napoca	Examinateur
Mr. Laurent VERCOUTER	Professeur, INSA de Rouen	Examinateur
Mr. Abdelaziz BENSRHAIR	Professeur, laboratoire LITIS , INSA de Rouen	Directeur de thèse
Mrs. Gabriela CZIBULA	Professeur, Université Babes-Bolyai, Cluj-Napoca	Co-Directrice de thèse
Mrs. Alexandrina ROGOZAN	Maître de conférences, laboratoire LITIS, INSA de Rouen	Encadrante de thèse

Thèse dirigée par **Abdelaziz BENSRHAIR, laboratoire LITIS** et **Gabriela CZIBULA, UBB**

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

SPMii ECOLE DOCTORALE

litis

Aknowledgements

First of all, I would like to express my deepest gratitude to my two PhD directors: Prof. Dr. Gabriela Czibula and Prof. Dr. Abdelaziz Bensrhair, for their excellent guidance, patience, scientific and financial support during my PhD studies. Without their help I would have not been able to complete this work.

I would like to give my special thanks to my PhD a Assoc. Prof. Dr. Alexandrina Rogozan who has also guided me through my research, encouraged me and who made possible the collaboration with INSA.

Second, I would like express my gratitude to the jury that accepted to review my thesis: Prof. Dr. Horia F. Pop (Babeş-Bolyai University), Prof. Dr. Daniela Zaharie (West University of Timisoara), Prof. Dr. Fawzi Nashashibi (INRIA, France) and Prof. Dr. Laurent Vercouter (INSA, Rouen).

Being part of two universities, I had the opportunity to meet wonderful persons who helped me in a professional or personal way. I am very grateful to Assoc. Prof. Dr. Laura Dioşan for her continuous guidance and for being always there for me, since my second year of university. I also want to thank my colleagues from Babeş-Bolyai University: Iuliana, Zsuszanna and Gabriel and from INSA: Alina, Vannee, Bassem, Fabian and Rawia for their help and friendship. It was great to feel part of such a team.

Last, but not least I would like to thank my husband Florin, my families (Sîrbu and Rus) and my friends for believing in me, for their long patience and their unconditional support during these difficult years.

Contents

List of Figures

List of Tables

List of publications

Publications in ISI Web of Knowledge

Publications in ISI Science Citation Index Expanded

1. Gabriela Czibula, Istvan-Gergely Czibula, **Adela Sîrbu** and Gabriel Mircea A novel approach to adaptive relational association rule mining. *Applied Soft Computing* published by *Elsevier*, volume 36, pp. 519–533, 2015

2. Bassem Besbes, Alexandrina Rogozan, **Adela-Maria Rus (Sîrbu**), Abdelaziz Bensrhair and Alberto Broggi. Pedestrian Detection in Far-Infrared Daytime Images Using a Hierarchical Codebook of SURF. *Sensors*, volume 15, no. 4, pp. 8570-8594, 2015, doi:10.3390/s150408570

Publications in ISI Conference Proceedings Citation Index

3. **Adela-Maria Sîrbu**, Gabriela Czibula and Maria-Iuliana Bocicor. Dynamic Clustering of Gene Expression Data Using a Fuzzy Approach. *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, SYNASC 2014, Timisoara, Romania, September 22-25, pp. 220-227, 2014

4. **Adela-Maria Sîrbu**, Alexandrina Rogozan, Laura Dioşan and Abdelaziz Bensrhair. Pedestrian Recognition by Using a Kernel-Based Multi-modality Approach. *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, SYNASC 2014, Timisoara, Romania, September 22-25, pp. 258-263, 2014

Papers published in international journals and proceedings of international conferences

5. **Adela Sîrbu** and Maria-Iuliana Bocicor. A dynamic approach for hierarchical clustering of gene expression data. *Proceedings of IEEE International Conference on Intelligent Computer Communication and Processing*, Cluj-Napoca, Romania, pp. 3-6,2013.

6. **Adela-Maria Rus (Sîrbu**), Alexandrina Rogozan, Laura Dioşan and Abdelaziz Bensrhair. Pedestrian recognition using a dynamic modality fusion approach. *Proceedings of IEEE International Conference on Intelligent Computer Communication and Processing*, Cluj-Napoca, Romania, September 3-5, pp. 393 - 400, 2015

7. **Adela-Maria Rus (Sîrbu**), Alexandrina Rogozan, Laura Dioşan and Abdelaziz Bensrhair. Pedestrian recognition by using a dynamic modality selection approach. *Proceedings of IEEE 18th International Conference on Intelligent Transportation Systems*, Las Palmas, Spain, September 15-18, pp. 1862 - 1867, 2015

8. Maria-Iuliana Bocicor, **Adela-Maria Sîrbu** and Gabriela Czibula. Dynamic core based clustering of gene expression data. *International Journal of Innovative Computing, Information and Control*, volume 10, no. 3, pp. 1051-1069, 2014

9. **Adela Sîrbu**. A study on dynamic clustering of gene expression data. *Studia Universitatis "Babes-Bolyai", Informatica*, Romania, Vol. LIX, Number 1, pp. 16-27, 2014.

10. Anca Andreica, Laura Dioşan, Radu Găceanu and **Adela Sîrbu**. Pedestrian recognition by using kernel descriptors. *Studia Universitatis "Babes-Bolyai", Informatica*, Romania, Vol. LVII, Number 2, pp. 77-89, 2013.

Introduction

This PhD thesis is the result of my research in the field of applying dynamic machine learning models for solving supervised and unsupervised classification problems. This research was started in 2012, under the supervision of my PhD directors: Prof. Dr. Gabriela Czibula and Prof. Dr. Abdelaziz Bensrhair and my PhD advisor Assoc. Prof. Alexandrina Rogozan.

Machine Learning (ML) is a subfield of artificial intelligence, focused on constructing computer programs that automatically improve with experience [Mitchell, 1997a]. Machine learning methods can be classified into: supervised, unsupervised and reinforcement learning. The aim of supervised learning method is to infer a function from the training data, consisting of pairs of input instaces with their target outputs. If the output of the function is continuous, we will have a regression problem or if it is a class label we will have a classification one. Thus, given a set of training examples, the supervised learning algorithms construct a model that is able to predict the output for new examples. Unsupervised learning aims to discover hidden patterns in data, without using a priori labels. A very popular unsupervised learning technique is clustering, which implies partitioning a certain data set in groups, whose components resemble each other according to a certain similarity measure [Jiang et al., 2012]. Reinforcement learning is the problem in which an agent must discover its optimal performance in an environment by maximizing the rewards [Kaelbling et al., 1996].

The research direction we are focusing on in the thesis is applying dynamic machine learning models to solve *supervised* and *unsupervised* classification problems. We are living in a dynamic environment, where data is continuously changing and the need to obtain a fast and accurate solution to our problems has become a real necessity. In this context, we aim to study and develop machine learning based models that take into account the dynamic aspect of data. There is a large variety of applications to machine learning techniques, which try to solve real life problems like: speech recognition, face recognition, natural language processing, medical diagnosis, email spam detection, fraud detection, pedestrian detection, bioinformatics, robotics etc. After a careful analysis, the particular problems that we have decided to approach in the thesis are pedestrian recognition (a supervised classification problem) and clustering of gene expression data (an unsupervised classification problem). The approached problems are representative for the two main types of classification (supervised and unsupervised) and are very challenging, having a great importance in real life.

The first research direction that we approach in the field of *dynamic unsupervised clas-*

sification is the problem of *dynamic clustering* of gene expression data. Gene expression represents the process by which the information from a gene is converted into functional gene products: proteins or RNA having different roles in the life of a cell. Modern microarray technology is nowadays used to experimentally detect the levels of expressions of thousand of genes, across different conditions and over time. Once the gene expression data has been gathered, the next step is to analyze it and extract useful biological information. Data mining is the field that offers the necessary methods to accomplish this task and one of the most used algorithms dealing with the analysis of gene expression data approaches the *clustering problem* [Stekel, 2006].

Clustering implies partitioning a certain data set in groups, where the components of each group are similar to each other [Jiang et al., 2012]. In gene expression data sets, each gene is represented by its expression values (features), at distinct points in time, under the monitored conditions. The process of gene clustering is at the foundation of genomic studies that aim to analyze the functions of genes [Song and Lee, 2012], because it is assumed that genes that are similar in their expression levels are also relatively similar in terms of biological function. A great number of clustering algorithms have been introduced in the literature, most of which deal with a given static data set (that is not subject to change during the clustering process). There are also some incremental approaches, meaning that the clustering algorithm was designed to take into consideration new instances of data as well, as these are added to the existing data set.

The problem that we address within the dynamic unsupervised classification research direction is the dynamic clustering of gene expression data. In our case, the term *dynamic* indicates that the data set is not static, but it is subject to change. Still, as opposed to the incremental approaches from the literature, where the data set is enriched with new genes (instances) during the clustering process, our approaches tackle the cases when new features (expression levels for new points in time) are added to the genes already existing in the data set. To our best knowledge, there are no approaches in the literature that deal with the problem of dynamic clustering of gene expression data, defined as above.

The second research direction that we approach related to *dynamic supervised classification* is *pedestrian recognition*, one of the most popular research directions in the domain of object detection and computer vision. Pedestrian safety is a critical issue with global impact. A World Health Organization report from 2015 [WHO, 2015] informs that traffic accidents are one of the most important causes of injuries and death around the world, being responsible for an estimated 1.25 million fatalities and millions more sustaining serious injuries. It was remarked that the majority of deaths are not of the car occupants, but of the vulnerable road users, consisting of motorcyclists, pedestrians and cyclists, while the number of pedestrian fatalities increases from high to low-income countries.

Traffic safety has become a priority both for the automobile industry and the scientific community, which have invested in the development of different protection systems. Initially, improvements involved simple mechanisms such as seat belts, but then more advanced systems

like antilock braking systems (ABS), electronic stabilization programs (ESP) and airbags were built. In the last years, the focus has changed to more intelligent on-board systems. Their goal is to anticipate accidents with the purpose of avoiding them, or reducing their severity. Such systems are called advanced driver assistance systems (ADAS), their goal being to assist the driver in making decisions, provide signals in potentially dangerous driving scenarios and perform counteractive measures [Gerónimo et al., 2010]. However, in order for the ADAS to work, it is highly necessary to develop efficient recognition systems.

There are main challenges in implementing such a system. Firstly, humans can have variable appearances because they can adopt a large variety of poses, wear different clothes, carry various objects and have different sizes. Secondly, pedestrians must be detected in outdoor urban scenarios which implies very cluttered backgrounds. Thirdly, the detector must work under a wide range of weather conditions and illumination, which vary from direct sunlight and shadows during the day to artificial or dim lighting during the night. Finally, partial occlusions made by common urban elements, such as parked cars, create further difficulties because only part of the object is visible for processing.

The problem that we approach within the dynamic supervised classification field is the development of dynamic pedestrian recognition systems, which are able to adapt to varying environmental conditions. In order to achieve this, we integrate information from multi-modality images such as intensity, depth and flow into dynamic models. It is known that the first need of a pedestrian recognition system is finding a robust feature set extracted from images, that allows cleanly discriminate the human form, even under difficult conditions. Taking this into account, we also aim to study a new technique for extracting features from images, by using kernel descriptors (KDs) [Bo et al., 2010], which obtained good results for visual recognition, but to our best knowledge, have not been used in pedestrian detection so far.

The thesis is organized in four chapters as follows.

In **Chapter 1** we present the background for the unsupervised classification problem approached in the thesis, the problem of dynamic clustering of gene expression data. We begin with an overview on the most important clustering methods, with focus on *k-means*, *fuzzy c-means* and *hierarchical clustering* algorithms, that we are going to further exploit in our proposed approaches. We continue with a special type of clustering, the dynamic clustering, along with a literature review in this direction. Finally, we address the problem of dynamic clustering of gene expression data, together with a short review of the existing approaches from the literature.

Chapter 2 is original and presents our work related to the problem of dynamic clustering of gene expression data. We begin by defining the problem that we approach and its importance in real life, then we introduce three dynamic clustering algorithms that can handle new collected data, by starting from a previous obtained partition, without the need to re-run the algorithm from scratch. In the same context of dynamic data, we also propose an algorithm for adaptive relational association rule mining of gene expression. We experimentally

evaluate our approaches on a gene expression dataset, analyze and compare our results with those obtained by other approaches from the literature. The obtained results emphasize the effectiveness of using our dynamic models.

In **Chapter 3** we present the background for the supervised classification problem approached in the thesis, the problem of dynamic pedestrian recognition. We begin by presenting the most important components in a pedestrian recognition system: the feature extraction and the classification components. Therefore, we give an overview of the most commonly used features, then we briefly present the most popular classifiers used in pedestrian detection. Finally, we provide a short literature review in the field, with emphasis on the fusion of image modalities, for which we are going to further introduce a dynamic approach.

Chapter 4 is original and addresses the problem of pedestrian recognition in single and multi-modality images. We begin with a comparison on several state-of-the-art features in far infra-red (FIR) spectrum, we continue with a literature review on kernel descriptors, then we present our studies on pedestrian recognition using these features for image representation.

In the first study we investigate how two learning algorithms, Support Vector Machines (SVM) and Genetic Programming (GP), are able to perform pedestrian recognition using kernel descriptors, extracted with three types of kernels: Exponential, Gaussian and Laplacian, while in the second one we study how kernel descriptors perform in single vs. multi-modality pedestrian recognition. We propose two dynamic models for pedestrian recognition, that are able to select the most discriminative modalities for each image in particular and further use them the classification process. Experimental evaluations on a pedestrian dataset confirm the performance of our dynamic models.

The original contributions from the thesis are presented in Chapters 2 and 4 and are the following:

- Three dynamic clustering algorithms, which can handle newly collected gene expression levels by starting from a previously obtained partition, without the need to re-run the algorithms from the beginning

  - A dynamic core based clustering algorithm, based on *k-means* clustering algorithm (Subsection 2.2.3.1.1) [Bocicor et al., 2014]; an heuristic to determine the optimal number of clusters in a gene expression data set (Subsection 2.2.2) [Bocicor et al., 2014], experimental evaluations of the dynamic core based clustering algorithm on a real life gene expression data set, analysis of the results and comparisons with results obtained by other dynamic approaches from the literature (Subsections 2.3.3.1 and 2.4.2) [Bocicor et al., 2014].
  - A dynamic algorithm for hierarchical gene expression clustering, based on *hierarchical agglomerative* clustering algorithm (Subsection 2.2.3.2.1) [Sirbu and Bocicor, 2013]; experimental evaluations of the algorithm for hierarchical clustering on a real life gene expression data set, analysis of the results and comparisons with

results obtained by our other dynamic approaches (Subsections 2.3.3.2 and 2.4.2) [Sirbu and Bocicor, 2013].

  - A dynamic algorithm for fuzzy clustering of gene expression data (Subsection 2.2.3.3.1) [Sirbu et al., 2014a]; experimental evaluations of the algorithm for fuzzy clustering on a real life gene expression data set, analysis of the results and comparisons with results obtained by our other dynamic approaches (Subsections 2.3.3.3 and 2.4.2) [Sirbu et al., 2014a]

- A dynamic algorithm for relational association rule mining of gene expression data, which can handle the newly arrived features by adapting previously obtained rules, without the need of re-running the mining algorithm from scratch (Subsection 2.4.3.2) [Czibula et al., 2015a]; experimental evaluations of the algorithm for relational association rule mining on a real life gene expression data set and analysis of the results (Subsection 2.4.3.3) [Czibula et al., 2015a]

- The usage of kernel descriptors for pedestrian recognition (Subsections 4.3.2 and 4.3.3) [Andreica et al., 2013, Sirbu et al., 2014b]

  - A comparison on how two machine learning algorithms: SVM and GP are able to learn based on KD features extracted by using three kernels: Exponential, Gaussian and Laplacian (Subsection 4.3.2) [Andreica et al., 2013]
  - A comparison on how KDs perform on single vs. multi-modality images, with parameters optimized independently on each modality: intensity, depth and flow (Subsection 4.3.3) [Sirbu et al., 2014b]

- Two dynamic machine learning based algorithms, capable to dynamically determine the most suitable modalities to classify an image

  - A dynamic modality selection algorithm which retains one suitable modality among intensity, depth and flow (Subsection 4.4.4) [Rus et al., 2015]; experimental evaluations of the algorithm on a pedestrian data set, analysis of the results and comparisons to other approaches from the literature (Subsection 4.4.4.1) [Rus et al., 2015]
  - A dynamic modality fusion algorithm which fuses the modalities considered suitable among intensity, depth and flow (Subsection 4.4.4) [Sirbu et al., 2015]; experimental evaluations of the algorithm on a pedestrian data set, analysis of the results and comparisons to other approaches from the literature (Subsection 4.4.5.1) [Sirbu et al., 2015]

Chapter 1

Unsupervised classification. Background.

In this chapter we are presenting the background knowledge related to the unsupervised classification problem approached in the thesis, the problem of *dynamic clustering of gene expression data*. Thus, in Section 1.1 we give a short overview on clustering, with emphasis on *k-means*, *fuzzy c-means* and *hierarchical clustering* algorithms, which stand at the basis of our proposed approaches introduced in Chapter 2. In Section 1.2 we present a special type of clustering, the dynamic clustering, together with some existing dynamic approaches from the literature. The problem of dynamic clustering of gene expression data is presented in Section 1.3 followed by several dynamic approaches existing in the literature for clustering gene expression.

1.1 Clustering

In this section we give a description of clustering, followed by a brief review of the main clustering algorithms [Jain and Dubes, 1988, Han and Kamber, 2006].

Clustering implies partitioning a particular data set in groups, whose components are similar to each other [Jiang et al., 2012]. According to Jain and Dubes [Jain and Dubes, 1988], clustering is a type of unsupervised classification applied on a finite set of instances (objects), between which the relationship is represented by a proximity matrix. Kendal and Stuart [Kendall and Stuart, 1966] suggest the term of *clustering* for techniques that group variables, and *classification* for techniques that group individuals. However, *clustering* represents the most important *unsupervised learning* problem in the machine learning domain.

Lance and Williams [Lance and Williams, 1967] propose a taxonomy of classification types (see Figure 1.1), where each leaf represents a different type of classification problem. The first level of the tree divides classification problems into *exclusive* and *non-exclusive*. The exclusive classification is a partitioning of a set of objects, in which each object is part of a single cluster. Non-exclusive classification can assign an object to multiple classes. A

type of non-exclusive clustering is *fuzzy* clustering, in which an object is assigned to all clusters with membership degrees ranging from 0 and 1. In the second level of the tree, exclusive classification is divided into *intrinsic* and *extrinsic*. If the classification uses only the proximity matrix it will be called an *intrinsic* classification. This type of classification is also called *unsupervised learning* because are not used category labels on objects representing an a priori known partition. Extrinsic classification uses both category labels and the proximity matrix in order to determine a discriminant surface that separates the objects into categories. Furthermore, in the last level of the tree, intrinsic classification is divided into *hierarchical* and *partitional*. Jain and Dubes [Jain and Dubes, 1988] explain hierarchical classification as a nested sequence of partitions, while partitional clustering as a single partition. The most representative partitional clustering methods are the *k-means* and the *k-medoids* methods.

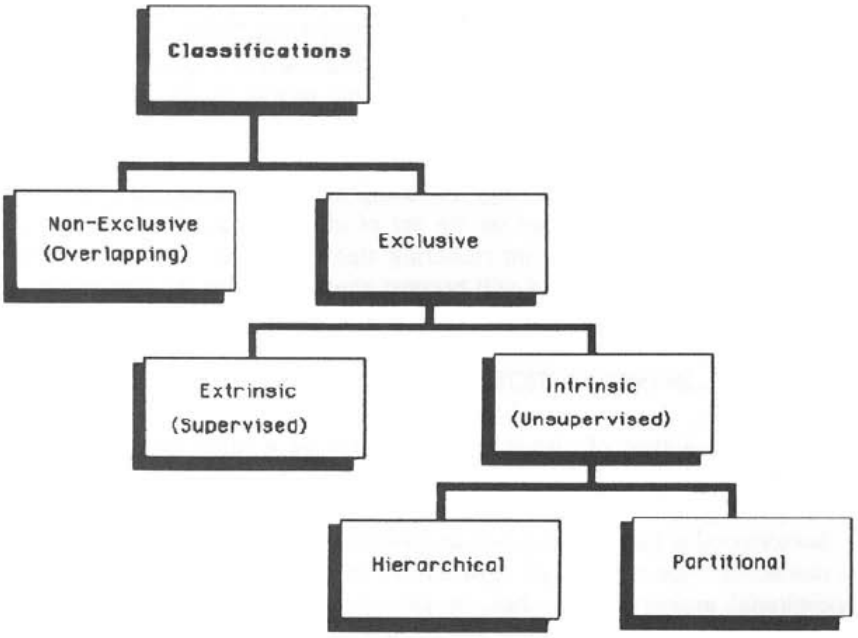


Figure 1.1: Tree of classification types [Jain and Dubes, 1988]

1.1.1 Distance metrics

An important step in clustering is determining the distance/dissimilarity among objects. There are numerous distance metrics for clustering available in the literature. In the following we are going to give a brief overview on them.

Let $Dist$ be a function, where $Dist : X \text{ x } X \rightarrow [0, \infty]$. It is called a metric of X if for each x, y, $z \in X$:

- $Dist(x, y) = 0$, if and only if $x = y$ (the identity axiom) [Singh et al., 2013];
- $Dist(x, y) = Dist(y, x)$ (the symmetry axiom) [Singh et al., 2013];
- $Dist(x, y) + Dist(y, z) \geq Dist(x, z)$ (the triangle axiom) [Singh et al., 2013].

To be noted that, in clustering, the distance measure can also be semi-metric (does not verify the triangle inequality).

Euclidean distance [Singh et al., 2013] computes the root of square difference between a pair of objects.

$$Dist(X,Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2} \tag{1.1}$$

Manhattan distance [Singh et al., 2013] sums the absolute differences between pair objects.

$$Dist(X,Y) = \sum_{i=1}^{n}|X_i - Y_i| \tag{1.2}$$

Chebychev distance [Singh et al., 2013], also known as maximum value distance, determines the absolute magnitude of the differences between a pair of objects.

$$Dist(X,Y) = Max_i|X_i - Y_i| \tag{1.3}$$

Minkowski distance [Singh et al., 2013] is a generalised distance.

$$Dist(X,Y) = (\sum_{i=1}^{n}|X_i - Y_i|^{\frac{1}{p}})^p \tag{1.4}$$

For p = 1, it represents the Manhattan distance and for p = 2 the Euclidean distance. Moreover, Chebychev distance is a variant of Minkowski distance when p=∞ [Singh et al., 2013].

Mahalanobis distance [Mahalanobis, 1936] is based on the correlation between variables. Unlike *Euclidean distance*, it takes into consideration the correlation of the data and is scale-invariant [Xu et al., 2013].

Let $X = (X_1, X_2, \ldots, X_n)^T$ be an observation from a set of observations with mean $\mu = (\mu_1, \mu_2, \ldots, \mu_n)^T$ and μ the covariance matrix. The Mahalanobis distance of X and S is computed as:

$$D_M(X) = \sqrt{(X-\mu)^T S^{-1}(X-\mu)} \tag{1.5}$$

Another way of defining Mahalanobis distance is as a dissimilarity measure between two random vectors $\vec{x}$ and $\vec{y}$ having the same distribution and the covariance matrix S:

$$d(X,Y) = \sqrt{(X-Y)^T S^{-1}(X-Y)}. \tag{1.6}$$

The Mahalanobis distance will be reduced to the Euclidean distance in the case when the covariance matrix is the identity one, and to normalized Euclidean distance:

$$d(X,Y) = \sqrt{\sum_{i=1}^{N}\frac{(X_i - Y_i)^2}{s_i}}, \tag{1.7}$$

if the covariance matrix is diagonal, where s_i represents the standard deviation of X_i and Y_i from the sample set [Xu et al., 2013].

1.1.2 K-means clustering

The *k-means* algorithm takes as parameter the number of clusters k and performs a partitioning of a set of n objects into k clusters, with the goal of achieving a high intracluster similarity and a low intercluster similarity. The similarity between clusters is computed against the mean value of the objects within the cluster, referred as its centroid [Han and Kamber, 2006].

The algorithm begins by randomly selecting k objects, as cluster centroids. Each of the remaining objects are assigned to the nearest cluster considering the distance between the object and the centroid of the cluster, then a new centroid for the clusters is computed. The process repeats until convergence. The commonly used convergence criterion is the square-error

$$E = \sum_{i=1}^{k} \sum_{o \in C_i} |o - m_i|^2 \tag{1.8}$$

where E is the sum of the square error for all instances in the data set, o an object represented by a point in space and m_i is the mean of cluster C_i. The objective of this criterion is to create clusters as compact and separate as possible [Han and Kamber, 2006].

The time complexity of the *k-means* algorithm is $O(lknm)$, where l represents the number of iterations, k denotes the number of clusters, n is the number of objects and m indicates the number of attributes. We have to mention that there are also variations of the classical algorithm, which apply different optimization techniques.

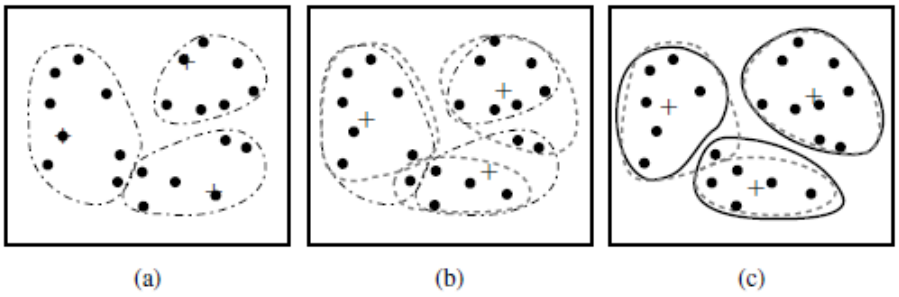


Figure 1.2: Clustering a set of objects using k-means algorithm [Han and Kamber, 2006]

1.1.3 Hierarchical clustering

The hierarchical clustering algorithms perform a partitioning of objects in a tree of clusters. Depending on how hierarchical decomposition is done, bottom-up by merging or top-down by splitting, hierarchical clustering algorithms can be classified into *agglomerative* or *divisive*.

The agglomerative clustering method, starts with a partition in which each object is placed in its own singleton. Then, the pair of closest clusters is merged into a single one, creating a partition and decreasing by one the number of clusters. This step is repeated until all objects belong to a single cluster. The divisive hierarchical clustering method performs the same steps, but in reversed order: starts with a partition where all objects are in a single cluster, and ends when all objects are placed in their own cluster.

The hierarchical clustering process has a tree-like representation, called *dendogram*, which illustrates how objects are grouped together step by step.

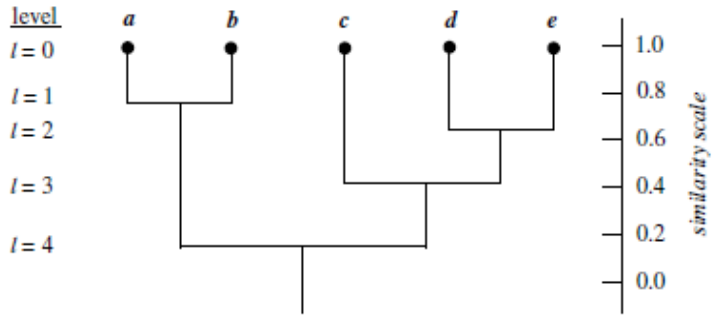


Figure 1.3: Dendogram for hierarchical clustering of five objects [Han and Kamber, 2006]

In order to decide which clusters to merge, the distance between clusters is computed. There are several measures used in the literature for the distance between two clusters, denoted by $d(C_i, C_j)$:

- **minimum distance** [Han and Kamber, 2006] : $d_{min}(C_i, C_j) = min_{o \in C_i, o' \in C_j} d(o, o')$
- **maximum distance** [Han and Kamber, 2006] : $d_{max}(C_i, C_j) = max_{o \in C_i, o' \in C_j} d(o, o')$
- **mean distance** [Han and Kamber, 2006] : $d_{mean}(C_i, C_j) = d(m_i, m_j)$
- **average distance** [Han and Kamber, 2006] : $d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{o \in C_i} \sum_{o' \in C_j} d(o, o')$

where $d(o, o')$ expresses the distance between two objects o and o', n_i represents the number of objects from C_i and m_i is the mean of C_i.

The clustering algorithm that measures the minimum distance between clusters is usually called *nearest-neighbour*. Moreover, if the clustering process ends when the distance between the closest clusters is greater than an established threshold, it is named as *single-linkage* algorithm. The clustering algorithm that measures the maximum distance between clusters is usually called *farthest-neighbour*. If the clustering process ends when the distance between the closest clusters is greater than an established threshold, it is named as *complete-linkage* algorithm.

In the context in which minimum and maximum distance measures are sensitive to outliers and noisy data, representing two extremes, the *mean* or *average* distance are oftenly chosen

as a compromise between the two. The mean distance is used for numeric data, whereas average distance can handle also categoric data.

The time complexity of *hierarchical clustering* algorithm depends on the linking type. In the case of *single-link* clustering the complexity is $O(n^2)$, while for *complete-link* is $O(n^2 \log n)$ where n is the number of objects.

1.1.4 Fuzzy C-means clustering

Fuzzy c-means clustering (FCM) [Albayrak and Amasyali, 2003], [Jain and Dubes, 1988], or Fuzzy ISODATA, represents a clustering method which is different from hard k-means clustering. *FCM* uses the idea of fuzzy partitioning, where a data instance (object) is assigned to all clusters with membership degrees (varying from 0 to 1).

FCM uses fuzzy sets in the clustering process, associating to each object a degree of membership to each cluster.

Denoting by k the desired number of clusters, a matrix U is used, where U_{ij} ($i \in \{1,,2...,k\}, j \in \{1,2...,n\}$ expresses the membership degree of object j to cluster i, such that

$$\sum_{i=1}^{k} U_{ij} = 1, \forall j \in \{1,2,...,n\} \tag{1.9}$$

Through an iterative process, FCM updates the cluster centroids and the membership degrees, in order to move the cluster means to the correct place within the data set. The convergence of *FCM* to the optimal solution is not ensured, because of the random initialization of the initial centroids i.e the initial values for matrix U. The algorithm reports the final values for the matrix U. Considering the final membership degrees given by the matrix U, an object O_j is usually assigned to the cluster $i = argmax_{l=1,k} U_{lj}$. Then, the following formula is used:

$$K_i = \{j \mid j \in \{1,...,n\}, U_{ij} > U_{rj}, \forall r \in \{1,...,n\}, r \neq i\},$$

in order to obtain the clusters in data after applying *FCM*. The number of clusters formed by *FCM* is at most k, because empty clusters could be obtained.

The time complexity for the *FCM* clustering algorithm is $O(nmk^2i)$, where i represents the FCM passes over the dataset, the number of objects is expressed by n, the number of clusters by k and the number of attributes by m.

1.2 Dynamic clustering

A clustering algorithm can be regarded as *dynamic* from several perspectives: operates on dynamic data sets, or/and adapts in some way the clustering process (e.g. adaptation from external feedback, dynamic thresholds). In the following we are going to briefly review

them, together with some existing approaches from the literature, with focus on the first type of dynamism, which will be tackled in this thesis.

Firstly, a clustering algorithm is considered *dynamic* if it is able to handle data sets that are subject to change. The term *dynamic* represents a generalization of the terms *adaptive*, *incremental* and *temporal* used in the literature to describe this particular type of clustering. Moreover, the dynamism of the data sets can be regarded from two perspectives: when new objects (instances) are added to the data set, or when new attributes are added to the existing objects. Several approaches in the literature address the problem of dynamic clustering when new objects are sequentially added to the data set.

In [Charikar et al., 2004] is proposed a hierarchical incremental clustering algorithm for document and image classification, which is able to maintain clusters of small diameter in an efficient manner, in the context in which new points are added.

In [Li et al., 2005] is proposed an adaptive clustering algorithm for network clustering (SACA), which adaptively forms clusters based on an accurate clustering measure called SCM, taking into account the connectivity of the nodes. In order to join or leave their clusters, the nodes must fulfill the condition of improving the SCM value of the whole network. Experiments prove the efficiency and accuracy of the method, even for large topologies.

In [Young et al., 2010] is presented a fast and stable incremental clustering algorithm that is based on a winner-take-all partition approach. The efficiency of the method considering the computational time and memory requirements is proved by simulations on several practical scenarios.

In [Kulic et al., 2008] is presented an incremental clustering based on the distance, in which newly acquired observations are represented using a Hidden Markov Model and then compared to existing clusters. The method was applied to incremental learning of human motion patterns using on-line observations. Experiments on a sequence of continuous human motion data confirm the performance of their approach.

In [Aaron et al., 2014b] is presented a dynamic two phase single-pass k-means clustering method, based on the k-means pyramid approach. In the first phase a large number of centers is used, then in the second one the centers obtained in the first phase are clustered into an established size. The performance of their approach is proved by experiments on color quantization. A fuzzy c-means version of the algorithm is introduced in [Aaron et al., 2014a].

In [Li and Chen, 2010] is proposed a fuzzy k-means incremental clustering approach, based on k centers and vector quantization. The authors use k centers in order partition data points into maximum k clusters, then vector quantization to assign new coming data points to clusters. Two centroids clustering is used to create a tree for k-means and reduce the time for calculating incremental data objects, also to decide if the new object should be added to an existing group or form a new cluster. Several data sets are used for validating the proposed approach.

In [Li et al., 2015] is introduced a batch dynamic incremental c-means clustering method

based on rough fuzzy set, BD-RFCM, in which new available data is not added one by one to the data set, but in form of cluster. The affiliation between a cluster from new incremental data and a cluster from original data is measured by the inclusion degree, which is similar to the membership degree of fuzzy set. Experiments on a synthetic data set confirm the effectiveness and correctness of the algorithm.

Most of the existing clustering methods, such as the *k-means* algorithm [Jain, 2010] or *hierarchical agglomerative clustering* algorithm (HACA) [Han and Kamber, 2006], start with a known set of objects, characterized by a set of attributes (features). All attributes are simultaneously used when computing objects' similarity. However, various applications where the feature set characterizing the instances increases exist, thus, re-clustering is required. An option in this situation would be to apply the clustering algorithm from the beginning on the attribute-extended objects, but would not be efficient. In order to overcome this problem, Şerban and Câmpan introduce in [Serban and Campan, 2005] and [Serban and Campan, 2006] two adaptive clustering algorithms that are able to identify a new partition of the set of objects, when the features set increases. The methods start from the set of clusters that was obtained by applying *k-means*, respectively *HACA* before the feature set was extended. This partitioning is adapted considering the newly added features. The authors show that the result is obtained more efficiently than applying *k-means*, respectively *HACA* from the beginning on the extended data set.

The idea of adaptive clustering introduced in [Serban and Campan, 2005, Serban and Campan, 2006] is used in chapter 2 and extended to handle the problem of *dynamic clustering* of gene expression data.

An earlier work [Wu and Gardarin, 2001] addresses the problem of gradual clustering by treating features sequentially in the clustering algorithm. Their algorithm, based on DBSCAN, reduces the number of distance calculations by using the triangle inequality. First, it stores the distances between a representative object and objects in n-dimensional space, then further use them to avoid distance calculations in (n+m)-dimensional space. Experimental evaluations confirm the efficiency of their approach.

Secondly, a clustering algorithm is considered dynamic if it performs an adaptation of the clustering. In [Su et al., 2009] is presented an incremental clustering algorithm which dynamically changes the radius threshold value and scanns the dataset only once according to the memory capacity. Experimental evaluations performed on mushroom dataset illustrate the effectiveness of the method.

In [Bagherjeiran et al., 2005] is introduced an adaptive clustering model which uses external feedback to improve cluster quality. Execution time is speeded up by using past experience in an adaptive environment, in which the reward values of successive clusterings are learned through Q-learning.

[Davidson et al., 2007] presents an efficient approach for incremental constrained clustering, that addresses the situation when the constraints are incrementally given. An efficiently update of the clustering is performed, in order to verify the new and the old constraints,

instead of re-partitioning the entire data set.

1.3 Clustering of gene expression data

Nowadays, microarray technology and the more modern next-generation sequencing technology allow the collection of veru large genetic data in the form of gene expression. These new technologies offer the possibility to measure the expression levels, or the activity of thousands of genes belonging to cells from different organisms, as the cells are exposed to different environmental conditions. The amount, time or location of expression of a certain gene are important characteristics to be determined, as they have a great impact on the well functioning of cells or of an organism, on a broader scale. One of the most notorious procedures used to analyse the gene expression data obtained from microarray or nextgeneration sequencing experiments is clustering [Stekel, 2006].

A great number of algorithms have been proposed for clustering gene expression data sets that are not subject to change. Among these, we mention approaches based on the k-means [Bagirov and Mardaneh, 2006] or the fuzzy k-means algorithms [Arima et al., 2003], on artificial neural networks [Yuhui et al., 2002] or methods using self organizing maps in conjunction with hierarchical clustering [Herrero and Dopazo, 2002], with k-means clustering [Yano and Kotani, 2003] or with particle swarm optimisation [Xiao et al., 2003].

Concerning clustering of dynamic gene expression data sets, to our knowledge, there are no approaches in the literature that deal with the dynamic clustering problem in the context when new expression levels are added to the existing genes. Although, as mentioned, no techniques exist that cluster gene expression data containing instances with an increasing number of features, there are a series of incremental clustering methods designed to work for data sets in which the number of instances may increase over time. We will present in the following some of these incremental approaches, as well as other studies that use dynamic or incremental clustering methods.

Sarmah and Bhattacharyya [Sarmah and Bhattacharyya, 2010] present a density based approach technique for clustering gene expression data, which can also be applied for incremental data. Their algorithm, GenClus [Sarmah and Bhattacharyya, 2010], obtains a hierarchical cluster solution and has as an advantage the fact that it does not require the number of clusters as input. InGenClus, the incremental version of GenClus, uses the result offered by GenClus and is able to update it when new genes are added to the data set, therefore decreasing the computational time. The algorithms are evaluated using real-life data sets and the reported results prove a good performance.

In [Das et al., 2009a] the authors propose an incremental clustering algorithm for gene expression data - incDGC, based on a clustering algorithm they had previously introduced. The main idea is that when a new gene is introduced into the data set, the current clustering should only be affected in the neighborhood of this gene. This algorithm does not need as input the number of clusters and it helps avoiding performing the clustering each time

the data set is updated. The algorithm was evaluated on three data sets and it proved to outperform other clustering algorithm, like as k-means or hierarchical clustering.

Lu *et al.* [Lu et al., 2004] introduce an Incremental Genetic K-means Algorithm (IGKA), which computes an objective value that the authors define and clusters centroids incrementally under the condition that the mutation probability from the genetic algorithm part is small, which leads to high costs of centroid calculation.

Bar-Joseph *et al.* present in [Bar-Joseph et al., 2002] a clustering algorithm for time series gene expression data that performs the clustering on continuous curve representations of the data, obtained using statistical spline estimation.

In [Das et al., 2009b] the authors present a clustering method that uses a dynamically computed threshold value to compute the membership of an object to a cluster.

In [Das et al., 2011] is presented a regulation based clustering approach, *PatternClus* [Das et al., 2011], for clustering gene expression data. An incremental version of PatternClus, in which clusters are incrementally identified in the context of a continuously increasing database, was also introduced. The proposed algorithm obtained better *z-score* values, when compared to k-means and hierarchical clustering.

An and Doerge [An and Doerge, 2012] introduce a novel dynamic clustering approach that deals with the time dependent nature of the genes so that the genes in the same cluster may have different starting and ending points or different time durations.

Chapter 2

New approaches for dynamic clustering of gene expression data

In this chapter, we address the problem of dynamic clustering of gene expression data and we propose three dynamic clustering algorithms, which can handle newly collected data, by starting from a previously obtained partition, without the need to re-run the algorithm from the beginning. In the same context of dynamic data, an algorithm for adaptive relational association rule mining of gene expression is also proposed [Czibula et al., 2015a]. The models introduced in this chapter are original, and were introduced in [Bocicor et al., 2014, Sirbu, 2014, Sirbu and Bocicor, 2013, Sirbu et al., 2014a, Czibula et al., 2015a].

The chapter is organized as follows. Section 2.1 defines the problem of dynamic clustering of gene expression data and its relevance, as well as some practical applications of solutions to this problem. Section 2.2 begins by presenting some theoretical considerations regarding our proposed approaches, based on two techniques that were proposed in the literature which handle the problem of clustering a set of objects, when the attribute set characterizing these objects increases [Serban and Campan, 2005, Serban and Campan, 2006] and continues in with an heuristic that we introduced foridentifying identify the optimal number of clusters in a gene expression data set, along with our dynamic approaches for clustering gene expression data.

Further, the proposed algorithms are experimentally tested on a real life gene expression data set and the results are reported in Section 2.3 [Bocicor et al., 2014, Sirbu, 2014, Sirbu and Bocicor, 2013, Sirbu et al., 2014a]. A discussion of these results, as well as comparisons of our algorithms with other techniques from the literature are given in Section 2.4 [Bocicor et al., 2014, Sirbu, 2014, Sirbu and Bocicor, 2013, Sirbu et al., 2014a]. In the same section we introduce an adaptive model for association rule mining of gene expression, which is able to adapt previously obtained rules when feature-set is extended, without performing re-mining from scratch [Czibula et al., 2015a]. Finally, Section 2.5 outlines the conclusions of this chapter and presents possible further work.

The original contributions of this chapter are the following:

- A dynamic core based clustering algorithm, which can handle newly collected gene expression data by starting from a previously obtained partition, without the need to re-run the algorithm from the beginning (Subsection 2.2.3.1.1) [Bocicor et al., 2014].
  - An heuristic to compute the optimal number of clusters in a gene expression data set (Subsection 2.2.2) [Bocicor et al., 2014].
  - Experimental evaluations of the dynamic core based clustering algorithm on a real life gene expression data set, analysis of the results and comparisons with results obtained by other dynamic approaches in the literature (Subsections 2.3.3.1 and 2.4.2) [Bocicor et al., 2014].
- A dynamic algorithm for hierarchical clustering of gene expression data, which can handle the newly arrived data by adapting a previously obtained partition, without the need of re-running the algorithm from scratch (Subsection 2.2.3.2.1) [Sirbu and Bocicor, 2013]
  - Experimental evaluations of the algorithm for hierarchical clustering on a real life gene expression data set, analysis of the results and comparisons with results obtained by our other dynamic approaches (Subsections 2.3.3.2 and 2.4.2) [Sirbu and Bocicor, 2013].
- A dynamic algorithm for fuzzy clustering of gene expression data, which can handle the newly arrived data by adapting a previously obtained partition, without the need of re-running the algorithm from scratch (Subsection 2.2.3.3.1) [Sirbu et al., 2014a]
  - Experimental evaluations of the algorithm for fuzzy clustering on a real life gene expression data set, analysis of the results and comparisons with results obtained by our other dynamic approaches(Subsections 2.3.3.3 and 2.4.2) [Sirbu et al., 2014a].
- A dynamic algorithm for relational association rule mining of gene expression data, which can handle the newly arrived data by adapting previously obtained rules, without the need of re-running the mining algorithm from scratch (Subsection 2.4.3.2)[Czibula et al., 2015a]
  - Experimental evaluations of the algorithm for relational association rule mining on a real life gene expression data set and analysis of the results (Subsection 2.4.3.3) [Czibula et al., 2015a].

2.1 Problem statement and relevance

In this section we aim at addressing the problem of dynamic clustering of gene expression data and its relevance, as well as some practical applications of solutions to this problem.

2.1.1 Problem definition and motivation

The emergence of microarray technology, that allows measuring the levels of expression of thousands of genes conducted to an exponential increase in the quantity of gene expression data. Still, all this data would be useless unless relevant biological information was extracted from it, therefore thorough exploratory analysis are usually required and performed. One of the mostly used techniques for this analysis and, most frequently, the first step, is clustering.

Clustering refers to creating a set of groups (clusters) and assigning each instance of a data set to one of these groups, according to a certain similarity measure. In what concerns gene clustering, the goal is twofold: firstly, by dividing the huge amount of gene expression data into clusters, this data becomes easier to process and analyse; secondly, but not less important, it is assumed that genes having similar expression patterns in a period of time (during the experiments) are likely to have similar biological functions and therefore clustering can also be considered an initial stage in the process of determining gene functions.

Gene expression data is usually collected in order to investigate the progress of different biological processes, as they evolve under different conditions and over time. Since biological processes are dynamic and time varying, they are best described by time series gene expression data [An and Doerge, 2012]. A time series data set consists of data obtained from biological experiments: samples of cells or tissues are prelevated from the same organism at different time points, through the evolution of the biological process. Therefore, for each of the targeted genes, the level of expression is measured at several distinct points in time. The data set will then be composed of thousands of genes (instances), each gene being characterized by a set of attributes (features): its expression levels (which can be quantified as real numbers) at different points in time.

However, there are some processes worth studying that may take days, or even months (e.g. diseases that progress over time), as well as experiments that are conducted over longer durations of time. For such cases, researchers must either wait until the experiment finishes and the expression levels of the genes are available at all moments of time, or analyse data gradually, as the experiment progresses. Gene expression clustering might be performed during the evolution of the experiment, at intermediate time steps, when genes would be characterized by only a subset of features. The main disadvantage is that as the experiments advance and new data becomes available (expression levels of the targeted genes for new points in time), the clustering process must once again start from scratch, which requires considerable time (especially as the number of the genes in such data sets is extremely high), in the end leading to a slower and more inefficient processing of the data.

2.1.2 Practical applications

Nowadays, in this postgenomic era, one of the greatest concerns of scientists is to deeply understand the functioning of organisms. One step towards this understanding is the analysis of genes and clustering offers one way to achieve this analysis.

Among the most important practical applications of gene clustering is the identification of gene functions. Genes with similar expression profiles will be clustered together, therefore facilitating the prediction of the functions of unknown genes or the identification of gene sets that are regulated by the same mechanism [Luan and Li, 2003]. The information offered by gene clustering is often used for genome annotation [Pejaver et al., 2011]. In practice, gene clustering is used for different applications. Among these, we mention the study of molecular mechanisms of plant-pathogen interaction to the goal of determining those genes in certain plants or vegetables that are resistant to different pathogens [Lopez-Kleine et al., 2013]. Further, another application is in the pharmaceutical industry, where biosynthetic gene clusters identified in microbial genomes could lead to the development of novel pharmaceuticals [Frasch et al., 2013].

Biological systems are inherently dynamic, therefore the gene information is extracted at different moments in time. Clustering can be used for analysis at any point and thus the conclusions extracted from the obtained partitions can be practically used as previously mentioned. The contribution of our algorithm is that it can obtain partitions as new data is gathered, using the partitions obtained in a former phase of data collection and without having to re-apply the clustering algorithm from the beginning. The final goal of the method is to cluster genes. The clustering result is useful for analyzing gene expression data and to apply this analysis for any of the practical purposes described above.

2.2 Methodology

2.2.1 Theoretical considerations

In the following we introduce our approaches for dynamic clustering of gene expression data. Our proposals start from the ideas of incremental clustering previously introduced in [Serban and Campan, 2005, Serban and Campan, 2006] that are extended in the following to handle the problem of dynamic clustering of gene expression data.

We first introduce some theoretical considerations, common for all methods that we propose.

We consider that $X = \{G_1, G_2, \ldots, G_n\}$ is the set of genes to be classified. A gene is measured m times and is characterized by an m-dimensional vector $G_i = (G_{i1}, \ldots, G_{im}), G_{ik} \in \Re, 1 \leq i \leq n, 1 \leq k \leq m$. An element G_{ik} from the vector characterizing the gene G_i represents the expression level of gene G_i at time point k.

Let $\{K_1, K_2, \ldots, K_p\}$ be the cluster set identified in X by using the *k-means*, hierarchical or *fuzzy c-means* algorithms. Each cluster is a set of genes, $K_j = \{G_1^j, G_2^j, \ldots, G_{n_j}^j\}$, $1 \leq j \leq p$. The mean, also called centroid, of the cluster K_j is denoted by f_j, where $f_j = \left(\frac{\sum_{k=1}^{n_j} G_{k1}^j}{n_j}, \ldots, \frac{\sum_{k=1}^{n_j} G_{km}^j}{n_j} \right)$.

The similarity measures or distances widely used for gene expression data are the *Eu-*

clidean distance and the *Pearson correlation* [Kim et al., 2007]. The measure we use for discriminating genes is the *Euclidean distance*: $d(G_i, G_j) = d_E(G_i, G_j) = \sqrt{\sum_{l=1}^{m}(G_{il} - G_{jl})^2}$. We have chosen this type of distance for the present study because it takes into account the magnitude of the changes in gene expression [Kim et al., 2007].

The set of attributes consisting of the m expression levels of the genes coming from m consequent measurements is then extended with s ($s \geq 1$) new features, coming from new measurements, numbered as $(m+1), \ldots, (m+s)$. The genes' extended vectors are $G'_i = (G_{i1}, \ldots, G_{im}, G_{i,m+1}, \ldots, G_{i,m+s}), 1 \leq i \leq n$.

Our aim is to analyse how the extended genes are grouped into clusters, starting from the grouping obtained before the attribute set extension. Our goal is to achieve an increased performance in comparison with the process of partitioning from the beginning.

The main idea is that, when the *k-means* or *fuzzy c-means* clustering process is ended, all genes are closer to the mean of their cluster than to the means of other clusters. In this way, for any cluster j and any gene $G_i^j \in K_j$, the inequality 2.1 below holds.

$$d(G_i^j, f_j) \leq d(G_i^j, f_r), \forall j, r,\ 1 \leq j, r \leq p,\ r \neq j. \tag{2.1}$$

By K'_j is denoted the set composed by the same genes as K_j, after the feature set was extended, and by f'_j, the centroid of the set K'_j ($\forall 1 \leq j \leq p$). After the feature set was increased, it is not certain that the sets $K'_j, 1 \leq j \leq p$ form clusters. The newly added features can change the genes' placement into clusters, obtained such that the similarity within each cluster to be maximized and similarity between clusters to be mininimized. Taking into account that the genes' features have equal weights (and normal data distribution) it is very likely that by adding new features to the genes, the old partitioning in clusters is similar to the actual one. Certainly, the clusters after extending the feature set could be obtained by applying the clustering algorithm (*k-means* or *HACA* or *fuzzy c-means*) on the set of extended genes. But, this process can be computationaly costly, that is why we focus on replacing it with one less expensive, which preserves the accuracy of the obtained results. In the following, we will refer the sets K'_j as clusters.

2.2.2 Identifying the number of clusters

It is well known that a problem with the *k-means* and *fuzzy c-means* algorithms is that the choice of the initial centroids may lead to the convergence of the squarred error value to a local minimum, instead of a global one. On the other hand, in *HACA*, the process of merging clusters must stop when a certain number of clusters is reached. To identify the optimal number of clusters in the gene data set, as well as the initial centroids for applying *k-means*, or *fuzzy c-means* algorithm, we propose the heuristic bellow.

For obtaining the number p of clusters in the data set, we are searching for p representative genes (one in each cluster). This process is an iterative one and is summarized bellow:

(i) The number p of clusters is initialized with 0.

(ii) The gene with the maximum average distance from all other genes is selected as the representative gene from the first cluster. The number p of representative genes becomes 1.

(iii) For selecting the next representative gene we reason as in the following. For each gene that was not previously selected as representative, the average distance ($davg$) beween that gene and the representatives (which were already chosen) is computed. The gene selected as the next representative will be the gene g which maximizes $davg$ and is greater than a given positive threshold denoted by *distMin*. In the case when such a gene is not found, the iterative process of selecting the initial centroids stops, otherwise p is increased and step (iii) is performed again.

We remark that step (iii) presented above assures that near genes will be placed in the same cluster, instead of being assigned to different clusters.

2.2.3 Our approaches

2.2.3.1 The Core Based Dynamic Clustering of Gene Expression (CBDCGE) Approach

The first step of this approach consists of applying the *k-means* algorithm on the initial data set, the one in which each gene is characterized by m expression values, at m time points. We start from the observation that, after the intial *k-means* clustering process is finished, the distance between each gene and its cluster mean is smaller than the distance to any other cluster's mean. Hence, for any cluster j and any gene $G_i^j \in K_j$, inequality 2.2 below holds.

$$d(G_i^j, f_j) \leq d(G_i^j, f_r), \forall j, r,\ 1 \leq j, r \leq p,\ r \neq j. \tag{2.2}$$

Considering the partitioning into clusters obtained on the set of genes before the feature set extension, our focus is to identify conditions in which an extended gene $G_i^{j\prime}$ remains correctly assigned to the cluster K_j'. Therefore, we compare the distance between the gene $G_i^{j\prime}$ and the mean of its cluster (f_j') with the distance to the centroids of the other clusters.

Starting from the approach introduced in [Serban and Campan, 2005] it can be easily proven that when inequality (2.2) is verified for an extended gene $G_i^{j\prime}$ and the cluster K_j', for each added feature ($\forall\, l \in \{m+1, \ldots, m+s\}$), then the gene $G_i^{j\prime}$ is closer to the centroid f_j' than to other centroid f_r' ($1 \leq j,\, r \leq p,\ r \neq j$).

$$G_{il}^j \geq \frac{\sum_{k=1}^{n_j} G_{kl}^j}{n_j}. \tag{2.3}$$

2.2.3.1.1 The CBDCGE algorithm

The CBDCGE algorithm which will be described in the following adapts the idea from [Serban and Campan, 2005] for the particular problem of dynamic gene expression clustering.

We will use the inequality (2.3) for determining within each cluster $K'_j, 1 \leq j \leq p$, the genes that are very likely to remain grouped together in their cluster, instead of moving to another group after extending the feature sete. These objects are considered to represent the *core* of their cluster. The idea of our approach is to compute, for each cluster K_j, its core, denoted by $Core_j$.

Let us denote by $StrongCore_j = \{G_i^{j\prime} | G_i^{j\prime} \in K'_j, G_i^{j\prime}$ satisfies the inequality (2.3) $\forall l \in \{m+1, \ldots, m+s\}\}$. We denote by $WeakCore_j$ the set of genes in K'_j verifying inequality (2.3) for at least the average number of features (computed from all genes belonging to K'_j) for which (2.3) holds.

For an added feature l ($m+1 \leq l \leq m+s$), and a cluster K'_j, the gene which has the maximum value for the feature l among all genes from K'_j verifies inequality (2.3). It is possible that cluster K'_j does not contain any gene which satisfies inequality (2.3) for all added features $m+1, \ldots, m+s$. But if such genes exist, these genes are nearer to the centroid f'_j than to the other centroids f'_r, $\forall 1 \leq r \leq p$, $r \neq j$) and they will form the $StrongCore_j$. In this case, $Core_j$ will be chosen as the nucleus $StrongCore_j$ of the j-th cluster in the CBDCGE algorithm. In the case when $StrongCore_j$ is the empty set, then we will consider as nucleus for the j-th cluster the most stable genes from K'_j (those genes that satisfy inequality (2.3) for as many features as possible). These genes will form the set $WeakCore_j$.

The *cluster cores*, selected as described above, will represent the nuclei in the dynamic clustering process. If the clusters remain unchanged, then all genes from $Core_j$ will certainly remain in the same cluster. This will not be the case for all core genes, but for most of them.

The *Core Based Dynamic Clustering of Gene Expression (CBDCGE)* algorithm will be presented below.

The algorithm begins with computing the cores for the old clusters. These cores are considered as the initial clusters in the adaptive process. Then, the CBDCGE performs the same steps as the *k-means* algorithm. We have to mention that if at a certain iteration a cluster from the partition becomes empty, it is removed from the partition.

2.2.3.2 The Dynamic Hierarchical Clustering of Gene Expression (DHCGE) Approach

This approach is based on the hierarchical aglommerative clustering algorithm (HACA) and on the idea described in [Serban and Campan, 2006]. Therefore, we begin from the partitioning into clusters of the data set with m expression values characterizing each gene obtained by *HACA*. Then, we analyse the conditions necessary for the extended gene $G_i^{j\prime}$ to be well located in the cluster K'_j. Thus, we compute the distances between $G_i^{j\prime}$ and the means of its old and new clusters (f_j and f'_j) and we compare them with the distances to

Algorithm 1 The *CBDCGE* algoritm [Bocicor et al., 2014], [Serban and Campan, 2005]

Algorithm Core Based Dynamic Clustering of Gene Expression is

Input: - $X = \{G_1, \ldots, G_n\}$ the m-dimensional genes

- $X' = \{G'_1, \ldots, G'_n\}$ the (m+s)-dimensional extended genes
- the distance metric d between the genes,
- $K = \{K_1, \ldots, K_p\}$ the partitioning of X,
- $noMaxIter$ a gived maximum number of iterations

Output: -$K' = \{K'_1, \ldots, K'_p\}$ the new partition of genes in X'

Begin

For each cluster $K_j \in K$

@ compute core $Core_j$ as the core of clusterK_j

Compute the centroid f'_j of K'_j

EndFor

While (K' is changed between two consecutive iterations) and

(the number of iterations is less than $noMaxIter$) do

For each cluster K'_j do

$K'_j := \{G'_i \mid d(G'_i, f'_j) \leq d(G'_i, f'_r), \forall r\,, 1 \leq r \leq p, 1 \leq i \leq n\}$

If $K'_j = \emptyset$ then

@ remove K'_j from the partition K'

EndIf

EndFor

For each cluster K'_j do

Compute the centroid f'_j of K'_j

EndFor

EndWhile

End.

the old and new means means of the other clusters K'_r ($\forall 1 \leq r \leq p,\ r \neq j$). The genes from the j-th cluster that are verifying the conditions from (2.3) are considered to be sufficiently close to each other and will remain in the same cluster. The rest of the genes from cluster K_j will be distributed each one in a singleton. This way, k' clusters will be obtained. For getting to the desired number k of clusters, the clusters are merged as in the classical *HACA* algorithm, but in a smaller number of steps, as we do not start from singletons in most of the cases. The "linkage metric" used in our experiments is "average-link".

We identify conditions necessary for an extended gene $G_i^{j\prime}$ to be "correctly" located in K'_j, considering K_j as the cluster which contains G_i^j by applying *HACA*. These conditions are a particularization of those from [Serban and Campan, 2006] for gene expression and illustrate when a gene that is placed in a particular cluster remains closer to its cluster than to any other clusters after its feature extension. For the mathematical representation of these conditions, as well as for the general theorem, we refer the reader to [Serban and Campan, 2006].

The first condition requires a gene $G_i \in K_j$ to be closer to its cluster's centroid than to any other centroid after the initial clustering process is over. This will not be fulfilled for every genes against the clusters formed by *HACA*. But, considering the fact that *HACA* uses the average-link linkage metric, there is a considerable chance that a lot of genes will comply to this condition. The second condition refers to the features that are added to each gene and expresses a mathematical inequality that has to be satisfied by the new components $(m+1), \ldots, (m+s)$ of the genes. All genes $G_i \in K_j$ which, after the initial hierarchical clustering process, are closer to their cluster's centroid than other centroids and whose extensions fulfill the requirements expressed in inequality 2.3 are similar enough to each other and sufficiently dissimilar to the genes from other clusters. From this reason, after the feature set was extended, they can be grouped together in the same cluster.

2.2.3.2.1 The *DHCGE* algorithm

Our *DHCGE* algorithm starts from the partition obtained by *HACA*. Further, our method is based on identifying cores [Serban and Campan, 2006] inside existing clusters. These cores are composed of those genes that are very likely to remain in the same group, after the new attributes are added to all instances of the data set.

As in [Serban and Campan, 2006], we denote by $StrongCore_j$ the set of genes from the cluster K'_j that fulfill both conditions expressed in inequalities 2.2 and 2.3 , therefore they are closer to the mean of their cluster than to any other mean, after the addition of the new attributes and implicitly, they are correctly placed in their cluster. Similar to [Serban and Campan, 2006], we denote by $WeakCore_j$ the set of genes that fulfill the following requirements: (i) they were closer to the mean of their cluster than to any other mean, before the extension; (ii) they satisfy the second condition expressed in inequality 2.3 after the extension, for at least the average number of clusters (computed for all genes belonging

to K'_j) for which all the genes in K'_j satisfy this second condition.

If, for cluster K'_j, the set $StrongCore_j$ is not empty, then it will be considered as the core $Core_j$ for cluster K'_j. Else, for $Core_j$ not to be empty, the most stable genes between all genes in K'_j will be taken as seed for cluster j, i.e. the genes from the set $WeakCore_j$. The genes in the $WeakCore_j$ might or might not be as stable as those in the $StrongCore_j$. If the two conditions from 2.2 and 2.3 are fulfilled, then it is sure that the genes are closer to the center of the cluster they already belong to than to any other cluster center, meaning that the conditions are *sufficient*; but they are not also *necessary*. Thus, there is the possibility that in also in $WeakCore_j$ there are genes being closer to the center of the cluster K'_j than to any other.

The *DHCGE* algorithm begins by identifying the cores of the clusters obtained by applying *HACA* on the initial data set. Then, when the feature set is extended, the algorithm begins the clustering process starting from these cores and continuing as the traditional *HACA*. The advantage over applying *HACA* from scratch is that *DHCGE* does not start again from clusters composed of one gene, therefore the clustering process is accelerated. We mention that the linkage metric we used to group two genes together in the hierarchical process, we used the *average link metric*. For two clusters K_i and K_j, the distance given by the average link metric is expressed by the following equation:

$$d(K_i, K_j) = \frac{\sum_{a \in K_i} \sum_{b \in K_j} d(a, b)}{\mid K_i \mid \times \mid K_j \mid}. \tag{2.4}$$

The *Dynamic Hierarchical Clustering of Gene Expression* algorithm is given in Algorithm 1. When the reached number of clusters is the formerly determined using the heuristic described in Section 2.2.2, then the algorithm stops.

2.2.3.3 The Fuzzy Dynamic Clustering of Gene Expression (FDCGE) Approach

The first step of this approach consists of applying *fuzzy c-means* on the initial data set, the one in which each gene is characterized by m expression values, at m time points. It is known that, when the *fuzzy c-means* clustering is completed, all genes are nearer to the mean of their cluster that to other means.

Considering the partitioning into clusters obtained on the set of genes before the feature set extension, our focus is to identify conditions in which an extended gene $G_i^{j\prime}$ is correctly located its cluster K'_j. Starting from the approach introduced in [Serban and Campan, 2005] it can be easily proven that when inequality (2.5) is verified for an extended gene $G_i^{j\prime}$ and its cluster K'_j, for each added feature ($\forall\, l \in \{m+1, \ldots, m+s\}$), then the gene $G_i^{j\prime}$ is nearer to its mean f'_j of cluster K'_j than to the means $f'_r, \forall 1 \leq j,\, r \leq p,\, r \neq j$.

$$G_{il}^{j} \geq \frac{\sum_{k=1}^{n_j} G_{kl}^{j}}{n_j}. \tag{2.5}$$

Algorithm 2 The *DHCGE* algoritm [Sirbu and Bocicor, 2013], [Serban and Campan, 2006]

Algorithm Dynamic Hierarchical Clustering of Gene Expression is

Input:

- $X = \{G_1, \ldots, G_n\}$ the m-dimensional genes
- $X' = \{G'_1, \ldots, G'_n\}$ the (m+s)-dimensional extended genes
- the metric d_E between the multi-dimensional genes
- $K = \{K_1, \ldots, K_p\}$ the initial grouping of genes

Output:

- $K' = \{K'_1, \ldots, K'_p\}$, the partition of extended genes from X'

Begin

@heuristically compute the number nc of clusters
For each cluster $K_j \in K$ do
  @ compute core $Core_j$ as the core of clusterK_j
  @ compute $OCore_j \leftarrow K_j \setminus Core_j$
EndFor
$K' \leftarrow \emptyset$
For i = 1 to nc do
  If $Core_i$ is not empty then
    $K' \leftarrow K' \cup \{Core_i\}$
  EndIf
  For each $G \in OCore_i$ do
    $K' \leftarrow K' \cup \{G\}$
  EndFor
EndFor
While $\mid K' \mid > p$ do
  @ select from K' two clusters C_1 and C_2 having the minimum distance $d_E(C_1, C_2)$
  @ merge clusters C_1 and C_2 into a cluster C
  @ remove from the partition K' the clusters C_1 and C_2 and add the cluster C
EndWhile

End.

2.2.3.3.1 The *FDCGE* algorithm

We will use the inequality (2.5) in order to determine inside each cluster the genes that are likely to remain in their cluster, instead of moving into other cluster after extending the feature set. These genes form the *nucleus* of their cluster. The idea of our approach is to compute, for each cluster K_j, its nucleus, denoted by $Nucleus_j$.

Let us denote by $StrongNucleus_j = \{G_i^{j\prime} | G_i^{j\prime} \in K'_j, G_i^{j\prime}$ satisfies the inequality (2.5) $\forall l \in \{m+1 \ldots, m+s\}\}$. We denote by $WeakNucleus_j$ the set of genes in K'_j satisfying inequality (2.5) for at least the average number of features (computed from all genes belonging to K'_j) for which (2.5) holds. The idea behind computing, for each cluster K'_j, the $StrongNucleus_j$, $WeakNucleus_j$ is the same as for the algorithm $CBDCGE$, described in Section 2.2.3.1.

The *cluster nuclei* selected as mentioned above, will be considered as starting point in the adaptive fuzzy clustering method. If the clusters remain unchanged, then all genes from $Nucleus_j$ will certainly remain the same cluster. This will not be the case for all genes in the nuclei, but for most of them.

The algorithm begins by computing the clusters' nuclei, which will be further used as initial clusters in the iterative process. Then, the FDCGE performs the same steps as the classical *fuzzy c-means* method. We have to mention that if at a certain iteration a cluster from the partition becomes empty, it is removed from the partition.

2.3 Experimental evaluation

In this section we aim to experimentally evaluate our dynamic clustering algorithms on gene expression data. The data set used in our experiments, the evaluation measures, as well as the obtained results are presented in the following sections. The results are original and were introduced in [Bocicor et al., 2014, Sirbu, 2014, Sirbu and Bocicor, 2013, Sirbu et al., 2014a]

2.3.1 Gene expression dataset

For the computational experiments performs for evaluating the performance of our methods a real-life data set was used. It is taken from [DeRisi et al., 1997] and chosen for the following reasons:

- It is publicly available.
- It is a time series gene expression data set.
- It has been experimented on by several works approaching the clustering problem, thus giving us the possibility to provide a comparison of our results with existing ones.

Microarray technology was used by the authors of [DeRisi et al., 1997] to measure the levels of expression of 6400 genes belonging to the organism *Saccharomyces cerevisiae*, during

its metabolic modification from fermentation to respiration. The expression levels for the enes were measured at seven time points during the process: 0, 9, 11.5, 13.5, 15.5, 18.5 and 20.5 hours.

Before proceeding with the evaluation of the dynamic clustering algorithm, a pre-processing step must be applied on this data. First, we exclude the genes with missing expression levels and then the genes that are not expressed or whose expression values do not change are filtered out. To this purpose, we used the MATLAB Bionformatics Toolbox [Henson and Cetto, 2005], which offers functions that allow us to remove genes having small variance over time or having very low absolute expression values, as well as genes with low entropy profiles. Following this pre-processing, the data set is reduced to a total number of 614 genes.

2.3.2 Evaluation measures

We present in the following a set of *evaluation measures* that will be further considered for computing the quality of the partitions provided by the proposed clustering algorithms. The measures (*IntraD*, *Dunn* and *Dist*) evaluate a partition from a clustering perspective and the (*Z-score*) evaluates a partition from a biological point of view.

We consider a partition $K = \{K_1, \ldots, K_p\}$, where each cluster consists of a set of genes.

A. *IntraD* - Intra-cluster distance of a partition

The *IntraD(K)* represents the *intra-cluster distance* of K and is defined as:

$$IntraD(K) = \sum_{j=1}^{p} \sum_{i=1}^{n_j} d^2(G_i^j, f_j)$$

where K_j is a set of genes $\{G_1^j, G_2^j, \ldots, G_{n_j}^j\}$ and f_j is the center of K_j. Better partitions (from a clustering perspective) are reflected in small $IntraD$ values .

B. *Dunn* - Dunn Index

The *Dunn index* [Pakhira et al., 2004] of a partition K is defined as:

$$Dunn(K) = \frac{d_{min}}{d_{max}}$$

where d_{min} represents the smallest distance between two genes from different clusters and d_{max} is the largest distance among two genes from the same cluster. The *Dunn index* takes values from the interval [0,∞]. The greater the value of this index, the better a partition is, therefore the *Dunn index* should be maximized.

C. *Dist* - Overall distance of a partition

The *overall distance* of K, expressed by *Dist(K)*, is computed as:

$$Dist(K) = \sum_{j=1}^{p} d_j$$

where d_j is computed as the sum of distances between all pair of genes from K_j, i.e

$$d_j = \sum_{G_1, G_2 \in K_j} d(G_1, G_2)$$

Better partitions (from a clustering perspective) are reflected in small $Dist$ values .

D. Z-score.

Z-score [Gibbons and Roth, 2002] is a figure of merit, indicating the relationship between a clustering result and the functional annotation of the used genes, within the gene ontology created by the Gene Ontology Consortium [Ashburner et al., 2000]. A higher value of the z-score shows that the obtained clusters are more significant from a biological perspective and therefore a more accurate clustering. To compute the z-score for a partition we used the ClusterJudge software, which implements the algorithm described in [Gibbons and Roth, 2002].

2.3.3 Results

Considering an initial number of features (denoted by m) characterizing the genes from the considered data set (Subsection 2.3.1), and different values for the threshold $distMin$ used for determining the initial centroids in the *k-means* and *fuzzy c-means* processes (see Subsection 2.2.2), the experiments are conducted as follows:

1. The initial number nc of clusters and the starting means are identified in data set using the heuristic presented in Subsection 2.2.2. The *k-means*, *HACA*, respectively *fuzzy c-means* algorithm is applied on the data set consisting of m-dimensional genes, starting from the identified centroids and a partition $\mathcal{K}$ is provided. In our implementation if at a certain iteration in the *k-means* algorithm a cluster becomes empty, it is removed from the partition.

2. The set of features is now increased with s new features, denoted as $(m+1), \ldots, (m+s)$. The *CBDCGE*, *DHCGE*, respectively *FDCGE* adaptive algorithm (Subsection 2.2.3.1.1, 2.2.3.2.1, 2.2.3.3.1) is now applied, by adapting the partition $\mathcal{K}$ and considering the instances extended with the newly added s features.

3. The partition into clusters provided by *CBDCGE* algorithm (denoted by $\mathcal{K}_{alg_name}$) is compared with the one provided by the $k-means$ algorithm applied from scratch on the $m+s$-dimensional instances (denoted by $\mathcal{K}'$). We mention that the initial centroids considered in the partitional clustering process are the centroids identified at step 1. The comparison of the obtained partitions is made considering the evaluation measures presented in Subsection 2.3.2 (considering the clustering and biological perspectives) and also the number of iterations performed by the clustering algorithms. Excepting

the first iteration, which involves the computation of cores, our dynamic algorithms perform the same operations as the tradional ones in the iterations.

2.3.3.1 CBDCGE algorithm

For testing the performance of the algorithm we conducted several experiments, in each one we start with a different number of features an then we add the rest of the attributes (up to seven which is the total number of attributes). The gene expression levels (features) are in chronological order.

Since *k-means* based clustering methods are very sensitive to the selection of initial centroids, it is very likely that the initial centroids may have an impact on the accuracy of the obtained results. Thus, we performed a comparative analysis on different methods for centroids' identification within the *CBDCGE* algorithm, as follows:

Heuristic 1

The first heuristic method for selecting centroids is the one described in Subsection 2.2.2.

Heuristic 2

The second heuristic method for determining the appropriate number p of clusters, introduced in [Sirbu, 2014], is based on selecting p representative genes, as following:

(i) p is initially set to 0.

(ii) The gene with the maximum average distance from all other genes is selected as the representative gene from the first cluster. The number p of representative genes becomes 1.

(iii) For selecting the next representative gene we apply the following reasoning. For each gene that was not previously selected as representative, the minimum distance ($dmin$) beween that gene and the representatives (which were already chosen) is computed. The gene selected as the next representative will be the gene g which maximizes $dmin$ and is greater than a given threshold ($distMin$). In the case when such a gene is not found, the iterative process of selecting the initial centroids stops, otherwise p is increased and step (iii) is performed again.

Actually, the difference between the two heuristics is that the first one uses the average distance ($davg$), while the second one the minimum distance ($dmin$).

Random

The third way of choosing centroids is a random selection of p centroids, p being the number of clusters heuristically identified as above.

No.	**No. of clusters**	**No.of iterations**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	$distMin = 3.231$ $nc = 63$					
$\mathcal{K}'$	62	21	411.7653	0.1345	13604.5546	5.4240
$\mathcal{K}_{CBDCGE}$	49	12	421.0210	0.1632	11319.7105	7.1500
2	$distMin = 3.233$ $nc = 62$					
$\mathcal{K}'$	61	21	417.6670	0.1472	15449.8604	5.2920
$\mathcal{K}_{CBDCGE}$	49	25	417.4004	0.1945	11119.4377	8.0740
3	$distMin = 3.26$ $nc = 61$					
$\mathcal{K}'$	60	19	423.3145	0.1356	15811.1460	5.6460
$\mathcal{K}_{CBDCGE}$	49	20	423.1767	0.1957	11976.0106	7.3780
4	$distMin = 3.44$ $nc = 47$					
$\mathcal{K}'$	47	24	440.9141	0.1586	17209.7174	6.2650
$\mathcal{K}_{CBDCGE}$	43	23	437.4579	0.18087	16150.0690	7.3010
5	$distMin = 3.45$ $nc = 46$					
$\mathcal{K}'$	46	26	444.9301	0.1586	18100.0294	6.0810
$\mathcal{K}_{CBDCGE}$	43	20	433.2875	0.1923	15451.7017	8.0780
6	$distMin = 3.47$ $nc = 44$					
$\mathcal{K}'$	44	21	448.1514	0.1586	17251.2868	7.8660
$\mathcal{K}_{CBDCGE}$	43	14	445.0609	0.1848	17734.6491	9.2680
7	$distMin = 3.51$ $nc = 42$					
$\mathcal{K}'$	42	23	451.5669	0.1655	19597.3680	7.7190
$\mathcal{K}_{CBDCGE}$	40	15	436.4608	0.1664	15665.8960	10.5900

Table 2.1: Results for the first experiment.

2.3.3.1.1 First experiment

In our first experiment, we are initially considering 5 attributes (i.e $m = 5$) and afterwards the set of features is extended with 2 attributes (i.e $s = 2$). Table 2.1 presents the results obtained in our experiment. Considering different values for $distMin$ we indicate the initial number nc of clusters (heuristically determined as indicated above), and for the partitions $\mathcal{K}_{CBDCGE}$ and $\mathcal{K}'$ we indicate: the number of clusters in the partition, the number of iterations performed by the algorithm and the values of the evaluation measures (indicated in Subsection 2.3.2). We mention that the values reported for *z-score* are averaged over ten repeated experiments, for each value of *distMin*.

Analyzing the results indicated in Table 2.1, we observe the following:

1. Excepting the second case and third case (lines 2 and 3 in Table 2.1) *CBDCGE* performs less iterations than performs *k-means* applied from scratch.

2. The partitions obtained by the *CBDCGE* method are better (considering all the evaluation measures presented in Subsection 2.3.2) than the partitions obtained applying *k-means* from scratch. In all the cases the *Dunn index* computed on the results obtained by the *CBDCGE* is greater than the one obtained by applying *k-means* from scratch, which denotes more compact and well separated clusters. The same holds for the *z-score* measure, implying that the partitions obtained by *CBDCGE* are biologically more relevant. In what concerns the *IntraD* and *Dist* measures, they also indicate better partitions, exception being only two cases when either one or the other are greater for the *k-means* algorithm applied from scratch (line 1 - for IntraD and line 6 for Dist).

Considering the previous analysis, we can conclude that for the first experiment the partitions obtained adaptively (applying *CBDCGE* method) are better than the ones obtained by applying *k-means* from scratch. Also, the number of iterations performed by the clustering algorithm (excepting one case) is smaller for the *CBDCGE* method.

2.3.3.1.2 Second experiment

In the second experiment we have performed the evaluation of the *CBDCGE* method, by initially considering 6 attributes (i.e $m = 6$) and afterwards extending the set of features with 1 attribute (i.e $s = 1$). Table 2.2 presents the results obtained in this experiment. Considering different values for $distMin$ we indicate the initial number nc of clusters (heuristically determined determined as indicated in Subsection 2.2.2) and for the partitions $\mathcal{K}_{CBDCGE}$ and $\mathcal{K}'$ we indicate: the number of clusters, iterations performed and the values of the evaluation measures (indicated in Subsection 2.3.2). As in the case of the first experiment, the values reported for *z-score* are averaged over ten repeated experiments, for each value of *distMin*.

Analyzing the results indicated in Table 2.2 we observe the following:

1. Excepting the first case (line 1 in Table 2.2) and the fourth case (line 4 in Table 2.2) *CBDCGE* method performs a smaller number of iterations than *k-means* applied from scratch.

2. The partitions obtained by the *CBDCGE* method are better (considering all the evaluation measure presented in Subsection 2.3.2) than the partitions obtained applying *k-means* from scratch. Excepting the second case (line 4 in Table 2.2) the *Dunn index* computed on the results obtained by the *CBDCGE* is greater than the one obtained by applying *k-means* from scratch. The *z-score* is always greater and both the *IntraD* and *Dist* measures are lower, all these indicating a better clustering result obtained by *CBDCGE*.

Considering the previous analysis, we can conclude that for the second experiment the partitions obtained adaptively (applying *CBDCGE* method) are better than the ones obtained by applying *k-means* from scratch. Also, the number of iterations performed by the clustering algorithm (excepting two cases) is smaller for the *CBDCGE* method.

No.	**No. of clusters**	**No.of iterations**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	$distMin = 4.38 \quad nc = 63$					
$\mathcal{K}'$	62	21	411.7653	0.1345	13604.5546	5.6500
$\mathcal{K}_{CBDCGE}$	54	22	401.7995	0.1763	9276.8868	8.5700
2	$distMin = 4.401 \quad nc = 62$					
$\mathcal{K}'$	61	21	417.6670	0.1472	15449.8604	5.5350
$\mathcal{K}_{CBDCGE}$	53	18	406.4398	0.1525	10003.3949	7.5120
3	$distMin = 4.6 \quad nc = 45$					
$\mathcal{K}'$	45	26	446.8450	0.1586	18359.2485	6.5640
$\mathcal{K}_{CBDCGE}$	40	12	437.5520	0.1896	14658.3710	9.6970
4	$distMin = 4.66 \quad nc = 42$					
$\mathcal{K}'$	42	23	451.5669	0.1655	18757.3680	7.8520
$\mathcal{K}_{CBDCGE}$	40	23	438.2164	0.1615	14074.2680	8.2860

Table 2.2: Results for the second experiment.

2.3.3.1.3 Third experiment

In order to decide the most appropriate heuristic for selecting the initial centroids in the adaptive clustering process, we conducted two experiments. In each one we started from a different number of initial features and then we added the rest of the attributes.

In both experiments the centroids were identified in three ways: using *Heuristic 1*, *Heuristic 2* and randomly. For the randomly chosen centroids, an average obtained by five consequent runs was provided.

Experiment 1

In this experiment, the initial data set contains $m = 5$ features and the remaining $s = 2$ features are added subsequently. The obtained results are presented in Table 2.3 and Figures 2.1a-2.3a.

From these results we can conclude the following:

- The minimum number of iterations and the smallest *Dist* value are achieved by using Heuristic 1, both in K' and K_{CBDCGE}.
- The smallest $IntraD$ value is achieved by randomly choosing centroids, both in K' and K_{CBDCGE}.
- The highest *Dunn* value is achieved by using Heuristic 2, both in K' and K_{CBDCGE}.
- The highest $Z-score$ value is achieved by randomly choosing centroids in K' and using Heuristic 1 in K_{CBDCGE}.

No.	**No. of clusters**	**No.of iterations**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	Heuristic 1 $distMin = 3.47$ $nc = 44$					
$\mathcal{K}'$	44	21	448.1514	0.1586	392.0747	7.717
$\mathcal{K}_{CBDCGE}$	40	14	445.0609	0.1894	412.4337	9.951
2	Heuristic 2 $distAvg = 1.13$ $nc = 44$					
$\mathcal{K}'$	44	11	440.1101	0.2686	20685.6718	6.042
$\mathcal{K}_{CBDCGE}$	41	12	448.1529	0.2102	17133.5209	7.654
3	Random $nc = 44$					
$\mathcal{K}'$	44	15	424.8114	0.1363	10912.56594	7.8844
$\mathcal{K}_{CBDCGE}$	43	14	427.7379	0.1535	11593.03042	9.22

Table 2.3: Results for the first experiment.

No.	**No. of clusters**	**No.of iterations**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	Heuristc 1 $distMin = 4.66$ $nc = 42$					
$\mathcal{K}'$	42	23	451.5669	0.1655	446.604	8.23
$\mathcal{K}_{CBDCGE}$	40	23	438.2164	0.1621	351.8567	8.5044
2	Heuristc 2 $distAvg = 1.51$ $nc = 42$					
$\mathcal{K}'$	42	18	445.1501	0.2664	21649.7054	7.288
$\mathcal{K}_{CBDCGE}$	40	18	436.7869	0.2163	16133.5829	9.244
3	Random $nc = 42$					
$\mathcal{K}'$	42	15	430.68062	0.14342	11368.6995	9.0144
$\mathcal{K}_{CBDCGE}$	41	14	428.36222	0.19498	11848.0839	9.0853

Table 2.4: Results for the second experiment.

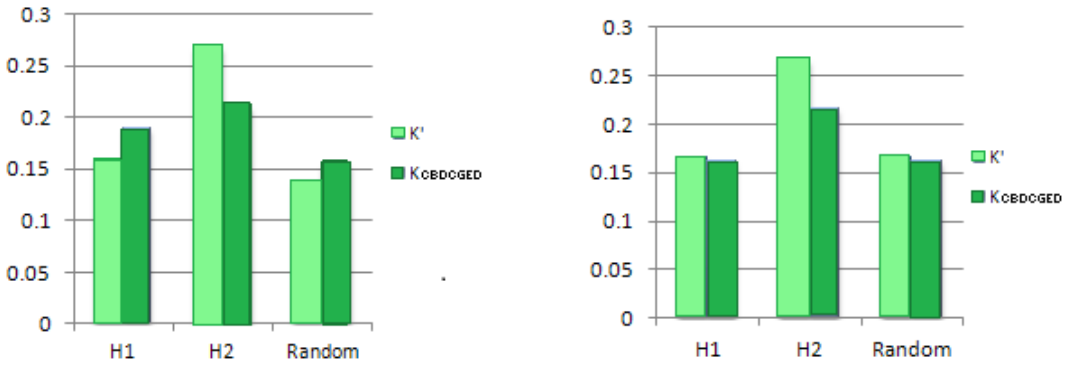
- From a biological point of view (considering the $Z - score$ evaluation measure), in all three cases the adaptive clustering outperforms the re-clustering from scratch process.

Experiment 2

In this experiment, the initial data set contains $m = 6$ features and the remaining $s = 1$ features are added subsequently. The obtained results are presented in Table 2.4 and Figures 2.1b-2.3b.

From these results we can conclude the following:

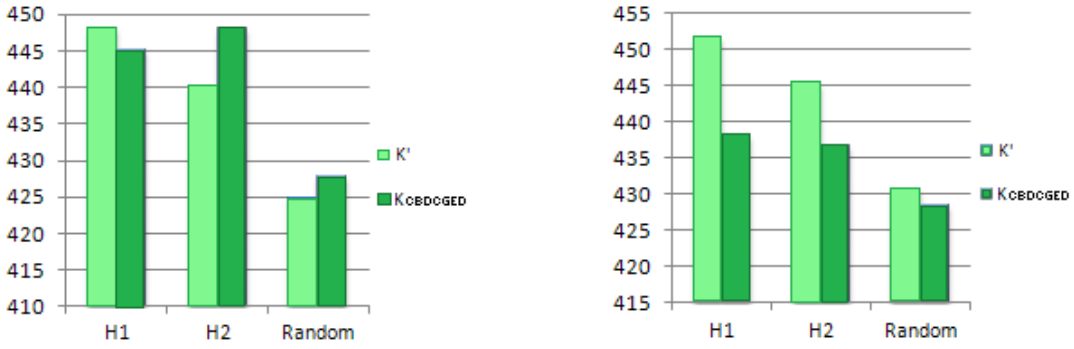
- The minimum number of iterations and the smallest $IntraD$ value are acheived by randomly choosing centroids, both in K' and K_{CBDCGE}.
- The highest $Dunn$ value is achieved by using Heuristic 2, both in K' and K_{CBDCGE}.



(a) First experiment. (b) Second experiment.

Figure 2.1: Illustration of the values of Dunn index obtained by using Heuristic 1, Heuristic 2 and random centroids, both for K' and K_{CBDCGE}.

- The smallest $Dist$ value is achieved by using Heuristic 1, both in K' and K_{CBDCGE}.
- The highest $Z-score$ value is achieved by randomly choosing centroids in K' and using Heuristic 2 in K_{CBDCGE}.
- The $Z-score$ evaluation measure, indicates in all three cases that the adaptive clustering outperforms the re-clustering from scratch.

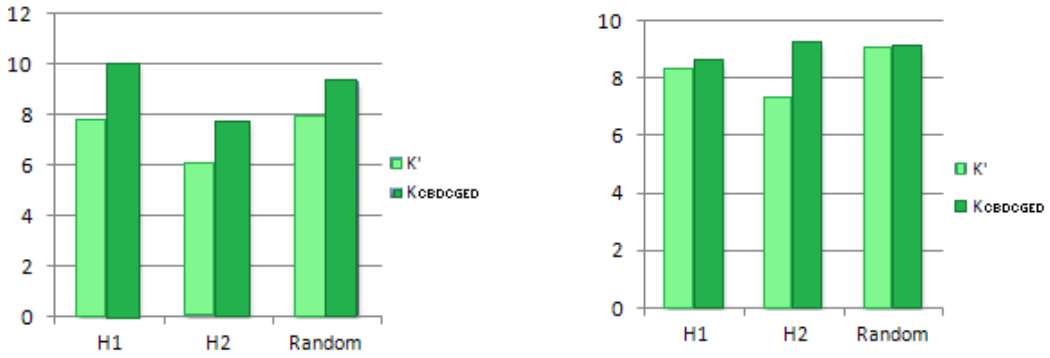


(a) First experiment. (b) Second experiment.

Figure 2.2: Illustration of the values of IntraD obtained by using Heuristic 1, Heuristic 2 and random centroids, both for K' and K_{CBDCGE}.

Statistical analysis

Since for the problem we approach in this thesis, gene expression clustering, the most relevant evaluation measure is the biological one, we performed a statistical analysis of Z-score values. We computed 95% Confidence Interval [Brown et al., 2001] for the average of the differences between the Z-scores obtained using the adaptive and from scratch approaches. All the Z-score values from the two experiments were considered. We obtained the (0.53,



(a) First experiment.

(b) Second experiment.

Figure 2.3: Illustration of the values of Z-score obtained by using Heuristic 1, Heuristic 2 and random centroids, both for K' and K_{CBDCGE}.

1.95) Confidence Interval for the average. Thus, there is a 95% confidence that the Z-score of the partition obtained adaptively exceeds the Z-score of the partition obtained by applying the k-means from scratch with a value that lies within the specified range.

Due to the variation of the results, we can not conclude which heuristic is the best. It depends on the evaluation measure (e.g. Heuristic 2 is the best from *Dunn* index perspective, but is not the best from $IntraD$ perspective), the type of algorithm (adaptive/from scratch), the number of features added (for the adaptive approach). Even if there are cases in which choosing centroids randomly gives better results than using heuristics, it does not represent a reliable option, as an inappropriate choice could strongly degenerate results.

Still, for both experiments we have performed, we can conclude that from a biological perspective (considering the $Z - score$ evaluation measure) a better approach is to use an heuristic for the initial centroids selection, instead of a random choice.

2.3.3.1.4 Discussion

Considering the experimental results presented in Section 2.3.3.1, an analysis of the *CBDCGE* algorithm is further provided. Furthermore, we present a study on the relevance of the considered features, as well as a comparison of our method to similar approaches in the literature.

a) Analysis of the results

The clustering technique that we proposed in this paper is generally suitable for dynamic data sets, in which the features characterizing the instances are continuously subject to change. Particularly, as biological processes and experiments are dynamic, clustering the gene expression data resulting from these led to the need of a dynamic approach.

Our *dynamic core based clustering* algorithm has two main advantages over the traditional *k-means* algorithm applied from the beginning: less iterations and better clustering accura-

cies. As can be seen in Tables 2.1 and 2.2, the number of iterations used by our dynamic algorithm for providing a solution is, on average, smaller that the one needed by *k-means*, run from the beginning, for the whole set of features. We have to mention that the running time of both algorithms is very small, around 3 seconds, therefore the dynamic algorithm performs similarly with the *k-means* from scratch in terms of execution time.

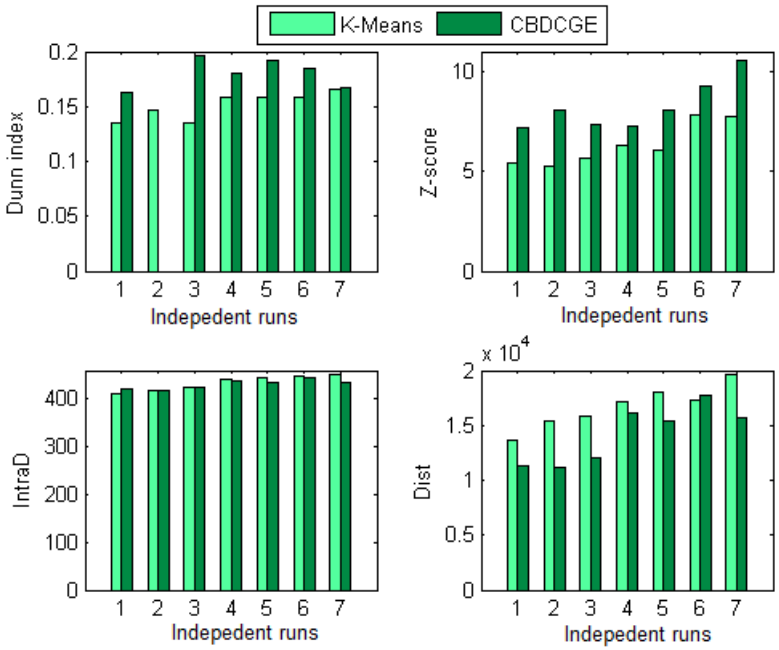


Figure 2.4: Illustration of the evaluation measures' values for the first experiment.

In what concerns the clustering accuracy, we used four evaluation measures (see Subsection 2.3.2) to help us evaluate each clustering result. Regarding the measures *IntraD* and *Dist* we mention that a decrease in their values signifies better partitions, while for the *Dunn* and for the *Z-score* greater values result from better clustering. The fact that our *CBDCGE* algorithm leads, in most cases, to better partitions than the *k-means* algorithm applied on the whole set of attributes, is illustrated in Figures 2.4 and 2.5. By analysing the top two plots of each of these figures, it can be clearly observed that, except for one case (Figure 2.5, *Dunn*, case 4), both the *Dunn* and the *Z-score* of our algorithm are greater than those obtained for *k-means*. The last two plots of the same figures show that, in almost all cases (except for Figure 2.4, *IntraD*, case 7 and *Dist*, case 6), the values of *IntraD* and *Dist* for *CBDCGE* are lower than those computed for *k-means*.

Another benefit of our approach, is that by using the heuristic presented in Subsection 2.2.2, it does not need the number of clusters. This value is computed by the algorithm, using a positive threshold, *distMin*, which represents the minimum distance to be used when deciding whether to assign genes to the same or to different clusters. By its definition, the increasing of *distMin* leads to the decreasing of the number of clusters. From Table 2.1 we note that the most biologically relevant clustering for the first experiment is obtained is for $distMin = 3.51$, while Table 2.2 indicates that for the second experiment, the best value of

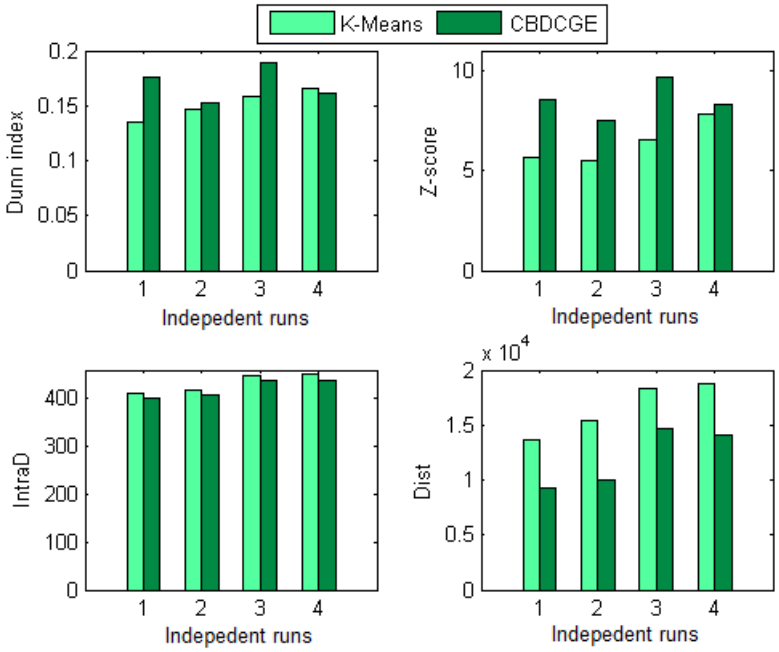


Figure 2.5: Illustration of the evaluation measures' values for the second experiment.

distMin is 4.6.

Regarding the number of new attributes, corresponding to new measurements, as the data set we used is composed of genes which were measured during a total of seven time points, we chose to use the first five time points for the initial partition and then incrementally add two attributes, in the first experiment or one, in the second. Table 2.5 presents the average values of the four evaluation measures for both algorithms (*k-means* applied from scratch and our adaptive algorithm *CBDCGE*) and for each of the two experiments. We remark that for the second experiment three out of the four considered evaluation measures, computed for *CBDCGE*, indicate that when only one attribute is added the obtained clustering is more accurate: apart from the *Dunn*, whose value is lower for the second experiment, compared to the first, the values of both the *IntraD* and *Dist* measures decrease and the *Z-score* is greater. The same table also demonstrates that the *CBDCGE* algorithm outperforms the *k-means* applied from scratch, as all the evaluation measures' values indicate better partitions: the values for *IntraD* and *Dist* are smaller, while those for the *Dunn* and the *Z-score* are greater.

b) Study on features' relevance

Information gain

In the following we aim at analyzing the influence of the *information gain (IG)* of the added features on the efficiency of the dynamic clustering process. The *information gain* measure is a measure from *information theory* and expresses the expected decrease in entropy determined by splitting the instances according to a given feature [Mitchell, 1997b]. For

Experiment	Algorithm	IntraD	Dunn	Dist	Z-score
First experiment	$\mathcal{K}'$	434.0442	0.1513	16717.7089	6.3276
	$\mathcal{K}_{CBDCGE}$	430.5522	0.1826	14202.4964	8.2627
Second experiment	$\mathcal{K}'$	431.9611	0.1515	16542.7579	6.4003
	$\mathcal{K}_{CBDCGE}$	421.0019	0.1700	12003.2302	8.5163

Table 2.5: Average values of the considered evaluation measures obtained for *k-means* and *CBDCGE* algorithms, after the two experiments.

computing the information gain of the attributes, the partition provided by applying *k-means* on the $m + s$-dimensional genes is used.

As gene expression levels take values in the $\Re$ space, for computing the information gain of the attributes we have to discretize their values. This was achieved by dividing their interval of variation into several sub-intervals.

Table 2.6 presents, for each experiment and each different number of sub-intervals we used, the features decreasingly ordered by their information gain (the added features are marked with bold), as well as a percentage indicating the information gain of the new features with respect to the one of the existing attributes.

Experiment	No. of subintervals	Order of feature	IG of new features / IG of old features (%)
First experiment	3	**7 6** 5 4 1 2 3	92.20%
	4	**7 6** 4 5 1 2 3	88.20%
	5	**7 6** 5 4 1 2 3	82.45%
	6	**7 6** 4 5 1 2 3	76.69%
Second experiment	3	**7** 6 5 4 1 2 3	51.37%
	4	**7** 6 4 5 1 2 3	50.78%
	5	**7** 6 5 4 1 2 3	51.25%
	6	**7** 6 4 5 1 2 3	50.66%

Table 2.6: The information gain measure for the attributes.

From Table 2.6 we can notice that the IG of the newly added attributes is rather high, when compared to the IG of the first set, especially for the first experiment. This may lead to a greater difficulty in adapting the partition (obtained by using the first set of attributes) for the *CBDCGE* algorithm. As mentioned before, the second experiment indicates that when only one attribute is added the obtained clustering is more accurate and this could be explained since in the second experiment the information gain introduced in the system is lower. Another conclusion is that that the IG of the attributes is monotonically related to the number of sub-intervals considered for the variation, as for both the existing and the new attributes the IG generally grows as the number of sub-intervals increases.

We have to mention two characteristics of the *CBDCGE* algorithm. First, the time complexity of the CBDCGE is not increased by the initial cores' computation step. The second characteristic is that the computation of the cluster' cores (using inequality 2.3) is based only on the current cluster.

Considering the above analysis of experimental results one can conclude that applying the adaptive *CBDCGE* method is effective for dynamic clustering of gene expression data.

Features' correlation

For studying how the sets of features are correlated with each other and to analyze how the correlation of the newly added attributes to the existing ones could influence the result of the clustering algorithm, we used the Pearson correlation coefficient [Tuffery, 2011]. This is a value ranging from -1 to 1 that measures the degree of linear correlation between two random variables. For each feature, we computed the Pearson correlation coefficient with the rest of the features. Figure 2.6 illustrates the average correlations among the features. By computing the mean M of the average correlations of initial features (the first five), we notice that the average correlations of the last two features (the new ones) are both higher than M. From this we can conclude that the adaptation process occurs in a simpler manner.

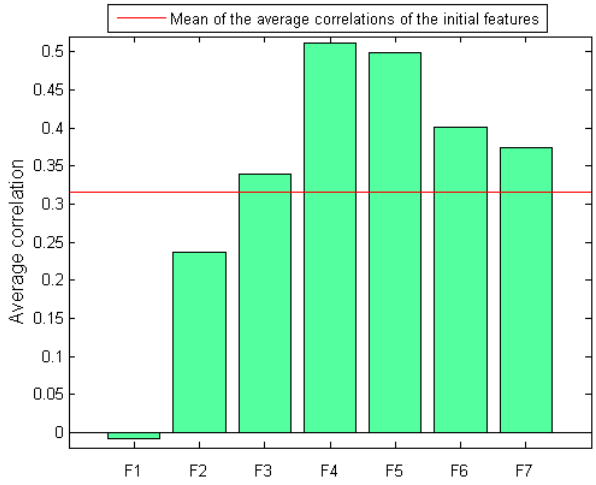


Figure 2.6: Illustration of the average correlations of the features.

Table 2.7 illustrates the ratios between the mean of the average correlations of the newly added features and the mean of the average correlations of the existing ones. We notice that for the second experiment this ratio is higher, thus indicating that when only one feature is added, the correlation between the new information and the existing one is stronger. This could be yet another reason for which the results obtained in the second experiment are more accurate than those obtained by the first.

Experiment	Mean correlation of new features Mean correlation of old features
First experiment	0.97
Second experiment	1.17

Table 2.7: Features' correlations.

2.3.3.2 DHCGE algorithm

For testing the *DHCGE* algorithm, we have conducted two experiments. In each one we started from a different number of initial features and then we added the rest of the attributes (up to seven, which is the total number of attributes).

Both experiments are subject to the process we describe next. We start with m initial features and we calculate the optimum number of clusters nc for each case, as presented in Subsection 2.2.2. Regarding the $dMin$ threshold (Subsection 2.2.2), we use several different values and for each such value the following steps are performed. The traditional hierarchical agglomerative clustering method is applied on the data sets containing the initial m features and this algorithm stops when the number of clusters reached is nc. The partition obtained so far is denoted by $\mathcal{K}$. Further, the data set is extended with s features: $(m+1), \ldots, (m+s)$. On this extended data set, we apply the *DHCGE* algorithm, which starts from the previously obtained partition $\mathcal{K}$. To compare the results obtained by our algorithm, the traditional *HACA* is also run (from scratch) on the $m + s$-dimensional data set, until the number of obtained partitions is equal to nc. The comparison is made using the evaluation measures presented in Subsection 2.3.2 and the number of iterations performed by each algorithm.

2.3.3.2.1 First experiment

In this experiment, the initial data set contains $m = 5$ features and the remaining $s = 2$ features are added subsequently. Table 2.3.3.2.1 shows, for each considered value of the threshold $dMin$, the computed optimum number of clusters nc, the number of performed iterations and the three considered evaluation measures, for both *DHCGE* (partition denoted by $\mathcal{K}_{DHCGE}$) and the *HACA* (partition denoted by $\mathcal{K}_{HACA}$) algorithms. We mention that *HACA* is applied from scratch on the new data set containing all the $m + s = 7$ attributes. For each value of $dMin$, the reported results for *Z-score* are averaged over 10 repeated experiments.

Table 2.3.3.2.1 shows that the number of iterations needed by the *DHCGE* algorithm is smaller than the one needed by *HACA*, in all four cases. We can also notice a reduction in the running time of our dynamic algorithm in all the cases. The Moreover, the three considered evaluation measures also prove that, generally, the obtained partitions are better for the *DHCGE*: *IntraD* is smaller in three out of the four cases and *Dist* is smaller in all four cases, this indicating a more accurate clustering; the *Z-score* indicates that in three

No.	No.of iterations	IntraD	Dist	Z-score	Time (s)
1	$dMin = 3.23$ $nc = 63$				
$\mathcal{K}_{HACA}$	551	471.6626	82162.4895	2.3710	28
$\mathcal{K}_{DHCGE}$	444	467.4943	55584.5283	2.7940	21
2	$dMin = 3.26$ $nc = 61$				
$\mathcal{K}_{HACA}$	553	475.0851	82376.9315	2.1000	29
$\mathcal{K}_{DHCGE}$	455	482.3278	69249.0666	3.2850	22
3	$dMin = 3.47$ $nc = 44$				
$\mathcal{K}_{HACA}$	570	529.1520	103157.8724	4.9500	29
$\mathcal{K}_{DHCGE}$	469	522.3940	86874.1104	3.9670	21
4	$dMin = 3.51$ $nc = 42$				
$\mathcal{K}_{HACA}$	572	558.0713	154399.2198	4.8390	29
$\mathcal{K}_{DHCGE}$	474	532.6680	94543.0836	5.1090	23

Table 2.8: Results obtained for the first experiment.

tests, the partitions obtained by *DHCGE* are more biologically relevant than the ones the traditional *HACA* achieves. We can therefore conclude that for this experiment, the quality of the partitions obtained adaptively by *DHCGE* is improved with regard to *HACA* and the result is reached after fewer iterations.

2.3.3.2.2 Second experiment

In this second experiment, we considered the initial data set as having $m = 6$ features and the remaining $s = 1$ feature is added subsequently. Table 2.9 shows, for each value of the threshold $dMin$, the computed optimum number of clusters nc, the number of performed iterations and the three considered evaluation measures, for both algorithms, *DHCGE* and *HACA*, the latter being applied from scratch on the new data set containing all the $m+s = 7$ attributes. For each value of $dMin$, the reported results for *Z-score* are averaged over ten repeated experiments.

As for the previous experiment, the number of iterations performed by *DHCGE* is less than the one *HACA* needs, in all four cases, as shown in Table 2.9. Still, the other evaluation measures show that the partitions obtained by our algorithm are not as good compared to those obained by *HACA*, as in the first experiment. *IntraD* indicates that *HACA* obtains better partitions in three out of four cases, but from Table 2.9 we notice that the difference in these cases is minor. *Dist*, however indicates more accurate clustering for *DHCGE*, in three out of four cases. Concerning the *Z-score*, we note that in two of the tests the clusters obtained by *DHCGE* have a higher biological relevance, while in the other two, those achieved by *HACA* are more relevant. We remark that for this second experiment, *DHCGE* clearly

No.	No.of iterations	IntraD	Dist	Z-score	Time (s)
1	$dMin = 4.38$ $nc = 63$				
$\mathcal{K}_{HACA}$	551	471.6626	82162.4895	2.3490	30
$\mathcal{K}_{DHCGE}$	369	472.7752	66836.8512	1.8320	18
2	$dMin = 4.40$ $nc = 62$				
$\mathcal{K}_{HACA}$	552	473.7939	82325.5096	2.4740	28
$\mathcal{K}_{DHCGE}$	369	474.5243	66908.912	2.1270	17
3	$dMin = 4.60$ $nc = 45$				
$\mathcal{K}_{HACA}$	569	527.4638	103154.4945	4.7480	29
$\mathcal{K}_{DHCGE}$	367	536.0378	110093.9625	5.2700	16
4	$dMin = 4.66$ $nc = 42$				
$\mathcal{K}_{HACA}$	572	558.0713	154399.2198	4.7960	30
$\mathcal{K}_{DHCGE}$	369	540.6549	109955.9118	5.5990	16

Table 2.9: Results obtained for the second experiment.

outperforms *HACA* (with regard to all three evaluation measures) in the last case, while for the other cases each measure indicates differently: in the first two cases *IntraD* and *Z-score* imply that *HACA* is more accurate, but *Dist* implies otherwise; in the third case, *IntraD* and *Dist* indicate that *HACA* obtains better results, while *Z-score* shows that the partitions obtained by our algorithm *DHCGE* are more biologically significant.

2.3.3.3 FDCGE algorithm

We have conducted three experiments for evaluating the performance of the *FDCGE* algorithm. In each of these, we start with an initial number of features m and subsequently add the new s features (considering that the total number of features for our experiments is 7).

For each experiment, we considered different values for the threshold $distMin$ and for each such value, the following steps are performed:

1. Using the heuristic presented in Subsection 2.2.2, we identified the initial centroids in the *fuzzy c-means* process and we computed the optimum number of clusters nc accordingly.

2. The *fuzzy c-means* algorithm is applied on the data set containing the genes characterized by the m initial features, starting from the centroids determined in the previous step. Following this process, a partition is obtained, which will be denoted by $\mathcal{K}$. We mention that if at a certain point during the running of the algorithm one cluster becomes empty, it is removed and therefore the number of clusters is decreased.

3. The new s features are added so that the genes are now instances in an $(m + s)$-dimensional space. Our fuzzy dynamic algorithm is applied on the new, feature enriched data set, using the previously obtained partition $\mathcal{K}$. The obtained result is denoted by $\mathcal{K}_{FDCGE}$.

4. The *fuzzy c-means* is applied from scratch on the data set containing the $(m + s)$-dimensional instances and a new partition is obtained, denoted by $\mathcal{K}'$. This result will be compared to the one obtained by our dynamic fuzzy algorithm, in order to analyze its performance with regard to traditional *fuzzy c-means*.

2.3.3.3.1 First experiment

In the first experiment we started with $m = 4$ features representing the gene values for the first four moments in time and we subsequently added the last $s = 3$. For each considered value of the threshold $distMin$, Table 2.10 shows the optimum initial number of clusters nc retrieved by the heuristic we used, as well as the values obtained for the evaluation measures presented in Subsection 2.3.2, both for our *FDCGE* algorithm and for the *fuzzy c-means* algorithm applied from scratch on the complete data set, with all the $m + s = 7$ features. For each value of $distMin$, the reported results for *Z-score* are averaged over ten repeated experiments.

No.	**No. of clusters**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	$distMin = 2.58$ $nc = 63$				
$\mathcal{K}'$	63	111.4800	0.0917	24927.8000	4.3400
$\mathcal{K}_{FDCGE}$	63	93.3800	0.0979	2470.8600	4.7450
2	$distMin = 2.59$ $nc = 62$				
$\mathcal{K}'$	62	111.4600	0.0917	25097.7800	4.1120
$\mathcal{K}_{FDCGE}$	62	92.7400	0.0979	2477.5400	6.0250
3	$distMin = 2.69$ $nc = 53$				
$\mathcal{K}'$	53	113.2700	0.0917	27486.7300	4.6330
$\mathcal{K}_{FDCGE}$	53	94.5700	0.1101	2889.7400	5.3410
4	$distMin = 2.8$ $nc = 45$				
$\mathcal{K}'$	45	110.4200	0.0935	26714.19	5.0990
$\mathcal{K}_{FDCGE}$	45	97.22	0.1027	3698.15	5.8830
5	$distMin = 2.82$ $nc = 42$				
$\mathcal{K}'$	42	111.6700	0.0861	30384.7600	5.0210
$\mathcal{K}_{FDCGE}$	42	97.8700	0.1141	4216.2700	5.4790

Table 2.10: Results for the first experiment.

We remark that for this experiment the evaluations measures indicate that in all cases, *FDCGE* is more performant than the *fuzzy c-means* applied from the beginning. The intra-cluster distance *IntraD* is smaller, while the *Dunn index* and *Z-score* are higher, in all 5 runs. Considerable improvement can be noticed in the case of the overall distance of a partition *Dist*, as its values computed for the clustering obtained by *FDCGE* are significantly lower than those reported for *fuzzy c-means*, this indicating compact clusters and better partitions.

2.3.3.3.2 Second experiment

In this experiment the data set contains genes represented by $m = 5$ features and the values corresponding to the last $s = 2$ moments in time are added afterwards. Table 2.11 illustrates the results obtained for this experiment, showing the same information as for the previous case: for each value of the threshold $distMin$ we retrieved the initial number of clusters, as well as the values of the evaluation measures (as before, the *Z-score* was averaged over 10 experiments for each value of $distMin$).

By looking at the columns corresponding to the *Dunn index* and *Dist*, we observe that according to these measures, *FDCGE* always outperforms *fuzzy c-means*, applied from scratch on the entire 7-feature data set. Regarding the *IntraD* we notice that except one test, all the values of this measure are smaller for our algorithm, this indicating better partitions. However, the *Z-score* is not as good as the other measures, as there are 3 cases in which the partitions obtained by *fuzzy c-means* are better, but nevertheless in the other 4 cases, the clustering result obtained by *FDCGE* is more biologically relevant. An important remark is that considering all the 7 tests made for this experiment, the average values of the evaluation measures indicate that our algorithm is more efficient than the traditional *fuzzy c-means*. These values are shown in Table 2.13, on the second row.

2.3.3.3.3 Third experiment

In this last experiment the initial set of features is composed of the genes' expression levels for the first $m = 6$ moments in time and the last $s = 1$ feature is further added. The obtained results are depicted in Table 2.12, which depicts the values of the evaluation measures and the heuristically obtained number of clusters for 4 considered tests. According to *IntraD* and *Dist*, *FDCGE* invariably outperforms *fuzzy c-means*, as the values for the partitions obtained by our dynamic approach are smaller, thus indicating better clustering. The *Dunn index* and *Z-score* are better for our algorithm only in 2 out of 4 tests, therefore suggesting that from the perspective of these two measures our algorithm and *fuzzy c-means* perform approximately equally. However, when computing an average of the values for each of these measures and each algorithm (see Table 2.13), we observe that 3 out of the 4 evaluation measures (among which the one indicating the biological significance of the clustering) prove that the *FDCGE* algorithm is more accurate.

We have to mention that the running time of *fuzzy c-means* and *FDCGE* algorithms is

No.	**No. of clusters**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	$distMin = 3.231$ $nc = 63$				
$\mathcal{K}'$	63	111.1700	0.0917	24964.4400	4.2890
$\mathcal{K}_{FDCGE}$	63	178.9000	0.1247	3027.6300	4.6560
2	$distMin = 3.26$ $nc = 61$				
$\mathcal{K}'$	61	111.6800	0. 917	24826.3700	4.4210
$\mathcal{K}_{FDCGE}$	61	92.4100	0.1238	2793.7600	4.3230
3	$distMin = 3.44$ $nc = 47$				
$\mathcal{K}'$	47	111.58	0.0935	27321.3000	4.3650
$\mathcal{K}_{FDCGE}$	47	98.2000	0.1195	3646.8700	6.4040
4	$distMin = 3.45$ $nc = 46$				
$\mathcal{K}'$	46	110.8700	0.0923	27659.66	5.6780
$\mathcal{K}_{FDCGE}$	46	97.8300	0.0994	3875.2900	5.4110
5	$distMin = 3.47$ $nc = 44$				
$\mathcal{K}'$	44	110.8700	0.0923	28898.2900	5.1910
$\mathcal{K}_{FDCGE}$	44	97.8300	0.0994	3690.66	4.4760
6	$distMin = 3.51$ $nc = 42$				
$\mathcal{K}'$	42	110.1900	0.0703	29405.2700	4.3430
$\mathcal{K}_{FDCGE}$	42	99.7300	0.1198	4492.8500	7.0180

Table 2.11: Results for the second experiment.

very small, around 3 seconds, therefore the dynamic algorithm performs similarly with the *fuzzy c-means* from scratch in terms of execution time.

2.4 Discussion

We provide an analysis of the dynamic clustering methods that we proposed and comparisons to related work from the literature, based on the experimental evaluation from the previous section.

2.4.1 Comparative analysis of our algorithms

To our knowledge, except for the dynamic clustering algorithms that we have proposed in [Bocicor et al., 2014, Sirbu and Bocicor, 2013, Sirbu et al., 2014a], there are no other approaches that deal with gene expression data sets for which new features (values for expression levels of genes at new time points) are added dynamically. For this reason, we will provide a comparison between the three models we proposed.

We cannot provide a thorough comparison of our results to existing ones, as except

No.	**No. of clusters**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	$distMin = 4.38 \quad nc = 63$				
$\mathcal{K}'$	63	111.6300	0.0917	24749.2000	4.1900
$\mathcal{K}_{FDCGE}$	62	95.1300	0.0490	2413.7900	3.3480
2	$distMin = 4.401 \quad nc = 62$				
$\mathcal{K}'$	62	111.4656	0.0917	25104.1767	4.1990
$\mathcal{K}_{FDCGE}$	62	95.7149	0.0663	2803.7793	4.2850
3	$distMin = 4.5 \quad nc = 53$				
$\mathcal{K}'$	53	112.98	0.0917	27429.8700	3.5050
$\mathcal{K}_{FDCGE}$	53	97.1700	0.1217	2705.8900	5.1240
4	$distMin = 4.6 \quad nc = 45$				
$\mathcal{K}'$	45	111.95	0.0935	28552.5400	5.8020
$\mathcal{K}_{FDCGE}$	45	97.5700	0.1179	3270.6300	5.2470

Table 2.12: Results for the third experiment.

Experiment	**Algorithm**	**IntraD**	**Dunn**	**Dist**	**Z-score**
First	$\mathcal{K}'$	111.6600	0.0909	26922.2520	4.6410
	$\mathcal{K}_{FDCGE}$	95.1560	0.1045	3150.5120	5.4946
Second	$\mathcal{K}'$	111.1685	0.0892	26832.7410	4.686
	$\mathcal{K}_{FDCGE}$	108.2114	0.1194	3471.4829	5.0100
Third	$\mathcal{K}'$	112.0064	0.0921	26458.9470	4.4240
	$\mathcal{K}_{FDCGE}$	100.3339	0.0887	8373.6217	4.5010

Table 2.13: Average values of the considered evaluation measures after the three experiments.

for the two techniques that we mentioned above, there are none that approach the case in which new features are added to existing genes. However, we can asses the performance of *FDCGE* with regard to biological relevance, as compared to other incremental clustering algorithms, in which new objects are dynamically added to the initial data set (as opposed to new features for existing instances). The algorithms presented in the papers [Sarmah and Bhattacharyya, 2010] and [Das et al., 2009a] report *z-scores* of 7.39 (GenClus algorithm [Sarmah and Bhattacharyya, 2010]) and 7.07 (incDGC algorithm [Das et al., 2009a]), while the best *Z-score* obtained by our algorithm is 7.01, which is satisfactory. Still, we intend to experimentally evaluate our method for different values of $distMin$, which may lead to higher values for the *Z-score*.

Regarding the running time of the dynamic algorithms, we mention that it is influenced by the time required for the cores' computation. This step is performed only once at the begining of the algorithm. The improvement in running time of the dynamic approaches is significant

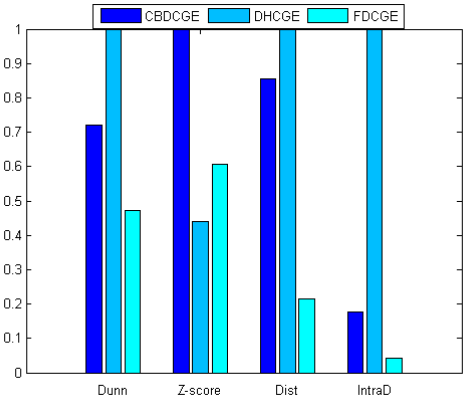


Figure 2.7: Comparison of *CBDCGE*, *DHCGE*, *FDCGE*, starting with $m = 5$ features

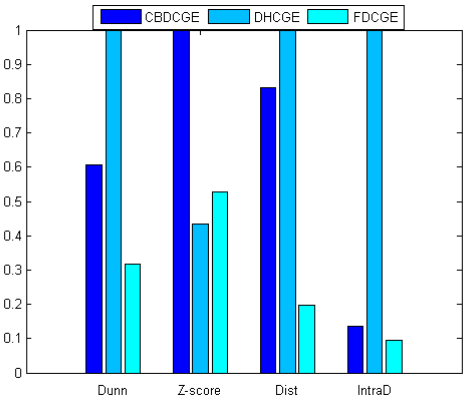


Figure 2.8: Comparison of *CBDCGE*, *DHCGE*, *FDCGE*, starting with $m = 6$ features

when the number of iterations performed by the classical clustering algorithms (non-dynamic) is large enough. In this case, an important reduction in the number of iterations brought by the dynamic approaches may lead to an important improvement in the execution time. This may be observed for the *DHCGE* algorithm. For the *CBDCGE* and *FDCGE* algorithms the improvement is not visible since the number of iterations performed is small.

2.4.2 Comparison to related work

Since there are no other algorithms that approach the problem of the dynamic clustering of gene expression when new expression levels are added to the existing genes, we cannot provide a thorough comparison of our results to other. However, we note that the biological relevance of the partitions obtained using *CBDCGE*, quantified in the *z-score*, is significant.

Although our algorithm was designed with the purpose of providing an adaptive clustering technique for dynamic gene expression data sets, instead of a novel clustering method, we remark that, in terms of *z-score*, it outperforms other existing incremental clustering algorithms proposed for gene expression data sets, which are changing, in the sense that they are enriched with new instances [Sarmah and Bhattacharyya, 2010, Das et al., 2009a]. Figure 2.9 illustrates the values of the *z-scores* reported by the algorithms *GenClus* [Sarmah and

Bhattacharyya, 2010] and *incDGC* [Das et al., 2009a] for the same data set that was used in our experiments. The same figure depicts the averaged *z-score* over all the experimental evaluations of our algorithm *CBDCGE*.

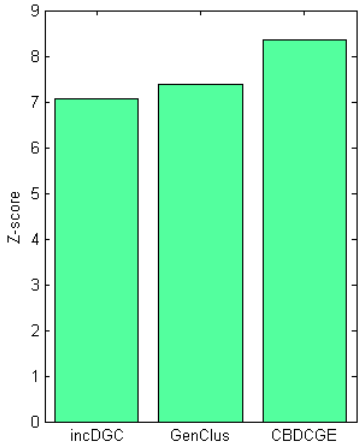


Figure 2.9: Comparative Z-scores.

Compared to applying the traditional *k-means* algorithm from the beginning over and over each time new values of gene expression become available, our *dynamic core based clustering* algorithm generally obtained better clustering accuracies, in terms of the considered evaluation measures. Moreover our algorithm needs a smaller number of iterations to achieve the solution. This can be seen in Tables 2.1 and 2.2. Still, to see how the number of iterations is being modified even for larger gene expression data sets, we experimented on the data set presented in Subsection 2.3.1, without applying all the pre-processing steps. More specifically, from the 6400 existing genes, we only eliminated the ones that have missing expression levels for certain time points, thus remaining 6276 instances. For this data set, we applied the traditional *k-means* and our *CBDCGE* algorithms, similar to what we have presented in the two experiments in Subsection 2.3.3.1. In first experiment we begun with 5 features and then added the last 2 and in second one we started with 6 features and subsequently added the last one. Table 2.14 illustrates the percent by which the number of iterations is reduced, showing two considered cases for each experiment. From this table we can conclude that the number of iterations used by our *CBDCGE* algorithm is reduced in all four cases, once even with almost 50%.

A lot of clustering algorithms that have been used to cluster gene expression data need the number of clusters as a priori information. Among these, we mention *k-means* [Bagirov and Mardaneh, 2006], self organizing maps or genetic algorithms. Compared to these techniques, our approach has the advantage that the number of clusters is not requires, but is computed according to the heuristic described in Subsection 2.2.2. We remark, however, that there exist other algorithms in the literature that do not need this number a priori: *GenClus* [Sarmah and Bhattacharyya, 2010] and *incDGC* [Das et al., 2009a].

Experiment	Algorithm	No. of clusters	No. of iterations	Iteration reduction (%)
Experiment 3	$\mathcal{K}_{CBDCGE}$	176	36	**2.77%**
	$\mathcal{K}'$	175	35	
	$\mathcal{K}_{CBDCGE}$	185	59	**47.45%**
	$\mathcal{K}'$	184	31	
Experiment 4	$\mathcal{K}_{CBDCGE}$	118	51	**7.84%**
	$\mathcal{K}'$	118	47	
	$\mathcal{K}_{CBDCGE}$	128	54	**12.96%**
	$\mathcal{K}'$	127	47	

Table 2.14: Reduction of the number of iterations.

2.4.3 Adaptive association rule mining of gene expression data

Starting from the idea of dynamic clustering of gene expression, when expression levels are added to the existing genes, we explored another dynamic process that takes place in the same context of feature-set extension: association rule mining.

Association rule mining [Calders et al., 2014] implies idemtifying attribute value conditions which appear often together in data [Tan et al., 2005, Vimieiro and Moscato, 2014]. Considering a set of instances characterized by a list of features, ordinal association rules [Campan et al., 2006a] are a specific type of association rules which determine ordinal relationships between the features values that are valid for a specified percentage of instances. Since the features can have various domains, ordinal association rules are not expressive enough [Czibula et al., 2012]. Therefore, relational association rules were introduced in [Serban et al., 2006] for capturing capture different types of relationships between features. A relational association rule is considered to be *interesting* if its support and confidence exceeds a given threshold. The *DRAR* method (*Discovery of Relational Association Rules*) was introduced for mining interesting relational association rules within data sets [Serban et al., 2006].

In this section we present the problem of association rule mining in the context of gene expression data and briefly describe our adaptive model for mining relational association rules.

2.4.3.1 Relational association rules on gene expression

In the following we give an example that illustrates how can the *DRAR* algorithm [Serban et al., 2006] be applied on the gene expression data set presented in Subsection 2.3.1, in order to discover interesting association rules.

The first 20 of 65 instances of the data set considered in our example is given in Table 2.15. Scaling to [0,1] was performed, using the *Min-Max* normalization method. As all

attributes in the experiment have float values, three possible binary relations between float valued attributes were defined: $=$, $<$, $>$.

a_1	a_2	a_3	a_4	a_5	a_6	a_7
-0.532	0.028	-0.333	-0.102	1.172	2.087	0.992
-0.538	-0.316	0.39	0.781	0.389	1.649	1.498
-0.358	0.138	1.14	0.813	0.121	2.018	1.495
0.126	0.105	0.086	0.557	0.784	1.838	1.613
0.029	-0.203	-0.16	0.719	0.852	2.062	1.082
-0.044	-0.129	0.14	0.817	0.66	1.957	1.086
-0.105	-0.266	0.326	0.769	0.655	1.725	0.785
-0.468	-0.837	0.186	1.354	0.809	1.887	1.674
0.005	-0.24	0.149	0.604	0.746	1.878	1.383
0.009	-0.079	-0.047	0.58	1.025	1.983	0.943
-0.01	-0.057	-0.137	0.584	0.841	2.196	1.344
-0.269	-0.676	0.034	0.9	0.568	2.02	1.42
-0.021	-0.222	-0.086	0.488	0.801	1.618	0.827
-0.269	-0.429	0.908	0.662	0.55	1.629	1.183
0.031	0.027	0.597	0.792	0.648	1.799	2.042
0.009	0.013	-0.2	0.629	1.576	1.802	1.906
0.148	0.408	0.116	0.727	0.717	2.063	1.822
-0.034	0.231	0.409	0.709	0.484	2.219	1.877
-0.758	0.087	0.558	0.382	0.451	1.758	1.592
-0.098	0.173	0.115	0.504	0.808	1.897	1.219
0.075	-0.057	0.296	0.524	0.157	1.731	1.529
0.281	0.079	0.039	0.417	0.608	1.921	1.381

Table 2.15: The first 20 of 65 instances of the gene expression data set

In Table 2.16 we depict all the interesting association rules and their confidence, which were discovered by applying *DRAR* algorithm, considering the minimum support as 1 and the minimum confidence as 0.6. In Table 2.17 are presented only the maximal interesting association rules. The attributes characterizing the instances are denoted by a_1, a_2,...,a_7.

Each line from Table 2.17 expresses a relational association rule of a certain length, which was discovered in the data set indicated in Table 2.15 with a specified confidence. For example, the first line in Table 2.17 refers to the relational association rule $a_1 > a_4$ of length **2** (i.e the rule contains two attributes) having a confidence of **0.83**. That is, the value of the attribute a_1 is greater than the value of the attribute a_4 in 83% of instances within the data set (i.e in **56** instances).

The results above show that interesting relational association rules can be found in the gene expression dataset. Further analysis of the discovered relational association rules by an

Length	Rule	Confidence
2	$a_1 > a_3$	0.86
2	$a_1 > a_4$	0.83
2	$a_1 > a_5$	0.8
2	$a_1 > a_6$	0.84
2	$a_1 > a_7$	0.63
2	$a_2 > a_3$	0.86
2	$a_2 > a_4$	0.8
2	$a_2 < a_5$	0.86
2	$a_2 > a_6$	0.83
2	$a_2 > a_7$	0.64
2	$a_3 <= a_4$	0.72
2	$a_3 <= a_5$	0.64
2	$a_3 <= a_7$	0.78
2	$a_4 > a_6$	0.66
2	$a_4 <= a_7$	0.61
2	$a_5 > a_6$	0.61
2	$a_5 <= a_7$	0.66
2	$a_6 <= a_7$	0.72
3	$a_1 > a_3 <= a_4$	0.69
3	$a_1 > a_3 <= a_5$	0.63
3	$a_1 > a_3 <= a_7$	0.69
3	$a_1 > a_6 <= a_7$	0.63
3	$a_2 > a_3 <= a_4$	0.66
3	$a_2 > a_3 <= a_5$	0.61
3	$a_2 > a_3 <= a_7$	0.69
3	$a_2 > a_6 <= a_7$	0.66

Table 2.16: Interesting relational association rules

expert in the field, may provide relevant information regarding genes' functions.

2.4.3.2 Adaptive association rule mining

The method *DRAR* for relational association rule mining starts with a set of instances, characterized by different features and discovers interesting relational association rules within a data set. But there are applications like dynamic gene expression data sets, where the feature set evolves. In this situation, the relational association rules could be obtained by applying the mining algorithm from scratch over and over again, when new features are available, but this can be inefficient.

In [Czibula et al., 2015a] we proposed a new adaptive association rule method, named

Length	Rule	Confidence
2	$a_1 > a_4$	0.83
2	$a_1 > a_5$	0.8
2	$a_1 > a_7$	0.63
2	$a_2 > a_4$	0.8
2	$a_2 > a_5$	0.86
2	$a_2 > a_7$	0.64
2	$a_4 > a_6$	0.66
2	$a_4 <= a_7$	0.61
2	$a_5 > a_6$	0.61
2	$a_5 <= a_7$	0.66
3	$a_1 > a_3 <= a_4$	0.69
3	$a_1 > a_3 <= a_5$	0.63
3	$a_1 > a_3 <= a_7$	0.69
3	$a_1 > a_6 <= a_7$	0.63
3	$a_2 > a_3 <= a_4$	0.66
3	$a_2 > a_3 <= a_5$	0.61
3	$a_2 > a_3 <= a_7$	0.69
3	$a_2 > a_6 <= a_7$	0.66

Table 2.17: Maximal interesting relational association rules

Adaptive Relational Association Rule Mining (ARARM), that is capable to efficiently mine relational association rules within the set of instance, when the feature set is extended. The *ARARM* method starts from the set of interesting rules that was established by applying *DRAR* before the feature set changed and adapts it considering newly added features. The result is obtained faster than applying *DRAR* on the gene data set after feature extension.

In the following, we introduce the adaptive relational association rule mining approach, as well as an algorithm called *ARARM* (*Adaptive Relational Association Rule Mining*) that is capable to efficiently mine relational association rules within a data set, when the feature set increases.

Let us consider a data set $R = \{r_1, r_2, \ldots, r_n\}$ consisting of n-dimensional *instances* (objects). An instance is described by a list of m features, $(a_1, \ldots, a_m)$ and is represented by an m-dimensional vector $r_i = (r_{i1}, \ldots, r_{im})$. Between the features values different types of relations can be defined. By $\mathcal{R}el$ we denote the set of all relationships which can be defined between the features values. As presented in Section 2.4.3.1, interesting relational association rules that are able to express relations (from the set $\mathcal{R}el$) between the features values may be discovered using the *DRAR* method [Serban et al., 2006].

The set of features is subsequently increased with s features, denoted by $m+1, \ldots, m+s$, the objects vectors becoming $r_i^{ext} = (r_{i1}, \ldots, r_{im}, r_{i,m+1}, \ldots r_{i,m+s})$, $1 \leq i \leq n$. The set of

extended instances is denoted by $R^{ext} = \{r_1^{ext}, r_2^{ext}, \ldots, r_n^{ext}\}$.

Considering certain minimum support and confidence thresholds (denoted by s_{min} and c_{min}), we want to analyze the problem of mining interesting relational association rules within the data set R^{ext}, i.e. after object extension, and starting from the set of rules discovered in the data R before the feature set extension. We aim at obtaining a better time performance compared to process of mining from the beginning. By $\mathcal{RAR}$ we express the set of interesting relational association rules from the data set R, and by $\mathcal{RAR}^{ext}$ the set of interesting relational association rules from the extended data set R^{ext} [Czibula et al., 2015b].

Certainly, the newly arrived features can generate new relational association rules. The new set of rules $\mathcal{RAR}^{ext}$ could be of course obtained by applying the *DRAR* method from scratch on the extended instance, aiming to replace this process with a less expensive one, while preserving the completeness of the rules generation process. More specifically, we will propose a method called *ARARM* (*Adaptive Relational Association Rule Mining*), which starts from the set $\mathcal{RAR}$ of rules mined from the data set before feature extension and adapts it (considering the newly added features) in order to obtain a set of interesting relational association rules from the set of extended objects R^{ext}. Definitely, through the adaptive process, we want to preserve the completeness of the *DRAR* method.

Let us denote by l the maximum length of the rules from the set RAR and by RAR_k $(1 \leq k \leq l)$ the set of interesting relational association rules of length k mined in the data set R (before the feature set extension), where $\mathcal{RAR} = \bigcup_{k=1}^{l} RAR_k$.

In the following we will briefly present the idea of discovering the set $\mathcal{RAR}^{ext}$ through adapting the set $\mathcal{RAR}$ of rules mined in the data set R before feature extension.

The *ARARM* algorithm identifies the interesting relational association rules using an iterative process which generates length-level rules, then verifies the candidates in order to comply the minimum support and confidence. *ARARM* performs multiple iterations over R^{ext}. During the first iteration, the algorithm computes the interesting 2-length rules. All the subsequent iterations over the data are performed of two phases. The k-length $(k \geq 2)$ rules from R^{ext} will certainly contain the k-length rules from $\mathcal{RAR}$ (the interesting rules discovered in the data set before extension) - if such rules exist. But, there is another possibility to obtain a k-length rule in the extended data set, through generating a candidate rule through joining two $k-1$-length rules from $\mathcal{RAR}^{ext}$ (generated at the previous iteration). In the second phase the support and confidence of the generated candidates are calculated. Only the interesting rules are kept and will be considered in the next step of the algorithm The process ends when no news interesting rules are found [Czibula et al., 2012].

At a certain iteration performed by the *ARARM* algorithm, the candidate generation process is essentially the same as the candidates generation process of the *DRAR* method [Campan et al., 2006b]. More specifically, for joining two $k-1$ length rules in order to obtain a k-length candidate rule, there are more possibilities (see [Czibula et al., 2015a]). Similarly to the proof presented in [Campan et al., 2006b] it may be proven that the candidate generation

process ensures the correctness and completeness of the *ARARM* algorithm.

Regarding the binary relations that may be defined between the attributes domains, we mention that we do not assume particular properties (such that the transitivity property), both *DRAR* and *ARARM* are working with general relationships between the attributes domains.

The most important step in the *ARARM* algorithm is the candidate generation process, which is also computationally expensive. This function has as a parameter a set *Rules* of k-length relational association rules and returns a set of $k+1$ length relational association rules generated from *Rules* through the join operations. The main idea of the candidate generation process is the following. All distinct combinations of two rules (r_1, r_2) from the set *Rules* are considered. If r_1 and r_2 match for join, the rule r_{join} obtained by joining r_1 and r_2 is constructed and is added to the resulting set of rules. Obviously, since r_1 and r_2 are k-length rules, r_{join} will be a $k+1$-length rule.

It has to be stated that running the *ARARM* method with $m = 0$ provides the set of interesting relational association rules mined in the data set of s-dimensional entities. Thus, this running is equivalent with applying the *DRAR* method on the data set of s-dimensional entities. Two rules are joined during the adaptive candidate generation process only if at least one rule has at least an attribute from the additional attributes, which are present only in the enlarged attribute set. Obviously, it is not necessary to join rules which contain only attributes from the original attribute set, since the joint rules are already known (these rule are in the set $\mathcal{RAR}$ of relational rules mined in the set of m-dimensional entities). This way, when generating the k-length relational association rules from the extended data set of $m+s$ dimensional entities, the candidate generation process is not applied on the set of $k-1$ length rules from the set off interesting rules extracted from the data set of m-dimensional entities. Thus, unlike in the *DRAR* algorithm applied from scratch on the $m+s$ dimensional entities, the join operations between the $k-1$ length rules from the data set of entities before the attribute set extension are skipped.

The time savings in the *ARARM* running time come from the time reduction of the candidate generation process as well as from an reduced number of support and confidence computations. Obviously, the step of computing the support and confidence for the rules from the set $\mathcal{RAR}$ is skipped, since for these rules we already have their support and confidence. Certainly, the reduction in the execution time of the *ARARM* increases with the increase of the set $\mathcal{RAR}$. This usually happens when decreasing the number s of added attributes. A smaller number of added attribute means a larger set of already known rules and this implies a smaller number of rules generated by the function that generates candidates, as well as less time for support and confidence computations.

2.4.3.3 Experimental evaluation

In the following we present a set of experiments ment to evaluate the effectiveness of *ARARM* of the gene expression data set presented in 2.3.1. Like in the dynamic clustering experiments, we are initially considering m attributes (measurements of gene expression levels) and afterwards we extend the set of features with s attributes. We consider in the experiments different values for the confidence threshold (c_{min}) and different type of relational rukes (maximal rules vs. all rules). The threshold s_{min} is set to 1. For each experiment, the set of interesting relational association rules on the $m+s$ dimensional instances are obtained in two ways:

1. by applying the *DRAR* method from scratch on the data set after the feature set extension (containing all $m+s$ features).
2. by adapting (through the *ARARM* algorithm) the rules obtained on the data set before the feature set extension (containing m features).

We mention that the same set of interesting relational association rules is discovered in data, independent to the way the rules were generated (1. or 2.), but, obviously, we are expecting the running time of the adaptive algorithm to be lower than the running time of the *DRAR* method applied from scratch. We also have to mention that two relations between the features values are considered in the relational association rule mining process: $\mathcal{R}el = \{\leq, >\}$. The experiments are performed using the *ARARM* API, introduced in [Czibula et al., 2015a].

Table 2.18 illustrates the performance of the *ARARM* method, for each of the performed experiment. The running times are given in milliseconds and for each experiment, the lower running time value is marked with bold. It can be observed that the time needed to obtain the rules adaptively is less than the time needed to obtain the rules from scratch, indicating that our approach is more efficient for identifying association relational rules when the feature set is extended than applying the mining process from the beginning.

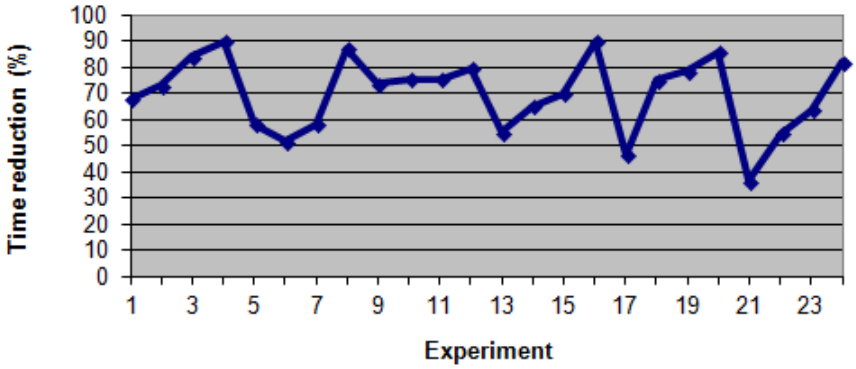


Figure 2.10: Time reduction using ARARM

For concluding about the efficiency of the adaptive method against the non-adaptive one, we depict in Figure 2.10, the reduction (in percentage) of the running time of *ARARM* with

Experiment	c_{min}	**No. of attributes** (m)	**No. of added attributes** (s)	**Type of rules**	**No. of rules**	**Time from scratch (ms)**	**Time adaptive (ms)**
1	0.3	3	4	Maximal	53	118	**38**
2	0.3	4	3	Maximal	53	118	**32**
3	0.3	5	2	Maximal	53	118	**19**
4	0.3	6	1	Maximal	53	118	**12**
5	0.3	3	4	All	114	31	**33**
6	0.3	4	3	All	114	31	**15**
7	0.3	5	2	All	114	31	**13**
8	0.3	6	1	All	114	31	**4**
9	0.4	3	4	Maximal	33	49	**13**
10	0.4	4	3	Maximal	33	49	**12**
11	0.4	5	2	Maximal	33	49	**12**
12	0.4	6	1	Maximal	33	49	**10**
13	0.4	3	4	All	64	20	**9**
14	0.4	4	3	All	64	20	**7**
15	0.4	5	2	All	64	20	**6**
16	0.4	6	1	All	34	20	**2**
17	0.5	3	4	Maximal	21	28	**15**
18	0.5	4	3	Maximal	21	28	**7**
19	0.5	5	2	Maximal	21	28	**6**
20	0.5	6	1	Maximal	21	28	**4**
21	0.5	3	4	All	38	11	**7**
22	0.5	4	3	All	38	11	**5**
23	0.5	5	2	All	38	11	**4**
24	0.5	6	1	All	38	11	**2**

Table 2.18: Results for the gene expression data set for $s_{min} = 1$

respect to the running time of *DRAR*.

As further work we intend to extend the proposed approach and apply it to supervised and unsupervised classification problems.

2.5 Conclusions and further work

This chapter presented three models for dynamic clustering of gene expression data in the context where expression levels for new time points are added to the existing genes. The algorithms are capable of adapting the previously obtained partitions when new features (measurements of gene expression levels) are added to the data set, without performing

re-clustering from scratch. In the same context of dynamic gene expression data we also introduced a method for adaptive relational association rule mining that is able to adapt the previously obtained set of interesting rules when new gene expression levels are added to the existing genes, without applying the mining algorithm from scratch.

These algorithms were presented in the original papers [Bocicor et al., 2014, Sirbu and Bocicor, 2013, Sirbu et al., 2014a, Czibula et al., 2015a]. The experimental evaluation that was performed on a real-life gene expression data set show that, in most of the cases, the clustering is reached more effectively and is also more accurate by using our proposed methods than by using the *k-means*, *hierarchical agglomerative clustering*, respectively *fuzzy c-means* algorithm from scratch on the feature-extended data. Also, the relational association rules are achieved faster using our method than applying the mining algorithm from the beginning.

Still, there are some cases when is better to perform a full-repartitioning of the extended set of objects, instead of adapting the existing partition. Examples of such situation could be: adding of a large number of expression levels for new time points or adding attributes with high information gain or contradictory information against the old attributes.

Further work can be done in order to determine situations when is more appropriate to adapt (using *CBDCGE*, *DHCGE* or *FDCGE*) the partition of the feature-extended set instead of recomputing the partition from scratch using a classic clustering approach. We also plan to extend the experimental evaluation on other publicly available data sets and to investigate methods to automatically identify the distance threshold for the clusters (e.g. supervised learning).

Chapter 3

Pedestrian detection. Background.

In this chapter we are presenting the background knowledge related to the supervised classification problem approached in the thesis, the problem of *dynamic pedestrian recognition* with onboard cameras, which represents a very challenging task with great importance in real life.

In the literature there are two important approaches for object detection: using the classical three step architecture, which involves the identification of regions of interest in an image, feature extraction and classification (see Figure 3.1) or using deep learning, which uses special neural networks like convolutional neural networks (see Figure 3.2) in order to learn multiple levels of features and classify. Unlike the classical architecture, deep learning works on entire images and needs large datasets with complex scenarios for training. Since in the thesis we focus on the classification part, we decided to exploit the first architecture, which is more flexible and proved high performances for pedestrian detection.

Therefore, we begin by presenting the most important components in a pedestrian recognition system that follows the first architecture: the feature extraction and the classification components. Thus, in Section 3.1 we give an overview of the most commonly used features, then in Section 3.2 we briefly present the most popular classifiers used in pedestrian detection. Finally, in Section 3.3 we provide a short literature review in the field, with emphasis on the fusion of image modalities, for which we are going to introduce a dynamic approach in Chapter 4.

3.1 Feature extraction

Object representation plays a very important role in an object recognition system. In our particular case of pedestrian detection, choosing the most pertinent features for pedestrian characterization is a very challenging problem, as it is still uncertain how the human brain performs the recognition based on visual information [Parra Alonso et al., 2007]. An important step before feature computing is finding a region of interest (ROI), an area where the potential pedestrian could be found. According to [Gavrila and Munder, 2007] there are several ways to obtain ROIs:

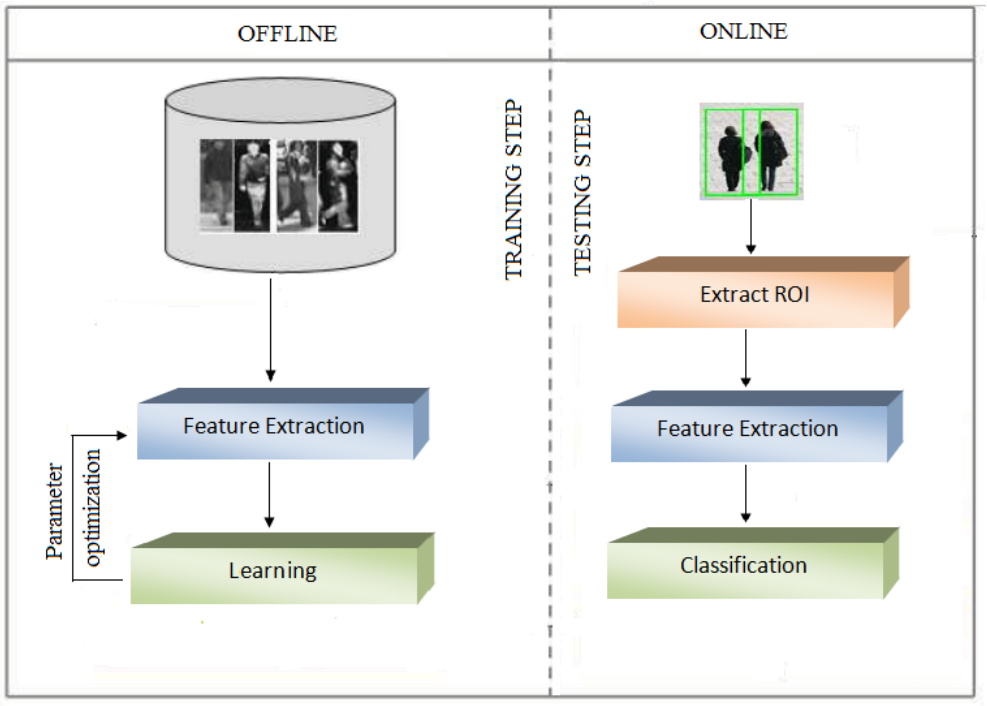


Figure 3.1: General steps in a pedestrian detection system

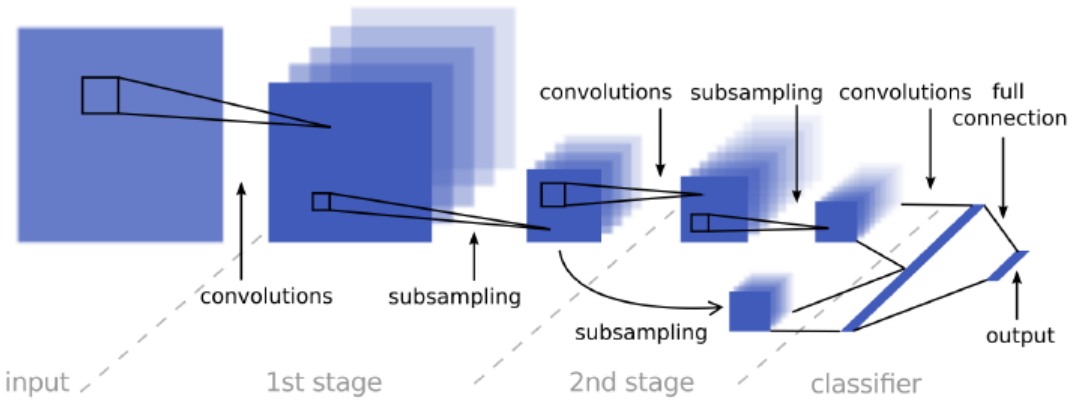


Figure 3.2: A multi-scale convolutional network [Sermanet et al., 2011]

1. *Standard background substraction* method, which can be used for surveillance applications, but is unsuitable for the case of moving camera
2. *Sliding window* approach, which implies shifting ROI windows of various scales over the images, while performing classification on each window
3. *Detection of independently moving objects* method, which is based on the idea of detecting deviations (from the expected background motion) in the optical flow field
4. *Stereovision*: based on disparity space.

After ROI have been identified, the feature extraction process can start. Features can be divided in global and local. Global features take the image as a whole. They include contour representations, texture features and shape descriptors, e.g. Local Binary Patterns, Histogram of Oriented Gradients and Haar features. Unlike global features, local features are calculated at several points in the image (points of interest), having the advantage of being more robust to occlusion and clutter [Lisin et al., 2005]. Examples of local features are Scale-Invariant Feature Transform, Speeded Up Robust Features, Haaris. There are also hybrid systems that combine local and global features to exploit the advantages of both representations [Lisin et al., 2005], [Besbes et al., 2015]. From another perspective, features can be classified into static (the ones mentioned above) and motion, which are able to capture motion from a sequence of images.

In the following we are presenting an overview on several global and local features, along with a family of motion features.

3.1.1 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005] are local feature descriptors commonly used in computer vision and pattern recognition, which are based on the idea that local object shape can, in most of the situations, be represented using the distribution of edge directions or local intensity gradients, even without exactly knowing the corresponding edge positions or gradients.

Their implementation supposes splitting the image window into small spatial cells (regions) and then computing a local histogram of edge orientations or gradient directions for the pixels within each cell. The final representation is obtained by combining the histogram entries. Contrast normalization of the local responses is performed for better invariance to illumination or shadowing, by collecting a measure of local histograms over larger spatial regions, called blocks, and using them for normalizing the cells in the block (see Figure 3.3) [Dalal and Triggs, 2005].

HOG features are used by most of existing pedestrian detectors.

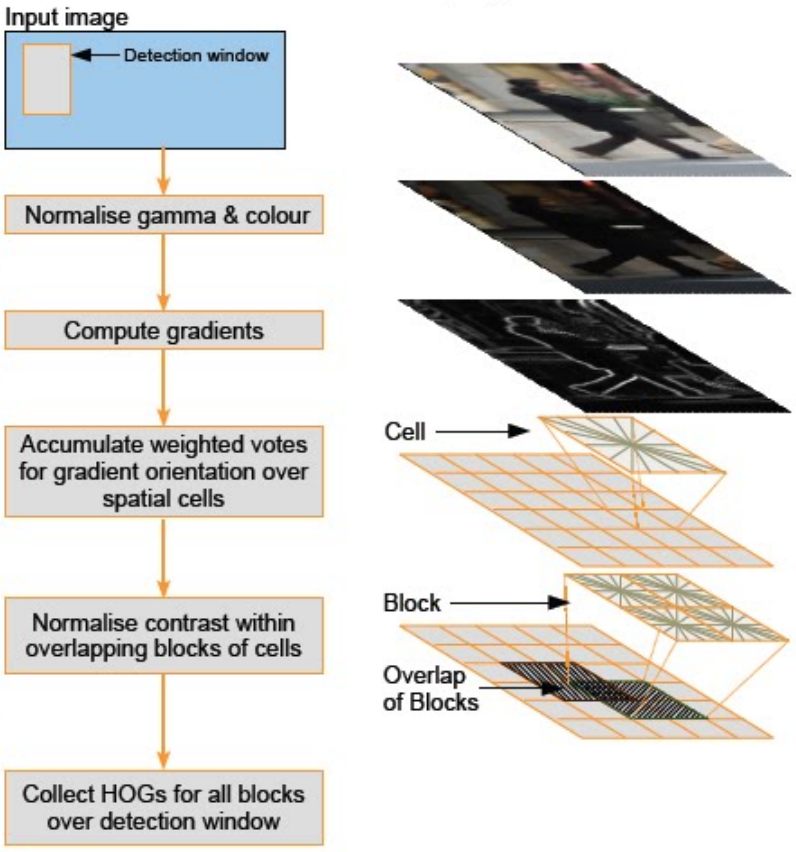


Figure 3.3: Feature extraction using HOG [Dalal and Triggs, 2005]

3.1.2 Histograms of Flow

Histograms of Flow (HOF) are motions features, introduced in [Dalal et al., 2006]. Their advantage compared to other motion features like the once introduced in [Viola et al., 2005] is that they are able to capture human motion from moving cameras against dynamic background, unlike the other ones that work only with static camera and background.

HOF use differential flow in order to cancel the effects of camera motion and voting similar to the one used in HOG to achieve a robust coding. The authors express by I^x and I^y, the images which contain horizontal, respectively vertical optical flow components, by (I^x, I^y) the 2D flow image and $I^x_x, I^x_y, I^y_x, I^y_y$ the corresponding x and y derivate differential flow images. Two two familes of schemes are used: Motion Boundary Histograms (MBH) and Internal Motion Histograms (IMH). In the first one, I^x and I^y are considered independent images for which they perform HOG like computations, while in the second one, they use (I^x_x, I^y_x) and (I^x_y, I^y_y) as pairs for angular voting and the derivates are raplaced by spatial differences taken at a larger scale. Based on these schemes, they introduce several descriptors: IMHdiff, IMHcd, IMHmd and IMHwd which proved to achieve high performances for pedestrian detection. Walk et al. [Walk et al., 2010a] introduced a lower-dimensional variant of HOF, IMHd2, and used it in the state of the art pedestrian detector MultiFtr+Motion from the benchmark presented in [Dollar et al., 2012].

3.1.3 Local Binary Patterns

Local Binary Patterns (LBP) [Ojala et al., 1996] are one of most widely used texture descriptors in computer vision, due to its invariance to gray level changes. The main steps in computing LBP are:

1. The ROI is divided into cells, e.g. 8×8 pixels.
2. Each pixel in a given cell is compared with the pixels from its neighbourhood (see Figures 3.4, 3.5) , obtaining a bit string ("1" if the center pixels value is smaller than the neighbours value and "0" otherwise)
3. A histogram is computed over each cell, based on the decimal valued of transformed bit-string.
4. The final feature vector is obtained by concatenating and normalizing the histograms of all cells

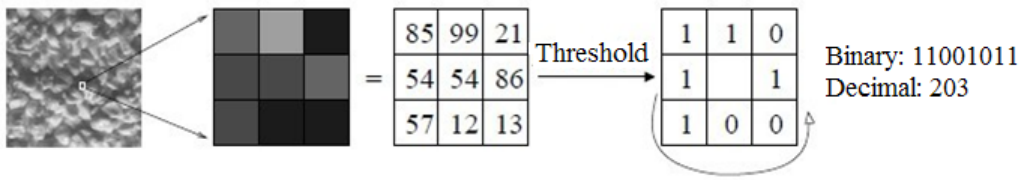


Figure 3.4: Basic LPB operator [Ahonen et al., 2006]

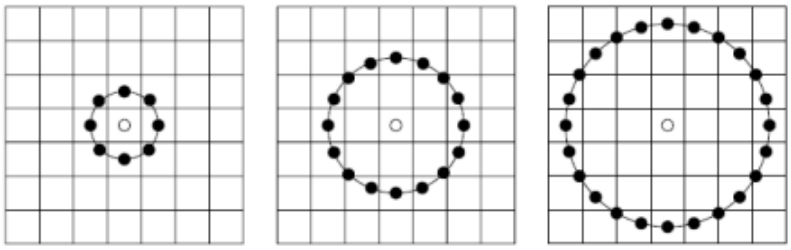


Figure 3.5: Examples of the extended LBP operator, with the circular (8, 1), (16, 2), and (24, 3) neighborhoods [Huang et al., 2011]

3.1.4 Scale-Invariant Feature Transform

Scale-invariant feature transform (SIFT) [Lowe, 2004] are local features, invariant to image scale and rotation, used in computer vision and pattern recognition.The main steps in SIFT computation, presented in [Lowe, 2004] are:

1. *Scale-space extrema detection.* All scales and image locations are searched through a difference-of-Gaussian function in order to determine potential points of interests which are invariant to scale and orientation.

2. *Keypoint localization*: Location and scale is determined at each candidate location, keypoints being selected based on measures of their stability.

3. *Orientation assignment*: Several orientations are assigned to each keypoint location, considering local image gradient directions. All following operations are performed on image data which has been previously transformed relative to the allocated orientation, location and scale, in this way achieving invariance to these transformations.

4. *Keypoint descriptor*: All local image gradients are computed at the determined scale in the regions surrounding the keypoints, being converted into a representation which can handle important local shape distortion and variation in illumination (see Figure 3.6).

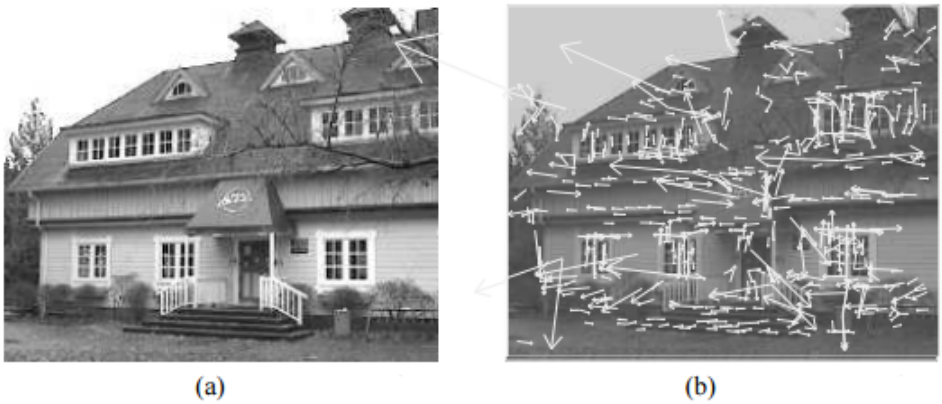


Figure 3.6: SIFT keypoints displayed as vectors illustrating scale, orientation, and location. [Lowe, 2004]

3.1.5 Speeded Up Robust Features

Speeded Up Robust Features (SURF) [Bay et al., 2006] (see Figure 3.7) are local features, scale and rotation invariant, used in computer vision for object recognition or 3D reconstruction. They are partly inspired by SIFT descriptors, but the authors proved that SURF detector is faster than SIFT, due to the integral images they make use of. The descriptor is based on 2D Haar wavelet responses sums around the point of interest.

3.1.6 Haar Wavelets

Haar wavelets [Oren et al., 1997] were the first features used in a real time vision system [Viola and Jones, 2001]. Their main advantage is the fast computation, making use of integral images. In [Oren et al., 1997] the main configurations were proposed: horizontal, vertical and corner (see Figure 3.8), then in [Viola et al., 2005] they were extended. The features are obtained by computing the difference between the sum of intensities in two rectangular areas with different configurations and sizes (see Figure 3.8).

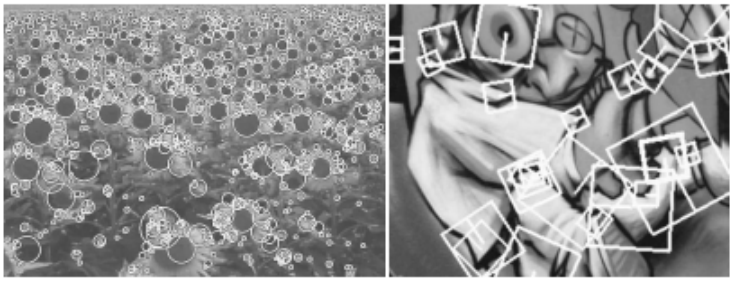


Figure 3.7: SURF keypoints for a sunflower field (a) and a graffiti (b) [Bay et al., 2006]

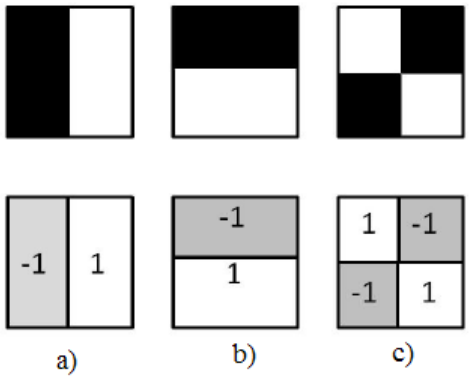


Figure 3.8: Haar wavelets configurations: vertical (a), horizontal (b) and corner (c) [Oren et al., 1997]

3.2 Classification

The performance of a pedestrian detection system is influenced by two important factors: the extracted features and the learning model. Support Vector Machines and boosting are widely used due to their good performance and extensibility [Dollar et al., 2012]. In the literature there are also approaches using Artificial Neural Networks and more recently, Deep Learning (e.g deep neural networks - DNNs), convolutional deep neural networks (CNNs), deep belief networks (DBNs) or recurrent neural networks (RNNs) [Deng and Yu, 2014].

3.2.1 Support Vector Machines

Support Vector Machines (SVM) [Vapnik, 1998] are supervised learning models used for classification and regression. Generally, classification is defined for the situation when there are m objects, each one belonging to one of the n classes, and a classification task would be to assign the belonging class to a new given object. In the case of binary classification using SVM, being given a set of training exampleswith their associated labels (one of the two categories), the SVM training algorithm constructs a model which is able to predict the category of a new example (see Figure 3.9).

SVMs apply linear classification to non-linear classification problems with a technique known as "kernel trick", which implies using a kernel function that maps data from the input space into a higher dimensional one, in which data becomes linearly separable. A separatting hyperplane is found by maximizing the margin between the different classes.

Let $T = (x_i, y_i)_{i=\overline{1,m}}$ be the training set, where $x_i \in \Re^d$ represents the input instance (data point) and y_i the class label of the object x_i $y_i \in \{-1, +1\}$. In the training phase of SVM, the input data points are mapped to a higher dimensional space. In this space, a maximal separating hyperplane is determined. In order to build a maximal margin classifier, the following convex quadratic programming problem has to be solved [Diosan et al., 2012]:

$$\begin{aligned} &\text{minimise}_{w,b,\xi} \tfrac{1}{2} w^T w + C \textstyle\sum_{i=1}^{m} \xi_i \\ &\text{subject to:} \quad y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ &\qquad\qquad\quad \xi_i \geq 0, \forall i \in \{1, 2, \ldots, m\}. \end{aligned} \tag{3.1}$$

The coefficient C, also known as *penalty error* or *regularization parameter*, is a parameter which controls the compromise between maximizing the margin and minimizing the classification error. For larger C values, the SVM might choose a separating hyperplane with smaller margin, as long as the number of correctly classified instances is maximized. On the other hand, if the margin is too small and the pattern of future data to predict varies more from the one used for training, the performance of the classification will be affected. Therefore, the choice of C parameter is crucial for achieving a high classification performance. The most popular manner to determine the optimal value of C is through cross-validation. The training data is randomly partitioned into k folds having approximately equal size. Then, $k-1$ folds are used for learning and the remaining one for testing. The process is repeated for all the folds (k times) and an average of the performances is computed [Olson and Delen, 2008].

The optimal hyperplane algorithm introduced originally by Vapnik in 1963 was a linear classifier [Vapnik, 1995]. Nevertheless, in 1992, Boser, Guyon and Vapnik [Boser et al., 1992] have proposed a way to build non-linear classifiers by using the *kernel trick*. The idea behind kernel methods is to map the input data points into a higher dimensional space, where the separating hyperplane must be found. The definition of this mapping ϕ implies describing an inner product for the feature space through a kernel function: $K(x, y) = \phi(x)^T \phi(y)$ [Schölkopf, 2000]. The goal is to determine the result of the inner product, for which is not needed to know the nature of the feature space or to have an explicit representation of ϕ. Therefore, the only requirement is to compute the kernel function on all the pairs of input instances [Olson and Delen, 2008].

In the literature there is a large variety of kernels that can be used. Examples of such kernels are: Linear, Polynomial, Normalised Polynomial, Laplacian, Gaussian or Euclidean. The choice of a suitable kernel function K is crucial for the learning process and must be performed depending on the particular problem that has to be solved. This should also consider a compromise between the model's complexity and the classification accuracy, since there are

situations in which a high speed of the classification is more important than obtaining a very high accuracy (e.g. real time applications).

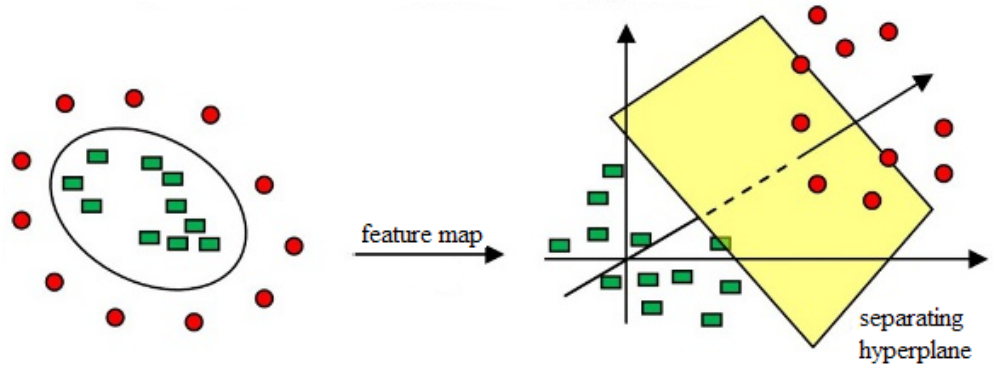


Figure 3.9: Classification using SVM

3.2.2 Adaptive Boosting

Adaptive Boosting (AdaBoost) [Freund and Schapire, 1997] is a machine learning algorithm that biulds a strong classifier by combining weak classifiers (often rule-of-thumb) in an iterative greedy mode. AdaBoost is well known due to its speed optimized by the use of cascades and can be combined with any classifier to find weak rules. Its disadvantage is that is sensitive to noise and outliers.

In the following we give a brief description of the main steps of the algorithm, as presented in [Schapire, 2013]. Given a set of n training examples $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$, where x_i is a feature and y_i a label, several iterations are performed. In each iteration, a distribution is computed over the training examples and a *weak* learning algorithm is applied in order to determine a weak hypothesis with low weighted error relative to the considered distribution. The final hypothesis is obtained by computing the sign of the weighted combination of weak hypotheses.

3.2.3 Artificial neural networks

Artificial neural networks (ANNs) are a type of machine learning model inspired by biological neural networks. They are composed by *neurons* organized in *layers* and interconnected by *axons*, allowing a large number of possible combinations. ANNs are able to provide a non-linear decision and can also work with raw data, without needing an explicit feature extraction process. Their advantage is that they are not sensitive to incomplete or noisy data, being able to learn complex patterns, but they need a high training time and have many parameters to be tuned.

3.3 Literature review

Significant research has been performed in the area of pedestrian detection and recognition. In [Enzweiler and Gavrila, 2009] the authors present the components of a pedestrian detection system and compare several state-of-the-art systems on the same datasets. Wavelet features are combined with AdaBoost, HOG is used in conjuction with SVM and Shape-texture with Neural Networks that use local receptive fields. The authors conclude that the classifier which uses HOG with SVM achieves the best results among the considered models. [Enzweiler and Gavrila, 2009].

In [Dollar et al., 2012] a challenging monocular visible dataset (the Caltech database) is proposed, together with an detailed comparison of several pedestrian detectors. Their study indicates that the best algorithms are the ones that make use of motion information.

A very recent research direction in pedestrian detection is oriented towards deep learning. In [Ouyang and Wang, 2013] is introduced a framework for deep learning that learns four components of a pedestrian detection system: feature extraction, deformation handling, occlusion handling and classification, in order to maximize their strengths. Experimental evaluations on Caltech data set indicate that their deep model outperforms the best-performing pedestrian detector at that moment. In [Angelova et al., 2015] is proposed a Large-Field-Of-View deep network for pedestrian detection, which is able to process larger image regions much faster than typical deep networks. The main idea of their method is to perform classification simultaneously at multiple locations. Experimental evaluations on Caltech data set show that their method is faster than joint deep one [Ouyang and Wang, 2013].

Even if deep learning is a promising approach for pedestrian detection, an important difficulty is finding appropriate data sets to train on. Since it works on the whole image, opposite to the classical approaches which involve the sliding window technique, it needs very large annotated data sets with complex scenarios.

Another recent direction of research, on which we are going to focus in the next chapter, approaches the combination of different features (multi-feature) and modalities (multi-modality), extracted from visible domain like intensity, motion information extracted from optical flow and depth information from the disparity map [Miron et al., 2015]. Some approaches combine depth and motion for hypothesis generation (see [Nedevschi et al., 2009], [Enzweiler et al., 2008]), but since in the thesis we address the classification problem, we are going to focus only on the ones which integrate them in the classification stage. Choosing a suitable fusion scheme in order to combine the information extracted is crucial. In the literature there are two classical fusion schemferes: the early fusion, at the low level of features and the late fusion, at the high level of matching scores. Of course, there are also approaches based on hybrid fusion schemes. In the following we are going to briefly present some representative approaches from the literature from each category.

In [Dalal et al., 2006] is proposed a detector that uses a combination of appearance descriptors obtained from a single frame of a video and motion descriptors obtained from

optical flow or spatio-temporal derivates of the succeeding frame. The authors apply both low-level and high-level fusion schemes. In the low-level fusion appearance and motion features are concatenated in a large feature vector that feeds a single classifier, while in the high-level fusion a two stage classification is used. First, individual classifiers are trained on each type of features, then a classifier is used to combine their outputs. Their experiments show that the high-level fusion approach is slightly better than the low-level one and could be improved by combining the components in a more complex manner.

In [Wang et al., 2009] is proposed a human detector with partial occlusion handling, that combines HOG and LBP into one feature vector, using a low-level fusion scheme. Experiments indicate that their model outperforms other existing detectors on the INRIA dataset.

In [Rohrbach et al., 2009] is proposed a high-level fusion of spatial features obtained from dense stereo and intensity images. Two classifiers are trained on features extracted from depth, respectively intensity images, then their outputs are combined utilizing a set of fusion rules: maximum, product, sum and SVM rules. Experiments on Daimler pedestrian data set show that their high-level fusion approach outperforms the state-of-art intensity only model and the low-level fusion approach using a joint feature space.

In [Oliveira et al., 2010] is presented an approach that combines HOG and local receptive fields (LRF) features at the classification level. Each type of feature is classified using both multilayer perceptrons (MLP) and SVM, then all outputs are fused using two fusion schemes: majority voting and fuzzy integral. Experiments on Daimler-Chrysler data set indicate good performances.

In [Walk et al., 2010b] are explored combinations of different classifiers: SVM and MPL-Boost, respectively HIKSVM and MPLBoost, on the same feature set obtained by joining appearance and motion features (HOG and histogram of oriented flow). Experiments on several pedestrian data sets indicate a performance improvement over the individual classifiers.

In [Enzweiler et al., 2010] are used for the first time together appearance, motion and stereo features for pedestrian recognition. The authors propose a multi-modality approach, which combines features extracted from images in three modalities: intensity, depth and flow into a mixture-of-experts framework. Later, in [Enzweiler and Gavrila, 2011] the framework is extended, being able to combine information from multiple features and cues. On pose-level are used shape cues based on Chamfer shape matching, on modality-level are considered intensity, depth and flow modalities and on feature level are used HOG and LBP. Individual expert classifiers on pose, modality and feature levels are integrated through a probabilistic model. Experiments on Daimler data set illustrate an important performance boost over the state-of-art classifier using intensity only and the joint feature approach.

In addition to the multi-feature and multi-modality approaches, there are models which fuse information from different domains (multi-domain) like visible and far infrared domains (see [Apatean et al., 2010], [Besbes et al., 2011]).

It was proved in the works above, that fusing information from multi-modality images leads to an important performance boost of the pedestrian recognition system. In the litera-

ture there are approaches using low-level fusion, by concatenating features from all modalities into a large feature vector, or high-level fusion at classification level. Even if they achieve good results, there are drawbacks to both approaches. The modality concatenation approach has the disadvantage of high dimensionality, sensitivity to miss-calibration and miss-functioning of on-board cameras and cannot be trained independently on each modality. The matching scores fusion approach like sum, product, majority vote, can only use different weights for the modalities, but not for each image in particular. With this in mind, our research question is the following. *Can a dynamic model eliminate the non-discriminative modalities for each image in particular, representing a bounding box within an image frame, for reducing complexity and improve performance*? In the next chapter we are going to introduce a few approaches for pedestrian recognition, including two dynamic models that aim to answer to this research question. Even if our models are applied to the multi-modality fusion, they can be generalized in order to handle multi-feature or multi-domain fusion.

Chapter 4

New approaches to pedestrian classification

In this chapter, we address the problem of pedestrian recognition in single and multi-modality images. The models introduced in this chapter are original, and were introduced in [Andreica et al., 2013, Sirbu et al., 2014b, Rus et al., 2015, Sirbu et al., 2015, Besbes et al., 2015].

The structure of the chapter is as follows. In Section 4.1 we define the problem of pedestrian recognition and its relevance, then we continue in Section 4.2 with a comparison of several state-of-the-art features in far infra-red (FIR) spectrum. In Section 4.3 we aim to investigate the effectiveness of using kernel descriptors, a generalization of HOG, for solving the pedestrian recognition task. Even if kernel descriptors proved to obtain good performances for visual recognition, to our best knowledge, they have not been used for pedestrian detection so far.

We begin by presenting a short literature review on kernel descriptors, then we continue with our studies on pedestrian recognition using these features for image representation. In the first study we investigate how two learning algorithms, Support Vector Machines and Genetic Programming (GP), are able to perform pedestrian recognition using kernel descriptors, extracted with three types of kernels: Exponential, Gaussian and Laplacian, while in the second one we study how kernel descriptors perform in single vs. multi-modality pedestrian recognition.

In Section 4.4 we propose two dynamic models for pedestrian recognition, that are able to select the most discriminative modalities for each image in particular and further use them the classification process. The first one, Dynamic Modality Selection (DMS) selects one suitable modality to classify an image, while the second one, Dynamic Modality Fusion (DMF) performs a fusion of the modalities that are considered suitable. Finally, in Section 4.4.6 we present our software built in order to train and test our dynamic models.

The original contributions of this chapter are the following:

- A comparison of several state-of-the-art features in FIR spectrum on a new dataset

(Subsection 4.2) [Besbes et al., 2015]

- The usage of kernel descriptors for pedestrian recognition (Subsections 4.3.2 and 4.3.3) [Andreica et al., 2013, Sirbu et al., 2014b]
  - A comparison on how two machine learning algorithms: SVM and GP are able to learn based on KD features extracted by using three kernels: Exponential, Gaussian and Laplacian (Subsection 4.3.2) [Andreica et al., 2013]
  - A comparison on how KDs perform on single vs. multi-modality images, with parameters optimized independently on each modality: intensity, depth and flow (Subsection 4.3.3) [Sirbu et al., 2014b].
- A dynamic modality selection algorithm which retains one suitable modality for the classification of an image among intensity, depth and flow (Subsection 4.4.4) [Rus et al., 2015]
  - Experimental evaluations of the algorithm on a pedestrian data set, analysis of the results and comparisons to other approaches from the literature (Subsection 4.4.4.1) [Rus et al., 2015]
- A dynamic modality fusion algorithm which fuses the modalities considered suitable for the classification of an image among intensity, depth and flow (Subsection 4.4.4) [Sirbu et al., 2015]
  - Experimental evaluations of the algorithm on a pedestrian data set, analysis of the results and comparisons to other approaches from the literature (Subsection 4.4.5.1) [Sirbu et al., 2015]

4.1 Problem statement and relevance

Pedestrian detection is one of the most popular research directions in the domain of object detection and computer vision. The number of vehicles has exponentially increased on the road over the last two decades. As a consequence the number of car accidents has increased too and along with that grew the need to develop better traffic safety mechanisms. Pedestrians represent a great part of the traffic and in order to protect them, different pedestrian detection systems were developed. The goal of these systems, called ADAS, is to improve driver's perception, in order to avoid collisions to pedestrians. A study performed by ABI Research shows that Mercedes-Benz, Volvo and BMW dominate the market of cars enhancing ADAS systems. It is very difficult to develop a very precise ADAS, mostly because of the way a pedestrian's appearance can vary. A pedestrian can change pose, carry different objects, have different shapes and heights, wear different clothes.

The main requirements of a pedestrian detection system are:

Figure 4.1: ADAS pedestrian detection

- *Robustness.* Such a system must be capable to recognize pedestrians with different appearances, heights, clothes, under occlusions and in difficult environment conditions (fog, snow, rain, dark).
- *Precision.* Unrecognized pedestrians can lead to dangerous situations, while many false alarms can reduce driver's confidence in the system.
- *Real time.* The system must detect the pedestrians very fast in order to give the driver enough time to react.
- *Cost.* The cost of a pedestrian recognition system should not be higher than the cost of the car. Sensors like Lidar or infrared cameras are very expensive. The system that we propose needs only a stereo camera.

As presented in the previous chapter, pedestrian detection is performed in several steps. Firstly, regions of interest (ROIs) are identified within an image frame (hypothesis generation), then ROIs are classified into *pedestrian/non-pedestrian* (hypothesis refinement). In this thesis we are focusing on the latter step, which is reduced to a binary classification problem.

Generally, *classification* is defined for the situation when there are m objects, each one belonging to one of the n classes, and a classification task would be to assign the belonging class to a new given object. The classification task is performed in two steps: a training step and a testing step.

In the training step, a classification model is learned from the training set, formed by tuples and their labels. Let $Im = \{Im_1, Im_2, \ldots, Im_n\}$ be the set of images to be classified in

our particular problem of pedestrian classification, each image belonging to one of the classes $C = \{pedestrian, non - pedestrian\}$. For each image (in our case a ROI within a frame) is computed a set of features, described by an m-dimensional vector, $F_i = (F_{i1}, \ldots, F_{im}), F_{ik} \in \Re, 1 \leq i \leq n, 1 \leq k \leq m$. An element F_{ik} from the vector characterizing the image Im_i represents the k^{th} feature belonging to i^{th} image. The training phase can be seen as the process of learning a mapping function $f : Im \rightarrow C$, for which $f(x) = y$, which are able to predict the label y for a given image x.

In the testing step, the learned model is used for classifying unseen tuples from the test set. For evaluating the classification performance, the class label predicted by the classifier is compared to the actual one.

In order to optimize classifiers' parameters, training is usually performed in two phases: a learning phase, in which a model is learned on a subset of the training set and a validation phase, where the model is tested on the remaining examples from the training set. The most widely used optimimization technique, cross-validation, extends this idea and applies this process k times on mutually exclusive subsets of data, taking into account the average of the performances.

4.2 Feature comparison in FIR domain

It is known that HOG features are among the best in the visible domain. In this section we aim to investigate how they perform in far infrared (FIR) images and also provide a comparison with other state-of-the-art features: HC Surf, Haar and Haar-like. This comparison is our contribution to the work presented in [Besbes et al., 2015], in which Bassem et al. introduce a new approach for pedestrian detection in FIR daytime images using a hierarchical codebook of SURF.

4.2.1 Dataset

In the experiments we use the Tetravision image database, provided by the Artificial Vision and Intelligent Systems Laboratory (VisLab) of Parma University [Bertozzi et al., 2006]. The images, representing daytime road scenes, were taken with on-board stereo-vision cameras in visible and FIR spectrum and then manually annotated. We chose for training a subset of images containing pedestrians with a large variability in appearances, scales and view points.

The training set contains 987 examples:

- 500 - pedestrians
- 487 - non-pedestrians

The test set contains a total number of 2092 examples:

- 1089 - pedestrians

- 1003 - non-pedestrians

4.2.2 Experimental evaluation

Haar, Haar-like and HOG features were extracted from resized bounding boxes (BBs) of 64×128 pixels in FIR images. From the Haar wavelet we collected 64 features computed at the fourth level decomposition. While the coefficients of Haar wavelets were used with RBF-SVM classifiers, the Haar-like features [Viola and Jones, 2004] were trained on a cascade of boosted classifiers with 11 stages. For HOG features, we computed histograms with 9 bins on cells of 8×8 pixels, with a block size of 2×2 cells overlapping by one cell size.

Features	HC Surf	Haar	Haar-like	HOG
Classifiers	RBF-SVM		boosted cascade	linSVM
F-measure (%)	**96.2**	91	95.43	95.7

Table 4.1: Comparison of state-of-the-art features in FIR domain [Besbes et al., 2015]

The results presented in table 4.1 show that, the HC local/global SURF-feature representation introduced in [Besbes et al., 2015] achieves, on FIR images, the highest F-measure allowing a significant improvement of performance when compared with Haar and wavelet based features. Whereas, we observe that the proposed feature representation slightly outperforms the state-of-the-art Haar-like and HOG features. One should recall that the presented results are obtained within the pedestrian detection framework introduced in [Besbes et al., 2015], since the features were extracted from the BB provided by the ROI generation component. For this comparison, we considered the state-of-the-art features, not in the visible spectrum where they have been proposed, but on FIR images.

4.2.3 Conclusions

It was proved that HOG features achieve very good results, both in visible and FIR images. Recently, in [Bo et al., 2010] is introduced a framework for feature extraction, kernel descriptors, considered by the authors a generalization of HOG. Since they seem very promising and, to our best knowledge, they have not been used for pedestrian recognition so far, we aim in the following section to investigate their effectiveness for pedestrian recognition, in both single and multi-modality images in visible spectrum.

4.3 Pedestrian recognition using kernel descriptors

In this section we begin by presenting a background on kernel descriptors, then we continue with a couple of studies that we performed in order to investigate their performance on the

particular task of pedestrian classification.

4.3.1 Background on kernel descriptors

Kernel descriptors (KDs) [Bo et al., 2010] can be seen as a generalization of orientation histograms (including HOG), which are a particular type of match kernels over *patches* (regarded as a collection of *blocks*). Moreover, kernel descriptors intend to overcome some drawbacks of histograms based approaches, in which similarity between different image regions is determined based on their histogram. For computing the histogram, pixel values as discretized into bins, therefore quantization errors might be introduced.

The advantage of kernel descriptors is that the pixel attribute values are not discretized, but are converted into rich patch-level features. Therefore, the similarity between different images regions is determined based on a match kernel function. For computation efficiency reasons, approximate, low dimensional match kernels are computed.

Kernel descriptors can also be applied hierarchically over sets of kernels. In the hierarchical approach [Bo et al., 2011b], image features are obtained by recursively applying kernel descriptors.

In [Wang et al., 2013], a supervised version of kernel descriptors is introduced, named Supervised Kernel Descriptors (SKDES). In this supervised approach, the authors use the bag-of-words (BOW) image classification pipeline and large margin criterion in order to learn the low level patch representation, making the patch features more compact and achieving better performance than KDs.

In [Bo et al., 2011a], Depth Kernel Descriptors are introduced for object recognition, consisting of a set of kernel features on depth images for modeling 3D shape, size and depth edges. The authors show that their features achieve a better performance than 3D ones like Spin and that they obtain an improvement of the capabilities of recognition in RGB-D and depth.

In the following, we are presenting three matching kernels: the gradient match kernel, the color kernel and the local binary pattern kernel, that were introduced in [Bo et al., 2010], and the approach used by the authors to learn compact features.

Image variantions can be identified by the **gradient match kernel**, K_{grad}, which is composed by three kernels: a kernel of magnitudes (represented by a linear one), an orientation kernel and a position kernel:

$$K_{grad}(P,Q) = \sum_{z \in P} \sum_{z' \in Q} \tilde{m}(z)\tilde{m}(z')k_o(\tilde{\theta}(z), \tilde{\theta}(z'))k_p(z,z') \tag{4.1}$$

where $k_p(z,z') = exp(-\gamma_p||z-z'||^2)$ is a Gaussian position kernel between two pixels (given by their position inside a region of the image), which is normalized to [0,1], and $k_o(\tilde{\theta}(z), \tilde{\theta}(z'))$ is a Gaussian kernel over orientations.

For the estimation of the difference among the orientations at pixels z and z', the normalized gradient vectors from (4.2) are used in the kernel function k_o

$$\tilde{\theta}(z) = [sin(\theta(z))cos(\theta(z))] \tag{4.2}$$

The normalized linear kernel weights the contribution of each pixel using gradient magnitudes, being the same as the one from orientation histograms. The aim of the orientation kernel k_o is to compute the distance between the gradients of two pixels, while the position kernel k_p is destinated to compute the spatial distance between two pixels [Bo et al., 2010].

The **color match kernel**, K_{col}, describes image appearance and is based on two kernels: a color kernel and a position kernel:

$$K_{col}(P,Q) = \sum_{z \in P} \sum_{z' \in Q} k_c(c(z), c(z'))k_p(z, z') \tag{4.3}$$

where $c(z)$ represents the color (the intensity or the RGB values) of the pixel from position z. The color kernel, $k_c(c(z), c(z')) = exp(-\gamma_c||c(z) - c(z')||^2)$ measures how similar two pixel values are [Bo et al., 2010].

The shape variations can be identified by the **shape match kernel**, K_{shape}, which is also composed by three kernels: one of standard deviations of pixel values in a neighbourhood, one of pixel value differences in a local window and a position kernel [Bo et al., 2010][Diosan and Rogozan, 2012].

$$K_{shape}(P,Q) = \sum_{z \in P} \sum_{z' \in Q} \tilde{s}(z)\tilde{s}(z')k_b(b(z), b(z'))k_p(z, z') \tag{4.4}$$

where $\tilde{s}(z) = s(z)/\sqrt{\sum_{z \in P} s(z)^2 + \epsilon_s}$, $s(z)$ measures the dispersion of pixel values in a local window (*e.g.* the 3 x 3 neighborhood around z), ϵ_s is a small fixed value, and $b(z)$ is a binary column vector, which binarizes the pixel value differences in local neighbouring region surrounding z.

The normalized linear kernel $\tilde{s}(z)\tilde{s}(z')$ weights the contribution of each locally binary pattern, and the Gaussian kernel $k_b(b(z), b(z')) = exp(-\gamma_b||b(z) - b(z')||^2))$ measures the shape similarity using local binary patterns [Bo et al., 2010].

The similarity between image patches is calculated in a principled way through match kernels, but when image patches are large, evaluating kernels might be computational expensive [Bo and Sminchisescu, 2009]. In order to efficiently obtain the corresponding representation of an image by using KDs, the authors have proposed a solution of two steps: firstly, several basis vectors are selected from the support region (selection being performed such as the match kernels to be approximated by a good precision) and, secondly, the compact basis vectors are learned (by using Kernel Principal Component Analysis)

4.3.2 KDs representation in SVM vs. GP

Since the most important conditions for achieving a performant classification algorithm are using an appropriate representation of objects to be classified and an effective algorithm for making decisions, we aim to investigate how efficiently can kernel descriptors represent

data for two different learning algorithms: SVM and GP. Features are extracted using three type of kernels: Exponential (4.5), Gaussian (4.6) and Laplacian (4.7).

$$K_{Exp}(x,y) = exp(-\frac{|x-y|}{2\sigma^2}) \tag{4.5}$$

$$K_{Gauss}(x,y) = exp(-\frac{|x-y|^2}{2\sigma^2}) \tag{4.6}$$

$$K_{Lapl}(x,y) = exp(-\frac{|x-y|}{\sigma}) \tag{4.7}$$

Why Kernels? Because, by definition, they catch the similarity between arbitrary inputs, being able, in the same time, to integrate invariance (that is present in the case of image processing), to capture dependencies and to perform an efficient computation and storage.

Why SVM? It is well known that linear SVMs are currently the most frequently used classifiers in Computer Vision [Lampert, 2009] since its training time is approximately linear in data dimensionality and, also, approximately linear in number of training examples. Furthermore, the evaluation time (per test example) is linear in data dimensionality and is independent of number of training examples. According to [Lampert, 2009] SVMs with non-linear kernel are usually used for small to medium sized Computer Vision problems because their training time is generally cubic in number of training examples, while their evaluation time is generally linear in number of training examples. Furthermore, the classification accuracy is usually higher compared to linear SVMs. Shortly, linear SVMs are very fast in training and evaluation, while non-linear kernel SVMs achieve better results, but do not scale well.

Why GP? There are several arguments that sustain the utility of GP-based methods in solving classification problems, in general, and object recognition, in particular. Firstly, a GP-based method is able to perform an implicit and automatic transformation of data (the original features can be pre-processed by different methods: selection of a subset of original attributes, weighting the original attributes, construction of new features as functional expressions involving the original attributes). The feature selection is part of the evolutionary process of GP that involves individuals encoded as variable-length chromosomes, while the feature construction benefits of the GP individual's ability to combine the attributes trough different operations. Secondly, GP is able to extract various models (like classification rules or discriminant functions) from data. Furthermore, due to its capability to evolve complex discriminant functions, GP is able to solve both linear classification problems and non-linear classification problems without apriori specifying the problem type (linear or non-linear).

When a linear classifier can not solve the problem, two solutions can be considered:

- a combination of several linear classifiers (as in the case of ANN or Boosting which actually encode decision functions which depend non-linearly on input data) or

- a data pre-processing step, when the original input data is transformed form the original space of representation (non-linear) into a new space that is, in general, a higher dimensional one and where the data becomes linearly separable. Usually this step is performed through the kernel functions (as in the case of SVMs). For instance a set of points can be non-linearly separable in Cartesian coordinates, but linearly separable in Polar coordinates.

Unlike other machine learning algorithms, GP automatically combines these two solutions during the evolution process, its individuals being able to automatically encode both type of classifiers (linear and non-linear). Related to how GP can solve Computer Vision tasks, the discriminant functions evolved by the GP algorithm are very akin to the kind of mathematical operations and transformations usually applied to image processing. GP are flexible. It is well known that GP individuals are able to represent a great variety of learning formalisms (eg. decision trees, classification rules, discriminant functions), but also learning mechanisms (like those involved in ANNs or SVMs). Flexibility also concerns the adaptability of GP techniques to various tasks through its elements (fitness function, genetic operation, evolving mechanisms). GP ensures interpretability of the evolved classifiers since the size of GP individuals influences the comprehensibility of the model; the bigger the classifier, the harder to interpret for humans. GP is able to ensure a competitive performance.

Therefore, we intend to study a GP-based classifier that is able to solve the given problem, obtaining improvements that concern several aspects:

- performance of the classification – a better classifier in terms of accuracy
- human-independent models – GP individuals are able to automatically decide in two very important aspects: knowledge and model representation. Instead of performing a distinct pre-processing step, the GP method is able to automatically and simultaneously select the most relevant features and construct a relevant model.
- less complex models – GP is able to automatically apply the principles of Occam's razor: if two models have the same performance (in our case if two decision algorithms ensure the same classification accuracy) the less complex model should be preferred.

Taking into account all these aspects, we have used in our numerical experiments the following methodology: several features are extracted directly from images through kernel descriptors and, afterwards, two algorithms (an SVM and a GP-based classifier) are considered in order to learn the decision model. In both cases, the learning takes place in a cross-validation framework.

4.3.2.1 Feature extraction

For extracting features we used the framework proposed by L. Bo in [Bo et al., 2010], for which we tested different kernel functions.

In the first step, based on the code developed by Xiaofeng Ren for kernel descriptors ($http://www.cs.washington.edu/ai/Mobile_Robotics/projects/kdes/$), various kernel functions for extracting local features from an image were tested. Since the data set we work with contains only gray images, we investigated only the gradient kernel descriptor [Bo et al., 2010], which is able to capture image variations. As presented in Section 4.3, the gradient kernel descriptor is based on three kernels: one of magnitudes, one of orientation and a position one.

The kernel of magnitudes is a linear one and we cannot replace it with another type of kernel, because of the equivalence to the histogram of gradients that must be preserved. The other two kernels from the composition of the gradient kernel descriptor, the orientation kernel which calculates the similarity of gradient orientations and the position kernel that quantifies the spatial closeness of two pixels, are in fact Gaussian kernels. Therefore, we changed the implementation in order to evaluate other kernels like Exponential and Laplacian for both orientation and position kernels.

4.3.2.2 Learning algorithms

In order to learn a classifier for discriminating between pedestrians and non-pedestrians, we used two machine learning algorithms: SVM and GP.

4.3.2.2.1 SVM

Regarding the SVM, we have considered the LibSVM tool [Chang and Lin, 2001] because it is reliable and has many features implemented. Unlike Bo's framework [Bo et al., 2010], which uses the primal formulation of SVM, we chose an implementation of the Sequential Minimal Optimization method [Platt, 1999], due to is capability to solve the quadratic programming optimization problem very fast.

In order to determine the most appropriate kernel for the classifier, we applied a Linear kernel, a Gaussian kernel, a Polynomial kernel and a Normalised Polyomial kernel with different parameters, deciding for the linear one.

4.3.2.2.2 GP

For the evolutionary classifier a linear and efficient GP version is actually utilized: Multi Expression Programming (MEP)[Oltean, 2005]. MEP uses a linear representation of chromosomes and a mechanism to select the best gene for providing the output of the chromosome, unlike other GP techniques which use a fixed gene for output. Furthermore, no extra processing in order to repair newly obtained individuals is required.

The advantages of dynamic-output chromosome over fixed-output chromosome are notable mostly in the situations in which the complexity of the target expression is not known. Variable-length expressions can be implicitly provided, while other techniques like Grammatical Evolution or Linear GP require special genetic operators that insert or remove chromosome parts for achieving such a complex functionality. Moreover, due to code reuse, the MEP has

exponential length while the encoded expression may have polynomial length [Oltean et al., 2009].

4.3.2.3 Experimental evaluation

Several numerical experiments about how the discussed learning algorithms (an SVM and a GP-based classifier) are able to solve a particular image classification task (pedestrian recognition) are presented in this section. To evaluate the performance of the considered classifiers, the Daimler-Chrysler (DC) crop wise data sets (18×36 pixels image size) have been used as provided in [Munder and Gavrila, 2006]. Actually, a binary classification problem was solved: separate the images that contain pedestrians from the images that do not. There are considered 4480 images: 2240 for training the classifier and 2240 for testing. **red**We could have improved the learning process by using more examples for training than for testing, or using cross validation and bootstrapping, but the purpose of the experiment was only to compare the performance of the kernels within the Gradient kernel descriptors.

For evaluating the classification performance, accuracy rate was actually computed, which represents the percent of correctly classified samples from a given data set (see Equation 4.8).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.8}$$

	computed positive examples	computed negative examples
real positive examples	True positive	False negative
real negative examples	False positive	True negative

Table 4.2: Confusion matrix

True Positives (TP), False Negatives (FN), False Positives (FP) and True Negatives (TN) represent components of confusion matrix given in Table 4.2.

However, the accuracy rate indicates performance of the classification in a confidence interval. For deciding if a system outperforms another one, the confidence intervals of their performance must be computed. A system is considered to outperform another one if their confidence intervals are disjoint. For our statistical analysis we considered a confidence interval of 95% (see Equation 4.9):

$$\Delta I = 1.96 \times \sqrt{\frac{Acc(100 - Acc)}{N}}\% \tag{4.9}$$

where N is the number of examples from the testing set.

In Table 4.3 are presented the accuracy rates (and their confidence intervals) by considering different kernels (when the image descriptors are actually constructed) and two learning algorithms (SVM and MEP). The performance measures are computed by taking into account the test images and the best identified classifiers (SVM with the best hyper-parameters and MEP with an optimal configuration).

	Gaussian	Exponential	Laplacian
SVM	0.657 ± 0.010	0.535±0.010	0.599 ± 0.010
MEP	0.682 ± 0.009	0.667 ± 0.009	0.737 ± 0.009

Table 4.3: Accuracy rates (%) obtained by SVM and MEP algorithms on images represented by different kernel descriptors.

Several remarks can be done by analysing the results indicated in Table 4.3.

Concerning the kernel descriptors, our results indicate that the Gaussian kernel seems to be able to extract the most relevant features from images when the SVM classifier is used, while the Laplacian kernel provides more significant information for MEP. Even if the Gaussian kernel is largely involved in feature extraction process [Bo et al., 2010], our results suggest that a deeper study should be performed regarding the proper selection of the kernel involved in the feature extraction process. This study might also reveal some criteria for selecting the most appropriate kernel descriptor to use for the input data of a particular problem.

Regarding the learning algorithm, the evolutionary one seems to be able to better generalise over unseen data compared to SVM, obtaining better accuracy rates for all three considered feature extraction methods. Even if GP obtained better accuracy rates that SVM, we are going to use further the SVM from the following reasons: GP has variable performance from run to run, caused by the initialization of chromosomes therefore introducing a bias, it needs a large amount of time for training due to complexity (each chromosome is a model). Moreover, we want to compare our results with the ones from the literature and to make sure that the improvement comes from or dynamic approaches and not from the learning algorithm.

4.3.2.4 Conclusions

A study on how two learning algorithms are able to perform pedestrian recognition in images is presented in this section. Daimler-Chrysler benchmark image dataset is involved in our numerical experiments.

The first step is to convert each image in a numerical representation relevant for the classifier. Several kernel descriptors are considered on this purpose: Exponential, Gaussian and Laplacian kernels. A statistical algorithm, SVM, and an evolutionary approach, MEP, are used for the learning phase for which the input data is represented by the previously extracted features.

Better accuracy rates are obtained when using the evolutionary model for all considered kernel descriptors. Regarding the kernel descriptors used, SVM learning indicates that the Gaussian is the best one, while MEP achieves the best results by using the Laplacian kernel. Therefore, we can not conclude which is the most efficient kernel descriptor and we intend to perform a further study of how the kernel selection influences the quality of recognition.

4.3.3 Single vs. multi-modality pedestrian recognition

In this section we aim at investigating how does the kernel descriptor based classifier work for single vs. multi-modality images. For the multi-modality approach we considered three modalities from visible domain: intensity, depth and flow. We decided to address only the visible domain, since it's less expensive and employs less complex models.

4.3.3.1 Feature extraction

For the feature extraction phase we chose kernel descriptors [Bo et al., 2010], presented in Section 4.3, because they overcome some drawbacks of histograms based techniques as mentioned in Section 4.3.

4.3.3.2 Learning algorithm

For the learning phase we use Support Vector Machines (SVM) [Vapnik, 1995]. There are two types of kernels used by SVMs: linear and non-linear. Even if the non-linear SVMs perform better than the linear ones, the improvement of results is paid with high computational costs and memory which is a big problem for pedestrian detection systems which must perform in real-time. From these reasons, we chose Liblinear [Fan et al., 2008], a very popular implementation of linear SVM.

4.3.3.3 Experimental evaluation

In this section we aim at presenting the experimental evaluation of single and multi-modality classifiers, using kernel descriptors for image representation.

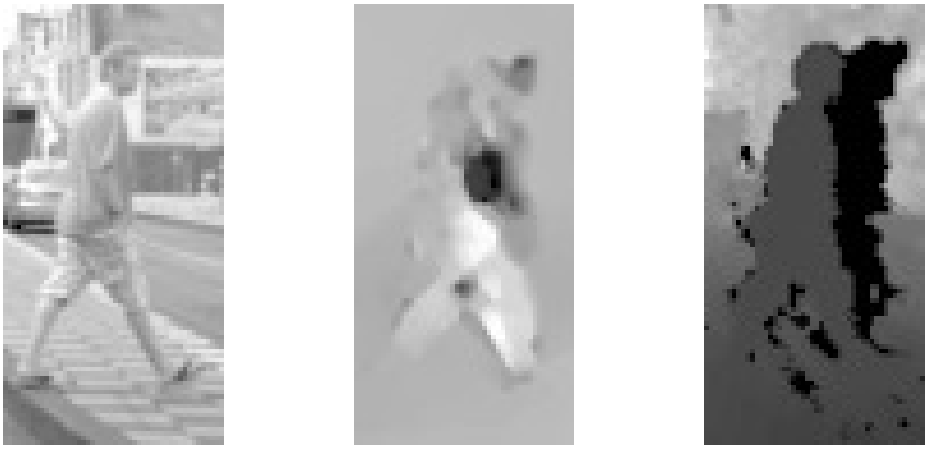
In the experiments we use Daimler pedestrian dataset [Enzweiler et al., 2010], which contains a collection of manually labelled bounding boxes containing pedestrians and non-pedestrians in images captured in an urban environment [Enzweiler et al., 2010]. Additional samples were created for each manually labelled pedestrian, by geometric jittering, while the non-pedestrian samples were obtained through a shape detection pre-processing. Dense stereo was calculated considering the semi-global matching algorithm [Hirschmuller, 2005], while dense optical flow using structure and motion-adaptive regularized flow [Wedel et al., 2009]. The dataset contains 48 x 96 px image crops in three modalities: intensity, depth and optical flow, which represents the reason for choosing this dataset. Figure 4.2 illustrates a pedestrian in each of these modalities.

The training set contains a total number of 84577 examples:

- 52112 – pedestrians
- 32465 – non-pedestrians

The testing set contains a total number of 41834 examples:

- 25608 – pedestrians



(a) Intensity (b) Flow (c) Depth

Figure 4.2: Pedestrian sample from Daimler dataset in (a) intensity, (b) flow and (c) depth images.

- 16235 – non-pedestrians

In our experiments we perform a fusion of features extracted by using KDs from three type of images: intensity, depth and flow. The features are extracted using the software from [Bo et al., 2010] and the learning is performed using Liblinear [Fan et al., 2008].

4.3.3.3.1 Evaluation measures

Designing classifiers for image recognition is a complex task, generally conducted by optimizing a single criterion: prediction accuracy. A such performance measure falls short of expectations when data are described by skewed class distributions or in the case of unbalanced training data. A solution to this problem is given by a more complex criterion utilised in the training phase of the classification: the Receiver Operating Characteristics (ROC) curve [Fawcett, 2006].

Some concepts must be utilised in order to define the ROC curve. Suppose that the input images and their characteristics are represented by a sample $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and their target classes $Y = \{y_1, y_2, \ldots, y_n\}$. Therefore, the classification algorithm must produce predictions as to the labels of those input samples.

The fraction of detected pedestrians out of the total actual pedestrians represents the true positive rate (TPR). The fraction of non-pedestrians classified as pedestrians out of the total actual non-pedestrians represents the false positive rate (FPR). In machine learning, the TPR is known recall (sensitivity), while the FPR is called the fall-out and can be computed as one minus specificity.

It is known that SVM is a scoring classifier. In order to obtain the corresponding labels, it is required to apply a threshold on those scores, tranforming the classifier into a discrete classifier. In the case of a single decision model, by varying the values of the threshold, all possible FP *vs.* TP rates are obtained. The ROC curve is obtained by representing

graphically the TP rates against the FP rates. Based on this graph it is possible to visualise the classification error rates per class, instead of just the general error of both classes [Fawcett, 2006].

Therefore, in order to measure the performance of our models, we plot the ROC curve, which shows the performance of the classifier when its threshold is varied. The ROC curve captures more information than the single accuracy measurement and facilitates the development of method which adapt to varying conditions [Fawcett, 2006]. Thus, this graph is a better objective for training classification algorithms than accuracy.

For comparing two models, we use the FPR at 90% detection rate, which is the reference value in the literature. Good performances are indicated by small values for FPR.

4.3.3.3.2 Parameter optimization

Since the performance on the classification process strongly depends on the parameters involved in both stages (feature extraction and learning), our approach involves an optimization phase dedicated to identify the best parameters of the decision model.

First of all, the parameters involved in the description computing are optimized. Taking into account that the images we work with are gray, we use the Gradient Kernel Descriptor. The orientation kernel and the position kernel are Gaussian kernels, as we concluded from the previous experiment that they achieve the best performance in conjunction with SVM classifier. In order to obtain the best representation for our particular images containing pedestrians and non-pedestrians, the parameters of the gradient kernel descriptor from Ren's implementation, *kdesdim* and *contrast*, have to be optimized on each type of image (intensity, depth and flow). The *kdesdim* parameter establishes the number of features that are extracted from each patch, whereas the gradient kernel usues the *contrast* parameter. These two parameters are optimized by using a cascade approach.

For each combination of parameters an SVM model is trained on a learning set and tested on a validation set. The best performance obtained on the validation set will indicate the best parameters of kernel descriptors. The optimization process is performed independently for each image modality (intensity, depth and flow).

After the optimization phase has been done, the optimal values of KD's parameters are utilised for determine the best image representation. These optimal characteristics are utilised by the SVM algorithm in order to construct a decision model by using the training data.

In order to optimize it's parameters, we extract from each type of image, KDs with varying *kdesdim* and *contrast* values.

- As the default value for *contrast* from Ren's implementation is 0.8, we search the optimum in the interval [0.65, 0.85] with step 0.01.
- For the *kdesdim* parameter we search in the interval [30, 70] with step 2.

- We mention that the boundaries of the search intervals have been empirically discovered through experiments.

The total number of features extracted by KD is *number of patches* * *kdesdim*. In a 48 x 96 px image with an 8 x 8 px KD grid, there are 55 patches, so the number of features ranges in the interval [1650, 3850]. Since our guide line is HOG descriptor [Dalal and Triggs, 2005] which extracts 1980 features, we search for *kdesdim* values that provide a number of features around this value.

For each combination of parameters an SVM model is trained on a learning set and tested on a validation set. The sets are obtained by dividing the training set presented in the previous section, in two subsets:

- Learning set: 2/3 from training set which contains: 34.741 pedestrians and 17.371 non-pedestrians
- Validation set: 1/3 from training set which contains: 17.371 pedestrians and 10.822 non-pedestrians

In the following we present the results achieved on the validation set, by varying *kdesdim* and *contrast* values. From space reasons, we present only the most relevant results. The parameters that minimize the FPR obtained at 90% detection rate (the ones in bold) are chosen for each type of image:

- $kdesdim = 54$ and $contrast = 0.78$ for intensity images
- $kdesdim = 54$ and $contrast = 0.76$ for depth images
- $kdesdim = 56$ and $contrast = 0.69$ for flow images

The various FPRs obtained by different KD's parameters and presented in Tables 4.4-4.9 indicate that the performance of the classification process in influenced by the KD's parameters and the optimization process is able to identify the values that are able of improving the pedestrian recognition.

4.3.3.3.3 Single-modality classification

In the previous section we presented the optimal KD parameters for each image modality. The values were discovered through an optimization process, which was performed on the training set (divided in learning and validation subsets). Using these parameters we extract KD features for each type of image from the entire training set and then we learn an SVM model.

The results obtained on the testing set for each of the three classifiers are presented in Figures 4.3 and 4.5. We can notice that the best results, reflected by the smaller FPR, are obtained on intensity images.

Kdesdim	FPR
40	0.009056
42	0.010164
44	0.010164
46	0.009148
48	0.009425
50	0.009887
52	0.009518
54	**0.008963**
56	0.009056
58	0.009056
60	0.008963

Table 4.4: FPR at 90% on **intensity** images for contrast=0.8 and different kdesdim values on validation set

Contrast	FPR
0.74	0.008224
0.75	0.008224
0.76	0.008409
0.77	0.008409
0.78	**0.008132**
0.79	0.008963
0.8	0.008963
0.81	0.008316
0.82	0.008316
0.83	0.008409
0.84	0.008871

Table 4.5: FPR at 90% on **intensity** images for kdesdim=54 and different contrast values on validation set

Kdesdim	FPR
40	0.026797
42	0.026797
44	0.026982
46	0.026797
48	0.026335
50	0.026335
52	0.025873
54	**0.025781**
56	0.026150
58	0.026243
60	0.025873

Table 4.6: FPR at 90% on **depth** images for contrast=0.8 and different kdesdim values on validation set

Contrast	FPR
0.73	0.024949
0.74	0.024949
0.75	0.025042
0.76	**0.024395**
0.77	0.025504
0.78	0.025504
0.79	0.025504
0.8	0.025781
0.81	0.026612
0.82	0.026520
0.83	0.026150

Table 4.7: FPR at 90% on **depth** images for *kdesdim*=54 and different contrast values on validation set

In order to analyse the results statistically we have used a 95% confidence interval for the FPR. Thus, Figure 4.3 depicts the confidence intervals obtained on the test set and reinforces our conclusion (related to best results obtained on intensity images) from a statistical point of view, also.

Kdesdim	FPR
40	0.055165
42	0.051469
44	0.052208
46	0.051377
48	0.050545
50	0.052208
52	0.051377
54	0.051654
56	**0.050083**
58	0.050083
60	0.048697

Table 4.8: FPR at 90% on **flow** images for contrast=0.8 and different *kdesdim* values on validation set

Contrast	FPR
0.66	0.048327
0.67	0.048512
0.68	0.048327
0.69	**0.048235**
0.7	0.048790
0.71	0.048512
0.72	0.049067
0.73	0.048882
0.74	0.048974
0.75	0.049436
0.76	0.049344

Table 4.9: FPR at 90% on **flow** images for *kdesdim*=56 and different contrast values on validation set

4.3.3.3.4 Multi-modality classification

After having analysed the effect of each modality independently for KD features, we now evaluate the effect of using modality fusion. The fusion is performed by joining the features extracted from intensity, depth and flow images. An SVM model is learned on the obtained feature vectors.

The results obtained on the testing set for the single-modality and the joint features classifiers are presented in Figure 4.5. We can observe an improvement when fusing the information provided by different modalities. Furthermore, the statistical analysis, based on confidence intervals (that can be observed in Figure 4.4), consolidates the improvement

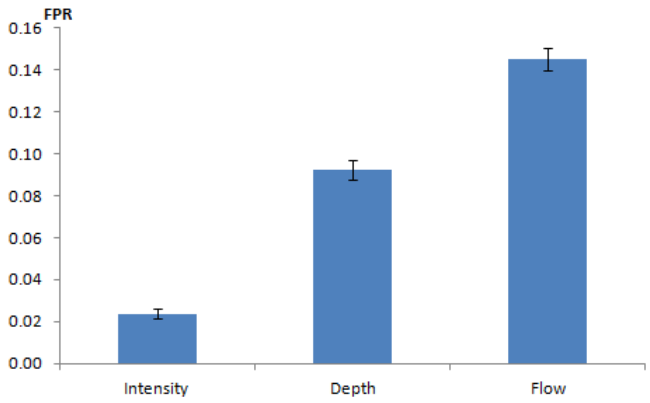


Figure 4.3: Single-modality classification performance on testing set. FPR at 90% detection rate and the corresponding confidence intervals.

achieved by the fusion of all three modalities over the single modality classifiers.

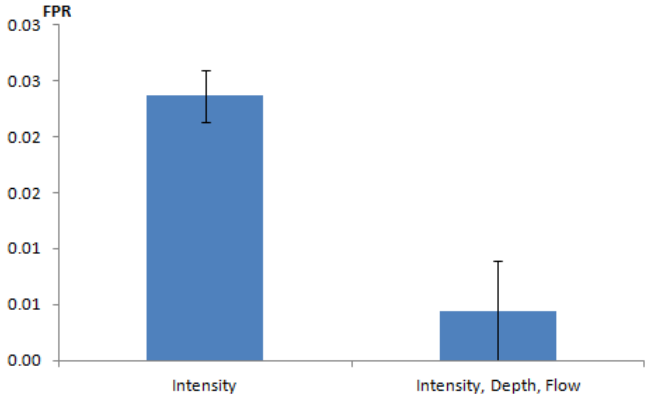


Figure 4.4: Single-modality *vs.* multi-modality classification performance on testing set. FPR at 90% detection rate and the corresponding confidence intervals.

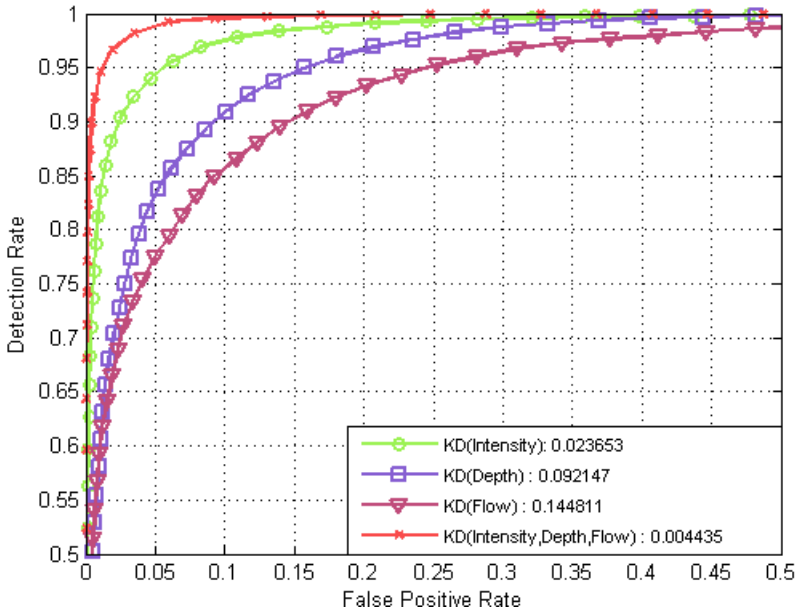


Figure 4.5: Single-modality *vs.* multi-modality classification performance on testing set, using KD features. FPR at 90% detection rate.

4.3.3.3.5 KDs vs. HOG in pedestrian recognition

Kernel Descriptors [Bo et al., 2010] are considered a generalization of HOG features [Dalal and Triggs, 2005], which are specific type of match kernels over patches.

In the following we aim to compare the performances of KDs and HOG on single and multi-modality classification. Therefore, in Figure 4.6 are presented the results obtained on Daimler data set using HOG features. By comparing the results achieved by KDs (Figure 4.5), with the ones obtained by HOG (Figure 4.6), we can conclude that KDs do not reach

the performances of HOG. This could be caused by the dimensionality reduction step that is performed within KDs using kernel principal component analysis. Further work could be carried out in order to optimize this step.

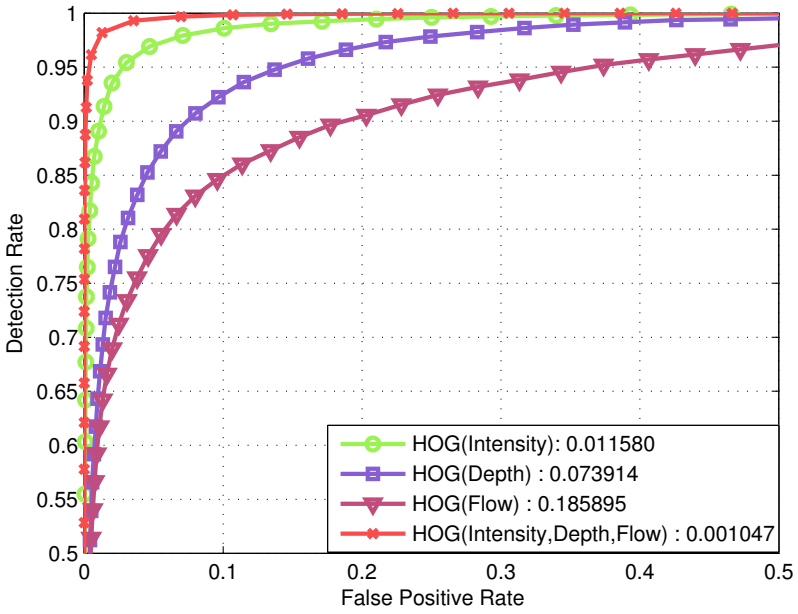


Figure 4.6: Single-modality vs. multi-modality classification performance on testing set, using HOG features. FPR at 90% detection rate.

4.3.3.4 Conclusions

We proposed an approach in which each image was represented by using Gradient Kernel Descriptor, whose parameters (*kdesdim* and *contrast*) were optimized by using a validation set of data. The extracted optimized features were incorporated finally into an linear SVM classifier in order to construct a decision model that can be utilised in order to label unseen images. In addition to the parameter optimization, a feature fusion was investigated by considering three modalities associated to an image: intensity, depth and flow.

We have studied how the multi-modality (intensity, depth, motion) usage vs. single-modality usage influence the results of pedestrian classification. Optimized KD-based features have different performances across modalities. The results obtained by using a single modality indicate that intensity has the best performance on tested dataset, followed by depth and flow. Nevertheless, the fusion of all modalities achieves to the most performant pedestrian classifier.

As further work we plan to perform an optimization of the Gaussian kernel used by Gradient Kernel Descriptor, also to investigate other types of fusion and to consider other datasets.

4.4 Pedestrian detection using dynamic modalities

It was proved that the fusion of features extracted from multi-modality images like intensity, depth and flow, improves the performance of pedestrian recognition (see Section 4.3.3). We have also proved it for KD features in Section 4.3.3.3. However, in some conditions, pedestrians are recognized only in a particular modality. For example, a pedestrian in an low contrast image is more difficult to be recognized in intensity than in other modalities.

In this section we propose two machine learning based algorithms that are able to dynamically select the most discriminative modalities for each image sample, representing a ROI within an image frame, that will be further used in the classification process.

4.4.1 Modality pertinence analysis

It is well known that the use of information from multi-modality images leads to a significant boost in the performance of a pedestrian recognition system. One of the best approaches so far is to concatenate the features extracted from each modality and build a large feature vector, but it requires strong camera calibration settings and non-discriminative modalities could lead to missclassification of some particular images.

In the following we present a comparative analysis of the classification performance of single and multi-modality classifiers, using KD features. As we can see in the Figure 4.7, illustrating the percent of correctly classified images using single modality classifiers: intensity, depth or flow, almost a quarter from the total number of images can be correctly classified using single modality classifiers in one or two modalities, but not in all three modalities. We have to mention that in this figure Intensity, Depth does not refer to a fusion of the two modalities, the interpretation being that 9.4% of the images were correctly recognized both in the intensity based and depth based classifiers. Moreover, we can see in Figure 4.8 that there are situations in which the classifier obtained by concatenating all the modalities is not able to classify correctly the images, while single or bi-modality classifiers do. In this figure, ID refers to the classifier that concatenates intensity and depth. This can be explained by the negative impact of non-discriminative modalities within a feature fusion.

We believe that the relative pertinence of the modalities is not a static parameter that could be estimated once for all, but varies dynamically from an image frame to another, from a bounding box to another, according to the environmental conditions (illumination, occlusions, cluttered backgrounds). Starting from this observation, we aim to design a dynamic model that, based on the features extracted from an image in a given modality representing a ROI within an image frame, is able to decide if that modality is suitable for the classification of that image. Our goal is to dynamically include the modalities in the fusion, frame by frame, in order to reduce complexity and improve performance. In the following, we shortly review the main fusion schemes from the literature, together with the disadvantages that we aim to overcome in our dynamic models.

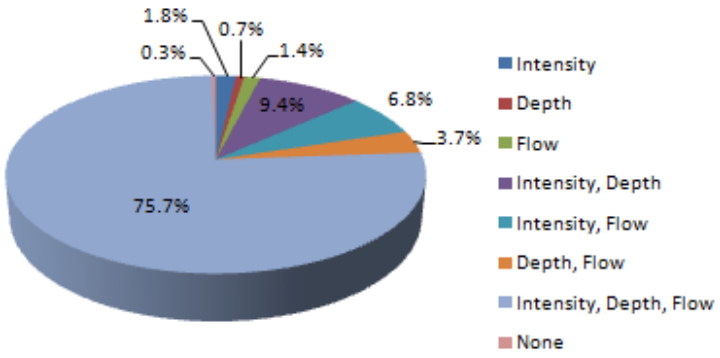


Figure 4.7: Percent of correctly classified images using single-modality classifiers on KD features

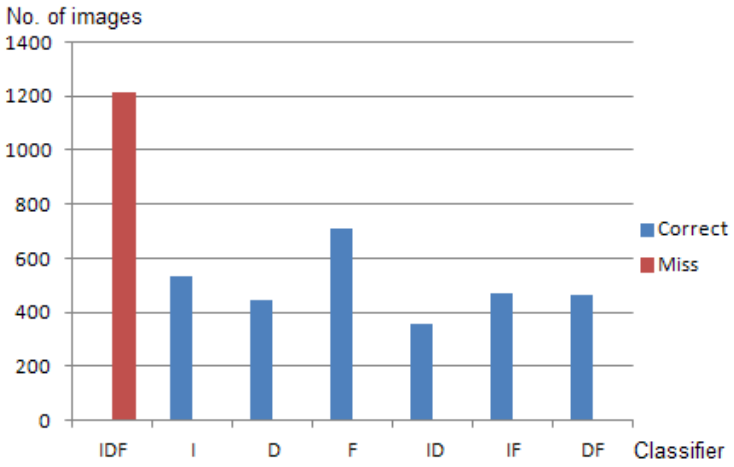


Figure 4.8: Number of images missclassified by the IDF classifier and correctly classified by single or bi-modality classifiers

4.4.2 Fusion models

Nowadays, developing complex systems requires the integration of several sources that provide complementary information, which can be achieved through a fusion process. In the literature, there are two classical fusion schemes applied for object classification:

- an early fusion, performed prior to classification, at the low-level of pixels (data fusion) or features (feature level, see Figure 4.10) and
- a late fusion, posterior to classification, at the high-level of matching scores (see Figure 4.11).

Since one of our dynamic models focuses on fusion, in the following we briefly describe one from each category, that has been already used in pedestrian recognition, together with some drawbacks of these approaches that we want to overcome in the dynamic models that

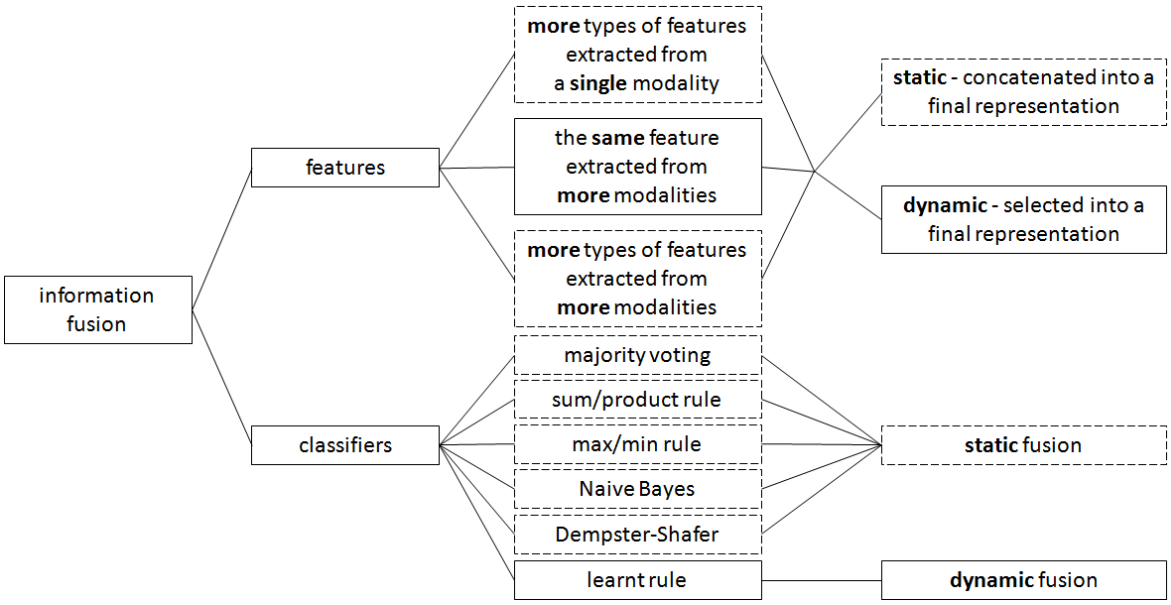


Figure 4.9: Information fusion in object recognition - possible taxonomy. Solid line are used in order to emphasize the fusion elements that our model approaches, while the doted lines denote other models from literature.

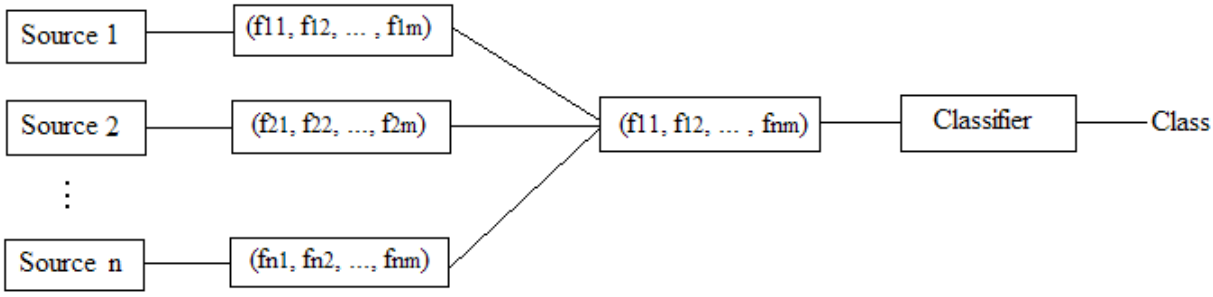


Figure 4.10: Early fusion general scheme - feature concatenation

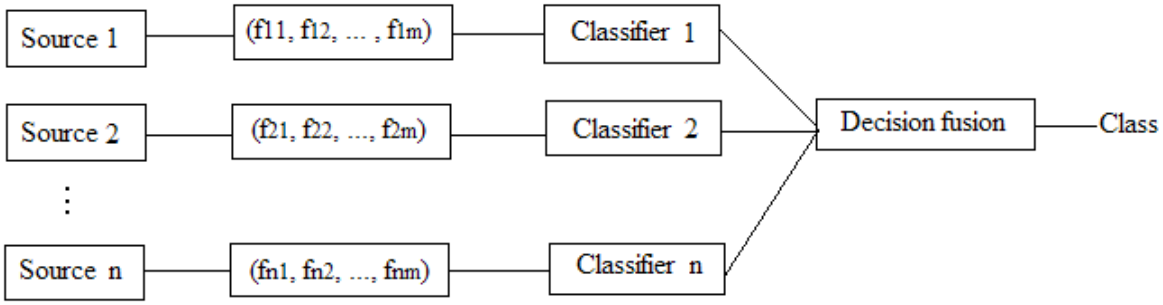


Figure 4.11: Late fusion general scheme - decision fusion

we propose. A more detailed classification of information fusion in object recognition can be seen in Figure 4.9.

4.4.2.1 Fusion by feature concatenation

Feature concatenation [Wang et al., 2009] is a low-level approach, which involves the construction of a joint feature space, that will forward feed the classifier. In the context of multi-modality pedestrian classification, the feature vector will concatenate the features extracted from all the modalities, e.g. intensity, depth and flow.

The drawback to this approach that we want to overcome in our dynamic models is that it cannot adapt to varying environmental conditions, which is crucial for a robust pedestrian detection system. This is caused by the fact that it uses the same modalities in all conditions, without being able to choose the most pertinent ones depending on the situation. Moreover, it is very sensitive to errors caused by the miss-calibration of on-board cameras or a miss-functioning of one camera among them. Even if the cameras are mounted on a fixed platform on the vehicle, passing over hills or holes could miss-calibrate them and a new calibration process would be required online.

4.4.2.2 Matching scores' fusion

There are several types of matching scores' fusion like: majority vote, sum, product, minimum, maximum, native Bayes or Demster-Shafer. Majority vote fusion [Oliveira et al., 2010] is a high-level approach, which involves a fusion of classification scores. In the context of multi-modality classification, the classifiers in each modality (intensity, depth and flow) give a vote (pedestrian/non-pedestrian). The final class (pedestrian/non-pedestrian) is the one that receives most of the votes.

The drawback to this approach in case of the fusion among intensity, depth and flow is a large number of false positives, since the FP rate increases from intensity to depth and flow (see [Enzweiler et al., 2010]). On the other hand, some pedestrians are recognized only in a particular modality, for e.g. in flow. For instance, if the classifiers in intensity and depth decide that the image contains a non-pedestrian, the majority vote will lead to a missclassification. Because some pedestrians can be discriminated only in a particular modality, the problem of missclassification will occur also in other types of fusion that weight the importance of a modality in the same manner for all images (bounding boxes within a given frame), in a static way, or use the confidence indicator of the classifier in each modality.

4.4.3 Modality selection component

The modality selection component represents the core of the dynamic models that we propose, being responsible of determining the most suitable modalities for classifying an image. Its design is based on a hybrid approach, which involves two types of classifiers: a pedestrian classifier, capable to distinguish between pedestrians and non-pedestrians, and a

modality pertinence classifier that decides whether a modality is suitable or not for a given image.

The main idea of the modality pertinence classifier is to learn from the experience of the pedestrian classifier in a particular modality if that modality is suitable or not for the classification of an image, representing a bounding box within a frame. We consider a modality suitable for an image, if the pedestrian classifier was able to correctly classify the image using that modality on a validation set.

Taking into account that the modality classifier learns from the experiences of the pedestrian classifier in a modality, we can intuitively assume that its performance is significantly influenced by the number of experiences, therefore we applied and further compared two learning methods.

4.4.3.1 Learning on a subset

In order to train the modality pertinence classifier we first need to train a pedestrian classifier, as its output will be considered in the modality pertinence classifier.

In the first step, we train a pedestrian classifier for each modality: intensity, depth and flow on the learning set, representing 2/3 from the training set, and applied the model on the validation set, representing 1/3 from the training set. The input of the pedestrian classifier consists of KD features extracted from images in intensity, depth or flow and the output is a class label: "pedestrian (P)" or "non-pedestrian (NP)".

In the second step, we train a modality pertinence classifier for each modality: intensity, depth and flow on the validation set. The input of the pertinence classifier consists of KD features extracted from intensity, depth or flow images and the output is a class label: "suitable (S)" if the class predicted by the pedestrian classifier is the same with the actual class, and "not-suitable (NS)" otherwise.

4.4.3.2 Learning with cross validation

This approach is meant to enlarge the sets used for training the pedestrian and modality pertinence classifiers, by performing cross validation (CV) with three folds. In the following we are going to detail this process.

We split the training set into three folds: $fold_A$, $fold_B$ and $fold_C$ and perform the following steps:

- learn a pedestrian classifier on $fold_A$ and $fold_B$ and apply on $fold_C$
- learn a pedestrian classifier on $fold_A$ and $fold_C$ and apply on $fold_B$
- learn a pedestrian classifier on $fold_B$ and $fold_C$ and apply on $fold_A$

At the end of this process, we will have the predicted classes for all the examples from the training data, which will be used for building the modality pertinence classifier, in the previously presented manner.

4.4.4 Dynamic modality selection

In this section, we introduce the first dynamic model that we propose for pedestrian recognition called Dynamic Modality Selection (DMS), which is able to dynamically select for each image, representing a bounding box in a frame, the most relevant image modality among intensity, depth and flow.

DMS uses the modality pertinence classifier presented in section 4.4.3 in order to find the modalities suitable for the current image and chooses one among them, according to a modality relative confidence indicator.

The steps performed by DMS are the following:

- check for all modalities if they are suitable for a given image, using the modality pertinence classifiers
- if there is more than one modality supposed to be suitable for the given image, choose the most confident one, which satisfies $Max\{|P_i^P - P_i^{NP}|, i \in \{I, D, F\}\}$, where P^P represents the probability of being a pedestrian, while P^{NP} the probability of being a non-pedestrian (modality relative confidence indicator)
- classify the image into *pedestrian* or *non-pedestrian*, using the most confident from the suitable modalities discovered in the first step, or the most confident among all modalities if none of them is considered suitable

In Figure 4.12 we present the architecture of DMS.

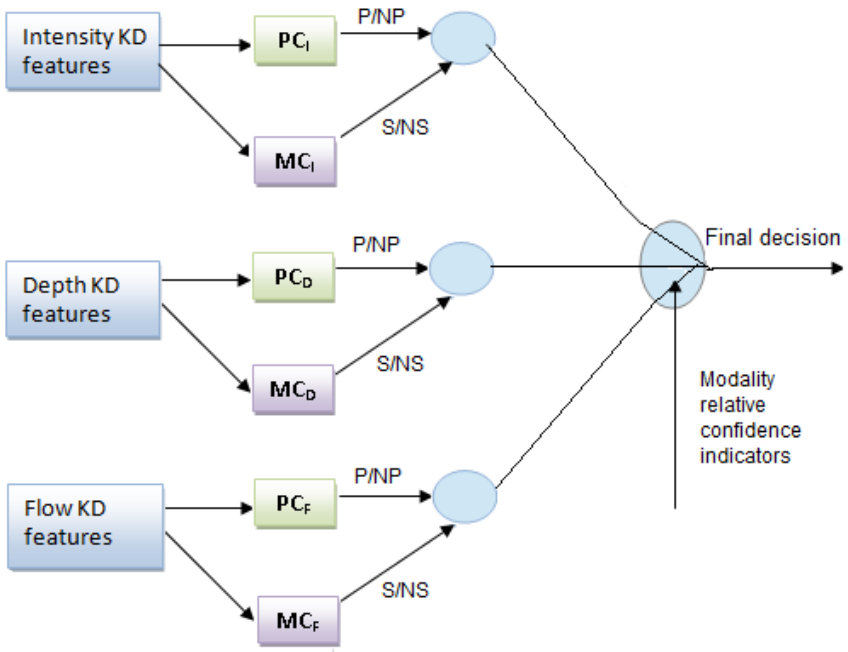


Figure 4.12: The architecture of DMS

4.4.4.1 Experimental evaluation

We conducted two experiments, in each one using a different mode for learning the modality classifiers. In the first one we trained the modality classifier on a subset of the training set (the validation set), while in the second one we performed the training on the whole training set, by cross validation.

The context of the experiments is the same as in the ones presented in Section 2: Daimler dataset, gradient KDs extracted from intensity, depth and flow images and support vector machines with Liblinear library.

In the training phase, which is performed offline, we train:

- three modality pertinence classifiers, one for each modality: MC_I, MC_D and MC_F on the training set (see section 4.4.3)
- three pedestrian classifiers, one for each modality: PC_I, PC_D and PC_F on the training set.

In the testing phase, which is performed online, for each image in intensity, depth and flow:

- we apply the modality pertinence classifier for each modality, resulting a set of suitable modalities, e.g. $\{I, D\}$
- we select one modality according to the modality relative confidence indicator, e.g. I
- classify with the classifier corresponding to the retained modality: e.g. PC_I

The modality selection component plays a significant role in the DMS classifier. As you can see in Figure 4.13 the performance of DMS increases together with the increase of the number of examples used for training the modality selection classifier, since the DMS trained with CV outperforms the DMS trained on a subset.

In Figure 4.14 we present the results obtained by the DMS classifier on the testing set, in comparison with the results achieved by the single-modality classifiers (intensity, depth and flow). In Figure 4.15 we compare our results to the ones obtained by the fusion models presented in section 4.4.2: the classifier trained on the concatenated modalities, evaluated for KD features in [Sirbu et al., 2014b], and the majority vote classifier.

We can notice that DMS achieves better performance than the single-modality classifiers and the majority vote classifier, and is very close to the performance obtained by concatenating the three modalities.

4.4.4.2 Conclusions

We proposed a dynamic single-modality selection approach which is able to select the most suitable modality to classify an image.

The advantages of our method over the joint modalities approach consist of:

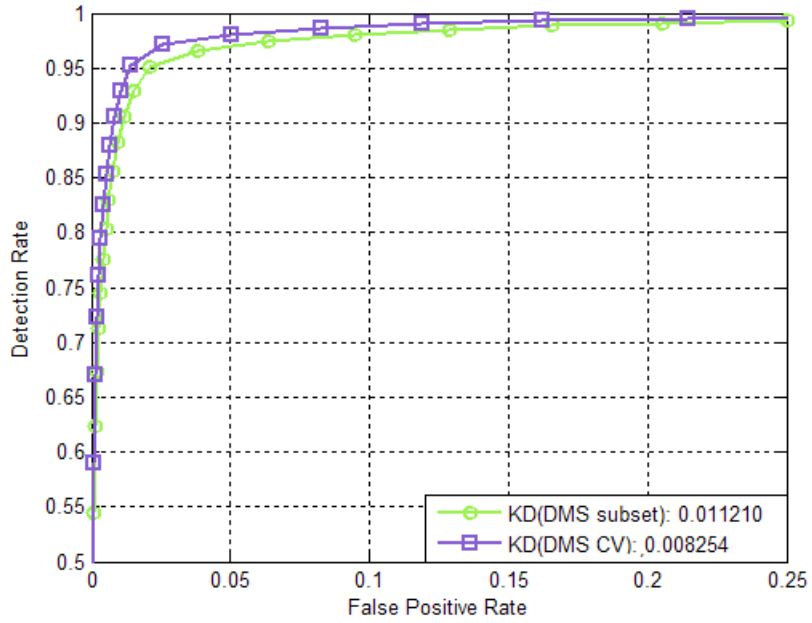


Figure 4.13: Classification performance for DMS with subset vs. cross validation modality selection, using KD features . FPR at 90% detection rate.

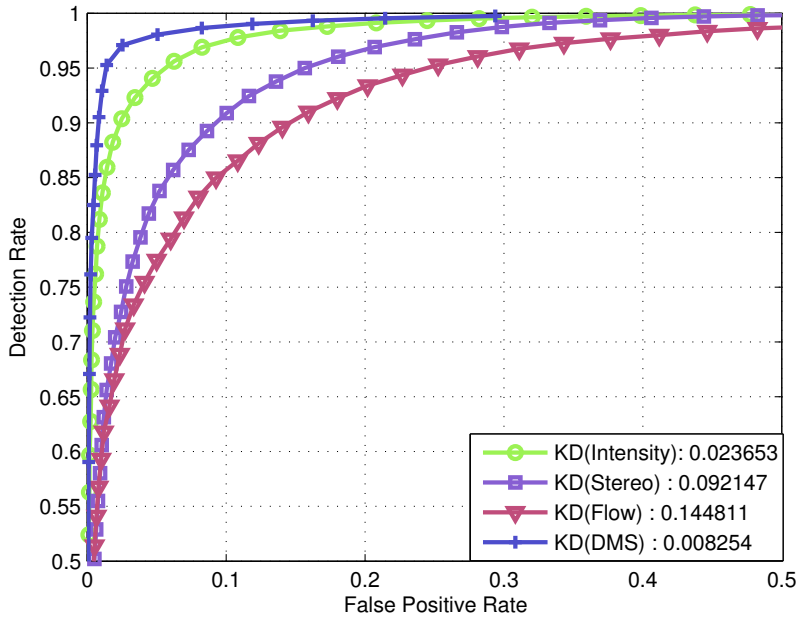


Figure 4.14: Single modality vs. DMS classification performance on testing set, using KD features. FPR at 90% detection rate.

- *lower complexity* – the amount of time needed for training individual classifiers is less than the one needed to train a classifier on a large feature vector, obtained by joining all modalities

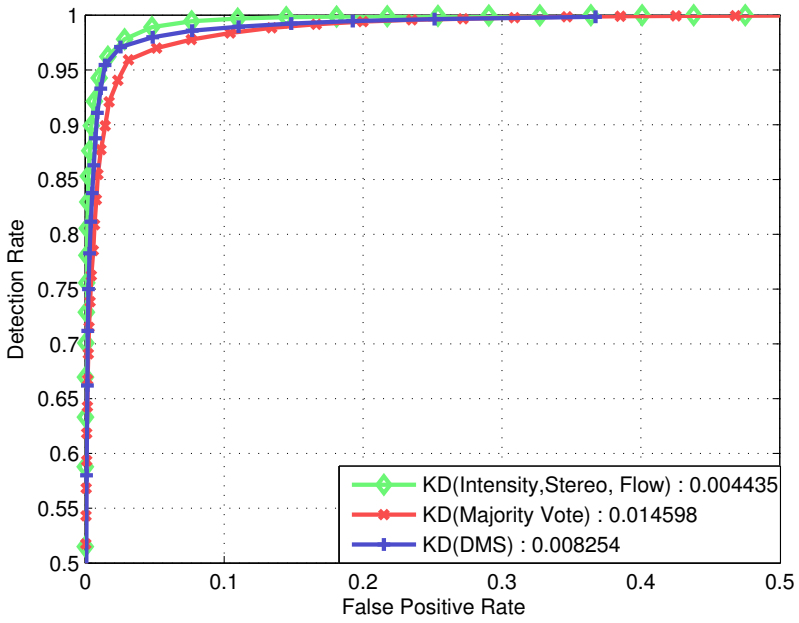


Figure 4.15: Modality fusion vs. DMS classification performance on testing set, using KD features. FPR at 90% detection rate.

- the individual classifiers can be *trained independently* on different datasets

The results achieved by the joint modalities classifier are slightly better than the ones obtained by DMS, but taking into account that its performance is strongly influenced by the amount of experiences of the pedestrian classifier in each modality (number of training examples for the modality classifiers) from which it could achieve a performance boost, we find DMS a promising alternative for the joint modalities classifier.

In the next section, we propose a model which combines the modality selection and the joint modalities approaches for obtaining a benefit from the advantages of both models.

4.4.5 Dynamic modality fusion

In this section we present the second dynamic model for pedestrian recognition, Dynamic Modality Fusion (DMF), which is able to dynamically determine the most suitable modalities for a given image and perform a fusion of the features extracted from them.

DMF also uses the modality classifier presented in section 4.4.3 in order to find the modalities suitable for the current image, but unlike DMS which selects only one modality, it takes into account all the modalities which were considered suitable and further uses them in the classification process.

The steps performed by DMF are the following:

- check for all modalities if they are suitable for the given image, using the modality classifiers

- if in the first step we get more than one modality that is supposed to be suitable for the given image, join the features extracted from all suitable modalities
- if no modality is considered suitable, join the features extracted from all modalities
- classify the image into *pedestrian* or *non-pedestrian*, using the modalities discovered in the first step

In Figure 4.16 we present the architecture of the DMF model.

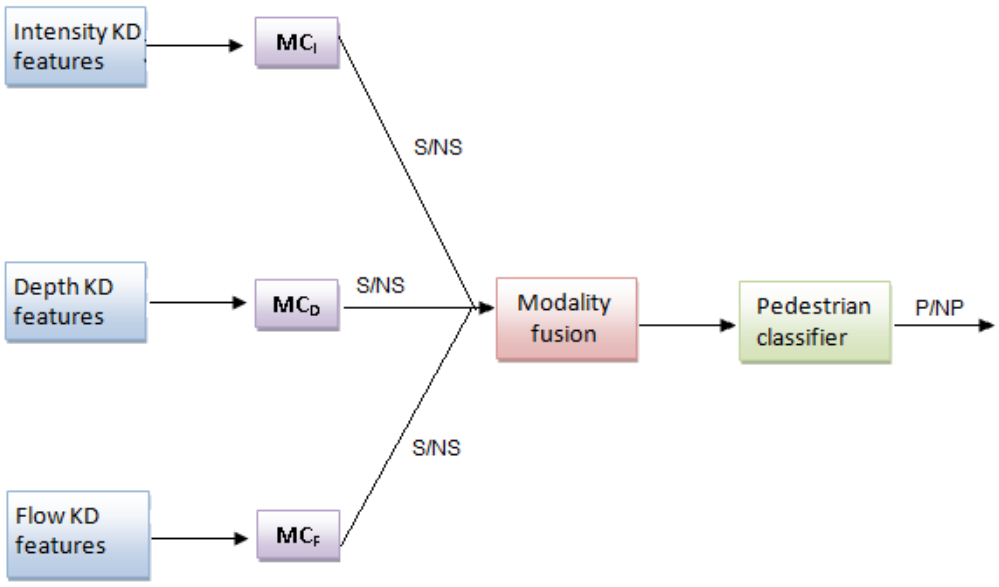


Figure 4.16: The architecture of DMF

4.4.5.1 Experimental evaluation

We conducted two experiments, in each one using a different mode for learning the modality classifiers. In the first one we trained the modality classifier on a subset of the training set (on a half of the training set), while in the second one we performed the training on the whole training set, by Cross Validation.

The context of the experiments is the same as in the ones presented in Section 2: Daimler dataset, gradient KDs extracted from intensity, depth and flow images and support vector machines with Liblinear library.

In the training phase, we trained:

- three modality classification models, one for each modality: MC_I, MC_F and MC_D, on half of the training set in the first experiment and on the whole training set in the second one

- seven pedestrian-non-pedestrian classification models one for each combination of modalities: PC_I, PC_D, PC_F, PC_{ID}, PC_{IF}, PC_{DF} and PC_{IDF}, on the training set.

In the testing phase, for each image in intensity, depth and flow:

- we applied the modality classifier for each modality, resulting a set of suitable modalities, e.g. $\{I, D\}$
- we join the KD features extracted from the suitable modalities, e.g $KD(I), KD(D)$
- classify with the classifier corresponding to the selected modalities: e.g. P_{ID}

The modality selection component plays a very important role also in DMF classifier. As you can see in Figure 4.17 the performance of DMF increases together with the increase of the number of examples used for training the modality selection classifier. The performance boost of CV approach over learning on a subset is less notable than for DMS due to the fact that in this experiment the error rate is very small and that we started from more training examples (1/2 of the training set) and added less new training examples.

The aim of the first experiment is to compare the performance of our dynamic model with those of classifiers based on a single modality. Both approaches use the KD extracted from images. Thus, in Figure 4.18 we present the results obtained by the DMF classifier on the testing set, in comparison with the results achieved by the single-modality classifiers (intensity, depth and flow).

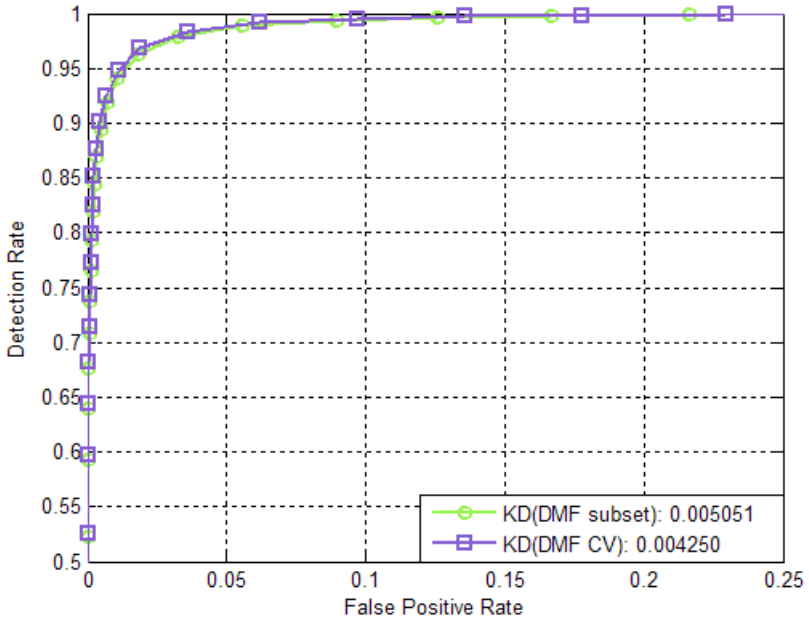


Figure 4.17: Classification performance for DMS with subset vs. cross validation modality selection, using KD features. FPR at 90% detection rate.

Then, a second experiment is dedicated to compare the classification results obtained by the proposed approach to the ones obtained by the fusion models presented in section 4.4.2.

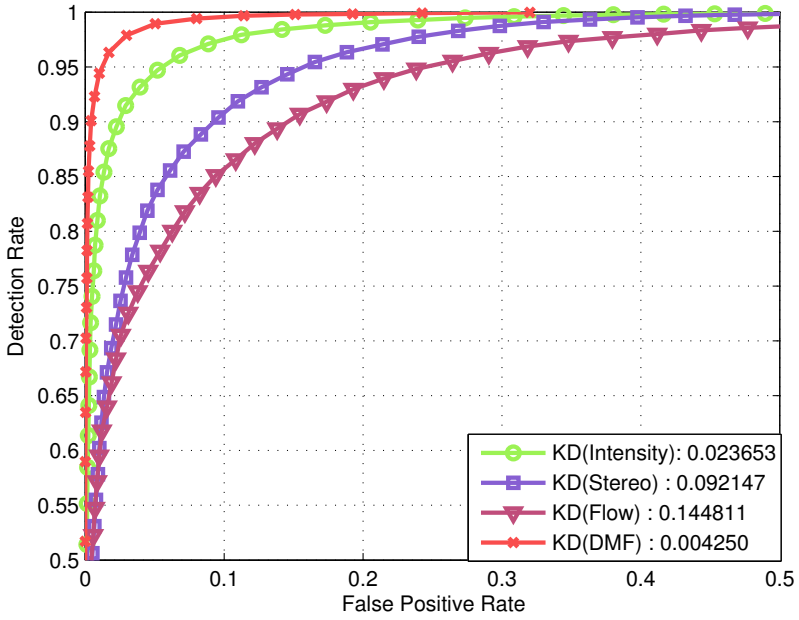


Figure 4.18: Single modality vs. DMF classification performance using KD features. FPR at 90%.

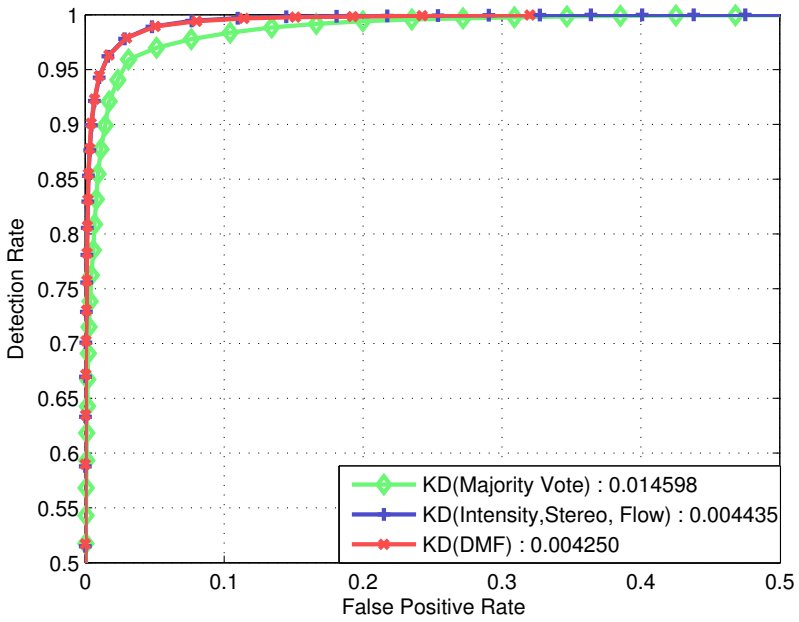


Figure 4.19: Static modality fusion vs. DMF classification performance on testing set, using KD features. FPR at 90%.

Again, all the compared classifiers use the KD features extracted from images. There are also other fusion models described in the literature [Enzweiler and Gavrila, 2011], but since they are based on other image descriptors, a fair comparison with our models seems impossible.

In Figure 4.19 the DMF model is compared to static fusion approaches where the classifier is trained on the concatenated modalities, evaluated for KD features in [Sirbu et al., 2014b], and for a majority vote classifier. We can notice that DMF outperforms both models, but its improvement over the joint modalities classifier is rather small. In the next section we are going to present some perspectives of improvement of the DMF.

4.4.5.2 Conclusions

We proposed a dynamic multi-modality fusion approach which is able to select the most suitable modalities to classify an image and join the features extracted from them.

The advantage of DMF over the joint modality approach is that is less prone to the errors caused by non pertinent modalities, while the advantage over the single modality classifiers rises from the situations when the individual modality classifiers cannot discriminate correctly between pedestrian and non-pedestrians, unless they join their features. Considering all the aspects mentioned above, we find DMF model, a promising approach for multi-modality pedestrian recognition.

Further work could be done to investigate methods to improve the modality selection component and also to apply DMF on other datasets. Some possible directions of improvement could be the integration of image based modality-pertinence indicators, a deeper optimization of kernel descriptors (the Gaussian kernels from Gradient Kernel Descriptor and the dimensionality reduction step) or the addition of other features like Shape Kernel Descriptors.

4.4.6 System architecture

In this section we present an overview on the software that we developed in order to train and test our dynamic models: Dynamic Modality Selection and Dynamic Modality Fusion. We used a layered architecture consisting of three layers: presentation layer, business layer and data layer (see Figure 4.20), which facilitates the extension of the system with other modules. We have to mention that this software was built for experimental purposes and includes only the feature extraction, classification and evaluation modules, but can be easily extended by adding a detection module.

In the following we are presenting the most important packages and classes of our system by layers, as shown in Figures 4.22 and 4.21. The implementation of the system implies other classes and methods that are not presented in the diagrams.

4.4.6.1 Business layer

The main packages of the business layer are the *classification* package and the *evaluation* package.

4.4.6.1.1 Classification package

This package contains the interface *IDynamicModality* and two implementation classes *DynamicModalitySelection* and *DynamicModalityFusion*.

The interface *IDynamicModality* contains methods for training the pedestrian classifiers and the modality pertinence classifiers and for predicting/testing.

The method *trainPedestrianClassifiers* receives as input the training files with intensity, depth, respectively flow features in CSV format and returns a list of SVM models, one for each image modality.

The method *trainModalityPertinenceClassifiers* receives as input the training files with intensity, depth, respectively flow features in CSV format and returns a list of SVM models, one for each image modality. There are several steps performed in this method. First, the training set is split into three folds, then a pedestrian classifier is learned on each combination of two folds and tested on the remaining one. Having the predicted values for the whole training set, they are compared with the actual labels. If the predicted label is the same with the actual one, an example containing the KD features and *suitable* label will be added to the modality pertinence classifier's training file, otherwise and example with *notSuitable* will be added. After having the training file, an SVM model is built. This process is performed for all image modalities.

The method *predict* receives as input the test files with intensity, depth, respectively flow features in CSV format, the modality pertinence classifiers and the pedestrian classifiers and the name of the file that will contain the SVM output and classifies the examples from the test files. The format of the output file is: *predictedClass probabilityPedestrian*.

There are two implementation classes for this interface: *DynamicModalitySelection* and *DynamicModalityFusion*, which have custom implementations for the *trainPedestrianClassifier* and *predict* methods.

In the *trainPedestrianClassifier* method from the the class *DynamicModalitySelection*, three SVM models are built, one for each modality, while in the method belonging to *DynamicModalityFusion* class seven models are built, one for each combination of modalities.

In the *predict* method from the class *DynamicModalitySelection*, a check is performed for all modalities in order to decide if they are suitable, using the modality pertinence classifiers. If they are, classification into *pedestrian* or *non-pedestrian* will be performed, using the pedestrian classifier corresponding to the chosen modality. If there is more than one modality considered suitable, the most confident one will be chosen and if none of them is considered suitable, the most confident among all modalities will be chosen.

In the the *predict* method from the class *DynamicModalityFusion* the suitable modalities are also discovered using the modality pertinence classifiers. If there is more than one modality considered suitable, their features are concatenated, then classification into *pedestrian* or *non-pedestrian* is performed, according to the pedestrian classifier that corresponds to that combination of modalities.

4.4.6.1.2 Evaluation package

This package contains the classes *Evaluation* and *EvaluationMeasure* and the enumeration *EvaluationMeasureType*.

The class *Evaluation* contains a method *evaluationPerformance* which receives as input the output file with the SVM predictions and a file with the actual class labels, each label on a new line and returns a list of evaluation measures: accuracy, F-measure, recall and precision. The *displayROC* method receives the same parameters and displays the ROC curve.

4.4.6.2 Data layer

The most important packages of the data layer are the *feature extraction* and the *data* package.

4.4.6.2.1 Feature extraction package

This package contains the interface *IFeatureExtraction* and its implementation *KDFeatureExtraction*. The interface *IFeatureExtraction* contains a method *extractFeatures* that is responsible for extracting features from a given image. The method receives as parameter the path of the image and returns a list of float values, corresponding to the features extracted. The class *KDFeatureExtraction* implements the *FeatureExtraction* interface for KD features.

4.4.6.2.2 Data package

This package contains the class *DataPreprocessor* which has a method *extractKDFeaturesToCSV* responsible for extracting KD features from a folder with images and writing them on a CSV file. This method is called for the folders with training and test images from the pedestrian dataset, for positive and negative examples.

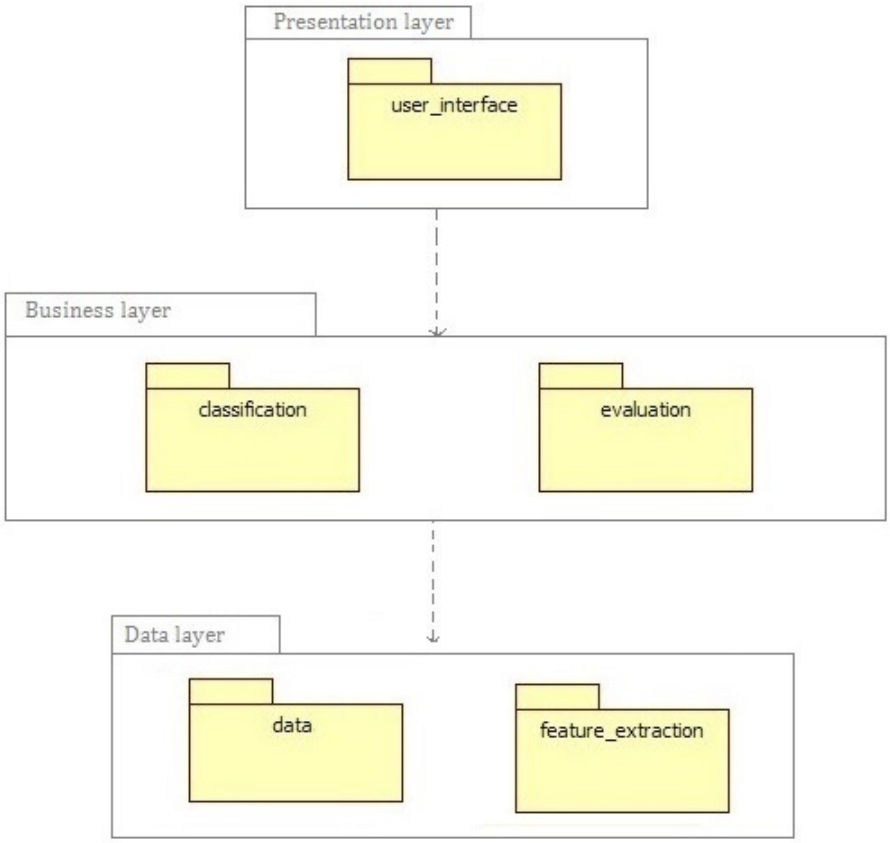


Figure 4.20: Highlevel system architecture

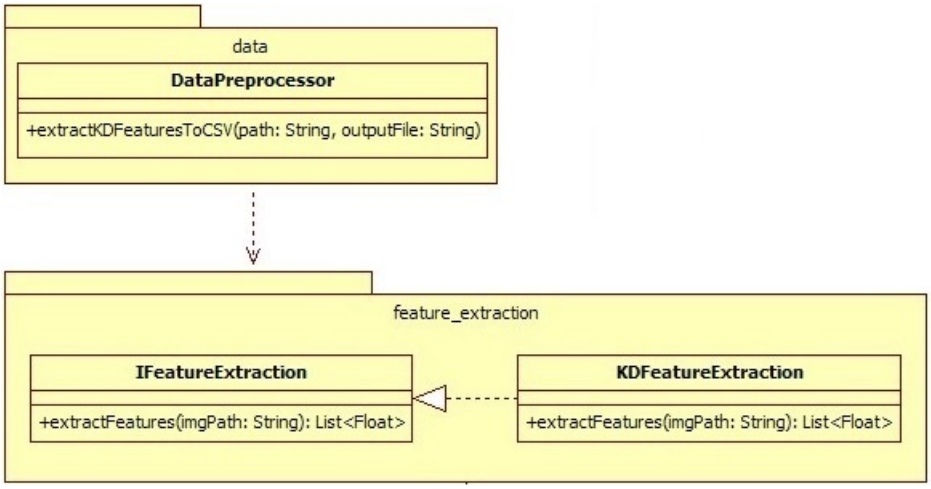


Figure 4.21: Simplified class diagram of the data layer

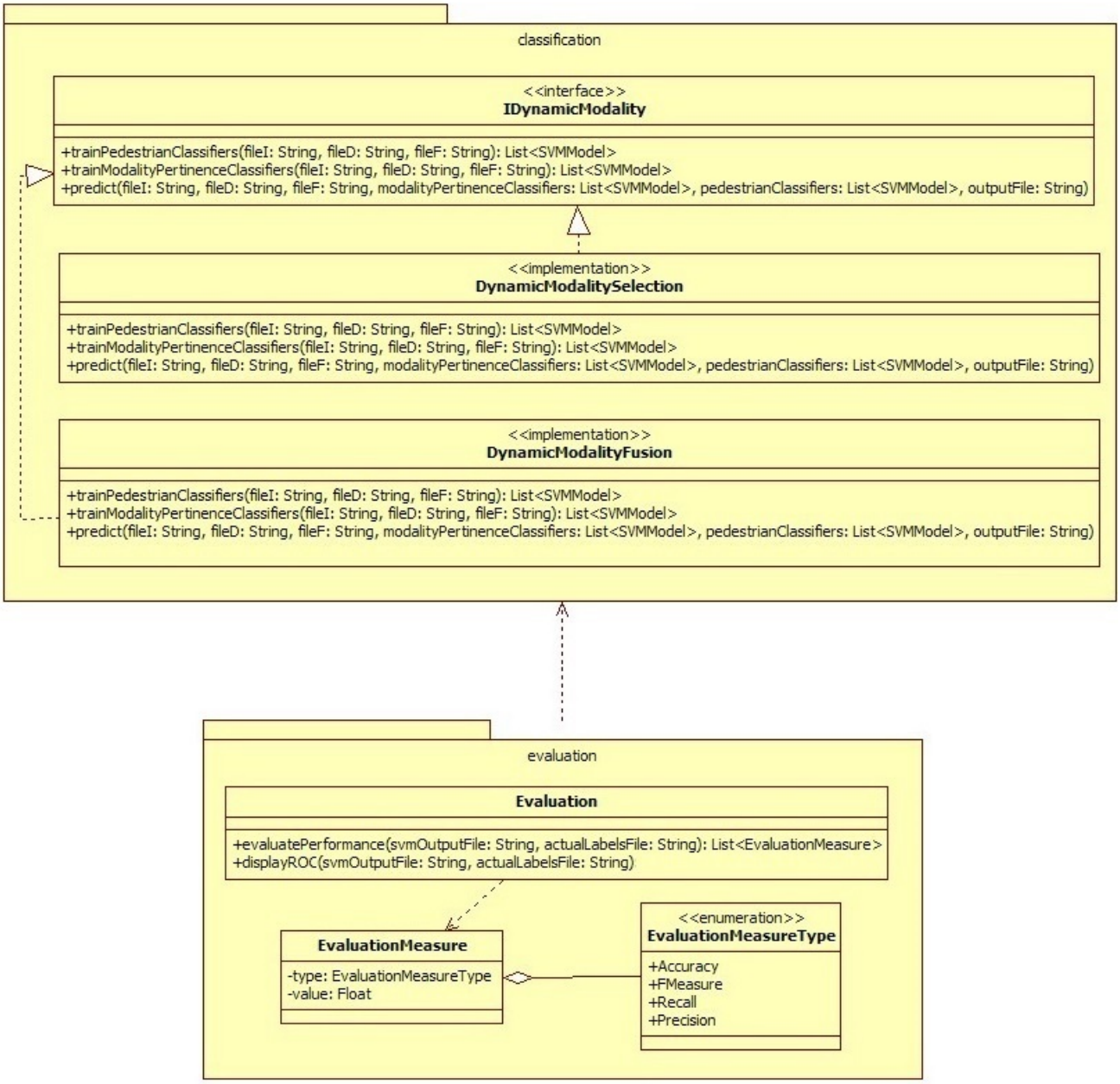


Figure 4.22: Simplified class diagram of the business layer

Conclusions

In this thesis we have focused on applying dynamic machine learning models to solve *supervised* and *unsupervised* classification problems. The particular problems that we have decided to approach are gene expression clustering and pedestrian recognition, because they are very challenging, have a great importance in real life and are representative for the two types of classification: unsupervised and supervised.

In the first research direction, we have introduced three dynamic models for clustering gene expression data, in the context where expression levels for new time points are added to the existing genes. The algorithms (CBDCGE, DHCGE and FDCGE) are capable of adapting the previously obtained partitions when new measurements of gene expression levels are added to the dataset, without performing re-clustering from scratch. The experimental evaluations that we have performed on a real-life gene expression data set show that, in most of the cases, the clustering is reached more effectively and is also more accurate by using our proposed methods than by applying the k-means, hierarchical agglomerative clustering, respectively fuzzy c-means algoritms from the beginning on the extended data. There are situations when the partitions are too difficult to adapt after the addition of new attributes and a full repartition should be considered.

In the same context of dynamic gene expression data, we have proposed a new adaptive association rule mining method (ARARM), which is capable to adapt the set of interesting rules discovered when new gene expression levels are added to the dataset, without performing re-mining from scratch. Experiments on the same gene expression dataset show that ARARM reaches the solution faster than applying the mining algorithm from the beginning.

Furher work in the first research direction could be done in order to determine in which cases it is more appropriate to adapt (using CBDCGE, DHCGE or FDCGE) the partition of the feature-extended object set than to recompute partition from scratch using a classic clustering approach. We also plan to extend the experimental evaluations on other publicly available datasets and to investigate methods to automatically identify the distance threshold for the clusters (e.g. using supervised learning).

In the second research direction, we have addressed two problems. First, we have performed a study on the efficiency of using kernel descriptors in pedestrian recognition, since they obtained good results for visual recognition and to our knowledge have not been used for this task so far. Then, we have introduced two dynamic algorithms, DMS and DMF, that

are able to determine the most suitable modalities among intensity, depth and flow to classify an image, representing a bounding box within an image frame, and further include them in the classification process.

Kernel descriptors have proved to achieve good performances both in single and multi-modality images. We have optimized the parameters of the gradient kernel independently on each modality, using a grid search algorithm. The selection of the most appropriate kernel (Exponential, Gaussian, Laplacian) is influenced not only by the images, but also by the learning algorithm. Even if they are considered a generalization of HOG, they do not reach their performances and this could be caused by the KPCA component used for dimensionality reduction.

Experimental evaluations on a pedestrian dataset show that the dynamic modality selection and fusion models, DMS and DMF, represent promising approaches for multi-modality pedestrian recognition. The first one has the advantages of lower complexity, individual training on modalities, while the second one achieves a higher performance boost. Moreover, the dynamic fusion schemes that we have proposed in our models are generic and could be applied to other problems which need a dynamic integration of the sources.

Further work in the second research direction could be done to extend and improve our dynamic models e.g. integrating image based modality-pertinence indicators, adding other features to the fusion like LBP, or using others from visible domain like HOG, HOF or from FIR domain, using other classifiers like Adaboost, and also to evaluate them on other datasets. Moreover, improvements could be brought to kernel descriptors by optimizing the Gaussian kernels (position kernel and orientation kernel for Gradient Kernel Descriptor) and adapting the dimensionality reduction process in order to retain the most relevant information.

Bibliography

[Aaron et al., 2014a] Aaron, B., Tamir, D. E., Rishe, N. D., and Kandel, A. (2014a). Dynamic incremental fuzzy c-means clustering. In *The Sixth International Conferences on Pervasive Patterns and Applications*.

[Aaron et al., 2014b] Aaron, B., Tamir, D. E., Rishe, N. D., and Kandel, A. (2014b). Dynamic incremental k-means clustering. In *International Conference on Computational Science and Computational Intelligence*.

[Ahonen et al., 2006] Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(12):2037–2041.

[Albayrak and Amasyali, 2003] Albayrak, S. and Amasyali, F. (2003). Fuzzy c-means clustering on medical diagnostic systems. In *Turkish Symposium on Artificial Intelligence and Neural Networks - TAINN*.

[An and Doerge, 2012] An, L. and Doerge, R. (2012). Dynamic clustering of gene expression. *ISRN Bioinformatics*, 2012:1–12.

[Andreica et al., 2013] Andreica, A., Diosan, L., Gaceanu, R. D., and Sirbu, A. (2013). Pedestrian recognition by using kernel descriptors. *Studia Universitas Babes-Bolyai, Seria Informatica*, LVIII(2):77–89.

[Angelova et al., 2015] Angelova, A., Krizhevsky, A., and Vanhoucke, V. (2015). Pedestrian detection with a large-field-of-view deep network. In *ICRA*, pages 704–711. IEEE.

[Apatean et al., 2010] Apatean, A., Rusu, C., Rogozan, A., and Bensrhair, A. (2010). Visible-infrared fusion in the frame of an obstacle recognition system. In *Automation Quality and Testing Robotics (AQTR), Cluj-Napoca*, pages 1 – 6.

[Arima et al., 2003] Arima, C., Hanai, T., and Okamoto, M. (2003). Gene Expression Analysis Using Fuzzy K-Means Clustering. *Genome Informatics*, 14:334–335.

[Ashburner et al., 2000] Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., and et al. (2000). Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25(1):25–29.

[Bagherjeiran et al., 2005] Bagherjeiran, A., Eick, C. F., Chen, C.-S., and Vilalta, R. (2005). Adaptive clustering: Obtaining better clusters using feedback and past experience. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 565–568, Washington, DC, USA. IEEE Computer Society.

[Bagirov and Mardaneh, 2006] Bagirov, A. and Mardaneh, K. (2006). Modified global k-means algorithm for clustering in gene expression data sets. In *Proceedings of the 2006 workshop on Intelligent systems for bioinformatics*, WISB '06, pages 23–28, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Bar-Joseph et al., 2002] Bar-Joseph, Z., Gerber, G., Gifford, D., and Jaakkola, T. (2002). A New Approach to Analyzing Gene Expression Time Series Data. In *Proceedings of the sixth annual international conference on Computational biology*, RECOMB '02, pages 39–48.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. J. (2006). Surf speeded up robust features. In *ECCV*, pages 404–417.

[Bertozzi et al., 2006] Bertozzi, M., Broggi, A., Felisa, M., and Vezzoni, G. (2006). Low-level pedestrian detection by means of visible and far infra-red tetra-vision. *IEEE Intelligent Vehicles Symposium*, pages 231–236.

[Besbes et al., 2011] Besbes, B., Ammar, S., Kessentini, Y., Rogozan, A., and Bensrhair, A. (2011). Evidential combination of SVM road obstacle classifiers in visible and far infrared images. In *Intelligent Vehicles Symposium*, pages 1074–1079. IEEE.

[Besbes et al., 2015] Besbes, B., Rogozan, A., Rus, A.-M., Bensrhair, A., and Broggi, A. (2015). Pedestrian detection in far-infrared daytime images using a hierarchical codebook of surf. *Sensors*, 15(4):8570.

[Bo et al., 2010] Bo, L., Ren, X., and Fox, D. (2010). Kernel descriptors for visual recognition. In *NIPS*, pages 244–252. Curran Associates, Inc.

[Bo et al., 2011a] Bo, L., Ren, X., and Fox, D. (2011a). Depth kernel descriptors for object recognition. In *IROS*, pages 821–826. IEEE.

[Bo et al., 2011b] Bo, L., Ren, X., and Fox, D. (2011b). Object recognition with hierarchical kernel descriptors. In *CVPR*, pages 1729–1736.

[Bo and Sminchisescu, 2009] Bo, L. and Sminchisescu, C. (2009). Efficient match kernel between sets of features for visual recognition. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *NIPS*, pages 135–143. Curran Associates, Inc.

[Bocicor et al., 2014] Bocicor, M. I., Sirbu, A., and Czibula, G. (2014). Dynamic core based clustering of gene expression data. *International Journal of Innovative Computing, Information and Control*, 10(3):1051–1069.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In Haussler, D., editor, *COLT*, pages 144–152. ACM Press.

[Brown et al., 2001] Brown, L., Cat, T., and DasGupta, A. (2001). Interval estimation for a proportion. *Statistical Science*, 16:101–133.

[Calders et al., 2014] Calders, T., Dexters, N., Gillis, J. J. M., and Goethals, B. (2014). Mining frequent itemsets in a stream. *Inf. Syst*, 39:233–255.

[Campan et al., 2006a] Campan, A., Serban, G., and Marcus, A. (2006a). Relational association rules and error detection. *Informatica*, LI(1):31–36.

[Campan et al., 2006b] Campan, A., Serban, G., Truta, T. M., and Marcus, A. (2006b). An algorithm for the discovery of arbitrary length ordinal association rules. In *DMIN*, pages 107–113.

[Chang and Lin, 2001] Chang, C. C. and Lin, C. J. (2001). *LIBSVM: a library for support vector machines*. Online.

[Charikar et al., 2004] Charikar, M., Chekuri, C., Feder, T., and Motwani, R. (2004). Incremental clustering and dynamic information retrieval. *SICOMP: SIAM Journal on Computing*, 33.

[Czibula et al., 2012] Czibula, G., Bocicor, M.-I., and Czibula, I. G. (2012). Promoter sequences prediction using relational association rule mining. *Evolutionary Bioinformatics*, 8:181–196.

[Czibula et al., 2015a] Czibula, G., Czibula, I.-G., Sirbu, A., and Mircea, G. (2015a). A novel approach to adaptive relational association rule mining. *Applied Soft Computing*, 36:519–533.

[Czibula et al., 2015b] Czibula, G., Marian, Z., and Czibula, I. G. (2015b). Detecting software design defects using relational association rule mining. *Knowl. Inf. Syst*, 42(3):545–577.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893.

[Dalal et al., 2006] Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *ECCV*, pages II: 428–441.

[Das et al., 2009a] Das, R., Bhattacharyya, D., and Kalita, J. (2009a). An Incremental Clustering of Gene Expression data. *World Congress on Nature and Biologically Inspired Computing. NaBIC 2009.*, pages 742–747.

[Das et al., 2009b] Das, R., Kalita, J., and Bhattacharyya, D. (2009b). A new approach for clustering gene expression time series data. *International Journal of Bioinformatics Research and Applications*, 5(3):310–328.

[Das et al., 2011] Das, R., Kalita, J. K., and Bhattacharyya, D. K. (2011). A pattern matching approach for clustering gene expression data. *IJDMMM*, 3(2).

[Davidson et al., 2007] Davidson, I., Ravi, S. S., and Ester, M. (2007). Efficient incremental constrained clustering. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 240–249, New York, NY, USA. ACM.

[Deng and Yu, 2014] Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3-4):197–387.

[DeRisi et al., 1997] DeRisi, J., Iyer, P., and Brown, V. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278(5338):680–686.

[Diosan and Rogozan, 2012] Diosan, L. and Rogozan, A. (2012). How can kernel influence image classification performance. *Studia Universitas Babes-Bolyai, Seria Informatica*, LVII(4):97–109.

[Diosan et al., 2012] Diosan, L., Rogozan, A., and Pecuchet, J.-P. (2012). Improving classification performance of support vector machine by genetically optimising kernel shape and hyper-parameters. *Appl. Intell*, 36(2):280–294.

[Dollar et al., 2012] Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761.

[Enzweiler et al., 2010] Enzweiler, M., Eigenstetter, A., Schiele, B., and Gavrila, D. M. (2010). Multi-cue pedestrian classification with partial occlusion handling. In *CVPR*, pages 990–997. IEEE.

[Enzweiler and Gavrila, 2009] Enzweiler, M. and Gavrila, D. M. (2009). Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2179–2195.

[Enzweiler and Gavrila, 2011] Enzweiler, M. and Gavrila, D. M. (2011). A multilevel mixture-of-experts framework for pedestrian classification. *IEEE Transactions on Image Processing*, 20(10):2967–2979.

[Enzweiler et al., 2008] Enzweiler, M., Kanter, P., and Gavrila, D. M. (2008). Monocular Pedestrian Recognition Using Motion Parallax. In *Proc. IEEE Symposium on Intelligent Vehicles*, pages 792–797.

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, (9):1871–1874.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

[Frasch et al., 2013] Frasch, H.-J., Medema, M., Takano, E., and Breitling, R. (2013). Design-based re-engineering of biosynthetic gene clusters: plug-and-play in practice. *Current Opinion in Biotechnology*.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS: Journal of Computer and System Sciences*, 55.

[Gavrila and Munder, 2007] Gavrila, D. M. and Munder, S. (2007). Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73(1):41–59.

[Gerónimo et al., 2010] Gerónimo, D., López, A. M., Domingo Sappa, A., and Graf, T. (2010). Survey of pedestrian detection for advanced driver assistance systems. *IEEE Trans. Pattern Anal. Mach. Intell*, 32(7):1239–1258.

[Gibbons and Roth, 2002] Gibbons, F. and Roth, F. (2002). Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Research*, 12(10):1574–1581.

[Han and Kamber, 2006] Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*, volume 54. Morgan Kaufmann.

[Henson and Cetto, 2005] Henson, R. and Cetto, L. (2005). *The MATLAB bioinformatics toolbox. Encyclopedia of Genetics, Genomics, Proteomics and Bioinformatics*. The MathWorks Inc., Natick, Massachusetts.

[Herrero and Dopazo, 2002] Herrero, J. and Dopazo, J. (2002). Combining hierarchical clustering and self-organizing maps for exploratory analysis of gene expression patterns. *Journal of Proteome Research*, 1(5):467–470.

[Hirschmuller, 2005] Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR*, pages II: 807–814.

[Huang et al., 2011] Huang, D., Shan, C., Ardabilian, M., Wang, Y., and Chen, L. (2011). Local binary patterns and its application to facial image analysis: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 41(6).

[Jain, 2010] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 31(8):651–666.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs.

[Jiang et al., 2012] Jiang, J.-Y., Cheng, W.-H., and Lee, S.-J. (2012). A dissimilarity measure for document clustering. *ICIC Express Letters*, 6(1):15–21.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[Kendall and Stuart, 1966] Kendall, M. G. and Stuart, A. (1966). *The Advanced Theory of Statistics, volume III*. Griffin, London.

[Kim et al., 2007] Kim, K., Zhang, S., Jiang, K., Cai, L., Lee, I., Feldman, L., and Huang, H. (2007). Measuring similarities between gene expression profiles through new data transformations. *BMC Bioinformatics*, 8(29).

[Kulic et al., 2008] Kulic, D., Takano, W., and Nakamura, Y. (2008). Combining automated on-line segmentation and incremental clustering for whole body motions. In *ICRA*, pages 2591–2598. IEEE.

[Lampert, 2009] Lampert, C. H. (2009). Kernel methods in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 4(3):193–285.

[Lance and Williams, 1967] Lance, G. N. and Williams, W. T. (1967). A general theory of classificatory sorting strategies II. Clustering systems. *The Computer Journal*, 10(3):271–277.

[Li and Chen, 2010] Li, T. and Chen, Y. (2010). Fuzzy K-means incremental clustering based on K-center and vector quantization. *JCP*, 5(11):1670–1677.

[Li et al., 2015] Li, W., Wang, R., Song, L., and Jia, X. (2015). Batch dynamically incremental c-means clustering algorithm based on rough fuzzy set. *Journal of Computational Information Systems*, 11(5):1553–1561.

[Li et al., 2005] Li, Y., Verma, S., Lao, L., and Cui, J.-H. (2005). SACA: SCM-based adaptive clustering algorithm. In *MASCOTS*, pages 271–279. IEEE Computer Society.

[Lisin et al., 2005] Lisin, D. A., Mattar, M. A., Blaschko, M. B., Benfield, M. C., and Learned-Miller, E. G. (2005). Combining local and global image features for object class recognition. In *In Proceedings of the IEEE CVPR Workshop on Learning in Computer Vision and Pattern Recognition*, pages 47–55.

[Lopez-Kleine et al., 2013] Lopez-Kleine, L., Romeo, J., and Torres-Avils, F. (2013). Gene functional prediction using clustering methods for the analysis of tomato microarray data. In *7th International Conference on Practical Applications of Computational Biology and Bioinformatics Advances in Intelligent Systems and Computing*, volume 222, pages 1–6.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[Lu et al., 2004] Lu, Y., Lu, S., Fotouhi, F., Deng, Y., and Brown, S. (2004). Incremental genetic k-means algorithm and its application in gene expression data analysis. *BMC Bioinformatics*, 5(172).

[Luan and Li, 2003] Luan, Y. and Li, H. (2003). Clustering of time-course gene expression data using a mixed-effects model with b-splines. *Bioinformatics*, 19(4):474–482.

[Mahalanobis, 1936] Mahalanobis, P. C. (1936). On the generalised distance in statistics. In *Proc. of the Nat. Inst. of Sci. of India*, volume 2, pages 49–55.

[Miron et al., 2015] Miron, A., Rogozan, A., Ainouz, S., Bensrhair, A., and Broggi, A. (2015). An evaluation of the pedestrian classification in a multi-domain multi-modality setup. *Sensors*, 15(6):13851.

[Mitchell, 1997a] Mitchell, T. (1997a). *Machine Learning*. McGraw-Hill.

[Mitchell, 1997b] Mitchell, T. (1997b). *Machine Learning (Mcgraw-Hill International Edit)*. McGraw-Hill Education (ISE Editions), 1st edition.

[Munder and Gavrila, 2006] Munder, S. and Gavrila, S. (2006). An experimental study on pedestrian classification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(11):1863–1868.

[Nedevschi et al., 2009] Nedevschi, S., Bota, S., and Tomiuc, C. (2009). Stereo-based pedestrian detection for collision-avoidance applications. *IEEE Trans. Intelligent Transportation Systems*, 10(3):380–391.

[Ojala et al., 1996] Ojala, T., Pietikainen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59.

[Oliveira et al., 2010] Oliveira, L., Nunes, U., and Peixoto, P. (2010). On exploration of classifier ensemble synergism in pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 11(1):16–27.

[Olson and Delen, 2008] Olson, D. L. and Delen, D. (2008). *Advanced Data Mining Techniques*. Springer.

[Oltean, 2005] Oltean, M. (2005). Improving the search by encoding multiple solutions in a chromosome. In Nedjah, N. and de Macedo Mourelle, L., editors, *Evolutionary Machine Design: Methodology and Applications*, Intelligent System Engineering, chapter 4, pages 85–110. Nova Publishers.

[Oltean et al., 2009] Oltean, M., Grosan, C., Diosan, L., and Mihaila, C. (2009). Genetic programming with linear representation: a survey. *International Journal on Artificial Intelligence Tools*, 18(2):197–238.

[Oren et al., 1997] Oren, M., Papageorgiou, C. P., Sinha, P., Osuna, E., and Poggio, T. (1997). Pedestrian detection using wavelet templates. In *CVPR*, pages 193–199.

[Ouyang and Wang, 2013] Ouyang, W. and Wang, X. (2013). Joint deep learning for pedestrian detection. In *ICCV*, pages 2056–2063. IEEE.

[Pakhira et al., 2004] Pakhira, M. K., Bandyopadhyay, S., and Maulik, U. (2004). Validity index for crisp and fuzzy clusters. *Pattern Recognition*, 37(3):487 – 501.

[Parra Alonso et al., 2007] Parra Alonso, I., Fernandez Llorca, D., Sotelo, M. A., Bergasa, L. M., Revenga de Toro, P., Nuevo, J., Ocana, M., and Garcia Garrido, M. A. (2007). Combination of feature extraction methods for SVM pedestrian detection. *IEEE Trans. Intelligent Transportation Systems*, 8(2):292–307.

[Pejaver et al., 2011] Pejaver, V., Lee, H., and Kim, S. (2011). Gene cluster prediction and its application to genome annotation. *Protein Function Prediction for Omics Era*, pages 35–54.

[Platt, 1999] Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA. MIT Press.

[Rohrbach et al., 2009] Rohrbach, M., Enzweiler, M., and Gavrila, D. M. (2009). High-level fusion of depth and intensity for pedestrian classification. In *DAGM-Symposium*, volume 5748 of *Lecture Notes in Computer Science*, pages 101–110. Springer.

[Rus et al., 2015] Rus, A.-M., Rogozan, A., Diosan, L., and Bensrhair, A. (2015). Pedestrian recognition by using a dynamic modality selection approach. In *IEEE 18th International Conference on Intelligent Transportation Systems, Gran Canaria, Spain, September 15-18*, pages 1862 – 1867.

[Sarmah and Bhattacharyya, 2010] Sarmah, S. and Bhattacharyya, D. (2010). An effective technique for clustering incremental gene expression data. *International Journal of Computer Science Issues*, 7(3):31–41.

[Schapire, 2013] Schapire, R. E. (2013). Explaining adaboost.

[Schölkopf, 2000] Schölkopf, B. (2000). The kernel trick for distances. In *NIPS*, pages 301–307, Cambridge, MA. MIT Press.

[Serban and Campan, 2005] Serban, G. and Campan, A. (2005). Incremental clustering using a core-based approach. In *Proceedings of the 20th international conference on Computer and Information Sciences*, ISCIS'05, pages 854–863, Berlin, Heidelberg. Springer-Verlag.

[Serban and Campan, 2006] Serban, G. and Campan, A. (2006). Hierarchical adaptive clustering. *Informatica*, 19(1):101–112.

[Serban et al., 2006] Serban, G., Campan, A., and Czibula, I. G. (2006). A programming interface for finding relational association rules. *International Journal of Computers, Communications and Control*, I(S.):439–444.

[Sermanet et al., 2011] Sermanet, P., Kavukcuoglu, K., and Lecun, Y. (2011). Traffic signs and pedestrians vision with multi-scale convolutional networks. In *In Snowbird Machine Learning Workshop*.

[Singh et al., 2013] Singh, A., Yadav, A., and Rana, A. (2013). K-means with three different distance metrics. *International Journal of Computer Applications*, 67(10):13–17.

[Sirbu, 2014] Sirbu, A. (2014). A study on dynamic clustering of gene expression data. *Informatica*, LIX(1):16–27.

[Sirbu and Bocicor, 2013] Sirbu, A. and Bocicor, M.-I. (2013). A dynamic approach for hierarchical clustering of gene expression data. In *Intelligent Computer Communication and Processing (ICCP)*, pages 3–6.

[Sirbu et al., 2014a] Sirbu, A.-M., Czibula, G., and Bocicor, M.-I. (2014a). Dynamic clustering of gene expression data using a fuzzy approach. In *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014, Timisoara, Romania, September 22-25, 2014*, pages 220–227. IEEE.

[Sirbu et al., 2014b] Sirbu, A.-M., Rogozan, A., Diosan, L., and Bensrhair, A. (2014b). Pedestrian recognition by using a kernel-based multi-modality approach. In *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014, Timisoara, Romania, September 22-25, 2014*, pages 258–263. IEEE.

[Sirbu et al., 2015] Sirbu, A.-M., Rogozan, A., Diosan, L., and Bensrhair, A. (2015). Pedestrian recognition using a dynamic modality fusion approach. In *Proceedings of IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, September 3-5*, pages 393 – 400.

[Song and Lee, 2012] Song, B. and Lee, H. (2012). Prioritizing disease genes by integrating domain interactions and disease mutations in a protein-protein interaction network. *International Journal of Innovative Computing, Information and Control*, 8(2):1327–1338.

[Stekel, 2006] Stekel, D. (2006). *Microarray Bioinformatics*. Cambridge University Press, Cambridge, UK.

[Su et al., 2009] Su, X., Lan, Y., Wan, R., and Qin, Y. (2009). A fast incremental clustering algorithm. In *Proceedings of the 2009 International Symposium on Information Processing*, pages 175–178.

[Tan et al., 2005] Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley.

[Tuffery, 2011] Tuffery, S. (2011). *Data Mining and Statistics for Decision Making*. John Wiley and Sons.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

[Vimieiro and Moscato, 2014] Vimieiro, R. and Moscato, P. (2014). A new method for mining disjunctive emerging patterns in high-dimensional datasets using hypergraphs. *Inf. Syst*, 40:1–10.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Robust real-time object detection. In *International Journal of Computer Vision*.

[Viola and Jones, 2004] Viola, P. and Jones, M. (2004). Robust real-time face detection. *International Journal of Computer Vision (IJCV)*, 57(2):137–154.

[Viola et al., 2005] Viola, P., Jones, M. J., and Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161.

[Walk et al., 2010a] Walk, S., Majer, N., Schindler, K., and Schiele, B. (2010a). New features and insights for pedestrian detection. pages 1030–1037. IEEE Computer Society.

[Walk et al., 2010b] Walk, S., Schindler, K., and Schiele, B. (2010b). Disparity statistics for pedestrian detection: Combining appearance, motion and stereo. In *ECCV (6)*, volume 6316, pages 182–195. Springer.

[Wang et al., 2013] Wang, P., Wang, J., Zeng, G., Xu, W., Zha, H., and Li, S. (2013). Supervised kernel descriptors for visual recognition. In *CVPR*, pages 2858–2865. IEEE.

[Wang et al., 2009] Wang, X., Han, T. X., and Yan, S. (2009). An HOG-LBP human detector with partial occlusion handling. In *ICCV*, pages 32–39. IEEE.

[Wedel et al., 2009] Wedel, A., Cremers, D., Pock, T., and Bischof, H. (2009). Structure- and motion-adaptive regularization for high accuracy optic flow. In *ICCV*, pages 1663–1668. IEEE.

[WHO, 2015] WHO (2015). Global status report on road safety. *World Health Organization.*

[Wu and Gardarin, 2001] Wu, F. and Gardarin, G. (2001). Gradual clustering algorithms. In *DASFAA*, pages 48–55. IEEE Computer Society.

[Xiao et al., 2003] Xiao, X., Dow, E., Eberhart, R., Miled, Z., and Oppelt, R. (2003). Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization. *In: Proc. 17th Intl. Symposium on Parallel and Distributed Processing.*

[Xu et al., 2013] Xu, G., Zong, Y., and Yang, Z. (2013). *Applied Data Mining.* CRC Press, Inc., Boca Raton, FL, USA.

[Yano and Kotani, 2003] Yano, N. and Kotani, M. (2003). Clustering gene expression data using self-organizing maps and k-means clustering. *Proceedings of SICE 2003 Annual Conference*, 3:3211–3215.

[Young et al., 2010] Young, S., Arel, I., Karnowski, T. P., and Rose, D. (2010). A fast and stable incremental clustering algorithm. In Latifi, S., editor, *Seventh International Conference on Information Technology: New Generations, ITNG 2010, Las Vegas, Nevada, USA, 12-14 April 2010*, pages 204–209. IEEE Computer Society.

[Yuhui et al., 2002] Yuhui, Y., Lihui, C., Goh, A., and Wong, A. (2002). Clustering gene data via associative clustering neural network. *In: Proc. 9th Intl. Conf. on Information Processing*, pages 2228–2232.