

ADVERTIMENT. La consulta d'aquesta tesi queda condicionada a l'acceptació de les següents condicions d'ús: La difusió d'aquesta tesi per mitjà del servei TDX (www.tesisenxarxa.net) ha estat autoritzada pels titulars dels drets de propietat intel·lectual únicament per a usos privats emmarcats en activitats d'investigació i docència. No s'autoritza la seva reproducció amb finalitats de lucre ni la seva difusió i posada a disposició des d'un lloc aliè al servei TDX. No s'autoritza la presentació del seu contingut en una finestra o marc aliè a TDX (framing). Aquesta reserva de drets afecta tant al resum de presentació de la tesi com als seus continguts. En la utilització o cita de parts de la tesi és obligat indicar el nom de la persona autora.

ADVERTENCIA. La consulta de esta tesis queda condicionada a la aceptación de las siguientes condiciones de uso: La difusión de esta tesis por medio del servicio TDR (www.tesisenred.net) ha sido autorizada por los titulares de los derechos de propiedad intelectual únicamente para usos privados enmarcados en actividades de investigación y docencia. No se autoriza su reproducción con finalidades de lucro ni su difusión y puesta a disposición desde un sitio ajeno al servicio TDR. No se autoriza la presentación de su contenido en una ventana o marco ajeno a TDR (framing). Esta reserva de derechos afecta tanto al resumen de presentación de la tesis como a sus contenidos. En la utilización o cita de partes de la tesis es obligado indicar el nombre de la persona autora.

WARNING. On having consulted this thesis you're accepting the following use conditions: Spreading this thesis by the TDX (www.tesisenxarxa.net) service has been authorized by the titular of the intellectual property rights only for private uses placed in investigation and teaching activities. Reproduction with lucrative aims is not authorized neither its spreading and availability from a site foreign to the TDX service. Introducing its content in a window or frame foreign to the TDX service is not authorized (framing). This rights affect to the presentation summary of the thesis as well as to its contents. In the using or citation of parts of the thesis it's obliged to indicate the name of the author



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Ph.D. Dissertation

Region-based face detection, segmentation and
tracking. Framework definition and application to
other objects.

Verónica Vilaplana Besler

Advisor: Prof. Ferran Marqués Acosta

Department of Signal Theory and Communications
Universitat Politècnica de Catalunya

Barcelona, October 2010

A mi mamá

Abstract

One of the central problems in computer vision is the automatic recognition of object classes. In particular, the detection of the class of human faces is a problem that generates special interest due to the large number of applications that require face detection as a first step.

In this thesis we approach the problem of face detection as a joint detection and segmentation problem, in order to precisely localize faces with pixel accurate masks. Even though this is our primary goal, in finding a solution we have tried to create a general framework as independent as possible of the type of object being searched.

For that purpose, the technique relies on a hierarchical region-based image model, the Binary Partition Tree, where objects are obtained by the union of regions in an image partition. In this work, this model is optimized for the face detection and segmentation tasks. Different merging and stopping criteria are proposed and compared through a large set of experiments.

In the proposed system the intra-class variability of faces is managed within a learning framework. The face class is characterized using a set of descriptors measured on the tree nodes, and a set of one-class classifiers. The system is formed by two strong classifiers. First, a cascade of binary classifiers simplifies the search space, and afterwards, an ensemble of more complex classifiers performs the final classification of the tree nodes.

The system is extensively tested on different face data sets, producing accurate segmentations and proving to be quite robust to variations in scale, position, orientation, lighting conditions and background complexity.

We show that the technique proposed for faces can be easily adapted to detect other object classes. Since the construction of the image model does not depend on any object class, different objects can be detected and segmented using the appropriate object model on the same image model. New object models can be easily built by selecting and training a suitable set of descriptors and classifiers.

Finally, a tracking mechanism is proposed. It combines the efficiency of the mean-shift algorithm with the use of regions to track and segment faces through a video sequence, where both the face and the camera may move. The method is extended to deal with other deformable objects, using a region-based graph-cut method for the final object segmentation at each frame. Experiments show that both mean-shift based trackers produce accurate

segmentations even in difficult scenarios such as those with similar object and background colors and fast camera and object movements.

Resum

Un dels problemes més importants en l'àrea de visió artificial és el reconeixement automàtic de classes d'objectes. En particular, la detecció de la classe de cares humanes és un problema que genera especial interès degut al gran nombre d'aplicacions que requereixen com a primer pas detectar les cares a l'escena.

A aquesta tesi s'analitza el problema de detecció de cares com un problema conjunt de detecció i segmentació, per tal de localitzar de manera precisa les cares a l'escena amb màscares que arribin a precisions d'un píxel. Malgrat l'objectiu principal de la tesi és aquest, en el procés de trobar una solució s'ha intentat crear un marc de treball general i tan independent com fos possible del tipus d'objecte que s'està buscant.

Amb aquest propòsit, la tècnica proposada fa ús d'un model jeràrquic d'imatge basat en regions, l'arbre binari de particions (BPT: *Binary Partition Tree*), en el qual els objectes s'obtenen com a unió de regions que provenen d'una partició de la imatge. En aquest treball, s'ha optimitzat el model per a les tasques de detecció i segmentació de cares. Per això, es proposen diferents criteris de fusió i de parada, els quals es comparen en un conjunt ampli d'experiments.

En el sistema proposat, la variabilitat dins de la classe cara s'estudia dins d'un marc de treball d'aprenentatge automàtic. La classe cara es caracteritza fent servir un conjunt de descriptors, que es mesuren en els nodes de l'arbre, així com un conjunt de classificadors d'una única classe. El sistema està format per dos classificadors forts. Primer s'utilitza una cascada de classificadors binaris que realitzen una simplificació de l'espai de cerca i, posteriorment, s'aplica un conjunt de classificadors més complexos que produeixen la classificació final dels nodes de l'arbre.

El sistema es testeja de manera exhaustiva sobre diferents bases de dades de cares, sobre les quals s'obtenen segmentacions precises provant així la robustesa del sistema en front a variacions d'escala, posició, orientació, condicions d'il·luminació i complexitat del fons de l'escena.

A aquesta tesi es mostra també que la tècnica proposada per cares pot ser fàcilment adaptable a la detecció i segmentació d'altres classes d'objectes. Donat que la construcció del model d'imatge no depèn de la classe d'objecte que es pretén buscar, es pot detectar i

segmentar diferents classes d'objectes fent servir, sobre el mateix model d'imatge, el model d'objecte apropiat. Nous models d'objecte poden ser fàcilment construïts mitjançant la selecció i l'entrenament d'un conjunt adient de descriptors i classificadors.

Finalment, es proposa un mecanisme de seguiment. Aquest mecanisme combina l'eficiència de l'algorisme *mean-shift* amb l'ús de regions per fer el seguiment i segmentar les cares al llarg d'una seqüència de vídeo a la qual tant la càmera com la cara es poden moure. Aquest mètode s'estén al cas de seguiment d'altres objectes deformables, utilitzant una versió basada en regions de la tècnica de *graph-cut* per obtenir la segmentació final de l'objecte a cada imatge. Els experiments realitzats mostren que les dues versions del sistema de seguiment basat en l'algorisme *mean-shift* produeixen segmentacions acurades, fins i tot en entorns complicats com ara quan l'objecte i el fons de l'escena presenten colors similars o quan es produeix un moviment ràpid, ja sigui de la càmera o de l'objecte.

Contents

1	Introduction	1
1.1	Problem description	2
1.1.1	Definitions and applications	2
1.1.2	Challenges of face detection	5
1.2	Overview of our approach	6
1.3	Summary of contributions	8
1.4	Outline of the Dissertation	9
2	One Class Classification	11
2.1	Classification	11
2.1.1	The design cycle of a classifier	12
2.1.2	Statistical learning	12
2.2	One class classification	16
2.2.1	One-class vs. two-class classification	16
2.2.2	One-class learning	17
2.2.3	One-class methods	20
2.2.4	Other approaches to one-class classification	21
2.3	Combining classifiers	22
2.3.1	Taxonomies	22
2.3.2	Fusion methods	23
2.4	Summary	24
3	Face Detection Systems	25
3.1	General Framework	25
3.2	State of the art	29
3.2.1	Techniques for pixel-based image models	31

3.2.2	Techniques for block-based image models	34
3.2.3	Techniques for compressed-domain image models	41
3.2.4	Techniques for region-based image models	43
3.2.5	Discussion	44
3.3	System Overview	49
3.4	Summary	51
4	The image model	53
4.1	Region-based image representations	53
4.2	Image representation based on Binary Partition Tree	57
4.3	Creation of the BPT	58
4.3.1	Merging criteria and region model	58
4.3.2	Methodology for the experiments	61
4.3.3	Definition of the merging criterion for the accuracy space	63
4.3.4	Definition of the search partition: Stopping criterion	64
4.3.5	Definition of the search space of the BPT	67
4.4	Summary	70
5	The face model	73
5.1	Color	74
5.1.1	The skin color cue	74
5.1.2	The skin color model	79
5.1.3	Color descriptors in BPT nodes	82
5.2	Texture	86
5.2.1	The texture cue	86
5.2.2	PCA-based classifiers	87
5.2.3	A two class classifier	94
5.2.4	Symmetry	96
5.3	Shape	101
5.4	Combination of classifiers	102
5.4.1	Diversity	103
5.4.2	Normalization	106
5.4.3	Combiners	109
5.5	Summary	111

6	Face detection on BPTs	115
6.1	Generic Descriptors	115
6.1.1	Geometry	116
6.1.2	Color	117
6.1.3	Texture	117
6.1.4	Shape	118
6.2	Simplification of the search space	118
6.3	Accurate object representation	124
6.3.1	Shape matching based on templates	125
6.3.2	Binary and chamfer distances	126
6.3.3	The extension	127
6.3.4	Experiments	127
6.4	Best nodes selection	129
6.5	Evaluation	130
6.5.1	XM2VTS	132
6.5.2	MPEG7	133
6.5.3	BANCA	134
6.5.4	BioID	137
6.5.5	Summary of results	137
6.5.6	Application to face recognition: i3Media	138
6.6	Summary	139
7	Face tracking	147
7.1	Object tracking	147
7.2	BPT based tracking	149
7.3	Mean shift tracking	152
7.3.1	Mean shift	152
7.3.2	Mean shift for tracking	154
7.3.3	Region-based mean shift	158
7.3.4	Experimental results	162
7.4	Summary	165
8	Beyond faces	175
8.1	Object detection and segmentation	175

8.1.1	System overview	177
8.1.2	Traffic sign detection	178
8.1.3	Sky detection	179
8.1.4	Licence plate detection	184
8.1.5	Summary of results	186
8.2	Object tracking	186
8.2.1	MRF energy formulation	188
8.2.2	Graph cuts for segmentation	190
8.2.3	Graph cuts in object tracking	192
8.2.4	Experiments	194
8.3	Summary	195
9	Conclusions and future work	203
9.1	Conclusions	203
9.2	Future work	204
A	Data sets	207
A.1	ORL	207
A.2	BioID	207
A.3	XM2VTS	208
A.4	Banca	208
A.5	MPEG-7	209
B	The image model: Similarity measures and stopping criteria	211
B.1	Definition of the merging criterion for the accuracy area	211
B.2	Definition of the search partition: Stopping criterion	212
B.3	Definition of the search space of the BPT	215
C	Texture classifiers	219
C.1	Probabilistic Visual Learning	219
C.2	Support Vector Data Description	220
C.3	Boosting Algorithms	222
D	Fusion of classifiers	225
E	Node extension for object detection	227

Bibliography

246

Chapter 1

Introduction

One of the central problems in computer vision is the recognition of object classes. The analysis of visual scenes is a high level task that humans perform subconsciously many times every day. We are able to detect instances of different objects irrespective of the high intra-class variability of these object classes. However, computer vision solutions to the problem of object detection in images and videos are far below human performance, both in accuracy and speed.

In particular, the detection of the class of human faces is a problem that generates special interest due to the large number of applications that require face detection as a first step (e.g. person recognition, gesture analysis, etc.).

The problem of face detection has been extensively studied, and different solutions have been proposed. Although state of the art systems present high detection rates, one important limitation is that they do not obtain the actual face area (at the pixel level). Their output is a rectangular subimage that contains only the main facial features, or a subimage that contains many pixels from the background.

Analogously, most approaches to object detection present the same type of output, a bounding box surrounding the object of interest that may even contain more background pixels than object pixels.

In this thesis we intend to approach the problem as a joint detection and segmentation problem, in order to precisely localize objects with pixel accurate masks. For that purpose, we rely on a region-based representation of the images, where objects are obtained by the union of regions in an image partition.

Our primary goal is to address the face detection and segmentation problems. However, in finding a solution we try to create a general framework, as independent as possible of the type of object being searched. These three premises, generality of the solution, accurate object definition and the intrinsic variability of the classes, lead us to use a generic image model based on regions, and to address the problem as a classification problem, where the system is

an ensemble of classifiers based on visual descriptors measured on regions (the object model).

The specificity for faces is achieved by training this object model with face samples. In the same way, it is possible to search for other objects by selecting and training the classifiers for another object class, with a different training set, but using the same image model.

In the course of this thesis we will focus on the face detection problem, with the ultimate goal of creating an application that can solve the problem with similar performance, in terms of detection and error rates, to state of the art systems, but also obtaining segmented faces. Therefore, the core part of the dissertation is devoted to the creation of the generic image model, to the definition of a face model, and to the development of a face detection system. At the end, we will show how to apply the proposed framework to the detection of other objects.

In this thesis we also address the problem of tracking. Following the same order as before, we first propose a solution to track and segment faces in videos and then extend this solution to track other deformable objects.

The rest of this chapter is organized as follows. Section 1.1 defines face detection and segmentation problems, and describes some of their applications and challenges. Next, Section 1.2 presents an overview of our approach. We conclude with a summary of contributions and the outline of the dissertation in Sections 1.3 and Section 1.4, respectively.

1.1 Problem description

In the first part of the thesis we address the problem of automatic segmentation of human faces in images, which is how to design a system to find, based on visual information, all the occurrences of faces in a given image, and extract them from the background.

If we look at the photograph in Figure 1.1 we can almost immediately say how many faces there are in the scene, where they are and which their extent and shape are. We can find faces of any size, orientation, pose and position, even if they are partially occluded. Face detection is a visual task that humans perform quite natural and easily. However, it is a challenging problem in image analysis and yet there is no solution with performance comparable to humans both in accuracy and speed.

1.1.1 Definitions and applications

We begin the discussion with definitions for face detection and other related problems. By *face detection* we understand: given an arbitrary image, decide whether there are any faces in the image, and if there are, return their position and extent in the image. The output given for each face is usually defined by a rectangular area within the image that contains the complete face with part of the background or only part of the face (including the main



Figure 1.1: Kids in the metro.

facial features) (see Figure 1.2(b)), or by a set of points corresponding to the position of some facial features (see Figure 1.2(c)).



Figure 1.2: Example of face detection and segmentation: (a) Original image, (b) and (c) face detection, (d) face segmentation.

A related but much simpler problem than face detection is *face localization*. In this case the input image contains exactly one face, and the task is to determine its position and sometimes its scale or extent. Face detection is a much harder problem than face localization, since in the former we need to find a reasonable model to describe faces and also a reliable mechanism to determine whether an image pattern resembles this model enough to be labeled as face. In the case of face localization, only a face model is needed, since the output is simply the best match between the image patterns and the face model [165]. The output of a face localization system is also a rectangle around the located face or a set of points indicating the main facial features.

Finally, a similar but more challenging problem than face detection is *face segmentation*, where the goal is, not only to detect all the faces present in an image, but also to find their

actual shape. Face segmentation aims to extract the contour of the faces and separate them from the background. In Figure 1.2 an example of face detection versus face segmentation is presented. Figure 1.2 (a) shows the original image, (b) and (c) are two typical outputs of face detection systems, whereas (d) illustrates the result of face segmentation. As this image contains only one face, (b) and (c) are also results of face localization. Note that in the first part of the dissertation we will loosely use the term ‘detection’ to refer to any of the three above mentioned problems, unless we need to describe a particular characteristic of location or segmentation systems.

Why is automatic face detection an interesting problem? There are a number of face processing tasks that require face detection, location or segmentation as a first step. Perhaps the most well known task is face recognition, where the goal is to identify one or more persons in the scene, by comparing the faces in the image with a stored database of faces [26]. Many face recognition techniques assume the availability of frontal faces of similar sizes with the same uniform background. This assumption only holds in very constrained scenarios, for example in passport photographs like the example in Figure 1.3(a). In more generic or realistic scenarios, for instance in the context of face recognition for video indexing, the faces may appear in different positions and in very complex backgrounds, as illustrated in Figure 1.3(b). Therefore, a previous processing is needed to detect and extract the face area from the background.



Figure 1.3: Simple vs. complex scenarios for face recognition

Other problem that requires face detection as a prior step is the extraction of facial features, such as eyes, eyebrows, nose, nostrils, mouth, lips and ears, for their use in applications like gaze estimation, emotion recognition or face recognition. Without knowing where faces are, most feature extraction algorithms will produce many false alarms, being therefore less useful.

Face detection is useful on its own (not only as first processing stage) in other applications like content-based image retrieval and indexing, selective video coding, crowd surveillance, security, and intelligent human-computer interfaces.

All the aforementioned applications can benefit from the accurate segmentation of the faces. Moreover, face segmentation is necessary in other applications like content-based image editing (e.g. automatic adjustment of tint in the facial regions), 3D human face model fitting for model-based coding, computer animation or morphing.

1.1.2 Challenges of face detection

Discerning the presence and finding the location and shape of faces in a scene are visual tasks that humans do almost effortlessly, even in very complex scenarios. However, the development of computer-based systems with similar capabilities is a difficult task, still not completely solved. The main reason is that image face patterns may have a great variability in appearance, and this variability is hard to characterize analytically.

There are many factors that alter face appearance [60]. Some variation is caused by the physical nature of faces and by the presence of external objects. Other types of variation are due to factors that arise because image face patterns are obtained by the 2D projection of 3D objects, and result from the interaction of light, the face and an observer (the camera). The main sources of variability of this kind are the presence of other objects, illumination, the viewing geometry and the imaging process. The following list details the main intrinsic and extrinsic sources of variation in facial appearance:

- *Physical variation:* Although faces are similarly structured, with the same spacial configuration of facial features, there can be significant differences among faces. Intrinsic sources of variation are identity, age, sex, ethnicity, hair style, facial expressions, etc.
- *Objects:* The presence of objects such as glasses, earrings, scarfs, hats and even make-up may produce partial occlusions and shadows.
- *Illumination:* Bad illumination conditions may produce shading, self-shadowing and changes in color. The variability in faces due to differences in illumination is usually dramatic. Variability in images of a same face due to illumination changes may be greater than variability from person to person under the same lighting. Indoor lighting conditions can be well controlled and hence face detectors may achieve very high performance in such conditions. However for outside scenes, lighting conditions are impossible to control and result in strong variations in facial appearance.
- *Viewing geometry:* The viewing geometry is responsible of the face pose, which is determined by the 3D position and the orientation of the observer with respect to the face. A change in pose is due to the relative rigid motion between the subject and the observer. Slight changes in the face position may lead to large changes in the face appearance in the image.

- *Imaging process*: the final image quality depends on characteristics of the camera such as its resolution, focus, or acquisition noise.

Finally, another factor that does not modify face appearance but has great influence in the difficulty of the detection is the *complexity of the background*. A face in a uniform background can be successfully detected with quite a simple detector, whereas faces in cluttered scenes are much more difficult to detect and require more complex systems.

Figure 1.4 illustrates the variability in face appearance due to several of the factors previously discussed. Most of the images are color photographs, but there are also graylevel images (c, h, i and k) and paintings (f, g). There are variations in pose from frontal to profile views (k, g, d), faces rotated in the plane (c, j, o), and facial expressions that modify face appearance (e, n). There are also partial occlusions due to objects like hats, hands or glasses (h, i, m, o), strong shadows (b, j, l, p) and changes in color due to particular lighting conditions (a, b).

The main challenge in face detection is, therefore, to build a system that can accurately account for the wide range of possible variations in face patterns.

1.2 Overview of our approach

In the first part of the thesis we address the problems of face detection and segmentation of frontal or nearly frontal faces that may be rotated in the image plane. Since the aim is to precisely localize faces (to segment the faces), we need to know exactly which pixels belong to the face. This problem can be solved adopting a region-based approach, and trying to find faces by the union of regions from a partition of the image. We use a hierarchical representation of the image, a Binary Partition Tree [145], which represents the image at different resolution levels. The tree is created by merging regions from a partition of the image, with the goal that some of the nodes in the tree represent complete faces or at least a good approximation of the faces. Then, face knowledge is applied to select the nodes in the tree that represent face instances.

While most intrinsic facial variations are difficult to model analytically, they can be modeled probabilistically through statistical learning. Thus, learning is the strategy followed to design the system, which is formed by two classifiers. The first classifier is a cascade of simple threshold classifiers based on low-level visual descriptors, that performs a first simplification of the problem by rejecting most of the nodes. Then a second classifier, an ensemble of more complex classifiers based on another set of descriptors, is used to classify the remaining candidate nodes and assign them a ‘face’ or ‘non-face’ label.

Next, we address the face tracking problem with two different strategies. The first one extends the detection system using the same concepts (the BPT and a face model). The need to reduce the computational cost of the tracker and, at the same time, to obtain a

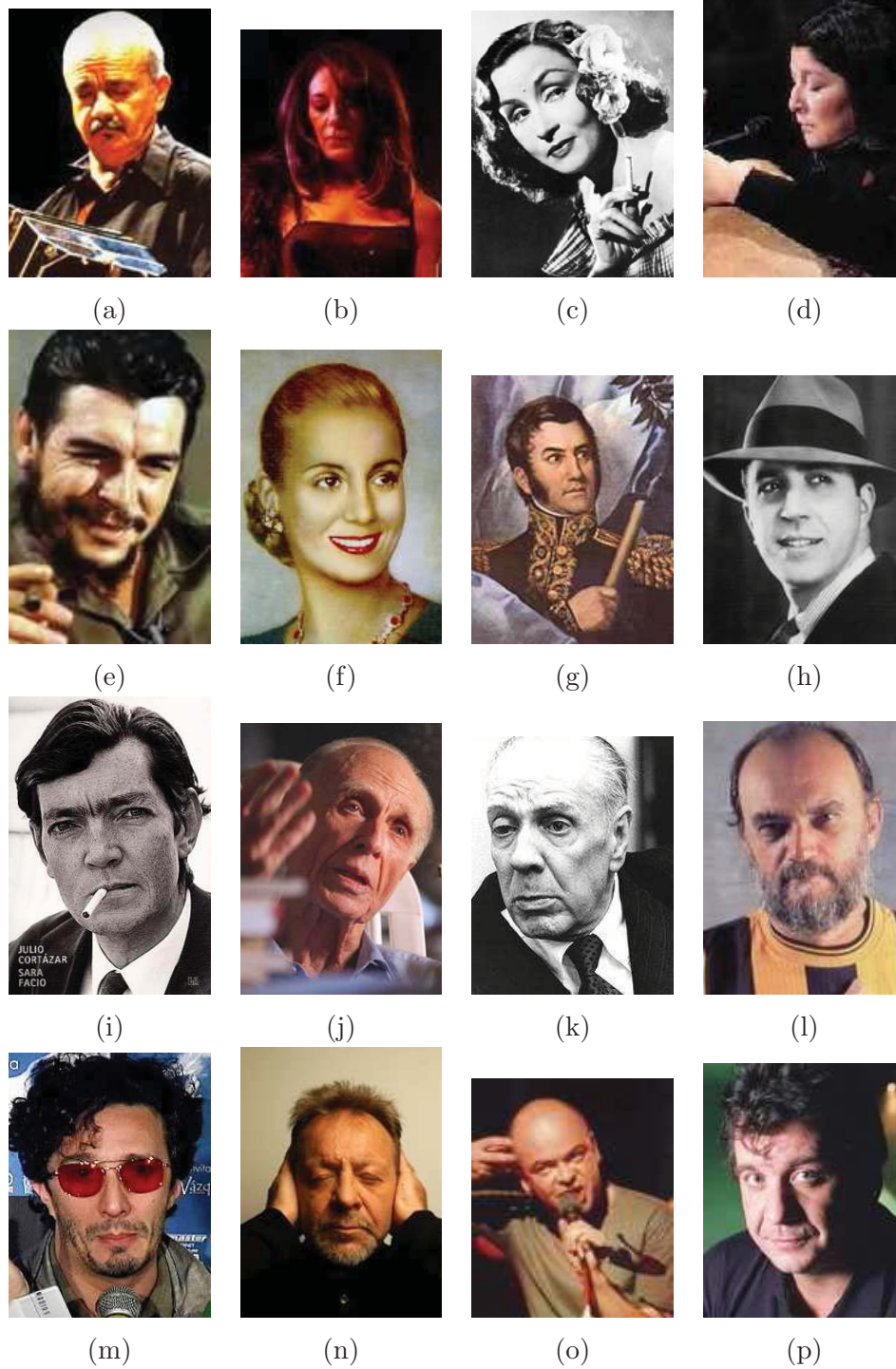


Figure 1.4: Variations in face appearance due to different facial expressions, illumination, viewing geometry, imaging process, external objects and backgrounds.

segmentation of the faces, motivates the second approach, which combines a region-based image model with a mean shift tracker.

Finally, we show that the detection framework proposed for faces is also valid to detect and segment other object classes. We also extend the second tracking technique to track objects with arbitrary, non rigid shapes.

1.3 Summary of contributions

The main contributions of this work are the following:

- The development of a technique for region-based detection and segmentation of frontal or near frontal faces. The system has been extensively tested on a set of face data sets, producing accurate segmentations and proving to be quite robust to variations in scale, position, orientation, lighting conditions and background complexity.
- The technique relies on a hierarchical region-based *image model*, the Binary Partition Tree developed by Salembier and Garrido [145]. In this work, this model has been optimized for the face detection and segmentation tasks. Different merging and stopping criteria have been proposed and compared through a large set of experiments.
- In the proposed system the intra-class variability of faces is managed within a learning framework. The *face class* is characterized using a set of descriptors measured on the tree nodes, and a set of one-class classifiers. The system is formed by two strong classifiers. First, a cascade of binary classifiers that perform a simplification of the search space, and then an ensemble of more complex classifiers that perform the final classification of the tree nodes.
- We show that the technique proposed for faces can be easily adapted to detect other object classes. Since the construction of the image model does not depend on any object class, different objects can be detected using the appropriate object model on the same image model. New object models can be easily built by selecting and training a suitable set of descriptors and classifiers.
- Finally, a tracking mechanism has been proposed that combines the efficiency of the mean-shift algorithm with the use of regions to track and segment faces through a video sequence, where both the face and camera may move. The method has been extended to deal with other deformable objects, using a region-based graph-cut method for the final object segmentation at each frame.

1.4 Outline of the Dissertation

The remaining chapters are organized as follows:

Chapter 2 briefly reviews the main concepts that appear in pattern recognition, defines the one-class classification problem and describes different models for one-class classifiers and their combination.

Chapter 3 details the state of the art on face detection from the perspective of the model used to represent the image. It also presents an overview of the proposed framework for face detection and segmentation.

Chapter 4 discusses the use of the Binary Partition Tree, a hierarchical region-based image representation, for object detection and, specifically, for face detection. Different merging and stopping criteria are objectively evaluated and compared.

Chapter 5 studies color, shape and texture attributes of faces, proposes a set of descriptors to measure them on image regions and associates each descriptor with one classifier. The individual performance of the classifiers is evaluated. Finally, different normalization and fusion rules are studied in order to build an ensemble of classifiers that improves the performance of each individual classifier.

Chapter 6 completes the description of the proposed face detection system, detailing the simplification, extension and final decision stages. A large set of experiments show the performance of the system in different scenarios.

Chapter 7 addresses the problem of face tracking with two different approaches. One is based on the use of the BPT and the other one is an extension of a mean-shift tracker that works with image regions.

Chapter 8 shows that the framework proposed for face detection and segmentation can be applied to detect and segment other semantic objects. It also extends the tracking mechanism proposed for faces to deal with other deformable objects, through the use of a region-based graph-cut algorithm.

Chapter 9 summarizes our approach and provides a discussion of the advantages and limitations of our work. It also suggests some directions for future research in this area.

Chapter 2

One Class Classification

This chapter briefly reviews definitions and problems encountered in pattern recognition applications, concepts that will appear in subsequent chapters. It also defines the one-class classification problem and describes different models of one-class classifiers. Finally, the last section discusses why and how to combine classifiers.

2.1 Classification

The problem of classification is to assign a new object (a *pattern*) to one of several known classes. Pattern recognition is the study of how machines can observe the environment, learn to distinguish patterns of interest from the background, and make sound and reasonable decisions about the classes of new patterns [77]. The goal in pattern recognition is to obtain, from a set of example data, models for the classes and learning rules to predict the labels of new, unknown objects; that is, to classify new objects.

The design of a classifier involves proposing some form or model for the classifier and using training patterns to learn or estimate the unknown parameters of the model. The learning can be performed in two main forms: *unsupervised* or *supervised* [41].

In unsupervised learning, the problem is to discover the structure of the data set, if there is any. Classes are learned based on the similarity of patterns. The system forms clusters or ‘natural groupings’ of the input patterns, where the concept of ‘natural’ is defined implicitly or explicitly in the clustering system itself. Given a particular set of patterns, different clustering algorithms may lead to different clusters.

In supervised learning each object in the data set comes with a label, and the classifier is trained using this information to do the labeling. For each sample in the training set there is a measure of how costly each classifier’s action on that sample is, and the goal is to reduce the sum of the costs for the training patterns.

A general classification problem can be solved following one of the two main approaches

to pattern recognition: *syntactic* or *structural* pattern recognition and *statistical* pattern recognition [77].

In the syntactic approach patterns are represented hierarchically in terms of simpler patterns or primitives and the relationships between primitives. Classes, which are described in terms of primitives and grammars or logical rules, are inferred from the available training data [53, 41].

In statistical pattern recognition the objects are described by a set of numerical characteristics called *features*. The goal is to choose those features that allow objects from different classes to occupy compact and disjoint regions in the feature space. The effectiveness of the representation space (the feature set) is determined by how well objects from different classes can be separated. Given a set of training objects from each class, the objective is to find decision boundaries in the feature space to separate objects belonging to different classes [54, 41].

In the following we will concentrate on supervised problems in statistical pattern recognition, since they provide a complete framework to understand and solve face detection and segmentation problems.

2.1.1 The design cycle of a classifier

A recognition system operates in two modes: training and classification (see Figure 2.1). In the training mode, a model for the classifier is proposed and the unknown parameters of the model are estimated from a set of labeled training objects. Objects are described by a set of features. Features may not be equally relevant, some of them may be important only in relation to others and some may be only noise in a particular context or application. Therefore, feature selection is used to improve the quality of the description. Feature selection, training and testing of a classifier model form the core of supervised pattern recognition. The cycle, which is shown in the top part of Figure 2.1, may be performed several times, changing the features, the parameters or even the classifier model until a satisfactory solution is reached [88].

In the classification mode, the trained classifier assigns the input object to one of the known classes, based on the measured features. The role of the preprocessing module is to segment the pattern from the background, remove noise, normalize the pattern and perform any other operation required to define a compact representation of the object.

2.1.2 Statistical learning

Let us assume that feature measures are real values. An object i is represented by the feature vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id}) \in \mathbb{R}^d$. $\mathcal{X} = \mathbb{R}^d$ is the feature space. Note that in the sequel we use the same term ‘object’ to indicate both the real object and the feature vector that represents

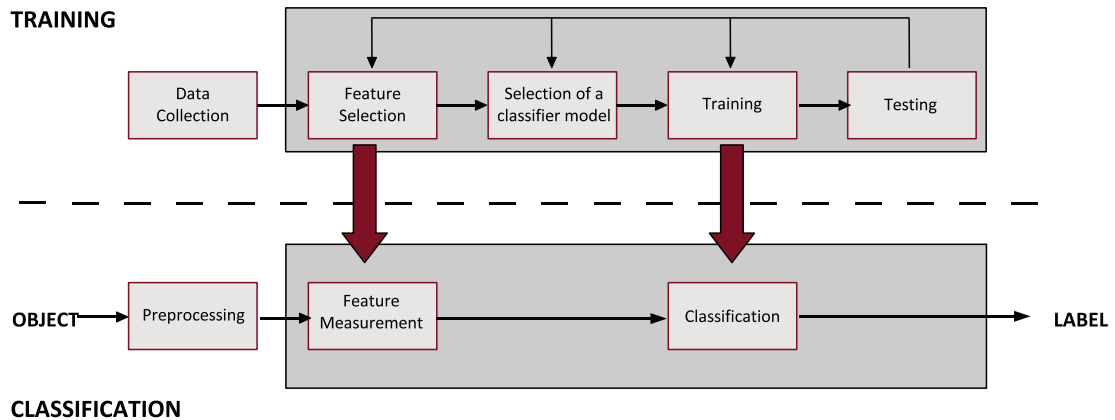


Figure 2.1: Training and classification modes of a supervised classification system.

it.

The choice of distinguishing features to achieve a ‘good’ object representation is a central aspect in any pattern recognition problem. The goal of feature selection is to characterize objects by measurements that are very similar for objects in the same class, but are very different for objects in other classes. It is assumed that the features are selected so that two objects that resemble ‘in real life’ are also near in the feature space. This is called the *continuity assumption*. Additional characteristics may be desirable for the representation such as robustness to noise or invariance to some transformations of the input (translation, rotation, scaling, changes of illumination, etc.). The feature selection process is clearly a domain-dependent task that requires prior knowledge about the problem (about the objects to classify).

Without loss of generality, let us assume a classification problem with two classes ω_1 and ω_2 and with labels -1 and 1 , respectively, where each object belongs to one and only one class. The information to design a classifier is usually in the form of a training data set, a set of N objects \mathbf{x}_i with associated labels $y_i \in \Lambda = \{-1, 1\}$:

$$\mathcal{X}^{tr} = \{(\mathbf{x}_i, y_i) | i = 1, \dots, N\} \quad (2.1)$$

From this training set, a function that estimates a label for each object has to be inferred. A *classifier* is, therefore, any function $f : \mathbb{R}^d \rightarrow \Lambda$ that assigns a label $y = f(\mathbf{x})$ to an object \mathbf{x} .

Usually the type of function f is chosen in advance, so the function depends on some parameters \mathbf{w} which have to be determined using the training set. Examples of these functions are linear classifiers, Gaussians densities, mixture models, support vector classifiers, etc.

The learning problem is, therefore, to find the optimal parameters for the function $f(\mathbf{x}, \mathbf{w})$; that is, the parameters \mathbf{w} that minimize the *expected risk*:

$$\mathcal{R}(f, \mathbf{w}) = \int \mathcal{L}(f(\mathbf{x}, \mathbf{w}), y) p(\mathbf{x}, y) d\mathbf{x} dy \quad (2.2)$$

where $\mathcal{L}(f(\mathbf{x}, \mathbf{w}), y)$ is a *loss function* that measures the discrepancy between assigned and true labels and $p(\mathbf{x}, y)$ is the true data distribution. Different definitions of the loss function are possible. For instance, when the outputs of f are discrete values, the 0 - 1 loss [169] can be used. If f is a function that estimates labels with real values $f : \mathbb{R}^d \rightarrow [-1, 1]$, two possible measures are the mean squared error between true and predicted labels or the cross entropy.

In most problems the true data distribution $p(\mathbf{x}, y)$ is unknown, and some approximation of the expected risk is minimized. One frequently used approximation is the *empirical risk*, which averages the errors of the training samples in \mathcal{X}^{tr} :

$$\mathcal{R}_{emp}(f, \mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(\mathbf{x}_i, \mathbf{w}), y_i) \quad (2.3)$$

A low empirical risk on the training set, however, does not guarantee a low expected risk. The classifier may be *overtrained* on the data. A sufficiently flexible function f can always fit the training data. The classifier is then entirely adapted to the data, including its noise, and may define structures which do not really characterize the classes. The overtraining or overfitting may reduce the *generalization capability* of the classifier. The generalization capability refers to the performance of the classifier in classifying test objects which were not used during the training.

In Figure 2.2(a) an example of a two class classification problem is given. The training objects, drawn from two banana-shaped data sets, are separated by a linear classifier, plotted as a solid line. In Figure 2.2(b), an example of overfitting on the same data is shown. A very flexible function is overtrained on the data and finds structures which are not really present.

A poor generalization can also be attributed to the use of a function $f(\mathbf{x}, \mathbf{w})$ with a large number of unknown parameters. The problems of overfitting and poor generalization become more severe when the number of features is too large relative to the number of training samples. Since the function has to be defined in the complete feature space, the volume to be described grows exponentially with d , the dimension of the space. This problem is called the *curse of dimensionality* [40].

The choice of the type of function $f(\mathbf{x}, \mathbf{w})$ depends on the true nature of the underlying probability distribution and on the amount of training data available. In broad terms, simple parametric functions with a few parameters introduce a strong bias, while functions with more parameters are more flexible but introduce high variance. The selection of a function type must address this tradeoff called the *bias-variance dilemma*. The best fitting function for a given sample is a tradeoff between the bias and variance contributions [58].

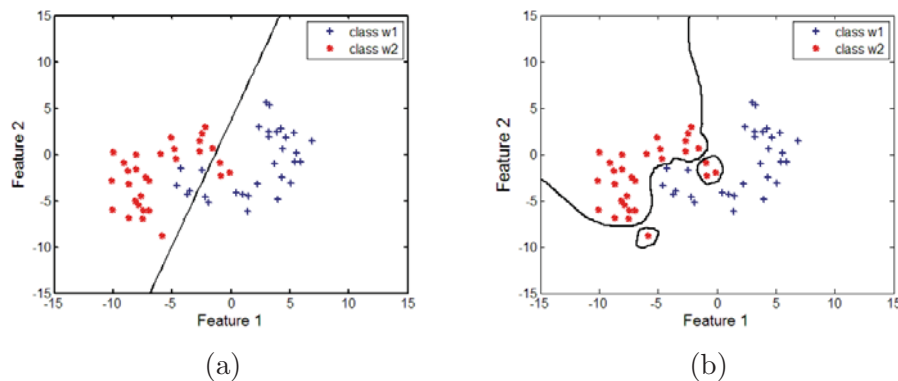


Figure 2.2: Scatterplot of the training set for a two class classification problem; a linear classifier (a) and a very flexible function that overtrains on the data (b).

The bias-variance dilemma may be avoided by introducing previous knowledge into the design of the function, such as constraints on the function form to decrease its complexity while preserving its flexibility. Another solution is to add an extra term to the empirical risk, \mathcal{R}_{struct} , and minimize a combination of empirical and *structural risk*. This structural risk should be a function measuring the complexity of the function model. Several approaches have been developed to define the structural risk, like counting the number of free parameters in the function f , minimizing the norm of the parameters \mathbf{w} , or considering the worst-case performance of the classifier using the VC-dimension (Vapnik Chervonenkis' measure of capacity) [12, 178, 189].

Another issue in pattern recognition is how to organize data for training and testing experiments. Typically two sets are used, the *training* set (seen data) to build the model and the *test* set (unseen data) to measure its performance. Sometimes, a third *validation* set is needed to tune the model (e.g. for pruning a decision tree). All data sets should have representative samples of the data to which the model will be applied. The main alternatives for making the best use of data are:

- *Holdout*: if a large data set is available, data can be split in two independent sets, one for training and the other one for testing. A problem may occur if certain class is not represented in the training set. A variation of this method, stratified holdout, samples data in such a way that each class is represented in both sets.
- *Cross-validation*: is a solution for small data sets. Data is divided into k folds (subsets) of equal size. The model is trained on $k-1$ folds, and the remaining fold is used for testing. The procedure is repeated k times, choosing a different fold for testing each time. The performance is averaged on the k test sets.

- *Bootstrap*: this method randomly selects several sets for training, with replacement. The error rate of the classifiers is averaged.

For a complete analysis of methods and algorithms in pattern recognition the reader is referred to [41, 88, 178, 11, 54, 53].

2.2 One class classification

2.2.1 One-class vs. two-class classification

The classification concepts and problems discussed in the previous section dealt with applications where objects were classified in one of several classes. However, there are other applications where there is only one class of objects of interest; applications where the goal is to distinguish objects in this class from any other object. This is called *one-class classification*.

As defined in [169], the problem in one-class classification is to make a description of a given set of objects and to detect which new objects resemble this training set. The difference with the conventional (two-class) classification problem is that in one-class classification it is assumed that only samples of one class are available. The objects of this class are called *target objects*. All other objects are defined as *outlier objects*. The task is then to define a boundary around the target class, using only samples of target objects, accepting as much of the target objects as possible, and minimizing the chance of accepting outlier objects (see Figure 2.3).

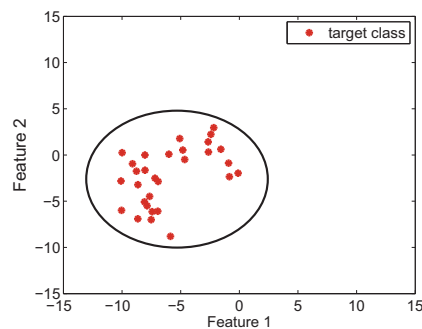


Figure 2.3: A one class classifier that encloses a training set.

One-class classification is useful in different applications. One application is outlier detection, where the goal is to detect suspicious or uncharacteristic objects from a data set. The outliers in the data may be caused by measurement errors in the feature values, resulting in much larger or smaller values than the values of other training objects. Another application is two-class classification problems where one of the classes is well sampled while the other is

not, because measurements on this class may be difficult or expensive to obtain. An example is a machine monitoring system, where the goal is to describe the normal operation condition of a running machine and to distinguish it from abnormal situations. Finally, the application that interest us, detection problems, where it is necessary to detect the presence of a certain class of objects. The problem is then to characterize this class using a training set of object samples, but without or with very few no-object samples.

One-class classification is also known in the literature with other terms, that originate from the different areas where it was applied: novelty detection [11], outlier detection [133] or concept learning in the absence of counter-examples [80].

The problem of one-class classification has been extensively studied by D. Tax; the notation and taxonomy of one-class methods presented in this section come from his work in [169, 172, 171].

2.2.2 One-class learning

Let ω_T and ω_O be the target and outlier classes, with labels 1 and 0, respectively. For one-class classification several models of $f(\mathbf{x}, \mathbf{w})$ have been proposed. Generally, this function has one of the following forms:

$$f(\mathbf{x}) = I(d(\mathbf{x}|\omega_T) < \theta_d) \quad (2.4)$$

when the classifier estimates a distance from an object \mathbf{x} to the target class, or

$$f(\mathbf{x}) = I(p(\mathbf{x}|\omega_T) > \theta_p) \quad (2.5)$$

when the classifier estimates a resemblance (or probability) of the object to the target class. θ_d and θ_p are thresholds and I is the indicator function. The classifier accepts the object if its distance to the target class is lower than the threshold θ_d or when the resemblance is larger than the threshold θ_p .

The methods differ in the definition of distance or resemblance measures, in their optimization and in the optimization of the thresholds with respect to the training set \mathcal{X}^{tr} .

A target object accepted by the classifier is correctly classified. If it is rejected, it is erroneously classified as outlier object. The fraction of target objects classified as target objects (*true positives*) is denoted by f_{T+} . The fraction of target objects classified as outlier objects (*false negatives*) is f_{T-} . The fraction of outlier objects classified as target (*false positives*) and outlier (*true negatives*) are denoted by f_{O+} and f_{O-} , respectively. Table 2.1 shows the four possible situations of classifying an object in one-class classification. Note that $f_{T+} + f_{T-} = f_{O+} + f_{O-} = 1$. Classification errors are the sum of false positives and false negatives.

The same problems that appear in the design of a general classifier are also present in one-class classification: the definition of the loss or error function, the curse of dimensionality, the

	Target object	Outlier object
Classified as target	true positive f_{T+}	false positive f_{O+}
Classified as outlier	false negative f_{T-}	true negative f_{O-}

Table 2.1: The four possible situations of classifying an object in one-class classification.

risk of overtraining and the generalization capability of the classifier. In one-class classification some of these problems may be worse. In a general two class classifier the decision boundary is supported from both sides by examples of each of the classes. In one-class classifiers the target class boundary has to be inferred from one side, using only true target objects. In this case it may be difficult to decide how tight this boundary should be.

When there are no outliers available, only f_{T+} and f_{T-} can be estimated whereas f_{O+} and f_{O-} are unknown. As a consequence, only f_{T-} , the fraction of target objects rejected by the classifier, can be minimized.

To avoid the trivial solution of accepting all the data, when there are no outlier samples in the training set it is necessary to make some assumptions about the distribution of outliers. For instance, to define the empirical risk \mathcal{R}_{emp} on target and outlier data, artificial outlier examples or a measure of the volume of the target data description is needed. Moreover, the definition of the structural risk \mathcal{R}_{struct} requires constraints on the smoothness of the function $f(\mathbf{x}, \mathbf{w})$ and also constraints to obtain a close boundary around the target data. A simple solution proposed by Tax [169] is to assume a bounded uniform distribution of the outliers around the target class.

The key point in one-class classifiers is the tradeoff between the fraction of the target class that is rejected f_{T-} and the fraction of the outlier class that is accepted f_{O+} , the tradeoff between false negative and false positive errors. An example is shown in Figure 2.4, where the elliptical training data is described with a circular boundary. The classifier makes some errors: part of the target data is rejected and some outliers are accepted. The figure shows the regions f_{T+} , f_{T-} and f_{O+} , assuming a uniform distribution of outliers in the rectangular (white) area. Increasing the volume of the description to decrease the number of false negatives increases also the number of false positives. Different tradeoffs between f_{T-} and f_{O+} can be obtained by varying the classifier's threshold on the distance or resemblance.

To compare two methods which differ in the definitions of probabilities $p(z)$ or distances $d(z)$, the target acceptance rate f_{T+} is fixed and then f_{O+} is measured for both methods. The method with the lowest outlier rejection rate f_{O+} should be preferred [169].

Different classifiers can also be compared through their receiving operating characteristic

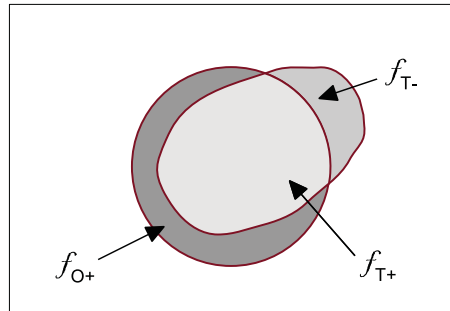


Figure 2.4: Regions in one-class classification. The target class is described with a spherical boundary. Outliers are uniformly distributed in the rectangle.

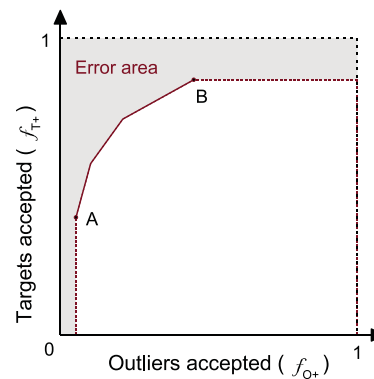


Figure 2.5: ROC curve and error area.

(ROC) curves. The ROC shows the fraction of outliers accepted by the classifier versus the fraction of targets accepted, as the discrimination threshold is varied. In most one-class classification methods one distance or resemblance is obtained independently of the threshold. In these cases, to compute each point in the ROC a threshold θ is fixed and f_{T+} and f_{O+} are measured on a test set for that threshold. Other methods derive the threshold directly from the training set. To calculate the ROC in these cases, for each value of f_{T+} , a threshold $\theta_{f_{T+}}$ is computed and then f_{O+} is measured on a set of outlier examples. An example of a ROC curve is shown in Figure 2.5, where the solid line A-B shows f_{T+} and f_{O+} for varying threshold values. A global measure of the classifier error (over all the possible threshold values) can be obtained by computing the area over the ROC curve (AOC, the colored area in Figure 2.5) or the area under the ROC curve (AUC).

2.2.3 One-class methods

There are three main approaches to one-class classification, that use different generalization principles:

- *Density based methods*, that estimate only a density on the target class
- *Boundary based methods*, which fit a boundary with minimal volume around the data
- *Reconstruction based methods*, that compute a distance to (some of) the target objects

Density methods

They are the most straightforward methods. Density methods estimate the probability density of the target objects $p(\mathbf{x}|\omega_T)$ using the training data and set a threshold on this density.

Using the Bayes rule, the posterior probability for the target class can be computed by:

$$p(\omega_T|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_T)p(\omega_T)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_T)p(\omega_T)}{p(\mathbf{x}|\omega_T)p(\omega_T) + p(\mathbf{x}|\omega_O)p(\omega_O)} \quad (2.6)$$

When no outliers are available, the distribution $p(\mathbf{x}|\omega_O)$ is unknown, and so are the priors $p(\omega_T)$ and $p(\omega_O)$. This means that the class posterior probability cannot be computed. The problem can be solved by assuming a distribution $p(\mathbf{x}|\omega_O)$ for the outlier class. If outliers are uniformly distributed around the target class, it can be proved that $p(\omega_T|\mathbf{x}_1) < p(\omega_T|\mathbf{x}_2)$ if and only if $p(\mathbf{x}_1|\omega_T) < p(\mathbf{x}_2|\omega_T)$. In this case, $p(\mathbf{x}|\omega_T)$ can be used instead of $p(\omega_T|\mathbf{x})$.

The most frequently used distribution models are Gaussian, mixtures of Gaussians and Parzen density estimation. The Gaussian model [12] is the most simple method, making a strong assumption on the model (unimodal and convex) for the data. A more flexible description can be obtained with a mixture of Gaussians [12]. This model requires more data for the training but has a smaller bias than the Gaussian model. Finally, the most flexible is the Parzen density estimation [122], a mixture of kernels (often Gaussian kernels) centered on the individual training objects. In this method a good description depends on how well the training set represents the true target distribution. All training objects are stored, so the training has almost no computational cost, though the testing is very expensive, particularly for large data sets in high dimensions, since distances to all the training objects have to be computed.

All these methods assume that the training data is a representative sample from the true target distribution. They have the problem that the estimation of a complete density function, particularly in high dimensions, requires a large set of training samples to overcome the curse of dimensionality. On the other hand, this approach works very well when a large set of sample data is available and when a flexible model is used.

Boundary methods

Boundary methods only focus on the boundary of the target class. In this sense, they are the most ‘pure’ one-class methods. They rely on distances between objects, and this makes them sensitive to the scaling of the features. The number of objects that is required for the training is smaller than in the case of density models.

The K-centers technique [200], the nearest neighbor method [169, 40] and the support vector data description [172] belong to this kind of approaches.

The main advantage of these methods is that they avoid the estimation of the complete probability density. The target class may be sparsely sampled: it is sufficient to know just some examples on the boundary of the class. The problem here is to know how to define the boundary and the threshold.

Reconstruction methods

The reconstruction methods have been developed to model data. These methods use previous knowledge and make assumptions about the generating process for the objects to select and fit a model to the data. Most of these methods make assumptions about the clustering characteristics of data or their distribution in subspaces. A set of prototypes or subspaces is defined and a reconstruction error is minimized.

It is supposed that outlier objects do not satisfy the assumptions about the target distribution. The outliers should be represented worse than true target objects and their reconstruction error should be high. The reconstruction error is therefore used as a distance to the target set.

Representative reconstruction methods are Principal Component Analysis (PCA)[82], mixtures of PCAs [175], K-means clustering [12], autoencoders and diabolito networks [169].

The advantage of reconstruction methods is that, due to the use of previous knowledge, they perform well and suffer less from poor generalization and low sample size. On the other hand, a large bias may be introduced if a wrong model is chosen.

2.2.4 Other approaches to one-class classification

We will mention here two other approaches that, although not being specifically designed for one-class classification tasks, may be adopted for object detection, which is the application of one-class classification that is the subject of this thesis.

Two-class methods

A simple solution to one-class problems is to generate outlier data around the target class and use a two-class classifier to discriminate target objects from outlier objects. The drawback of this strategy is that it requires this set of outliers, since in some applications this data may

be difficult to obtain. Even if a set of outliers is collected, it may happen that it is not a representative sample of the true outlier distribution.

A common solution is to find a set of ‘near-target’ objects in the outlier class. The near-target data can be obtained, for instance, with outlier objects wrongly classified as target objects in early stages of the training process.

Template matching

Template matching is one of the simplest and earliest approaches to pattern recognition. Matching is a generic operation which is used to measure the similarity between two entities of any kind such as points, curves or shapes. In its basic form, a template or a prototype of an object to be recognized is available. The candidate objects are matched against the stored template. The similarity measure should take into account translations, rotations and changes in scale of the input patterns. A training set may be used to optimize the similarity measure as well as to obtain the template.

Template matching can be used to detect target objects in one-class classification problems. Here patterns are compared against only one prototype. It can be understood as a limit case where the target class is modeled with only one sample, the template, and an acceptance threshold is learned from a set of training target objects.

The technique may fail for large variations among patterns or changes in viewpoint. Deformable template approaches [79] can be used to deal with this variability.

2.3 Combining classifiers

In many pattern recognition problems different classifiers may implicitly represent different useful aspects of the problem or the input data, while no one classifier represents all useful aspects. In those cases, combining several classifiers in the hope of improving the overall classification accuracy is a common practice.

2.3.1 Taxonomies

There are various taxonomies proposed to summarize the work in the field of classifier combination [88]:

- *Fusion and selection*: Perhaps the most accepted structure, it divides methods in two groups: fusion and selection. In classifier fusion the decisions of individual classifiers are combined into a final decision. In classifier selection, each member in the ensemble is supposed to know well some part of the feature space and is responsible for this part. One type of selection classifier which is frequently used is a cascade classifier, where only one classifier is active at a time. Given a new input, the first classifier tries to

make a decision, if it is certain the input is labeled and the procedure stops. If not, the input is passed to the next classifier. In [77] a similar division is proposed, according to the architecture of the combination scheme: parallel, where individual classifiers are used independently and their decisions are combined, cascade or serial, where classifiers are applied in a serial sequence, and hierarchical, where classifiers are combined into a structure (similar to a decision tree).

- *Decision or coverage optimization*: Decision optimization methods choose and optimize the combiner for a fixed set of base classifiers, while coverage optimization methods assume a fix combiner and create diverse base classifiers.
- *Trainable or non-trainable combination*: Some combiners do not need training after each classifier in the ensemble has been trained individually (e.g. majority voting). Other combiners need additional training (e.g. weighted average). A third group builds the combination at the same time that trains the individual classifier. A typical method in this group is Adaboost [51].

In what follows we focus on fusion methods, i.e. combining classifier decisions or mixture of experts, since most combination schemes in the literature belong to this category.

2.3.2 Fusion methods

Fusion methods use the outputs of the individual classifiers to find the overall support value for each class and label each input with the largest support. Alternatively, the outputs of the classifiers may be treated as features in a new feature space, where another classifier is trained to produce the final class label. In this way, layer upon layer of classifiers can be built. As pointed in [88], the important issue in this case is how to train these classifiers so that the increased complexity is justified by a corresponding gain in accuracy.

There are several ways to integrate the outputs of the classifiers in an ensemble, which depend on the information given by individual members. Usually three types of output or levels of fusion are distinguished [190]:

- *The abstract level*: each classifier produces a class label. These classifiers offer the minimum amount of information for the fusion, as no information about potential alternatives are available. There is no information about the certainty of the guessed label.
- *The rank level*: the output of each classifier is a subset of labels, with the alternatives ranked in order of their likelihood. This level is useful for systems with a large number of classes, for example for biometric recognition systems.

- *The measurement level:* each classifier produces a real value output for each class, which is the support for the hypothesis that the sample to be classified comes from this class.

The most commonly used strategies for combination at the abstract level are majority voting and variations such as unanimity or weighted majority. Other techniques within this group are the Naive Bayes approach [37], which assumes independent outputs, or multinomial methods like the behavioral knowledge space (BKS) [72] or the Wernecke's method [188], which are based on estimates of the posterior probabilities.

Combinations at the rank level include the Borda Count method [13], which is a generalization of majority voting, or schemes like the intersection, union or highest rank rule proposed in [70].

For soft level outputs, if the outputs are estimated posterior probabilities or likelihoods, typical combination rules are sum/average, product or order statistics [88]. When the outputs are interpreted as fuzzy membership values, believe or evidence values, then fuzzy rules, believe functions or Dempster Shafer techniques can be used [140, 190].

2.4 Summary

In this chapter we have reviewed the main definitions and issues of pattern recognition. We have focused on one-class classification problems, the approach that will be used to address face detection in Chapter 5, as well as the detection of other objects in Chapter 8. We have also reviewed different strategies to combine classifiers. Classifier fusion will be used in Chapter 6 and Chapter 8 to improve the overall performance of the detection system.

Chapter 3

Face Detection Systems

This chapter details the state of the art on face detection and presents an overview of our technique. In the first section we propose a general scheme for a face detection system and analyze its components. The scheme allows a better understanding of the structure and operation of a face detector and of the different approaches that can be considered to solve the problem. In the second section we review the most well known face detection techniques from the point of view of this scheme; the analysis using this general view facilitates the comparison among the different methods. In the last section we make a proposal for a face detection system, which is developed in the following chapters.

3.1 General Framework

Face detection is a classification problem. To find face instances in an image, different parts or areas in the image are analyzed. In the sequel, these areas are called *patterns*. For each pattern, features are measured and used to decide whether the pattern is an instance of a face. In other words, each pattern is classified as face or no-face.

In any face detection system we can identify three basic elements: a description of the object being searched or *Face Model*, a procedure that simplifies the search by selecting some parts of the image where the search is to be performed, that is, a *Selection of Candidates* and, finally, a mechanism to *Classify* or decide which of the candidates are face instances. The scheme of a general detector is depicted in Figure 3.1.

In addition to the previous elements, there are other two fundamental parts in the system, which we call *Image Model* and *Application*. Since we are working with images, it is necessary to choose an image representation and to define which patterns in the image will be analyzed. This is the task of the Image Model block. The *Application* block makes explicit the idea that the convenience of a particular image or face model may be an application-dependent issue. Low or high resolution images, indoor or outdoor environments, simple or cluttered

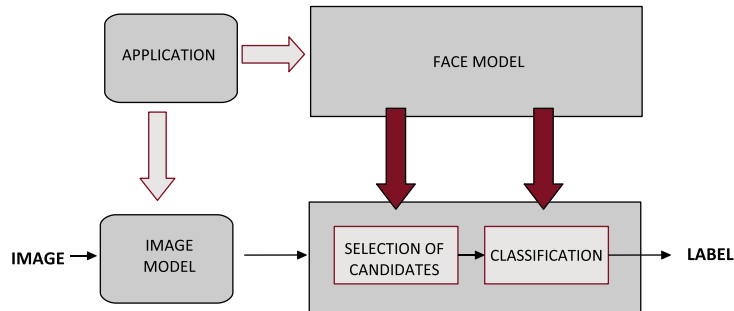


Figure 3.1: General face detection system.

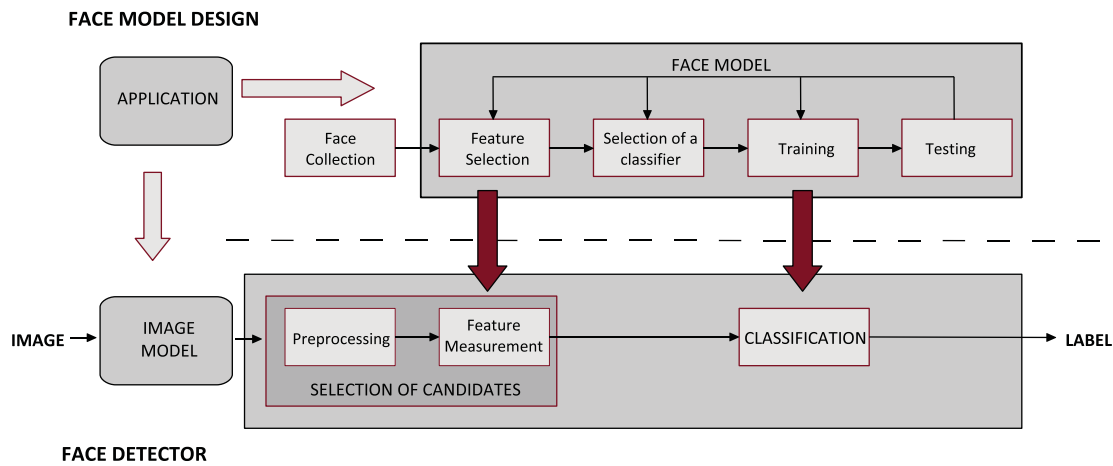


Figure 3.2: Face detection as a classification system.

backgrounds are very different scenarios with varying complexities which may require different strategies concerning the image or the face models.

The relationship between each part of a face detector and the general classifier system described in the previous chapter is illustrated in Figure 3.2. The *Face Model* block is what we called the training or design cycle of a classifier. It comprises the selection of a set of representative features, the proposal of a classifier model and the training and evaluation of the classifier. The *Selection of Candidates* gathers the preprocessing and the feature measurement on input patterns, and the *Classification* block performs the same task in both schemes. In the following, each part of the system shown in Figure 3.2 is described.

The Image Model: it defines the type of representation for the images and how to create it. The Image Model, therefore, characterizes the area of support of the patterns. The model can be one of the following:

- *Pixel based*: The image is understood as a set of independent pixels.
- *Block based*: The image is seen as a set of rectangular arrays of pixels. The rectangular areas are defined by scanning the image with a sliding window of fixed size.
- *Compressed domain based*: The image is seen as the set of coefficients in a particular transform domain, such as the DCT.
- *Region based*: The image is represented as a set of homogeneous connected components. The regions are obtained by segmenting the original image following a certain criterion of homogeneity.

The image model defines a set of potential candidates (e.g. pixels, blocks or regions). For example, this set is formed by all the possible rectangular subimages in a block-based image model (at all positions, of any size). We call this set the *search space*.

The Face Model: it is a characterization of the face class in terms of a set of attributes that can be measured on the image. This model gathers all the knowledge about faces that is needed to build the detector. Its definition is what in pattern recognition terms is called *the design cycle* or the *training mode* of a pattern recognition system. Most of the problems mentioned in Chapter 2 have to be addressed at this stage: how to select a good set of features, which classifier model to use, how to collect a ‘good’ training set, the risk of overtraining, how to measure the performance of the classifier, etc. The main issues to consider are:

- *A one class or a two class problem*: Face detection can be approached as a problem of classification in two classes or in one class. Two-class classification implies that a candidate pattern has to be assigned to one of two labels: face or no-face. In this case, the two classes have to be modeled, training samples of face and no-face patterns are needed, and a decision boundary between the two classes has to be inferred. In one class classification, only information about the face class is assumed. The training set contains only examples of face patterns, which are used to define a boundary around the face class.
- *Feature selection*: The goal is to describe the face class by measurements whose values are very similar for instances of the face class and very different for no-face instances. That is, the attributes should allow a good discrimination between faces and no-faces. On the other hand, since face patterns present large variability due to diverse factors such as facial expressions, illumination, viewing geometry, etc., features should also deal with as much of this intra-class variability as possible. When the two-class approach is followed, a characterization of a no-face class is also needed.

- *The face class:* The face class is described in terms of a set of *attributes*. In some cases, attributes are explicitly defined, using prior knowledge about how faces look like and a set of visual characteristics measured on the images (geometry, color, texture). Attributes are measured in the image by their associated *descriptors* and a feature vector that represents the pattern is obtained for further classification. In other cases, the attributes are implicit, in the sense that no assumptions are made about the structure, color or shape of faces and the discriminating features are selected through the learning process. For example, a common representation is a feature vector obtained by a raster scan of the pattern. During the training stage, the feature vectors of a set of labeled samples are input to the classifier. The classifier adjusts its parameters while it learns to discriminate face samples from no-face samples, without making explicit which features (within the feature vector) are the most discriminative.
- *The no-face class:* When the two-class approach is adopted, it seems very difficult, if not impossible, to explicitly describe the complete no-face class, so the modeling is usually performed implicitly through the learning. When the detector learns from examples, many examples are needed in order to get an accurate separation between classes, particularly for feature spaces of high dimension. In the case of faces, although there is a huge amount of negative examples (no-face patterns), not all of them are useful from a learning point of view. One strategy that is commonly used instead of modeling the complete no-face class is to model only a ‘near-face’ class, that is, a set of no-face images that lie close to the face class in the selected feature space, and which may be easily confused with faces.
- *Selection and training of the classifier:* The next steps in the design of the face model are the choice of the classifier type, its training and testing using a set of sample face (and possible non-face) patterns. The difficulty of the classification depends on the variability in the features’ values for patterns in the same class. There is also a relationship between the complexity of the feature selection process and the complexity of the classifier. As Duda and Hart say in [41], an ideal feature selection would yield to a representation where the job of the classifier is trivial, and conversely, an ideal classifier would not require a complex feature selection.

The Selection of candidates: this block may perform several tasks with the final goal of defining a set of candidate patterns that will be evaluated by the classifier. Not all the steps are present in a particular system.

- *Simplification* of the search by reducing the initial set of patterns. This step is not always applied: in some systems all the patterns defined by the Image Model are evaluated (e.g. sliding windows).

- *Precise definition* of the region of support for the remaining patterns (e.g. by masking the patterns with an elliptical mask, or modifying the area of support of a pattern to match the face shape).
- *Preprocessing* or normalization of patterns, with histogram equalization, illumination correction, etc.
- *Measurement* of features on the normalized patterns.

Figure 3.3 illustrates the analysis of an image using the four different image models and different strategies for candidate selection. In the first example, a pixel-based image model, pixels with colors very different from skin color are discarded, the remaining pixels are grouped and an ellipse is fitted to each group to define a candidate. The second example shows a block-based approach, where rectangular subimages of fixed size are extracted and masked. In the third example MPEG macroblocks with (average) skin color are selected and the smallest groups of macroblocks are discarded. Finally, the last example illustrates a simple strategy for a region-based image model. The image is segmented and skin colored regions are chosen as candidates.

Classification: in this block, candidate patterns are analyzed: the feature vector representing each candidate is input to the classifier that makes the final decision and assigns a ‘face’ or a ‘non-face’ label to the candidate. The Selection of candidates and Classification blocks define what, in the previous chapter, we called the classification mode of operation of the system (see Figure 2.1), since the classifier has already been trained during the Face Model creation.

Application: the choice of the image model may depend on the application. For instance, for creating a human face avatar from a photograph of a person, region-based models may be the most appropriate, whereas if the goal is to count the number of persons in the scene, a block-based representation will be enough. Also the complexity of the face model is different for color or gray-level images, simple or complex backgrounds, one or several faces per image, etc. The difficulty in obtaining training or test samples or constraints on execution time may also condition the design of the face model.

3.2 State of the art

Although being the first step in any face recognition system, the problem of face detection was relatively unexplored for its own sake until the mid 1990s. Early works were mainly devoted to the problem of localization in very constrained domains: mug shots, driver’s licences and passport images where there is one and only one face and the backgrounds are relatively uncluttered. An excellent review which focuses on face recognition but also describes detection approaches developed prior to 1995 is presented in [26].

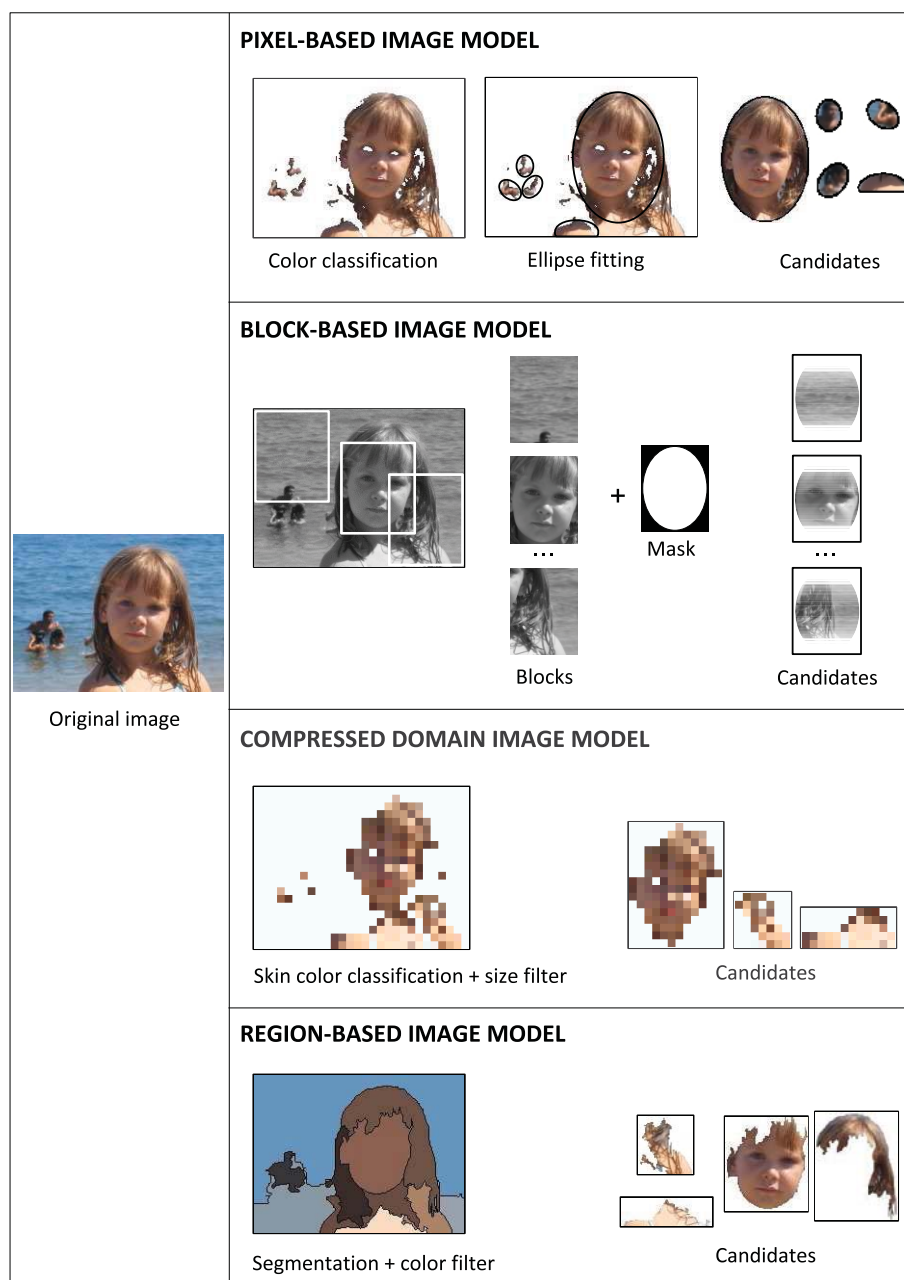


Figure 3.3: The four image models and selection of candidates.

Research activities on face detection have intensified since then. The interest is motivated by its application in very active and relatively new areas such as security, content-based image retrieval, video coding, crowd surveillance and intelligent human-computer interfaces. Such applications have different constraints in terms of processing requirements and thus pose a wide range of different technical challenges.

In this section we review the most important works on face detection. The problem has been approached with a diversity of strategies. This fact and the huge amount of publications related to the task make it difficult to organize the different approaches. Two detailed surveys on face detection techniques (developed before 2001) are [69] and [193]. In [193] they are classified in four categories: *knowledge-based*, *feature invariant*, *template matching* and *appearance-based* methods, though categories overlap, since several methods fall in more than one category. In [69], methods are organized in two broad categories distinguished by how they utilize face knowledge: *feature-based* methods, that build what we called explicit face models, and *image-based* methods, that incorporate knowledge implicitly through learning.

However, we understand that the main differences between strategies originate from the different models that are used to represent images and patterns. Therefore, we present the methods in four categories according to the four image models mentioned in Section 3.1. For each of the image models, we give a general description of the main parts of the systems: search strategy, selection of candidates, attributes used in the face models and type of classifiers. Then, details on the most representative techniques as well as references on similar works are provided. The comparison between techniques is postponed to Section 3.2.5.

3.2.1 Techniques for pixel-based image models

There are two main strategies for pixel-based image representations, one based on the detection of facial features and the other one based on pixel classification.

Detection of local facial features

These were the earliest methods. They start by detecting facial features and then infer the presence of a face according to the configuration of the features. Facial features such as eyebrows, eyes, nose, mouth, hair-line or face contours are commonly extracted using edge detectors. Anthropometric measures or statistical models that describe the relationship between features are used to verify the candidate configurations. The main problem of these techniques is that the features may be corrupted due to noise, illumination and occlusion. Several examples of these algorithms are proposed in the literature [62, 199, 198, 92, 101].

Their main limitation is that they were designed principally for face localization in very constrained environments. Under non-constrained scene conditions, such as presence of a complex background and uncontrolled illumination, most of the local facial features based



Figure 3.4: Classification of pixels into skin and no-skin classes in pixel-based methods. Original image (a) and skin-colored pixels (b).

methods are not stable.

Since in this thesis we are concerned with the more general problems of face detection and segmentation, we will not analyze these localization techniques. The interested reader is referred to [192, 69] for a complete review on facial-feature based approaches.

Grouping pixels

The heaviest part of the systems within this category is the selection of candidates. Explicit attributes of faces are employed to simplify the search by defining a very reduced set of candidate patterns, which are then classified by relatively simple classifiers.

Most of these techniques segment faces in color images. They use the color attribute of faces as a first cue and classify each pixel in the image into two classes, skin and no-skin, using a skin color model estimated with a training data set. An example of this initial step is shown in Figure 3.4. Then, skin-like pixels are grouped using connected component analysis or clustering algorithms. Next, a simple shape descriptor is used: if the shape of a connected region has an elliptic or oval shape, the pattern becomes a face candidate and is evaluated by a final classifier. Some methods use also a size descriptor to introduce additional constraints (application related) and discard too small or too large candidates. The following are some representative works that have adopted this approach.

Chai and Ngan use color information for face localization and segmentation in ‘head and shoulders’ type images in [24, 25]. They work in the YCbCr color space. Skin color values are modeled with two ranges in Cr and Cb components. Image pixels are classified as skin color pixels if both Cr and Cb values fall inside these ranges. The spatial distribution of detected skin-color pixels is analyzed, and morphological erosion and dilation are applied to

keep the largest clusters and discard small isolated objects, defining the candidate patterns.

For the classification, the variance of the luminance values is analyzed and candidate regions that are too uniform to be part of a face are discarded. Finally, geometric correction and contour extraction stages are used to correctly define the facial region.

Garcia and Tziritas also use a generic color descriptor as a first cue to find faces [56]. Color clustering is performed on the original image in order to extract a set of dominant colors and quantize the image according to this reduced set of colors. From a set of skin color samples in the YCbCr space, the envelope of the cluster is estimated. With this information, quantized pixels are classified as skin or no-skin colored.

A merging stage is iteratively applied on the set of homogeneous skin color regions in the color quantized image, in order to provide a set of candidate areas. Shape (aspect ratio of bounding box) and size descriptors are used to reduce the number of candidates.

For the classification, a textural analysis is performed on face candidates to remove false alarms caused by objects with color and aspect ratios similar to faces. The intensity component is analyzed by performing a wavelet packet decomposition on each candidate. Each candidate is described by band filtered images containing wavelet coefficients. A set of simple statistical data is extracted from these coefficients, in order to form the feature vectors, and a probabilistic metric derived from the Bhattacharyya distance is used to classify the feature vectors.

The face class is therefore modeled explicitly, using simple color, shape and size descriptors for the selection of candidates and specific texture descriptors for the final classification.

Other works: Terrillon et al. [174, 173] model skin color with a Gaussian distribution, classify pixels and group them in clusters. The smallest clusters are discarded and for the remaining clusters invariant Orthogonal Fourier-Mellin moments are extracted. These features are used with two classifiers: a multilayer perceptron neural network in [174] and SVMs with different kernels in [173].

Sobotka and Pitas [161] follow a similar approach for frontal face localization and find skin color clusters working in the HSV color space. In a second stage, facial features inside the components are extracted by evaluating the topographic gray-level x-relief and y-relief. For each face candidate, the vertical symmetry, the distances between facial features and the assessment of each individual facial feature are analyzed. The best candidate is chosen.

Other techniques based on color pixel classification are [68, 141, 143, 71]. Li et al. [93] perform face detection in sequences, and combine color with motion information to perform a first classification of pixels, assuming that faces are moving objects in a static background. Motion information helps to find a robust definition of patterns of interest.

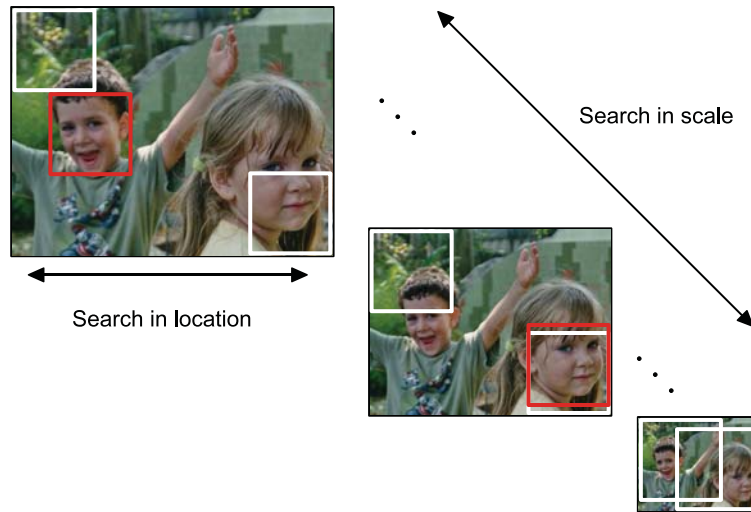


Figure 3.5: Scanning strategy for block-based image models.

3.2.2 Techniques for block-based image models

These techniques usually work with grayscale images, and represent patterns as squared windows. The input image is scanned with a sliding window of fixed size (typically 19x19 or 20x20). In order to detect faces of different sizes, the input image is repeatedly down-scaled, and the analysis is performed, with a window of the same size, at every scale. An example of this strategy is shown in Figure 3.5. At each scale of resolution, the system looks for faces that have the same size as the sliding window. White boxes in the pictures mark the first and last pattern analyzed at each scale, whereas red boxes show the two detected faces.

Some methods apply an exhaustive search, and analyze patterns over an exhaustive range of locations and scales in the image. In others, the search space is sampled and a scaling factor and steps for horizontal and vertical displacements of the scanning window are defined. This way, the analysis is faster but the localization is less accurate. The size of the window, the sampling rate and the step size may vary depending on the method and the need for a computationally efficient system.

The selection of candidates is quite simple. Each pattern defined by the image model is a final candidate after being preprocessed, that means that there is no simplification or reduction on the number of patterns to analyze. Patterns are usually preprocessed with masking, illumination gradient correction and histogram equalization to remove part of the background, to reduce noise, extra-light, shadows or to improve contrast on the image. Finally, a feature vector is extracted from the normalized patterns and is fed into the classifier.

A problem that arises with window scanning techniques is overlapping detections. This problem is usually solved with two heuristics: (i) counting the number of detections in a small

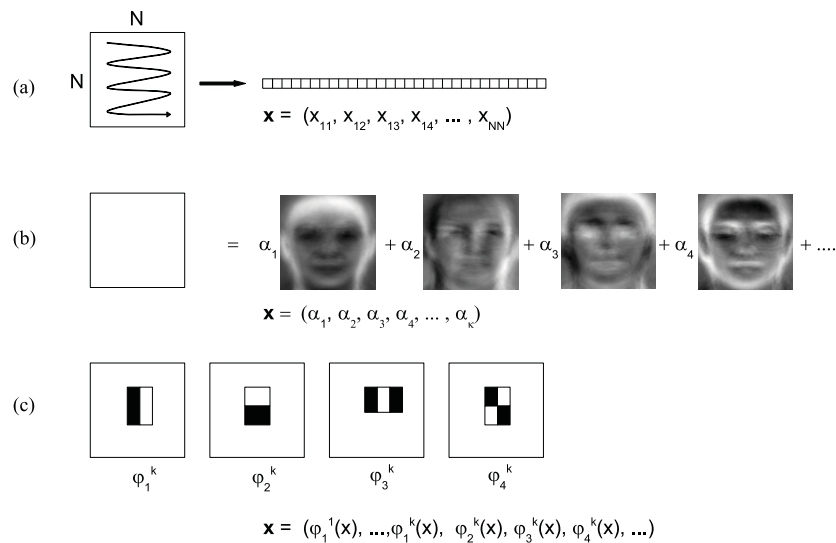


Figure 3.6: Features for classification in block-based methods. In (a), pixel features obtained by a raw by raw reading of the pattern. In (b), a global approach where the feature vector is formed by the coefficients of the projection of the pattern on a PCA computed from training data. In (c), local features computed within two, three and four rectangular areas; at each position k of the rectangles, the sum of the pixel values in the white rectangles are subtracted from the sum in the dark rectangles.

neighborhood of the location of the current pattern and output a detection if this number is higher than a threshold, and (ii) eliminating overlapping detections when a pattern is detected following strategy (i) [69].

To create the face model, a dataset of face images is used. Faces are scaled to the window size, aligned and preprocessed in the same way as the patterns.

For this group of techniques, the most important part of the system is the classifier, since it has to analyze a very large number of candidates.

For the review, we organize the techniques by the locality of the features used for the classification. In the simplest case, all the pixels in the (preprocessed) pattern are used for the classification. The feature vector is formed by a raster scan of the candidate pattern. This strategy is illustrated in Figure 3.6(a). A second group assumes a holistic approach and works with global features. Global representations like PCA encode a pattern in terms of basic functions that account for the variance in the face data set, as shown in Figure 3.6(b). More recent strategies exploit the idea that each pixel has significant statistical dependence only with a reduced set of neighbor pixels and compute a set of local features, as illustrated

in Figure 3.6(c). Finally, a fourth category is presented, describing techniques that tackle the computational burden inherent in block-based strategies with a coarse to fine cascade of classifiers, independently of the type of features.

Using pixel features

In these methods the feature vectors are formed with the intensity values obtained by a raster scan of the preprocessed patterns (see Figure 3.6(a)). In all the cases, the face and no-face classes are modeled with implicit attributes since the discriminating features are selected during the learning process.

Rowley, Baluja and Kanade report in [137] one of the earliest methods to detect upright frontal faces in grayscale images with a neural network based system. They approach the task as a two class classification problem, where the classes are implicitly modeled through learning.

The system has two components: a set of multilayer neural networks that detects face patterns and a decision making module that renders the final decision from multiple detection results. All the networks have the same form and are trained in a similar manner, but with random initial weights and permutations in the order of presentation of the training images. In this way, the networks have different biases and make different errors. Several arbitration schemes are proposed to produce the final decision ('and', 'or' and voting). The decision module merges overlapping detections and arbitrates between the multiple networks.

The method is extended in [138] to detect faces at any degree of rotation in the image plane. This procedure uses a separate neural network, called a 'router', to analyze the input window before it is processed. The router returns a rotation angle that is used to rotate the window to an upright position before passing it to the classifier.

Osuna, Freund, Giroi propose the first approach to face detection using Support Vector Machines (SVM) in [117]. They detect frontal, un-occluded faces in grayscale images by estimating the decision boundary between the face and non face classes.

An SVM is used to classify each candidate. The SVM is trained with a database of 19x19 face and non-face patterns, using a second order polynomial as kernel function. Once the decision surface has been obtained, the system is used over images that do not contain faces and misclassifications are used as negative examples in subsequent training phases. As training an SVM using large data sets is a very difficult problem, they propose a decomposition algorithm based on the iterative solution of subproblems and the evaluation of optimality conditions that are used to generate improved iterative values and to establish the stopping criteria for the algorithm.

Other works: Feraud et al. present in [46] a method based on a constrained generative model, Lin et al. [96] propose a system based on probabilistic decision-based neural network, Roth et al. [136] apply an architecture called SNOW (sparse network of Winnows).

Using global features

These methods assume that images of human faces (of the same size) lie in a subspace of the overall image space. Principal component analysis, linear discriminant analysis or factor analysis are used to represent this subspace. The feature vector is extracted from the representation of the pattern in the subspace (see Figure 3.6(b)).

Moghaddam and Pentland propose in [111, 110] a one-class approach, a probabilistic learning method for face localization based on density estimation to model the face class and locate faces in grayscale images.

They assume a Gaussian distribution for frontal face patterns (and a mixture of Gaussians for faces in multiple views) in the image space. Then, principal component analysis is applied to find an estimate of the face likelihood of an input pattern using only the projections of the pattern into the subspace spanned by the first eigenvectors. The face likelihood of each pattern relies on an estimate of the Mahalanobis distance between the pattern and the face class.

The estimated probability density is used to formulate the face localization problem in a maximum likelihood framework. Given an input image, a saliency map is computed by estimating, at each spatial position, the face likelihood of the subimage centered at that point. The ML estimate of the position of the face is given by the position that leads to the maximum value in the map.

Sung and Poggio follow a two class classification approach to detect multiple frontal or slightly rotated faces in grayscale images by modeling the class distributions [166].

The distribution of a face class and a ‘near-face’ class are modeled using a training set of face and face-like patterns. In both cases they apply a modified k-means algorithm to find six Gaussian clusters. The face training set is formed by frontal face patterns resized to 19x19 pixels, and the same patterns slightly rotated and mirrored. Principal component analysis is applied to each cluster to reduce the dimensionality of the representation; only the projection of the cluster in the subspace spanned by the first 75 eigenvectors is considered.

For each candidate, a feature vector of 12 distances between the normalized pattern and the 12 cluster centroids is computed. The distance between the pattern and each cluster has two components: the normalized Mahalanobis distance between the pattern and the cluster centroid, and the Euclidean distance between the pattern and its projection in the

subspace. Finally, the feature vectors are input to a multilayer perceptron network trained with backpropagation that performs the final classification. They also report tests using a nearest neighbor classifier.

Other works: A technique based on mixtures of factor analyzers and another one based on Fisher's linear discriminant are proposed by Yang, Ahuja and Kriegman in [195]. Other linear subspace methods are presented in [86, 108]. Popovicy and Thiran train an SVM with PCA features in [127].

Using local features

Techniques within this group are based on the idea that (aligned) face images have a sparse structure of statistical dependence. Each variable (pixel) has a strong statistical dependency with a small number of other variables and a negligible dependency with the remaining ones [154].

Papageorgious and Poggio propose in [120] a new representation describing patterns in terms of a large set of local oriented intensity differences between adjacent areas in the image. The features are computed with an over-complete Haar wavelet transform.

A support vector machine classifier is trained with a large set of positive and negative samples. The system has a relatively slow processing speed since it works with a large number of local features (1740 features for each pattern of size $19 \times 19 = 361$) and performs an exhaustive search on the image.

Schneiderman-Kanade present in [157] a system based on a naive Bayes classifier. They choose a functional form of the posterior probability function that captures the joint statistics of local appearance and position. Using a discrete representation of local appearance the overall statistical model is estimated by counting the frequency of occurrence of the patterns over training images. The model is applied to the detection of frontal and profile faces.

Other works: Colmenarez and Huang also use local statistics in a system that models classes with a family of discrete Markov processes and selects the process that optimizes the information-based discrimination between the two classes in [29, 30].

Coarse to fine processing or rejection based methods

A drawback of traditional block-based approaches is that the application of complex classifiers on an exhaustive search can be very time consuming. One way to reduce the computational

cost is to build the classifier as a cascade of classifiers. At each stage, a classifier makes a decision to either reject the pattern or to continue the evaluation using the next classifier. Usually the complexity of the classifiers increases from the first to the last classifier. This strategy is designed to remove most of the candidates with a minimum amount of computation. Most of the image is quickly rejected as background, and more complex classifiers concentrate on the most difficult face-like patterns. A scheme of a rejection-based classifier is shown in Figure 3.7.

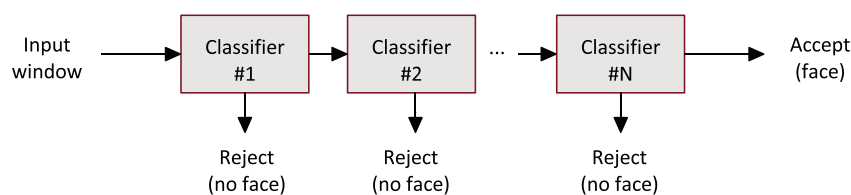


Figure 3.7: Cascade of classifiers

The most recent state-of-the-art classifiers combine this coarse to fine strategy with the use of local features to reach high performance, both in speed and detection rates. We will briefly describe the most relevant works that follow this approach.

Fleuret and Geman propose in [48] a face detector based on a nested family of classifiers. The detection is based on edge configurations: the global face form is decomposed into a set of correlated edge fragments corresponding to the presence of some facial feature. Each classifier is a succession of tests dedicated to certain locations and orientations in the input pattern. When going through the different levels in the hierarchy, the classifiers are more dedicated to particular locations and orientations. The approach is coarse to fine both in terms of pose and complexity of the classifiers.

Viola and Jones combine the idea of a cascade of classifiers of [48] with efficient computation of local features and a learning algorithm based on Adaboost [51] in the system proposed in [185], leading to one of the current state-of-the-art methods.

They work with an intermediate representation of the image, called ‘integral image’ that allows a fast computation of a set of local, Haar-like features. The search is done by scanning this integral image and applying the classifiers at every location. A very large number of simple features is computed from the integral image. Each feature is associated with a ‘weak’ (a very simple) classifier that applies a threshold on that feature. Three types of rectangle features are used, that compute differences between the sum of the pixel values within rectangles (in

two, three and four rectangles as shown in Figure 3.6(c)). This produces over 180.000 features for patterns of size 24x24, a many times over-complete set of features.

The approach is based on combining a set of weak classifiers in order to build a stronger one. At each step of the training, and among all possible weak classifiers, one is selected such that its error rate on the training set is minimal. The parameters of this classifier are estimated and the global (strong) classifier is given as a linear combination of the selected weak classifiers. In this iterative process, weak classifiers which are selected early yield minimum error rates while weak classifiers selected in further iterations make larger errors as the discrimination task becomes harder. The approach is coarse-to-fine in terms of complexity of the classifiers.

Fast processing is achieved by combining successively stronger classifiers in a cascade. At the beginning a simple two-feature strong classifier is used to filter many background structures. The number of features in different steps of the cascade increases and this makes the classifiers more and more discriminating (and more expensive in terms of training) than the classifiers at the earlier steps. The thresholds of strong classifiers close to the beginning of the cascade are adjusted to have no miss-detections.

Each classifier in the cascade is trained on the negative samples of the previous classifiers. This combination allows background zones in the image to be quickly discarded while spending more computational resources on 'face-like' candidates.

The work is extended in [83] to handle profile views and rotated faces. Different detectors are built for different views and poses, and a decision tree is trained to determine the point of view for each candidate, so that only the detector for the selected view is run, an approach similar to Rowley's in [138].

The technique is very fast due to computation of features through the integral image and the use of the cascade to rapidly reject most of the candidates, and has a high performance thanks to the training with Adaboost. However, it has some weak points: it is very difficult and time-consuming in training, and requires an extremely large amount of no-face samples, since the classifiers in the cascade are trained on false detection samples from previous classifiers.

H. Sahbi, D. Geman and N. Boujemaa present in [144, 142] a face detector system for grayscale images based on a hierarchy of SVM classifiers.

For each pattern, a feature vector with 8x8 low frequency coefficients of the Daubechies wavelet transform is computed. The hierarchy of SVM classifiers, structured as a tree, provides a coarse to fine search. The complexity and discrimination of the classifiers increase in proceeding from the root to the leaves. The hierarchy is built by dividing the set of possible poses in a nested family of sets. To build each classifier, a simplified SVM decision function is learned using a subset of support vectors. The number of support vectors, which defines the complexity of each classifier, depends on its level in the hierarchy.

A candidate is a face if there is a complete chain of positive answers -through a complete path from the root to one leaf- and it is a non-face if there is a collection of negative responses for a set of classifiers that covers all poses.

Schneiderman-Kanade present in [158] a system of three classifiers to detect faces in three poses: frontal, left profile and right profile. Each classifier is based on the statistics of localized ‘parts’. Each part is a transform from a subset of wavelet coefficients to a discrete set of values. A total of 17 parts (each consisting of 8 wavelet coefficients) are extracted, capturing various combinations of locality in space, frequency, and orientation. Class conditional statistics of the parts are computed from a training set and represented with tables. The classifiers are trained with a modified Adaboost technique. Each classifier computes the part values for each candidate and makes a decision by applying a likelihood ratio test. For efficiency, the classifier evaluates this likelihood ratio in stages. At each stage, the classifier compares the partial likelihood ratio to a threshold and makes a decision about whether to stop evaluation (labeling the input as non-face) or to continue further evaluation.

Other works: Zhang et al. propose in [203] a modification to Viola and Jones’ technique, using global features derived from PCA in the later stages of boosting (each weak classifier is based on a single feature, one of the PCA coefficients) to increase the detection rate of the last classifiers in the cascade.

Romdhani et al.[134] merge the SVM approach of Osuna with the idea of a cascade of evaluations of Viola and Jones and the Antifaces of [85]. The goal is to use the cascade to build a system faster than those based only on SVMs. A reduced set of vectors is used to obtain a set of SVM classifiers of increasing complexity. The classifiers are combined in a cascade. They use Gaussian kernels in the SVMs, and bootstrapping for the learning.

A semi-naive Bayes classifier that decomposes the input pattern into subsets and represent statistical dependency within each subset is described by Schneiderman in [154]. The technique is extended in [156] to work with a Bayesian network. The same author proposes in [155] a cascade method analogous to [185] that uses simple fast feature evaluation in early stages of the cascade and more complex discriminative features at later stages. A ‘feature-centric’ computational strategy is proposed to re-use feature evaluations across multiple candidate windows.

3.2.3 Techniques for compressed-domain image models

Motivated by the need of fast and efficient search and index algorithms for visual information stored in a compressed form in large databases, some researchers proposed to work directly in the compressed domain. Generic features (mainly for color and shape attributes) are extracted

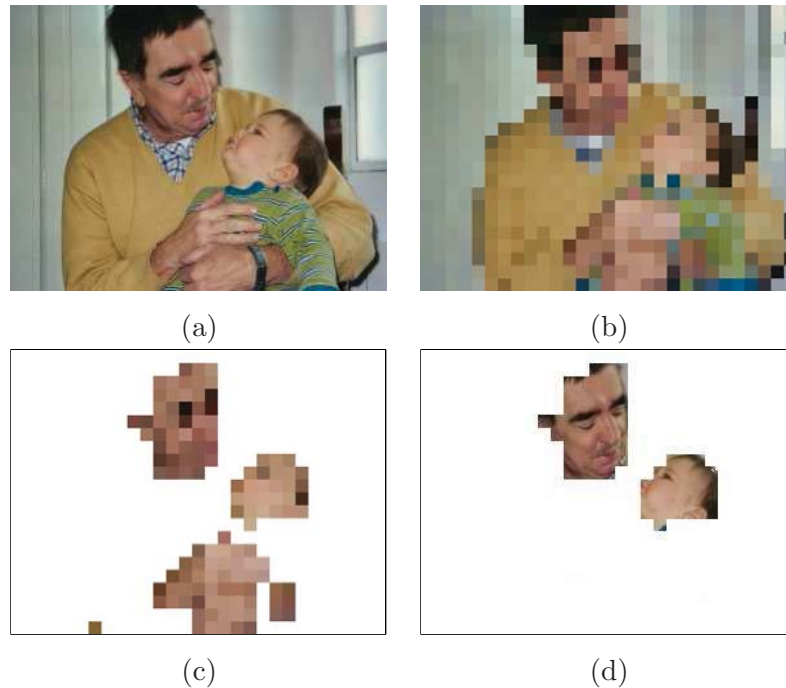


Figure 3.8: Compressed-based method. Original image (a), macroblocks represented with the average color (b), skin-colored macroblocks (c) and final faces (d).

from a downsampled version of the image, like in pixel-based methods. Then, specific features (texture) are extracted and evaluated at full resolution.

Wang and Chang work with MPEG compressed video in [187], describing a fast algorithm to detect faces in I frames. The technique works at the macroblock level, working directly with the DCT coefficients. Therefore, a minimal decoding of the video is required. The method is illustrated in Figure 3.8.

The selection of candidates is performed by analyzing color and shape information. The distribution of skin color in the CbCr chrominance plane as well as face aspect-ratio constraints are learned from training data. The DC values of Cb and Cr blocks are used to represent the average chrominance of each macroblock. In a first stage, macroblocks are classified as skin or non-skin colored. Then, morphological operations are performed to eliminate noise and fill holes in the resulting skin colored (block-based) regions, and shape information is used to select the final set of candidates (constraints on aspect ratio, minimum and maximum face size). The final classification is performed with two simple classifiers based on the energy distribution of the luminance DCT coefficients over different frequency bands.

Other works: Lu and Eleftheriadis [98] combine the previous system [187] with the texture classifier of Sung and Poggio [166] mapped from the pixel domain to the DCT domain. They do not scale the image to detect faces of different size, as commonly performed in block-based approaches. To avoid the hard problem of resolution transforms in the DCT domain they train different classifiers for several image scales. Other techniques that work on the compressed domain are [49, 118].

3.2.4 Techniques for region-based image models

Within this group we find techniques that partition the image into a set of homogeneous regions using a generic segmentation algorithm, and then merge and analyze the regions using visual attributes in order to find face instances. There are few approaches following this strategy. We briefly describe one technique for gray-level images and another two that segment the image and select candidates based on skin-color information.

Vilaplana and Marques propose a region-based technique for the segmentation of faces in gray-level images with simple backgrounds [182]. The image is represented by means of a max-tree and a min-tree [148], where leaves are the regional maxima and minima of the image, respectively, and the remaining nodes represent flat zones, ordered by the grey level value differences between regions. The face class is modeled with a Gaussian distribution in the image space, and principal component analysis is applied to find an estimate of the face likelihood of each region in terms of the projections into the subspace spanned by the first eigenvectors. A filtering stage analyzes the tree branches in the max-tree, and selects the nodes with highest likelihoods (avoiding local maxima), obtaining the bright components of the faces. The results are combined with a dual operator, that selects face dark components. Since max and min trees are extremum oriented, nodes do not represent complete objects in the scene. A post-processing stage is applied on the selected components to extract the complete face.

Yang and Ahuja [194] perform a multiscale segmentation [2] to generate homogeneous regions at multiple scales. Using a Gaussian skin color model, regions of skin tone are extracted by merging regions from the coarse to the finest scale and grouped into elliptical shapes. A candidate is classified as a face if facial features (eyes and mouth) exist within the elliptic region. Neither details on the merging of skin regions nor about the extraction of facial features are given.

Liu, Yang, Peng present in [97] a technique where the image is first segmented using a watershed algorithm. Then, pixels are classified as skin or non-skin colored, and the result of the classification is combined with the segmentation to obtain skin-colored regions. Facial

features are extracted and used to refine the skin-like regions. A merging algorithm is applied on the remaining regions, and the facial likeness of original and merged regions is evaluated using a set of fuzzy membership functions devised for a number of features (aspect ratio, orientation, compactness, density, area). The regions with likelihood larger than a threshold are accepted as face instances. Since several regions may be accepted for the same face, further analysis is performed to select the most representative region for each face.

Other works: A very preliminary version of the system presented in this thesis is [104], where only one PCA-based classifier is used to classify nodes from the original Binary Partition Tree [145].

3.2.5 Discussion

When comparing the different methods for face detection, the most important criteria are probably detection rate and localization precision. However, the performance criteria, which are application dependent, may cover other aspects of practical importance, like algorithm complexity, hardware requirements, scalability, training cost, execution time, etc. [128]

A fair comparison between the different techniques is not always possible due to the lack of standard performance evaluation procedures: even if results are presented on the same test set, the criteria of what a correct detection means usually differ, as well as the terminology used to describe the results.

When evaluating and comparing the different methods for face detection, the following aspects should be considered:

Face definition and system output: the type of output is different for systems that perform localization, detection or segmentation. Moreover, in most systems the adopted image model defines the output. In block-based methods outputs are rectangles around the main facial features. Pixel-based methods that search for facial features output a set of points that indicate the location of these features. Pixel-based methods that use visual cues to focus attention and also region-based methods define precisely the facial contours. Finally, compress-domain methods output an approximation of the facial area with a localization error due to the use of macroblocks as basic units.

Also the lack of standard terminology to describe results makes it difficult to compare algorithms. First, the criteria to define what a successful detection is may differ. While one algorithm may consider a successful detection if the bounding box contains eyes and mouth, other may require the entire face (including forehead or hair) to be enclosed in the bounding box. Most face detection algorithms do not clearly define what a successful detection is. Second, the performance measures are diverse. Most of the published works only provide detection and error rates to show the quality of their systems, but they

Image Model	Strategy	Representative works
Pixel-based	Local facial features	Govindaraju [62] Other works [199, 198, 92, 101]
	Grouping pixels	Chai and Ngan [24, 25] Garcia and Tziritas [56] Terrillon [174, 173] Sobotka and Pitas [161] Li, Gong et al. [93] Other works [68, 141, 143, 71, 93]
Block-based	Using pixel features	Rowley, Baluja and Kanade [137] Osuna, Freund and Giroi [117] Other works [46, 96, 136]
	Using global features	Moghaddam and Pentland [111, 110] Sung and Poggio [166] Other works [195, 86]
	Using local features	Papageorgious and Poggio [120] Schneiderman and Kanade [157]
	Rejection-based methods	Fleuret and Geman [48] Viola and Jones [185, 83] Sahbi et al. [144, 142] Schneiderman and Kanade [158] Other works [85, 203, 134, 154, 156, 155]
Compressed-domain		Wang and Chang [187] Other works [98, 49, 118]
Region-based		Vilaplana and Marques [182] Yang and Ahuja [194] Liu et al. [97] Other works [104]

Table 3.1: Face detection strategies.

do not mention the way they count detections and errors to compute those rates. A good detection for one system may not be sufficient for others. In general, two kinds of methods are used to count the detections: manual, where correct detections are manually counted, and automatic, where the database is annotated (e.g. with the eye positions) and then a correct detection is accounted if some measure of distance between the detected face and the ground truth is below a threshold.

Finally, the goodness or precision of the detection also depends on the type of output. A displacement of a few pixels in a rectangular representation may be tolerated whereas the same displacement on a facial feature point or a badly fitted contour may be visually unacceptable.

System design and training cost: The previous review was already based on design issues of the methods; in this subsection, therefore, we only highlight the most important differences.

Besides localization techniques based on the detection of facial features, the earliest detection systems are block-based and have to analyze hundreds of patterns per image with the same classifier. For instance, for a QCIF image (176x144), a system that performs an exhaustive search at full size with a sliding window of 20x20 pixels has to analyze 19625 candidates. The analysis is then repeated at lower scales. Therefore, a huge number of patterns is classified with the same classifier, a very time consuming process with the risk of producing a very high number of false positives.

However, most of the candidates are no faces and can be rejected with a relatively simple classifier. This idea led to the use of cascades of classifiers of increasing complexity, with the goal of rapidly discarding most of the candidates. This solution can be understood as a simplification task performed by the first classifiers in the cascade. The last classifiers concentrate on the most 'difficult' (face-like) patterns. Systems of this kind, once trained, are very fast and accurate, achieving real-time performance. The costly part is the training, both in time and in the size of the training data that is required to obtain a well trained system.

On the other side, pixel, region and compressed based methods also perform a simplification but at an early stage, and spend more efforts on defining a very reduced set of candidates. The final classifiers tend to be simpler and easier to train than cascades. In these systems, the most costly part is this simplification: the classification and grouping of pixels or macroblocks in pixel and compressed based methods, respectively, and the initial image segmentation in region-based strategies. In general, they are not so fast as block base approaches but the localization is much more precise. Table 3.1 summarizes the reviewed techniques in terms of design.

Training and test sets: Most face detection techniques require two types of data sets of

images, one for training and one for testing.

For the training, data sets originally designed to measure the performance of face recognition or verification methods are commonly used. As images in these sets were collected for recognition, each image contains only one individual, and there are several images per subject. In general terms, images within each set were captured with the same camera, under controlled conditions, with uncluttered backgrounds, and images present slight variations in facial expressions, pose and illumination. The most well known data sets are FERET [124], AT&T (former ORL dataset) [150], Purdue AR [106], M2VTS [126], XM2VTS [109] and Banca [9], which are briefly described in Table 3.2.

On the contrary, there are really very few publicly available data sets specifically designed to evaluate face detection or segmentation algorithms. Therefore, researches usually test their systems with their own sets, which are not available to other researchers making it difficult the comparison of results between techniques. Two public data sets that where collected specifically for face detection on grayscale images are the CMU & MIT set [137, 166], a difficult data set with several faces per image, and images of varying quality, and BioID [81], a set with one face per image and variability in backgrounds, face size and illumination. The CMU & MIT set has emerged as standard testbed for block based methods working with grayscale images. Tables that show systems and performances on this test set can be found in [69, 193]. To test systems that work with color images, usually BANCA or XM2VTS are used, in addition to images collected from other sources which are not available to other users. Examples of some of these data sets are shown in Appendix A.

Performance metrics: It is not easy to compare the performances of face detection systems, even when they produce the same type of output or use the same data sets for testing. The main problems are the lack of a standardized testing procedure and the inaccurate or incomplete information in the publications [69].

In order to have a fair comparison, results should report detection rate, false positive and false negative rate, besides the description of how these values are computed. They should also give training and execution times, and whenever possible, a ROC curve to analyze the tradeoff between false positive and false negative rates. Many of these questions are left unanswered in most of the papers, introducing a degree of uncertainty in the systems performance.

In the last years, however, there have been some efforts towards the definition of standard procedures and metrics, like [128], or works in the context of the CHIL project [130] and the US ARDA VACE program [177].

Dataset	Description
FERET [124]	Grayscale face images (14051) with views ranging from frontal to left and right profiles.
AT&T (former ORL)[150]	Grayscale, 40 subjects, 10 images per subject
Purdue AR [106]	Color, 3267 faces with different facial expressions, illumination and occlusions.
M2VTS [126]	Color face sequences with simple background and audio for multimedia identification, 37 subjects, 5 shots per subject.
XM2VTS [109]	Multimodal, contains color face images, audio, video sequences and 3D models for multimodal identification, 295 subjects.
Banca [9]	Color, 208 individuals (52 from each country - France, G. Britain, Italy, Spain), 3 scenarios (controlled, adverse, degraded), 4 sessions per scenario, 10 frontal face images per person per session (a total of 6240 images).
CMU & MIT [137, 166]	Grayscale, 130 images with 507 faces, high and low resolution images with multiple faces of varying size in complex backgrounds.
BioID [81]	Grayscale frontal face images of 23 subjects with large variety of illumination, background and face size (total of 1521 images).

Table 3.2: Face datasets.

3.3 System Overview

In this section we present an overview of our face detection and segmentation system, following the framework proposed in Section 3.1.

To jointly detect and segment the faces, we work with a region-based image model. The choice of this type of model introduces new problems. First, we need to find a good segmentation of the image, where region contours correspond to face contours, so that the faces can be found by the union of regions in the partition. Second, since the analysis of all possible combinations of regions would probably be intractable, we need a mechanism to reduce the number of possible unions to be analyzed. At the same time, we must ensure that we can find the faces among the proposed unions. In Chapter 4 we will show that Binary Partition Trees are a good solution for these problems.

Moreover, the construction of the BPT does not depend on any object class, making them useful for other applications. For example, pre-computed BPTs can be used to detect different object classes (Chapter 8) or in a query by region example application, where the user selects or marks a region in the image and the system searches for a similar region among the nodes of the BPTs.

Concerning the learning of the face model, we understand that the one-class paradigm (Chapter 2) describes the face detection problem better than the two-class approach, since the class of non-face images potentially consists of all other occurring patterns. Therefore, the model is built in terms of one-class classifiers based on region descriptors. Details on the face model are discussed on Chapter 5 and Chapter 6

A block diagram of the complete system is shown in Figure 3.9. In the following, we briefly explain its parts.

Image model

Each input image is represented with a Binary Partition Tree. The tree is a hierarchical representation of the image at different resolution levels which proposes a simplification of the search space. The search is performed at the scales (region size) and location (region position) defined by the tree nodes.

Face model

The face model characterizes the face class in terms of descriptors and classifiers. Taking into account its use we organize descriptors into three groups: generic descriptors, a shape descriptor and specific descriptors.

- **Generic descriptors:** they are related to low-level visual attributes which are common to any object. For instance, a color descriptor like the average color of a region or a

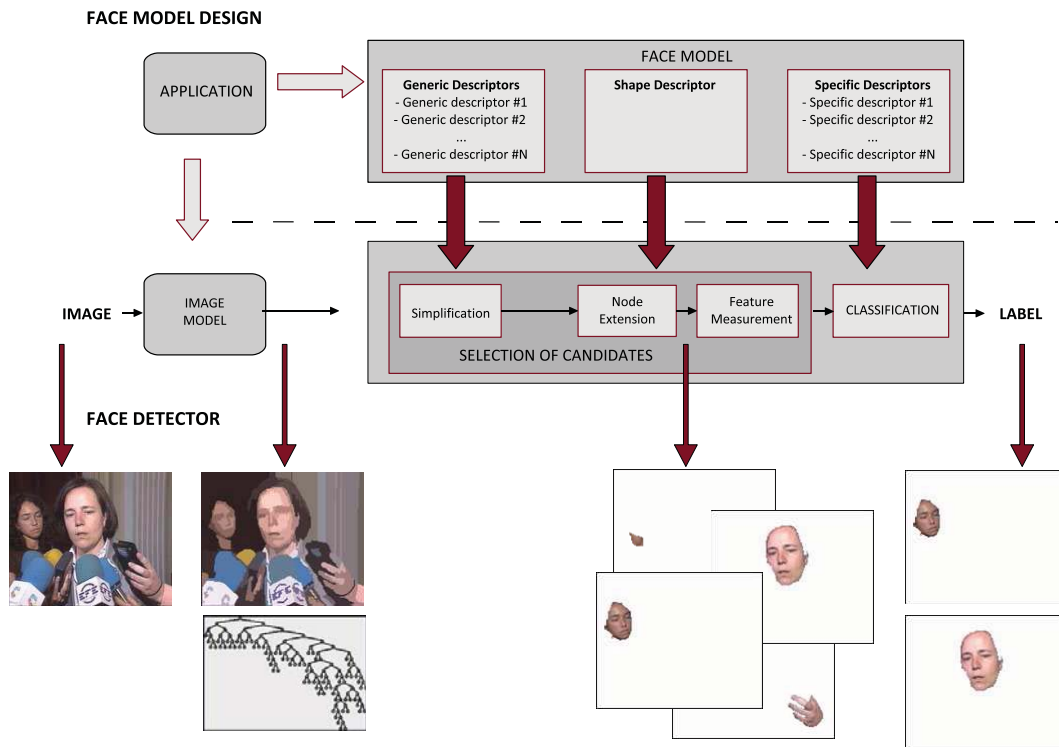


Figure 3.9: Our face detection system.

geometric descriptor like the aspect ratio belongs to this group. In general, they are simple and relatively easy to measure. Each descriptor is associated with a simple binary classifier which is trained to accept as many face patterns as possible (Section 6.2).

- **Shape descriptor:** it is an approximate model of the face shape (an ellipse), used to improve the node representation (e.g. to solve possible segmentation errors) (Section 6.3).
- **Specific descriptors:** they are descriptors related to attributes that are specific to faces. Examples of specific descriptors may be a measure of reflectional symmetry or coefficients of the PCA computed for a training set of face samples. They are more complex and costly to compute than generic descriptors but they are evaluated only on a small set of regions. Each specific descriptor is associated with a soft classifier (a classifier with continuous output values) (Chapter 5).

Selection of candidates

- **Simplification:** each node passes through a cascade of classifiers. Each classifier has a binary output (0 or 1), and it is based on one of the generic descriptors. If one node

is rejected by one classifier, it is not analyzed by the next classifiers in the series.

We call *active node* a node that is accepted by all classifiers in this step. Therefore, generic descriptors allow a further simplification of the search space. This is a hard selection stage, where we try to eliminate patterns that are very different from faces in terms of the generic descriptors and the constraints introduced by the application. For example, patterns with colors very different from skin color or patterns that are too small or too wide may be rejected (see Section 6.2).

- **Node extension:** an ellipse is fitted to each active node (shape matching based on distance transforms), and the area of support of the node is modified by adding or removing small regions to conform to the shape model. We call *extended node* the new area of support of each active node. The set of extended nodes define the set of face candidates (see Section 6.3).

Classification

Candidates are evaluated and assigned a ‘face’ or ‘non-face’ label. Each specific descriptor has an associated classifier that outputs, for each candidate, its likelihood of being a face instance. The classifiers are combined in an ensemble that assigns to each candidate a face likelihood. Finally, candidates with large enough likelihood are selected as face instances (see Section 5.4 and Section 6.4).

3.4 Summary

In the first part of the chapter, we have proposed a general scheme to analyze and compare face detection systems, identifying its main components as parts of a classification system: the Image Model, the Face Model, the Selection of Candidates, Classification and Application blocks. The state of the art on face detection has been reviewed taking into account the four possible image models commonly used by the techniques (pixel, block, compressed or region-based). We have seen that a fair comparison among them is not possible since there are no standard evaluation procedures, and also because published results differ on the evaluation criteria, test sets and type of output. In this chapter we have also provided an overview of the face detection system that will be developed in the next chapters.

The review work presented in this chapter has contributed to the following book chapters:

- M. Pardàs, V. Vilaplana, C. Canton, *Image and Video Processing Tools for HCI*, in *Multimodal Signal Processing for Human Computer Interaction*, pp. 93-118, Elsevier, 2009, ISBN 978-0-12-374825-6.

- M. Gurban, V. Vilaplana, J.P. Thiran and F. Marqués, *Face and speech interaction*, in *Multimodal user interfaces: from signals to in-teraction*, edited by D. Tzovaras, pp. 102-141, Springer, 2008, ISBN 978-3-540-78344-2.

Chapter 4

The image model

In this fourth chapter we discuss the use of Binary Partition Trees (BPT) for object detection and, in particular, for face detection. Binary Partition Trees are hierarchical region-based representations of images that define a reduced set of regions that covers the image support and that spans various levels of resolution. In this chapter several issues related to the construction and usefulness of the BPT for, initially, face detection, and afterwards, object detection are studied. In the first sections we analyze the compromise between computational complexity reduction and accuracy, which leads us to define two parts in the BPT, one providing accuracy (the BPT accuracy space) and one representing the search space for the object detection task (the BPT search space). Then, we objectively compare various similarity measures and stopping criteria for the tree construction.

4.1 Region-based image representations

Any object detection strategy relies on a particular image representation or *image model* that defines a set of patterns to analyze. The image representation should compact all the information in the scene in a small number of elements. At the same time, it should be as generic as possible so that this representation can be reused in other contexts (e.g.: searching the same image for different objects). As discussed in Chapter 3, common image models are (i) pixel-based representations, where the image is understood as a set of independent pixels, (ii) block-based representations, where the image is seen as a set of rectangular arrays of pixels, (iii) region-based representations, where the image is represented as a set of homogeneous connected components, and (iv) compressed domain representation, where the image is seen as a set of coefficients of a particular transform domain.

While pixel- and block-based representations are simple to define, they produce a large number of elements to be analyzed. Detecting objects on these representations is a difficult task. In pixel-based representations the elementary unit, the pixel, provides extremely local

information. Pixels have to be grouped and isolated pixels or small groups discarded. In block-based representations the unit is a rectangular array, and although the information is not so local as in the previous case, the size of the block is fixed a priori, without taking into account the visual content of the signal. Therefore, the scale of representation is in most cases too far from the interpretation scale. Moreover, since the task is to detect not only the presence of the object but also its position and its scale, usually an exhaustive search at different scales and different positions is required, and algorithms must deal with a very large number of blocks.

On the contrary, the definition of region-based and compressed-domain representations involves complex image processing operations but largely reduces the analysis space. In particular, region-based image representations provide additional advantages: regions may provide a first level of abstraction towards a semantic representation of the image, and the results may be more robust as features are computed from regions that follow some homogeneity criterion [147].

Region-based image representations provide a simplification of the image in terms of a reduced number of representative elements, which are the regions. In a region-based image representation, objects in the scene are obtained as the union of regions from an initial partition. Since an arbitrary partition may contain a large number of regions, the number of possible unions among these regions can be very large. Actually, a region-based representation implies a compromise between accuracy (related to the number of regions in the partition) and processing efficiency (related to the number of unions to be analyzed).

One approach to solve this compromise is to reduce the number of possible unions of regions by proposing the most likely ones, given a specific criterion. This can be performed by creating a hierarchy of regions representing the image at different resolution levels¹. This region-based multi resolution representation allows the analysis of the image at multiple scales [147]. The idea is to have not only a single partition but a universe of partitions representing the image at various resolutions. In this context, object detection algorithms only need to analyze the image at the positions and scales that are proposed by the regions in the hierarchy. From this set of regions at various resolution levels, an application dependent algorithm can select the most convenient one(s) for its concrete application. The selected region(s) may represent the complete object to be detected or only provide a good approximation. In this case, this approximation can be used as an anchor point for launching a refining process which leads to the complete object.

There exist different approaches to hierarchical region-based image representation, usually related to tree data structures. In these structures, tree nodes are associated with regions

¹In this work we are not referring to multi resolution as a multiple representation in which, at each resolution, the image is represented by a different number of pixels (see [21]). In the region-based framework, the multi resolution notion is related to the number of regions at each level.

in the image whereas tree links represent the inclusion relation. In the quad-tree image representation [151] [125], the image is recursively subdivided into four equal rectangular regions (quadrants) and, therefore, each node has either four children or no children (a leaf). As the subdivision strategy does not depend on the image content, many false contours are introduced in the image representation and it is difficult to use it for general object detection tasks.

The min-tree and max-tree representations [148] adapt to the image content because they describe the image flat zones (the largest connected components of the image where the image is constant). The leaves of these trees are the regional minima and maxima of the image, respectively for the min-tree and the max-tree. The remaining nodes, which represent flat zones, are ordered by the gray level value differences between regions. Therefore, min-tree and max-tree representations are devoted to dark and bright image components, respectively. As a result, the regions associated with their nodes may not conform to real objects in the scene. Note however that many real objects may not coincide with minima or maxima of the image.

The component tree [112] or tree of shapes [10] merges the min-tree and max-tree into a single representation. It represents an image in such a way that maxima and minima can be simultaneously handled. Nevertheless, since the resulting tree is still extremum oriented, nodes commonly do not represent objects in the scene.

The Binary Partition Tree (BPT) [145], unlike the previous approaches, reflects the similarity between neighboring regions. It proposes a hierarchy of regions created by a merging algorithm that can make use of any similarity measure. Starting from a given partition, that can be as fine as assuming that each pixel or each flat zone is a region, the region merging algorithm proceeds iteratively by

1. computing a similarity measure (*merging criterion*) for all pair of neighbor regions
2. selecting the most similar pair of regions and merging them into a new region
3. updating the neighborhood and the similarity measures

The algorithm iterates steps (2) and (3) until all regions are merged into a single region.

The BPT stores the whole merging sequence from an initial partition to the one-single region representation. The leaves in the tree are the regions in the initial partition. A merging is represented by creating a parent node (the new region resulting from the merging) and linking it to its two children nodes (the pair of regions that are merged). An example of BPT is shown in Figure 4.1. As with the other tree representations, the nodes of the tree represent regions and the links the inclusion relationship.

The BPT represents a set of regions at different scales of resolution and its nodes provide good estimates of the objects in the scene. As previously said, classical object detection

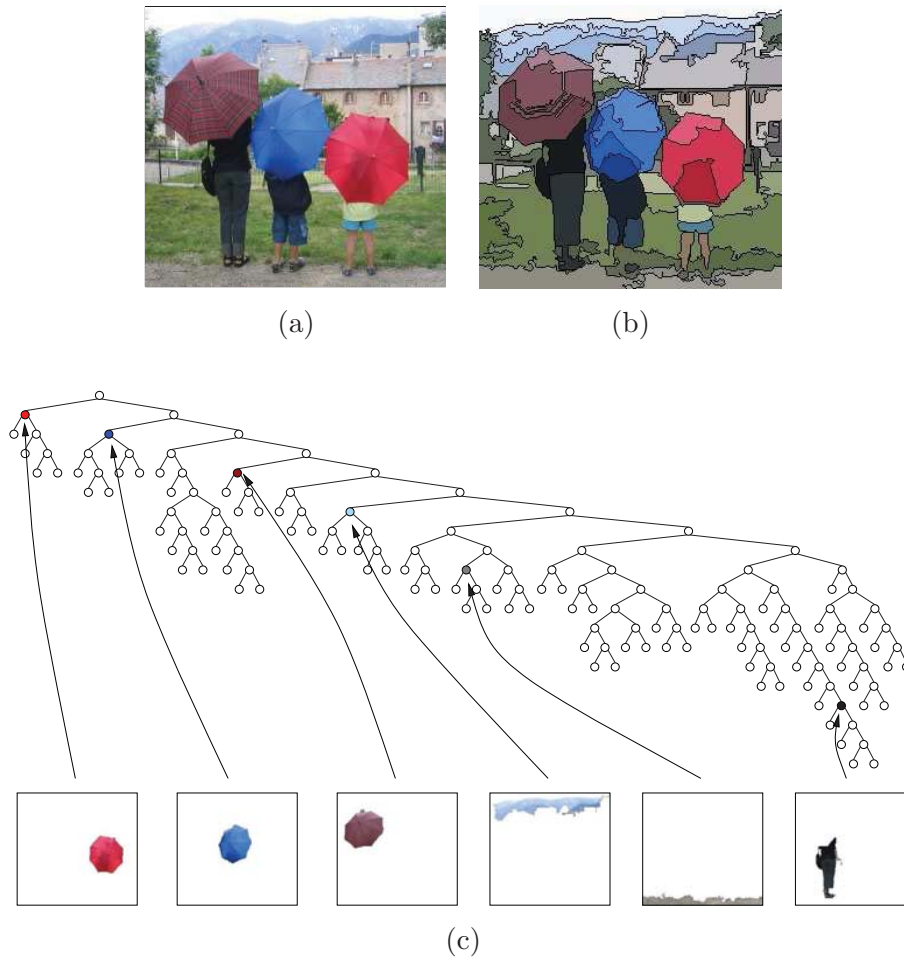


Figure 4.1: Example of BPT and illustration of its ability to represent objects in the scene: (a) Original image, (b) initial partition with 80 regions, where each region has been filled with its mean color and (c) Binary Partition Tree and examples of nodes representing objects in the scene.

algorithms scan the image at numerous positions and scales looking for the possible representations of the object in the scene. Using the BPT representation, the image has to be analyzed only at the positions and scales that are proposed by the BPT nodes. Therefore, the BPT can be considered as a means of reducing the search space in object detection tasks.

In the following sections we study the use of the BPT for object detection. Although the ultimate goal of this thesis is face detection and segmentation, we will show in Chapter 8 that the proposed representation is generic and can be reused to detect different kinds of objects. We propose strategies to provide accuracy (definition of the initial partition) and efficiency (selection of the nodes to be analyzed) to the representation. To achieve this goal, we highlight two set of nodes in the BPT, a set providing accuracy and another set defining

the search space. We also analyze various similarity criteria that can be used for the BPT construction and objectively assess their performance working with two different databases: MPEG-7 [113], for the face detection application, and ©Corel, for generic object detection.

4.2 Image representation based on Binary Partition Tree

The image representation and its main features in the context of face detection are illustrated in Figure 4.2. The BPT is constructed from its leaves by successive merging steps. The leaves of the tree form a partition of the space that is commonly named the initial partition. The initial partition can be made of individual pixels or of flat zones. This is, however, neither very useful nor practical in most object detection applications as the size of the resulting BPT is quite large. In most applications, the use of a very accurate partition with a fairly high number of regions is more appropriate. This initial partition can be created with any segmentation algorithm. For the experiments reported in this chapter, the initial partition is created with the same merging algorithm used to create the BPT (see Section 4.3). If this merging algorithm is used to define a partition, a *stopping criterion* is assessed at each merging step. Since in the context of object detection this partition is used to ensure an accurate representation of the objects in the scene, this partition is called in the sequel the *accuracy partition* (see Figure 4.2).

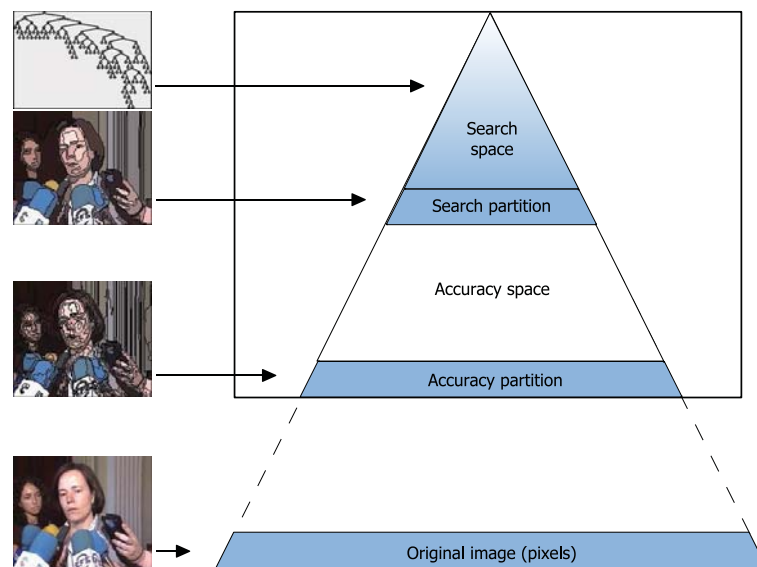


Figure 4.2: Image representation based on the BPT.

The BPT represents the sequence of merging of neighboring regions assuming that the most similar pair of regions is merged first. This image representation is a hierarchical region-based representation of the image. At lower scales, that is at scales close to the initial

partition, there is a very large number of small regions. This set of nodes contains information about the image details and provides accuracy to the image representation. As regions are merged, higher scales are created. They represent regions of the image that are progressively larger and possibly more meaningful. Note that, as the region size increases, it becomes possible to robustly measure and analyze some of the region features.

In the context of object detection, it is useless to analyze very small regions because they cannot represent meaningful objects and, measurements are not be reliable. Moreover, the number of very small regions may be fairly high. If the application involves severe restrictions on the computational complexity of the object detection scheme, the search space should be as small as possible. Taking into account the previous aspects, we distinguish two zones in the BPT: the *accuracy space* providing preciseness to the description (lower scales) and the *search space* for the object detection task (higher scales). These zones can be defined by selecting a point on the merging sequence. Regions that are created after this point belong to the search space. A specific point of the merging sequence is obtained by assessing a stopping criterion. The partition that is obtained at this point of the merging process is called the *search partition* (see Figure 4.2).

The following section discusses the creation of the image representation exemplified in Figure 4.2. It focuses on how to define the accuracy and the search partitions and on the possible similarity and stopping criteria for the merging steps.

4.3 Creation of the BPT

4.3.1 Merging criteria and region model

Several approaches can be followed to create the BPT. In this work we use a general merging strategy proposed in [146, 57], which allows the implementation of segmentation algorithms and filters. It works on a region adjacency graph (RAG), a graph where each node represents a connected component of the image (regions or flat zones) and whose links connect two neighboring nodes. A RAG represents a partition of the image. A merging algorithm on this graph is a technique that removes some of the links and merges the regions associated with the corresponding nodes. The merging is done in an iterative way. In order to completely specify the merging algorithm three notions are defined:

1. The *merging criterion*, $O(R_1, R_2)$, which defines the similarity of neighboring regions and therefore the order in which regions are going to be merged. $O(R_1, R_2)$ is a real valued function of each pair of neighboring regions R_1 and R_2 .
2. The *region model*, M_R , that specifies how regions are represented and how to compute the model of the union of two regions.

3. The *stopping criterion*, which defines when to stop the merging process.

A simple example of the merging algorithm is shown in Figure 4.3. The algorithm first constructs the region adjacency graph (c) of the initial partition (b) which has 6 regions. For each pair of neighboring regions, a homogeneity measure based on color similarity is computed (eq. 4.5). The algorithm starts merging the pair of neighbors with lowest distance (the most similar regions in the sense of the homogeneity criterion). In the example, the pair of most similar regions, 1 (sky) and 2 (mountain), is merged to create region 7. Then, distances between nodes are recomputed. Next, region 4 (glacier) is merged with region 5 (lake) to create region 8 and distances are recomputed. This process is iterated until only one region is obtained, 11 in the example, which is the whole image. In this example, the merging sequence is $(1, 2)|(4, 5)|(7, 8)|(3, 9)|(6, 10)$. This merging sequence progressively defines the tree shown in (e).

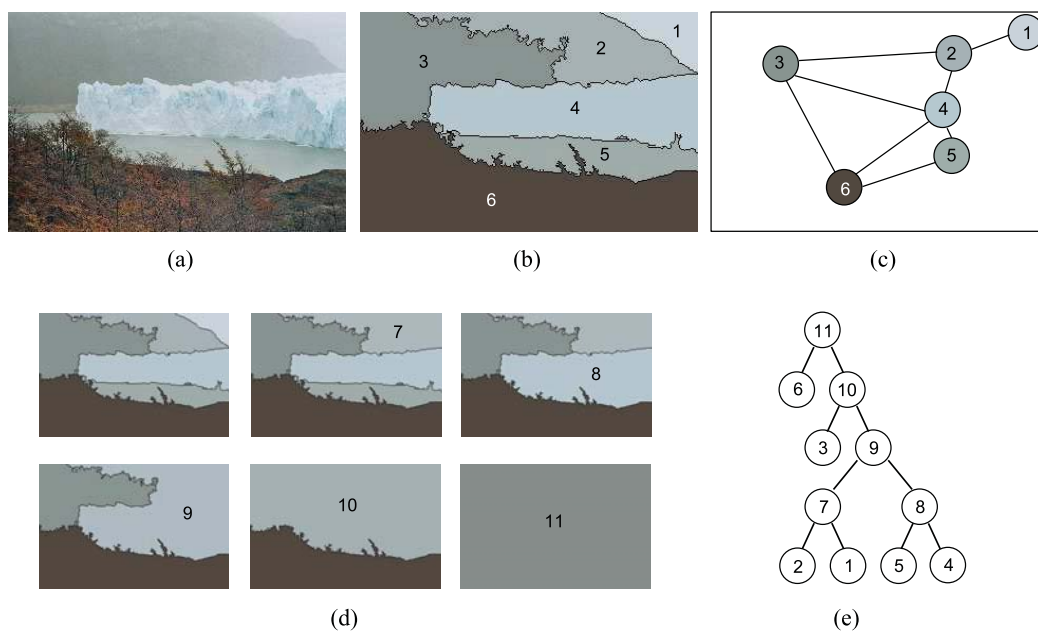


Figure 4.3: Example of merging algorithm. Original image (a), initial partition (b), RAG (c), merging steps (d) and Binary Partition Tree that represents the sequence of mergings (e).

In this work, the region model M_R is assumed to be constant within the region R , and is the vector formed by the average values of all pixels $p \in R$, in the YCbCr color space²:

$$M_R = \frac{1}{N} \sum_{p \in R} I(p) \quad (4.1)$$

²The motivation for using this color space is explained in Chapter 5

where N is the number of pixels of region R .

The similarity measure or merging criterion is computed for each pair of neighboring regions, R_1 and R_2 , according to a selected homogeneity criterion. The basic criterion used in most still image segmentation approaches is color homogeneity. Some of the measures are size independent, like the mean squared error (MSE) between the merged region and its model [57]

$$O_{MSE}(R_1, R_2) = \sum_{p \in R_1 \cup R_2} (I(p) - M_{R_1 \cup R_2})^2 / (N_1 + N_2) \quad (4.2)$$

or the Euclidean distance (ED) between the region models [186].

In general, size independent color-based measures tend to produce partitions with few large regions and a large number of extremely small regions. Trying to avoid this problem, other measures have been proposed taking into account the region sizes, as the squared error (SE) [57]

$$O_{SE}(R_1, R_2) = \sum_{p \in R_1 \cup R_2} (I(p) - M_{R_1 \cup R_2})^2 \quad (4.3)$$

or the weighted Euclidean distance (WED) [1]. When using these size dependent criteria, the cost of merging for small regions decreases, forcing small regions to merge together first and encouraging the creation of large regions.

As a compromise between the two types of measures, two other measures are analyzed here. They compare the models of the original regions with the model of the region obtained after the merging [57]. The weighted squared distance between region models (WSDM) is defined as:

$$O_{WSDM}(R_1, R_2) = N_1 \|M_{R_1} - M_{R_1 \cup R_2}\|_2^2 + N_2 \|M_{R_2} - M_{R_1 \cup R_2}\|_2^2 \quad (4.4)$$

and the weighed Euclidean distance between region models (WEDM) as:

$$O_{WEDM}(R_1, R_2) = N_1 \|M_{R_1} - M_{R_1 \cup R_2}\|_2 + N_2 \|M_{R_2} - M_{R_1 \cup R_2}\|_2. \quad (4.5)$$

Finally, and based on experimental evidences, we observed that while luminance information is necessary to define visually relevant contours, chrominance information is extremely important in the definition of objects. Since our final goal is to have regions in the search space that are as close as possible to the objects in the scene, we introduce another distance which modifies the WEDM by increasing the relevance of chrominance information. We also introduce a second change based on the use of contour information. Since most 'real' objects are regular and compact (that is, tend to have simple contours), the analysis of shape complexity can provide additional information for the merging. Therefore, we include in the similarity measure a term related to the contour complexity of neighboring regions.

As a consequence, the proposed merging criterion (which implies a normalized weighted Euclidean distance between models with contour complexity and, for simplicity, is referred to as NWMC in the sequel) has two terms. One term is based on color similarity, where the color difference in each component is normalized by the dynamic range of the component in the image. This way, the measure adapts to the chrominance variability of the image. For each image component, we compute the weighted Euclidean distance between models normalized by the dynamic range in each component:

$$O_{color}(R_1, R_2) = N_1 \|w(M_{R_1} - M_{R_1 \cup R_2})\|_2 + N_2 \|w(M_{R_2} - M_{R_1 \cup R_2})\|_2 \quad (4.6)$$

where w is a vector, and w_i is the inverse of the dynamic range of the image component $i \in \{Y, Cb, Cr\}$ (that is, the difference between the Max value and the Min value of the image component).

The second term is related to the contour complexity of the merged regions. After analyzing several approaches, the adopted measure computes the increase in perimeter of the new region with respect to the largest of the two merged regions: $\Delta P(R_1, R_2) = \min(P_1, P_2) - 2P_{12}$, where P_1 and P_2 are the R_1 and R_2 perimeters, respectively, and P_{12} is the common perimeter between the regions. The term that measures contour complexity is

$$O_{cont}(R_1, R_2) = \max(0, \Delta P(R_1, R_2)) \quad (4.7)$$

which sets to 0 negative increments that occur when a region is partially or totally included in the other. Color and contour similarity measures are linearly combined to form the NWMC:

$$O_{NWMC}(R_1, R_2) = \alpha O_{color}(R_1, R_2) + (1 - \alpha) O_{cont}(R_1, R_2) \quad (4.8)$$

The use of the α parameter in the NWMC criteria is analyzed in appendix B.3, where it is shown (see Figure B.6) that the sensitivity of the segmentation process is very low with respect to variations of the alpha parameter around $\alpha = 0.5$ and, therefore, this value is used throughout.

In the sequel the proposed distances are investigated in the context of BPT construction for generic object detection. We will discuss the creation of the image representation presented in Section 4.2 focusing on how to define the accuracy and the search partitions and on the similarity and stopping criteria for the merging steps. The quality of the representation is analyzed in terms of partition-based metrics, working with different sets of images. First, we detail the methodology used for the experiments.

4.3.2 Methodology for the experiments

To assess the quality of the representation we analyze the proposed criteria in terms of partition-based metrics. We use two different databases for assessment purposes. The objects

in these databases have been manually segmented and the resulting object partitions are used as ground truth in the experiments:

- To analyze generic features, a set of 100 images from the ©Corel image database is used. The set contains 10 images of 10 different complexity classes which are grouped in the following way: *tigers, horses, eagles, mountains, fields, cars, jets, beaches, butterflies* and *roses*. The objects in this Corel subset (160 in total) have been manually segmented in the context of the SCHEMA project (<http://www.iti.gr/SCHEMA/>).
- For the face detection application, a set of 100 images from the MPEG-7 database [113] has been selected. These images contain human faces in scenarios of different complexity that have been manually segmented leading to a total of 116 objects.

There exist several proposals for comparing image partitions [33] [184]. Most of these techniques compare two simple partitions (that is, partitions with only a few regions) where each region represents an object in the scene. This type of partition is commonly named *object partition*. In our case, in addition to this type of partition comparison, we want to assess how well the regions in a dense partition (that is, a partition with a large number of regions) conform to the shape of the objects represented in a ground truth object partition.

We have selected the approach proposed in [23] since it provides a single framework for both cases. Initially, a symmetric distance is proposed in [23], $d_{sym}(P, Q)$, which is defined in terms of the number of pixels whose labels should be changed between regions in P to achieve a perfect matching with Q (P and Q become identical).

The definition is extended in [23] to an asymmetric distance, $d_{asy}(P, Q)$, so that the distance between a partition P and any partition Q , finer than P , is zero. A partition Q is said to be finer than a partition P if the intersection of P and Q is equal to Q . Therefore, the asymmetric partition distance is the minimum number of pixels whose labels should be changed so that partition Q becomes finer than partition P . In our work, the asymmetric distance will be used for assessing how well an object partition (P) can be matched by the result of merging regions from a denser partition (Q).

In [23], the number of label changes for both distances is normalized by the image size to obtain the final distance values. In our work, given that we are always using a ground truth object partition, we can normalize the distance values using the object size. An example of how the asymmetric distance is computed is presented in Figure 4.4. Partitions shown in Figures 4.4.(b) and 4.4.(c) are compared in terms of the number of pixels whose labels should be changed so that partition (b) is finer than (c). Figure 4.4(e) shows the pixels that require a label change.

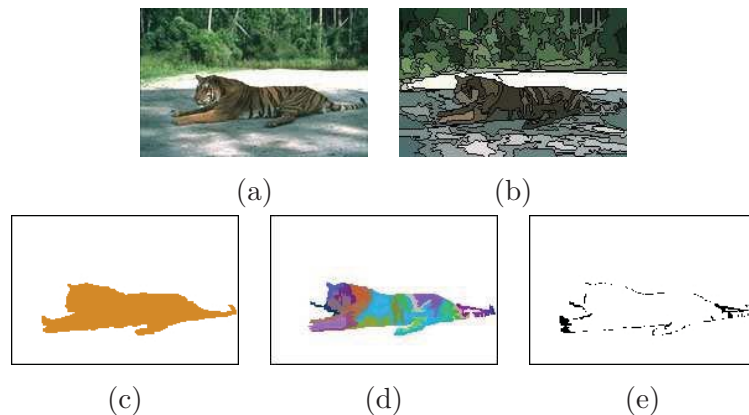


Figure 4.4: Illustration of asymmetric distance: Original image (a), example of search partition (regions filled with their mean value) (b), object partition (c), regions from the search partition partially contained in the object region (d), pixels requiring a label change (e).

4.3.3 Definition of the merging criterion for the accuracy space

In this subsection we analyze the merging criteria used to create the accuracy space. In this analysis, the starting point for the merging algorithm is the pixel level instead of the accuracy partition. Note, however, that almost any of the proposed merging criteria could be applied to define a useful accuracy partition using a simple stopping criterion, such as reaching a fair number of regions (e.g. 500 regions for face detection).

In the first experiment (see Table 4.1), the criteria are compared in terms of mean values of the final PSNR between the image created by filling each region with its model and the original image, and of the variance of the region sizes. The values are obtained for a set of 100 images from the Corel database. To decouple the effects of the merging and stopping criteria, that is, of the creation of the accuracy space and the search partition, a simple stopping criterion is used: merge up to 50 regions.

	MSE	SE	WSDM	WEDM	NWMC
PSNR	22,99	24,72	25,33	24,87	20,16
σ_{reg}^2 on region size	16,90	2,86	4,74	1,51	2,18

Table 4.1: Merging criteria comparison on the Corel subset. PSNR values are given in dBs. σ_{reg}^2 values are divided by 10^5 .

In terms of PSNR, the WEDM criterion improves all criteria except the WSDM which is only slightly better. However, these results for the WEDM are obtained while largely outper-

forming the other criteria in terms of variance of the region sizes. This is a relevant feature since it ensures that regions obtained with the WEDM will be adequate for a subsequent robust feature estimation: large enough and homogenous in size while presenting similar PSNR values than previous measures and leading to visually good representation. This behavior is illustrated in the second row of Figure 4.6 with images of different complexities.

Figure 4.5 illustrates the behavior of the WEDM as well as the importance of each image component in equation 4.5. In this example, the plots show the evolution of the similarity values for the merging process starting from an accuracy partition containing 500 regions (b) up to achieving 100 regions (c). The plot on the top shows the value of the global merging order; that is, combining the three image components as described by equation 4.5. The next three plots show the percentage of each component in the global measure given by equation 4.5. Note that most of the mergings are performed based on luminance similarity (the luminance percentage is high) and the contribution of the chrominance components remains quite stable through the merging sequence.

In a second experiment, the quality of the obtained partitions is assessed in terms of their accuracy in representing objects. The asymmetric distances between the partitions obtained with each of the five merging criteria (using as stopping criterion $N_{reg} = 50$ regions) and the object partition (ground truth) are computed for each object. In this case, the global behaviors of the SE and WEDM criteria are very similar, outperforming those of MSE, WSDM and NWMC criteria. The complete description of the experiment and its results are presented in Appendix B.1.

Summarizing the results of the experiments, it can be observed that the WEDM merging criterion largely outperforms the MSE, WSDM and NWMC criteria, while improving the SE one. Therefore, we propose the WEDM as merging criterion for the definition of the accuracy area of the BPT.

4.3.4 Definition of the search partition: Stopping criterion

We distinguish two zones in the BPT: the accuracy space, which provides to the description and the search space for the object detection task. These zones are defined by a stopping criterion that selects a point in the merging sequence or, equivalently, a partition (the search partition) that is obtained at this point of the merging process.

Typical stopping criteria deal with reaching an a priori value of a parameter such as the final number of regions in the partition or the PSNR between the image created by filling each region with its model and the original image. However, as we are defining the search space above the search partition, the objective is to create regions corresponding to parts of the objects in the scene whilst avoiding the creation of regions spanning more than one object. Thus, the stopping criterion has to take into account the complexity of the scene.

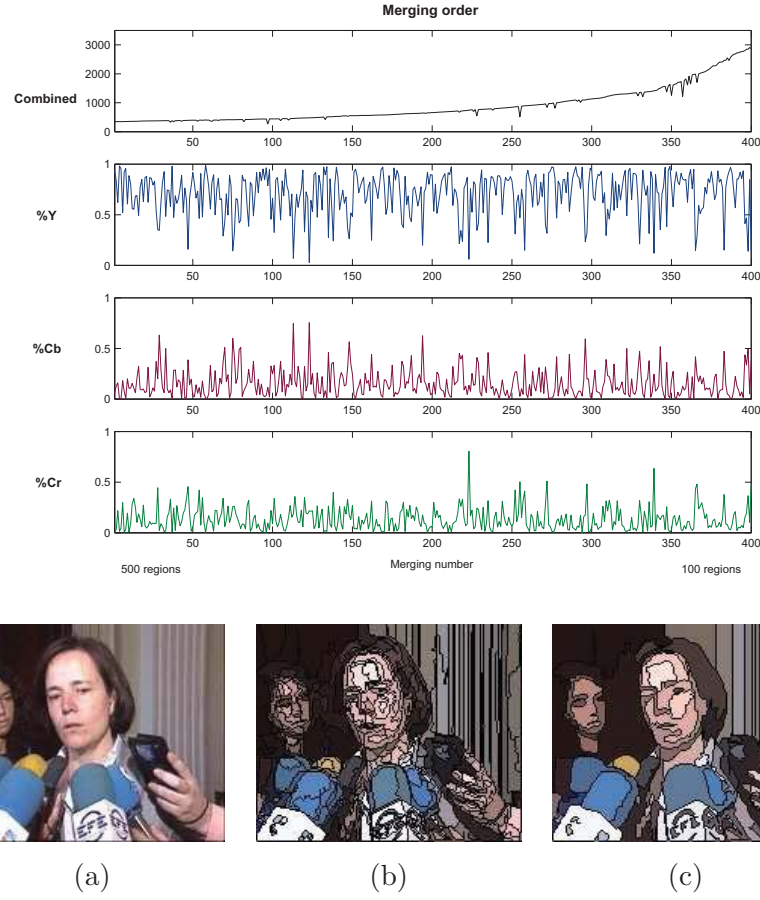


Figure 4.5: Relevance of the image components and evolution of region sizes during the merging process using the WEDM. Original image (a) and partitions with (b) 500 and (c) 100 regions.

We propose a procedure to estimate this complexity based on the sum of the similarity measures used to merge regions, the *accumulated merging cost*. Let $O(k)$ be the cost of the merging at iteration k , that is the similarity measure between the regions merged at iteration k . The accumulated merging cost (AMC) is defined as

$$AMC(m) = \sum_{k=1}^m O(k). \quad (4.9)$$

The stopping criterion is defined as a fraction $T_{AMC} \in [0, 1]$ of the total AMC (which is $AMC(N - 1)$, where N is the number of regions in the initial partition). This criterion stops the merging process at iteration \bar{m} , where

$$\bar{m} = \min\{m/AMC(m) > AMC(N - 1)T_{AMC}\}. \quad (4.10)$$

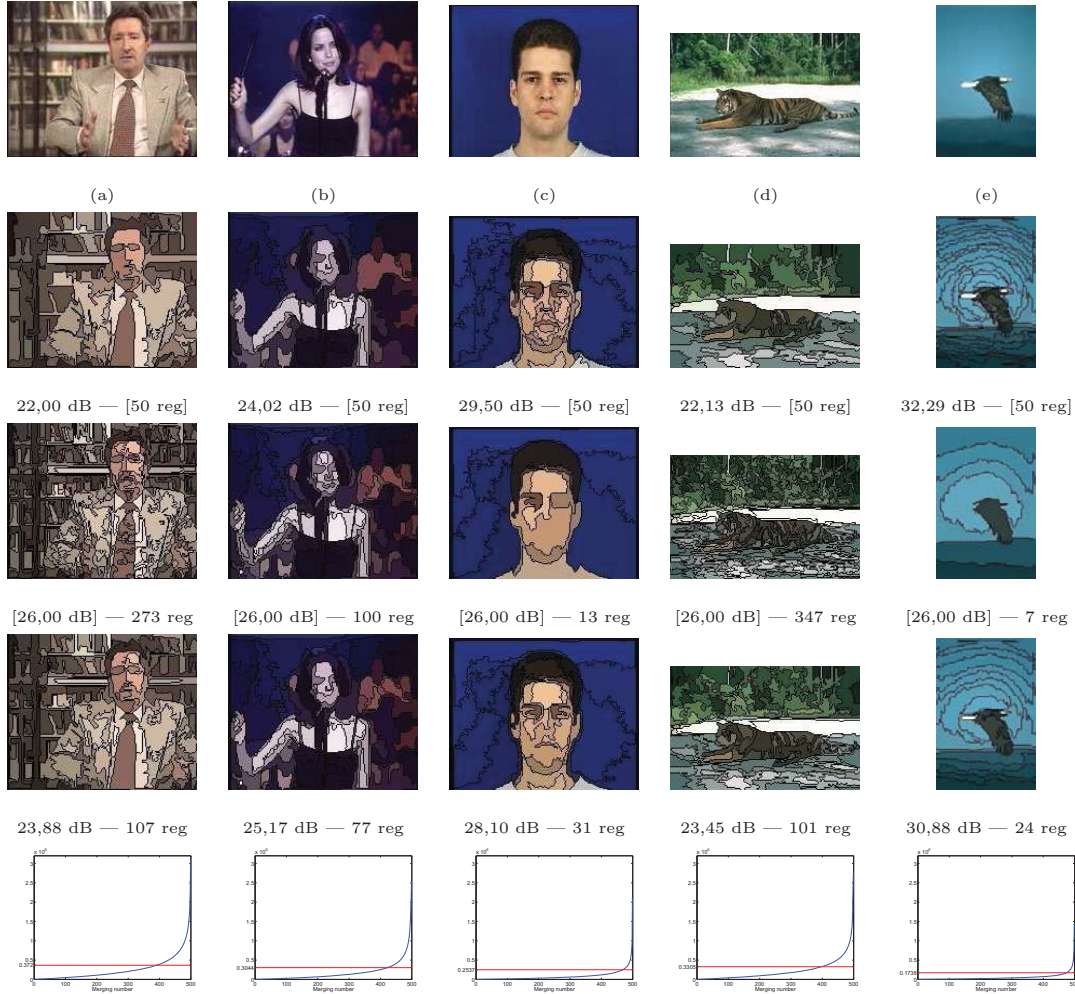


Figure 4.6: Comparison of the different stopping criteria. First row: original images. Second row: SC1: $N_{reg} = 50$. Third row: SC2: PSNR = 26 dB. Fourth row: SC3: $T_{AMC} = 0.12$. Fifth row: AMC(m).

Note that if the similarity measure used in the merging process is the weighted square difference of region models (WSDM) [57] and the initial partition defines each pixel as a different region, then the accumulated cost equals the squared error and the stopping criteria becomes a threshold relative to the maximum PSNR.

A stopping criterion based on the analysis of the accumulated cost was also proposed in [1]. However, their approach is not useful in our case since the method in [1] aims at achieving final partitions with a very reduced number of regions.

Figure 4.6 compares, for a set of images with different complexity, the results obtained by the most common stopping criteria: final number of regions N_{reg} , final PSNR and the proposed AMC criterion. In all cases, the merging criterion is the WEDM measure. The

second row presents the partitions obtained using as stopping criterion a fixed number of regions, $N_{reg} = 50$. In the third row, the final PSNR is fixed to 26dB whereas the fourth row shows the results for the new criterion, with $T_{AMC} = 0.12$. As it can be seen, the proposed criterion adapts to the image complexity; that is, it avoids oversegmentation as well as undersegmentation effects, obtaining partitions in which the main objects in the scene are correctly represented.

The last row of Figure 4.6 shows the accumulated costs plotted for each iteration of the merging process, starting with an accuracy partition composed of 500 regions. Note that images with different complexity lead to AMC(m) plots of different behavior. Plots also show the thresholds obtained for $T_{AMC} = 0.12$.

This value of T_{AMC} has been selected after analyzing the robustness of the system using the Corel database subset, in terms of final PSNR, average number of regions and average distance to object partitions, with respect to variations of T_{AMC} . It is used in all the experiments performed in the next chapters. Also the stopping criterion is quantitatively assessed using the Corel subset and the asymmetric distance (see Appendix B.2, Experiment 3).

The merging costs are not accumulated from the very first merging (in the original image) but starting from the accuracy partition. The accuracy partition is defined by a fixed (and rather large) number of regions (e.g. between 500 and 1000 for face detection). The AMC criterion is computed starting from this accuracy partition. The accuracy partition represents the highest resolution in the hierarchy and, therefore, should ensure a sufficient definition in the scene representation. The evolution of PSNR and asymmetric distances for a varying number of regions in the accuracy partition is analyzed in Appendix B.2, Experiment 4.

4.3.5 Definition of the search space of the BPT

Here, we analyze the construction of the hierarchical structure from the regions defined by the search partition. The discussion is centered on the similarity measure. Ideally, nodes in the search space should be objects or parts of objects with a semantic meaning. Therefore, the similarity measure should be related to a notion of object. Several approaches to segmentation try to create ‘meaningful’ partitions incorporating geometric features into the segmentation process, like measures of proximity, compactness, inclusion or symmetry (see [1] and [47]). However, the integration of this information is difficult to analyze and evaluate, since there is a strong overlap between various geometrical features (e.g. adjacency, contour complexity and quasi-inclusion). Moreover, in some cases the homogeneity assumptions implied by these features are too strong to lead to generic segmentations; that is, the resulting partitions are already biased to a specific type of object. This analysis led us to the proposal of the NWMC criterion defined in Section 4.3.1.

Figure 4.7 shows the priority values for the sequence of mergins performed to create the upper part of BPT for the image and the search partition presented in Figures 4.5(a) and

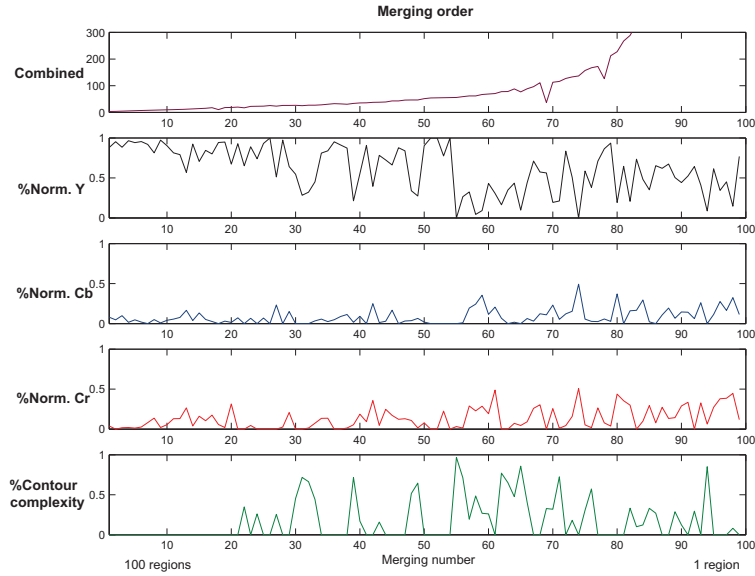


Figure 4.7: Example of priority values used at each step during the merging process to build a BPT from a search partition with 100 regions.

4.5(c), respectively, using the NWMC criterion. The plots represent the combined measure, the contribution (percentage) of the normalized color measures for each component and the contribution of the contour complexity term. The use of the term related to contour complexity favors the merging of regions with partial or total inclusions, unless they are very different in color. The first merging steps are performed mainly based on luminance differences. Note that near iteration number 30, 40 and 50, the merging steps are performed between regions that produce a large increase in perimeter but are very similar in color and that were not merged before due to the high value of the contour term.

Figures 4.8 and 4.9 present examples of BPTs and search partitions created with the NWMC and the WEDM criteria, respectively. These images exemplify the case of objects in the scene being correctly gathered in single nodes if the NWMC criterion is used. Using the WEDM measure, the object information is split among various nodes. To clearly illustrate this point, nodes related to parts of the human faces and their associated subtrees are shown. Note that each face is correctly represented with a single node in the NWMC case (see Figure 4.8) whereas in the WEDM case, neither of the two faces is represented with a single node and face regions are split among various nodes (see Figure 4.9).

More examples of the improvement achieved with the NWMC criterion are presented in Figure 4.10. In this case, we do not present the complete BPT but only those nodes related to the faces. Row (a) in Figure 4.10 shows the original images. Row (b) presents the nodes that represent face regions for the original WEDM criterion (eq.4.5), row (c) shows the nodes obtained by the criterion based on normalized color but without the contour complexity term

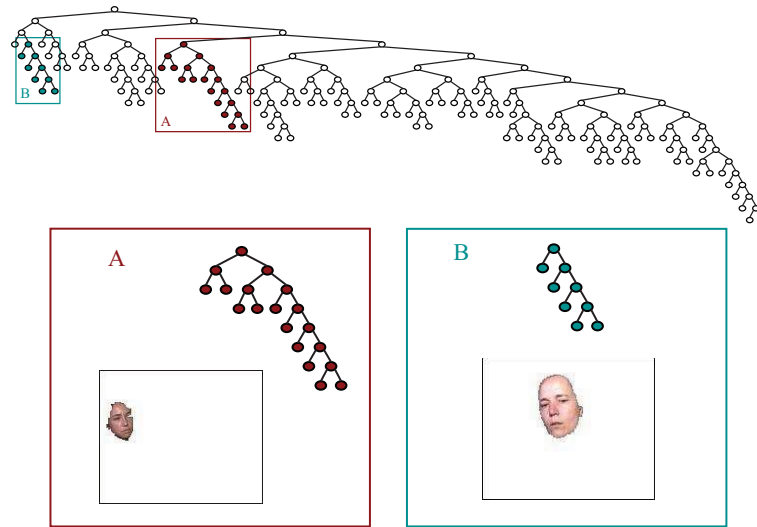


Figure 4.8: Search space of the BPT created with NWMC.

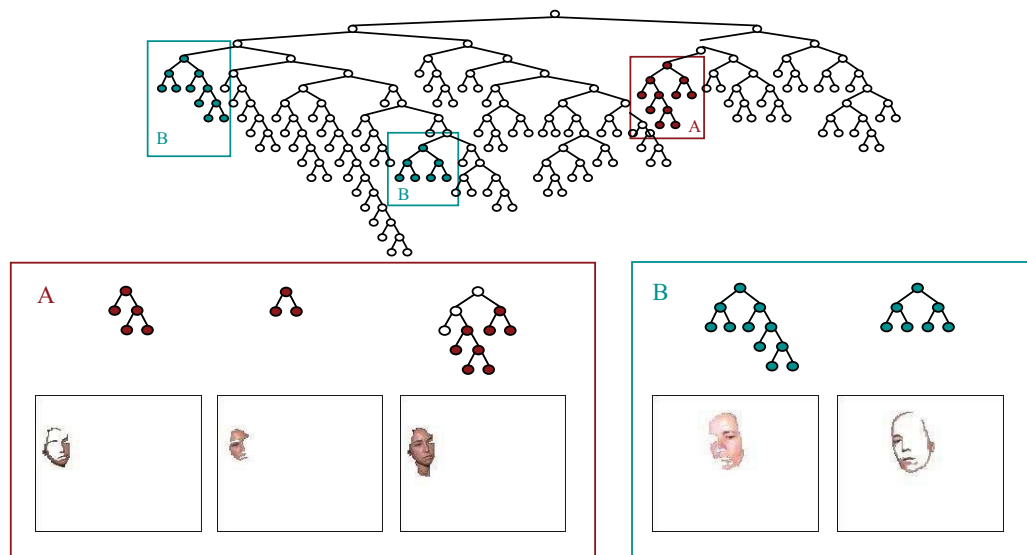


Figure 4.9: Search space of the BPT created with WEDM.

(eq.4.6), and row (d) the nodes obtained by the proposed NWMC criterion (eq.4.8). As it can be seen, the use of a normalized color term already improves the quality of the nodes (row (c)) which is further improved by the inclusion of the contour complexity term (row (d)).

The behavior illustrated in the previous figures is further analyzed in Experiments 5 and 6 (see Appendix B.3) using the Corel subset and the MPEG-7 face subset, respectively. For each image, the node in the BPT leading to the smallest symmetric distance with respect to each manually segmented object was selected. Statistics of the symmetric distances achieved with the nodes selected from BPTs created using the WSDM, WEDM and NWMC merging

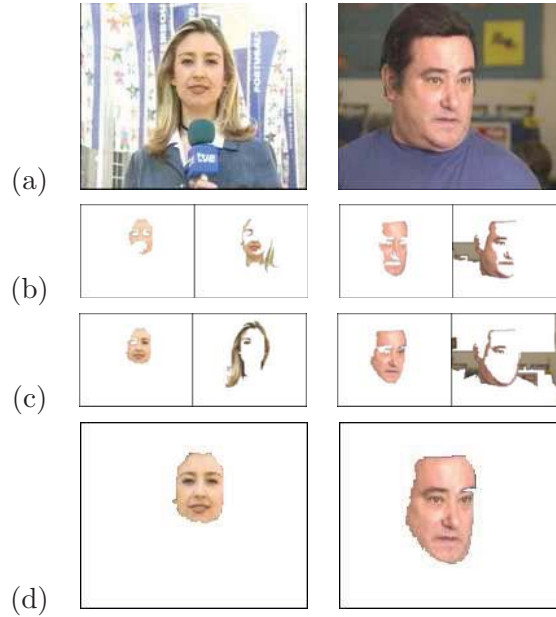


Figure 4.10: Face nodes: row (a) Original images, row (b) WEDM, row (c) a criterion based on normalized components and row (d) a combination of color and contour complexity: NWMC.

criteria confirm the improvements introduced by the NWMC criterion.

Finally, to complete the study of the image representation creation, the influence of the value of the parameter α in the performance of the NWMC merging criterion is analyzed in Appendix B.3, Experiment 7. The conclusion is that the sensitivity of the segmentation process is very low with respect to variations of the alpha parameter around $\alpha = 0.5$ and, therefore, this value is used in the experiments presented in the next chapters.

4.4 Summary

This chapter was devoted to the image model. We have discussed the use of the Binary Partition Trees for face and object detection. Concerning the tree construction, we have analyzed the compromise between computational complexity reduction and accuracy. This analysis led to the definition of two parts in the BPT: the accuracy space, which provides accuracy to the representation, and the search space, for the object detection task.

Next, we have objectively compared various similarity measures for the tree creation. We have concluded that different similarity criteria should be used for the two parts of the BPT. The Weighted Euclidean Distance between region Models (WEDM) may be used for the definition of the BPT accuracy space, and the Normalized Weighted Euclidean distance between Models with Contour complexity (NWMC) for the BPT search space. The transition

between the accuracy and the search spaces in the BPT is defined by the so-called Accumulative Merging Cost (AMC). The analysis of the various similarity measures and the influence of the different parameters is completed with the experiments reported in Appendix B.

The usefulness of this image model will be exhaustively studied in Chapter 6 for the task of face detection, and demonstrated for the detection of other semantic classes in Chapter 8.

The results presented in this chapter appear in the following papers:

- V. Vilaplana, F. Marqués, P. Salembier, *Binary Partition Trees for Object Detection*, IEEE Transactions on Image Processing, vol. 17, n.11, pp. 2201-2216, 2008. ISSN: 1057-7149.
- V. Vilaplana, F. Marqués, *On building a hierarchical region-based representation for generic image analysis*, Proceedings of ICIP 2007, IEEE International Conference on Image Processing, pp. IV.325-IV.328, San Antonio, USA, September 2007.
- V. Vilaplana, F. Marqués, *Region-based hierarchical representation for object detection*, Proceedings of CBMI 2007, 5th International Workshop on Content-Based Multimedia Indexing, pp. 157-164, Burdeos, France, June 2007.

Chapter 5

The face model

In this chapter we study how to characterize the face class, that is how to acquire a model suitable for face detection. Following a region-based approach, we define the *face model* in terms of a set of explicit attributes that are measured on image regions.

The detector is formed by two main classifiers which make use of the face model (see Figure 5.1, an extension of the block diagram presented in Figure 3.9). The first classifier is a cascade of very simple classifiers. Each classifier in the cascade has binary output (0 or 1), and it is based on one simple descriptor which describes a property related to a low-level visual attribute of the pattern. The goal of this classifier is to *simplify* the search space, that is, to rapidly discard most of the nodes which are related to outliers, while accepting all target objects. The remaining candidates are input to a second classifier, a combination of continuous-valued classifiers based on more complex region descriptors, which performs the final *classification* into face or non-face. Between both classifiers, there is an intermediate optional block that modifies the area of support of candidate regions to improve the object representation, a process that we call *node extension*.

This chapter is devoted to the analysis of face specific attributes and their related descriptors. We study color, texture and shape attributes of faces, propose a set of descriptors (*specific descriptors*) to measure them on the image regions and associate each descriptor with one classifier. The individual performance of these classifiers is analyzed. Finally, different normalization and fusion rules are studied in order to build an ensemble of classifiers that improves the accuracy of each single classifier.

Therefore, in this chapter we concentrate on specific face descriptors and on the second classifier, since this is the core part of the system. We postpone the details concerning the simplification (based on *generic descriptors* and a cascade of simple classifiers) and the node extension stages to Chapter 6.

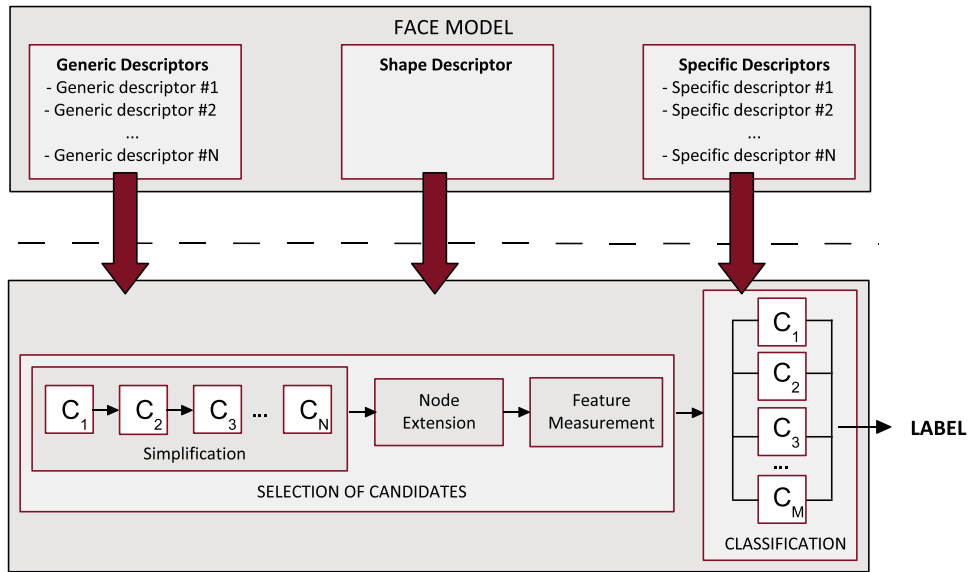


Figure 5.1: Face detection system.

5.1 Color

5.1.1 The skin color cue

Color is frequently used as a fundamental cue in face detection systems. Compared to face detection approaches that work with gray scale images, the use of color provides complementary information which is invariant to rotation, scaling and partial occlusions. Face detection systems that work with color images usually begin the search by detecting skin-colored areas or employ features based on skin color. They assume that faces, under ‘normal’ illumination conditions, have a characteristic ‘skin color’.

By skin color it is understood any perceived color that resembles or has the appearance of skin. However, this definition is vague in the sense that it depends on the observer. Humans see skin colors as quite stable and constant under a wide range of illumination conditions. The human visual system has the ability to automatically disregard the effects of widely varying illumination. This ability helps to keep stable the appearance of object’s colors. On the other hand, cameras do not have this mechanism. They cannot discriminate changes in the object reflectance from changes in illumination [107].

There are several challenges related to the use of color information in general object detection tasks. First, there may be problems with the robustness of the detection due to changes in illumination, since the same color may have different appearance under different illuminations. Second, even under the same illumination, color distribution differs from one camera to another depending on the camera sensor characteristics. In addition, there are

another two factors that affect face detection systems which are based on skin color detection. One is that skin color, even viewed under constant illumination, may vary between individuals. The other one is that there are many other objects that have the same color appearance as skin, since skin reflectance is not unique, particularly when it is viewed by ‘low resolution’ observers like humans or CCD cameras [164].

In the following we analyze the above mentioned problems to see how to approach them. For that, we show the appearance of human skin color under changing illumination and then we discuss the two primary steps for skin color representation: (i) the choice of a suitable color space and (ii) the modeling of skin color in the selected space.

Color acquisition in CCD cameras

A color signal is a light spectra produced by a light source or by the interaction between the illuminations’ spectra and the reflectance properties of materials. A CCD camera may be described as a filter that transforms continuous color signals to three descriptor values (e.g. ‘red’, ‘green’ and ‘blue’) of a limited range. Since the spectral data for a point is described with only three values, it is only an approximation of the true, incoming color signal spectra. Therefore, color samples with different spectra may become metameric, that is, they can appear as two different colors under a certain illumination but they can produce the same descriptor values when viewed under another illumination, being then indistinguishable [107].

Cameras are calibrated in order to disregard the effects of the illumination. The goal is that a ‘white’ object appears white in the image regardless of the current illumination conditions. This is called *white balance* or *white calibration*.

The following images show how much skin color appearance changes with illumination and camera calibration and what happens when cameras are not white balanced to the actual illumination. The examples belong to the Physics-based Face Database [105]. Figure 5.2 shows 16 images of the same face taken under four typical real world light sources and with different camera calibrations. Illuminants are daylight source 6500K (D65), incandescent A 2856K (A), horizon sunlight 2400K (H) and fluorescent (TL84). The light sources used in the white balancing are on the x-axis and the prevailing illuminants appear on the y-axis. Illuminants are ordered so that their color temperature increases from left to right and from top to bottom. The images taken under the calibration illumination (in canonical conditions) lie on the diagonal axis. Note that there are evident differences in color even between them.

We can see in the examples that when the camera is balanced to one illuminant, changes in the illumination greatly modify the color appearance. The same thing happens when the illumination is fixed and the reference illuminant varies. For high color temperatures skin has more bluish content, whereas for low color temperatures the skin tone shifts towards red.

The previous examples illustrate how image colors, in particular skin colors, depend on the characteristics of cameras and on the illumination. Despite these sources of variability, color

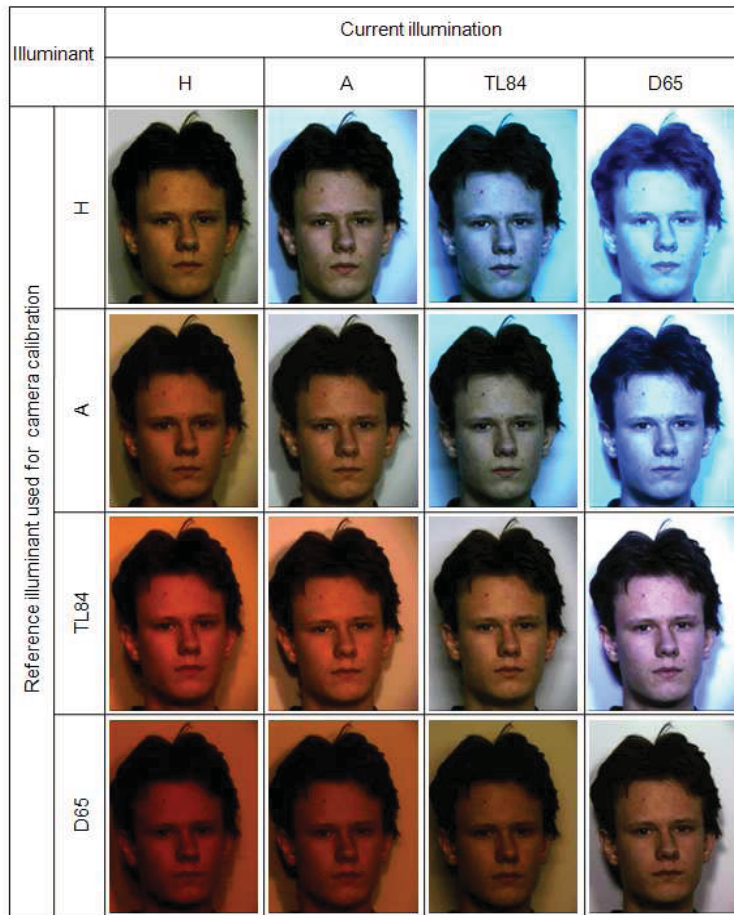


Figure 5.2: Images taken under several illuminations using different reference illuminants for camera calibration. Illuminant 'H' is horizon sunlight H for incandescent 2300K, 'A' is incandescent A for CIE A 2856K, 'TL84' is fluorescent TL84 corresponding to fluorescent F11 and 'D65' is a daylight source 6500K for modeling CIE D65.

is largely used in machine vision, and color models are learned and associated with specific objects or object features. In particular, it is often assumed that these variations are relative stable in a given application, and skin color models are learned and used successfully for skin detection [60]. Skin color information can be considered a very effective tool for identifying facial areas provided that the underlying skin color pixels can be represented, modeled and classified accurately [84].

Color spaces and skin color models

The problem of skin detection can be understood as a one class or as a two class classification problem. In any case, the primary steps for skin detection using color information are:

1. the choice of a suitable color space to represent image pixels, and
2. the modeling of skin color using an appropriate distribution in the selected color space

Color spaces: A color space is a specification of a coordinate system and a subspace within that system where each color is represented by a single point [61].

Usually the output of CCD cameras is expressed as digitalized values in the RGB color space. RGB is a *device-oriented* space, in the sense that it is associated with input, processing and output devices. Other device-oriented spaces are non-linear normalized RGB or NCCrgb, ratios of RGB components, etc.

RGB values are intensity dependent so any color distribution in this space varies with the scene brightness. One way to achieve a level of intensity invariance to scene brightness is to convert RGB signals into a color space which makes the intensity explicit. This is accomplished, for instance, with *user-oriented* spaces such as YIQ, HSI, HSV, HSB, HSL, etc. In these spaces colors are more ‘intuitively’ defined by their hue, saturation and intensity values. A 2D distribution in the chrominance components (hue and saturation) of these spaces provides a color model with a degree of invariance to scene brightness.

Other spaces that separate intensity from chrominance are *color-difference coding* spaces, which are used to reduce the amount of data to be transmitted for video and television. They were developed taking into account the human color perception: the human visual system has considerably less spatial acuity for color information than for brightness. Therefore, the image can be coded with a wide-band component for brightness and two narrow-band color components, where each color component has less resolution than brightness [129]. Examples of color difference spaces are YCbCr for digital video and YPbPr for analogue video.

Finally, another group of *device-independent* color spaces such as CIE Luv or CIE Lab may be used to represent color signals independently of a particular device and application. These spaces are perceptually uniform, in the sense that the Euclidean distance between two colors is similar to the perceptual distance between them.

Almost all these color spaces and variations obtained by linear or non-linear transformations of them have been proposed and used for skin detection (see [181, 84, 3]).

Skin color model: One of the easiest and often used methods to model skin color is to define skin color cluster decision boundaries for the different color components. Single or multiple ranges of threshold values for each color component are defined and color samples that fall within that range(s) are classified as skin colored. Another common approach is to use 2D or 3D color histograms, since histograms are stable representations unaffected by occlusion and changes in view. Samples for which the corresponding color likelihood is greater than a predefined threshold are classified as skin colored. Skin and non-skin histograms are

also frequently used in histogram-based Bayes classifiers. Another group of representative works use Gaussian distribution models. The advantage of these parametric models is that they can generalize well with less training data and also have very less storage requirements. It is assumed that under controlled illuminating conditions skin colors of different individuals cluster in a small region in the color space, and the distribution can be modeled by a multivariate normal distribution. Also Gaussian mixtures have been used to describe complex shaped distributions to account for varying illuminating conditions. Gaussian models are used in one or two class approaches. Finally, other classifiers such as multi layer perceptrons, self organizing maps, maximum entropy classifiers, or Bayesian networks are also used.

References for these models can be found in [164] and [107], while two surveys on pixel-based skin color detection techniques are [181] and [84].

Although there are works that compare the performances of some skin models on different color spaces, it is very difficult to make a fair evaluation and comparison of all the methods. First, because most of the techniques do not take into consideration the modeling of the illumination or camera properties. In most of the databases used for skin detection there is no information reported on illumination conditions or type of camera used and, usually, changes in illumination refer only to changes in illumination direction. Second, the data sets consist of images collected from different sources, and the concept of variability in skin color appearance is somehow vague. In most cases a particular type of classifier is selected, as well as training and test sets of skin and no skin samples. Then, the classifier is trained for each color space and the performances of the classifiers are compared. The conclusions, therefore, strongly depend on the type of classifier and also on the training set, since classifiers may be overtrained on the data.

Adaptive approaches to deal with changing lighting conditions, have been proposed in [131, 143, 160]. In turn, techniques that take into account the image formation process have been proposed in [164] and [107].

After reviewing the literature we may conclude that there is no color space or skin color model that clearly outperforms the others, but the choice depends on the application or scenario.

In this thesis we work with archive images or databases, assuming that all the available information is contained in the image data itself. Therefore, we will not use information on the illumination of the scene neither on the type of camera used. We will assume controlled environments, expecting that images are taken in more or less ‘normal’ conditions (where skin color in the images look like skin). However, a given skin model will be valid only for those conditions under which it was generated. In changing environments, the skin model should be adapted to cope with the variations in scene illumination.

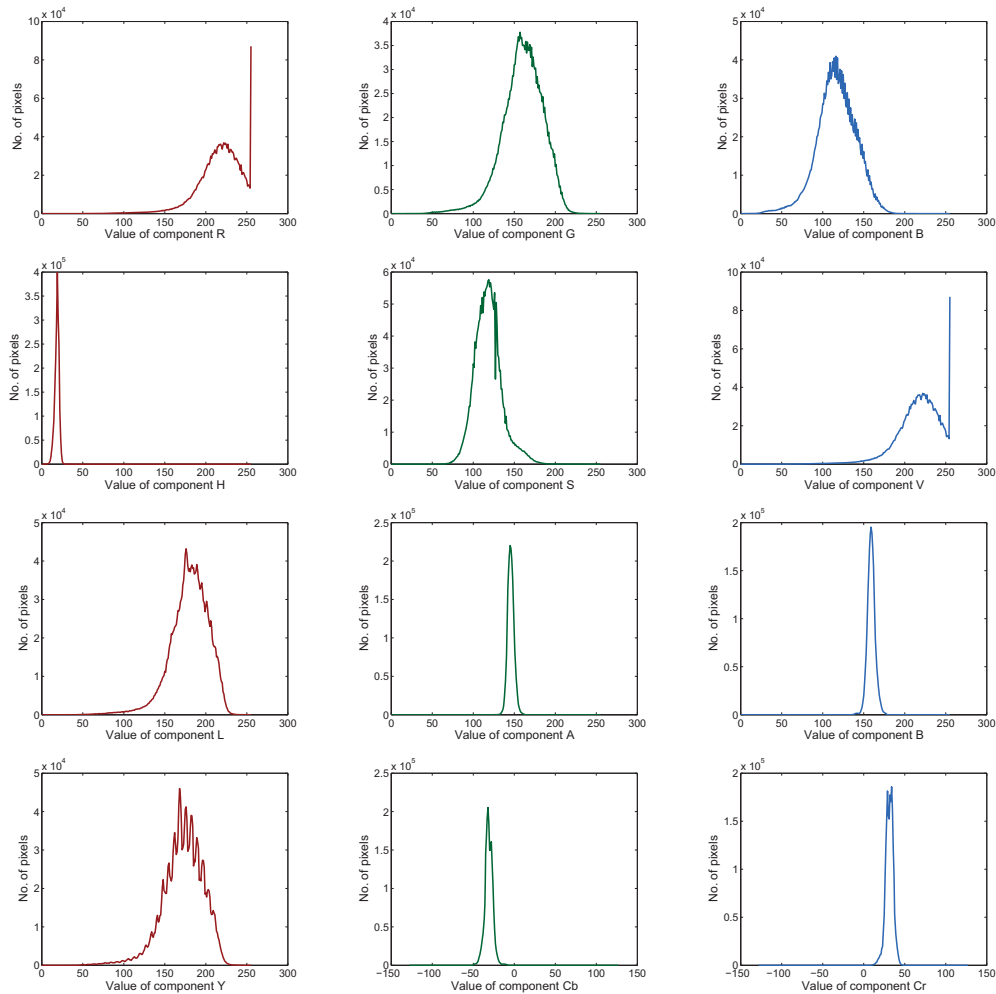


Figure 5.3: Histograms of skin color components in RGB, HSV, CieLab and YCbCr.

5.1.2 The skin color model

In order to select a color space and a suitable skin model, we take into account the following issues:

- Skin detection should be robust, to some extent, to changes in the scene brightness.
- The skin model should deal, as much as possible, with skin tone variability between people from different ethnic groups.
- The skin colors of different individuals should cluster in a small region in the color space provided that the images are taken under illumination controlled environments (a feature space where skin color pixels are grouped in a compact cluster).

- The cluster should be described with a simple (parametric) statistical model (few parameters, easy to train).
- The computational complexity in terms of color space transformation, training and use of the model should be low.
- The model should be easy to adapt to different illumination conditions.

The first requirement, robustness to changes in brightness, leads to the use of color spaces where chrominance is decorrelated from intensity, to discard the intensity component and to use the chrominance components to build a 2D skin color model. The same type of space satisfies the second requirement. Several works like [191, 64] have shown that nearly all the variation in skin color for different ethnic groups is in the intensity component, while chrominance values remain clustered.

We illustrate these properties with the help of the XM2VTS face database [109]. For the experiments, a set of more than 2 million sample points were collected from face images of 100 different people. The images were manually segmented and only skin pixels were extracted; that is, no eyebrows, eyes or mouth pixels were included in the set. The database images have been acquired under controlled illumination conditions. Figure 5.3 shows the histograms for each component in four different color spaces, one for each of the groups of spaces listed above: RGB, HSV, CIE Lab and YCbCr. In RGB, the variances are quite large in all the components whereas in the other three spaces, where there are separated components for luminance and chrominance, skin colors are grouped in narrow clusters. Among these three spaces we prefer YCbCr, since it is a color space widely used in image coding, particularly in MPEG digital video and therefore skin detection can be performed without any color transformation from one color space to another. We will then work with the two components, Cb and Cr.

The next example illustrates the observation that skin tone variability between ethnic groups lies in luminance rather than in chrominance values. Figure 5.4 shows four images of people with different skin tones (images from the XM2VTS data set). We observe that skin chromaticities overlap on the same region of the CbCr plane, whereas there is a considerable difference among luminance histograms.

Finally, the last example shows how strongly the color distribution depends on the illumination. The Y-histogram and chrominance distribution of skin pixels for images of the same face under four different illuminations are presented in Figure 5.5. The skin chromaticities for each image form a compact cluster, but as the illumination varies the clusters are localized on different parts of the CbCr plane. If we compare these clusters with those presented in Figure 5.4 we see that chrominance variability between the same face under different illumination is much higher than in the previous example, with faces of four different people under the same illumination.

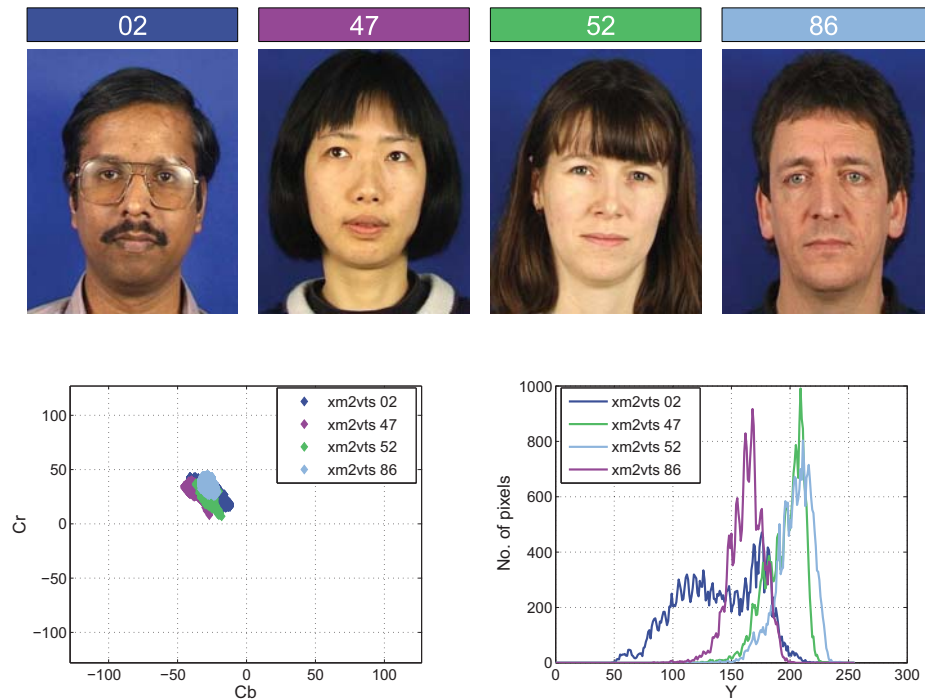


Figure 5.4: Chrominance and luminance for different skin tones.

We next turn to the problem of the choice of a classifier model in the CbCr subspace. Among the one-class classifier methods described in Chapter 2, density methods are the most appropriate due to the low dimensionality of the data and the large number of samples that are available. If the training data is a representative sample from the true skin color distribution, that is, if training and test sets are acquired under similar conditions, the complete density function can be accurately estimated.

Skin color distribution can be effectively modeled through a unimodal Gaussian function in the chrominance space:

$$p(\mathbf{c}|S) = \frac{\exp\left[-\frac{1}{2}(\mathbf{c} - \mu_S)^T \Sigma_S^{-1}(\mathbf{c} - \mu_S)\right]}{[2\pi|\Sigma_S|]^{\frac{1}{2}}} \quad (5.1)$$

where $p(\mathbf{c}|S)$ is the probability density function of a color vector $\mathbf{c} = (c_{cb}, c_{cr})$ given the skin class S . The mean μ_S and covariance matrix Σ_S are estimated with a training set of skin colored pixels.

Figure 5.6(a) shows the chromatic distribution of skin colored pixels (as a joint histogram of Cb and Cr components and its intensity map) for skin pixels collected from three data sets, XM2VTS, MPEG7 and Banca (controlled subset) images, using 108635, 126453 and 117625 pixels, respectively. For this set, the estimated mean and covariance values are given in table 5.1.

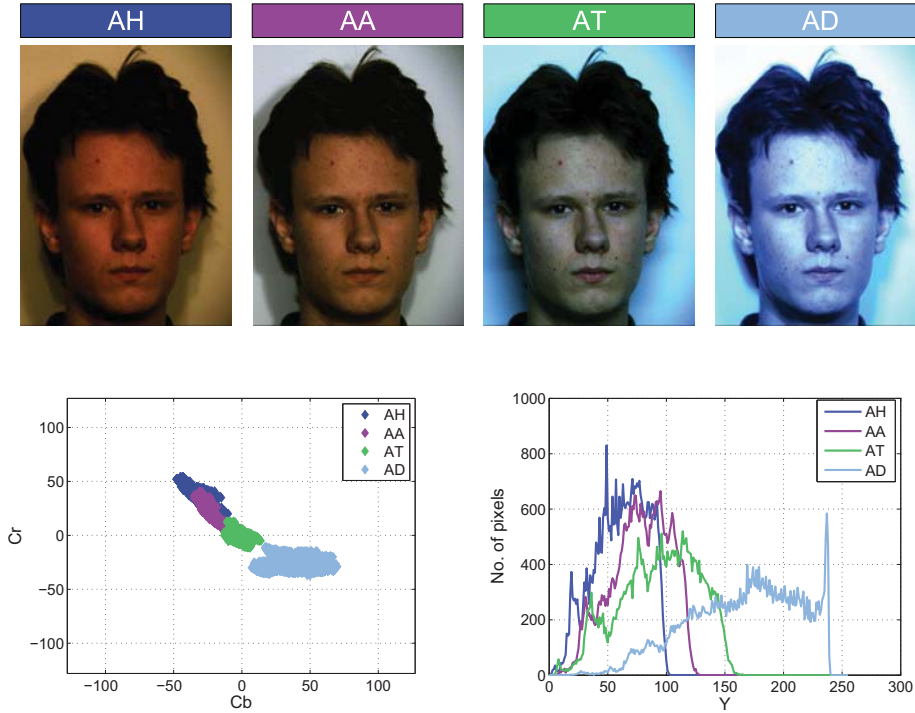


Figure 5.5: Chrominance and luminance for the same face under different illuminants.

	μ_{cb}	μ_{cr}	$C_{cb,cb}$	$C_{cb,cr}$	$C_{cr,cr}$
SKIN	-23.31	28.35	97.87	-65.25	80.46

Table 5.1: Mean and covariance values for skin pixels from XM2VTS, MPEG and Banca images.

5.1.3 Color descriptors in BPT nodes

Among the large number of color descriptors proposed in the literature, we choose the mean color and the set of dominant colors, due to their ability to characterize the perceptual color similarity and the low complexity of their associated extraction procedures. In the following we assume that we work in the YCbCr color space.

The *mean color descriptor* for a region R computes the average of the pixel values in each component:

$$MEANC(R) = \left(\frac{1}{N} \sum_{(x,y) \in R} Y(x,y), \frac{1}{N} \sum_{(x,y) \in R} Cb(x,y), \frac{1}{N} \sum_{(x,y) \in R} Cr(x,y) \right) \quad (5.2)$$

where N is the number of pixels in region R .

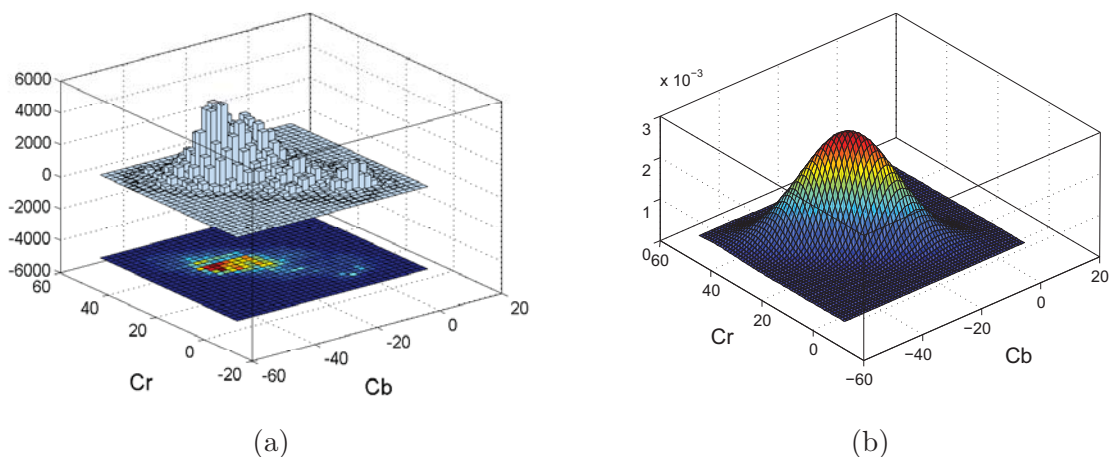


Figure 5.6: Histogram of skin chrominance (a) and estimated Gaussian distribution (b).

Clearly, this descriptor is not sufficient to represent the region texture or even its visual appearance accurately. However, it is very simple and fast to compute using the hierarchical tree structure, since the mean color of a region can be computed by averaging the color descriptors attached to its children nodes.

The *dominant color descriptor* [103] provides a compact description of the representative colors in an image or image region. Its main target applications include similarity retrieval in image databases and browsing of image databases based on single or several color values. Unlike histogram based descriptors, the representative colors are computed from each image or region instead of being fixed in the color space, thus allowing the color representation to be accurate and compact.

The descriptor for a region R is

$$DOMC(R) = \{ \{ (\mathbf{c}_i, p_i, v_i) \}_{i=1, \dots, N}, s \} \quad (5.3)$$

where N is the number of dominant colors. Each dominant color value \mathbf{c}_i is a vector of color component values. The percentage p_i is the fraction of pixels in the region corresponding to color \mathbf{c}_i , and $\sum_i p_i = 1$. The optional color variance v_i describes the variation of the color values of the pixels in a cluster around the corresponding representative color. The spatial coherence s is a single number that represents the overall spatial homogeneity of the dominant colors in the image. The number of dominant colors N can vary from image to image (region to region). A maximum of eight dominant colors was found to be sufficient to represent an image or an image region [103].

The extraction of the set of dominant color is performed in [36] through a splitting-clustering process in CIE Luv color space. The procedure is initialized with one cluster and

one representative color (the center of mass of the cluster). The algorithm follows a sequence of splitting clusters and updating centroids steps until a stopping criterion (maximum number of iterations or minimum distortion) is met. The fraction of pixels in the region that belong to each of the quantized colors is calculated. A measure of color coherence is computed with a connected component analysis that identifies groups of pixels of the same dominant color that are spatially connected.

Our extraction algorithm is different since it takes advantage of the accuracy partition of the image, and computes the dominant colors of the tree nodes using the color descriptors attached to the regions of the accuracy partition. For each region in this partition, its mean color is computed. Then, the dominant colors of a region R are obtained through a merging process performed on the YCbCr color space, using the mean colors of the fine regions (regions in the accuracy partition) that form the region R . Distances between all pairs of colors associated with the fine regions are computed. The two most similar colors are merged, and a new representation for the color is obtained as the weighted average of the merged colors, where the weights are the sizes of the regions whose colors are being merged. Next, distances between colors are recomputed.

We could draw an analogy between this color merging process and the region merging algorithm explained in Chapter 4. Note, however, that in this case we do not take into account spatial relationships since we deal with colors (points in a 3D color space), not regions. Distances between all pairs of colors are computed. The similarity measure between two colors \mathbf{c}_1 and \mathbf{c}_2 is

$$D(\mathbf{c}_1, \mathbf{c}_2) = \sum_j w_j (c_1^j - c_2^j) \quad (5.4)$$

where w_j is a weight associated to the color component j [103].

The colors are merged until a termination criterion is reached. We work with two criteria: one, that limits the number of dominant colors in a region (so mergings are performed until a fixed number of colors is obtained), and a second criterion that stops the fusions when the distance between the most similar pair is above a given threshold. This last criterion avoids the merging of very different colors.

Two examples of mean and dominant colors are presented in Figure 5.7. The 8 dominant colors are represented with a colored image where bar i is painted with color \mathbf{c}_i and has width proportional to p_i .

Experiments

To illustrate and compare the usefulness of both descriptors, we compute them for a set of manually annotated target and outlier regions, estimate the skin-likelihood associated with each region and compute the ROC curves. If we assume that the non-skin colors are uniformly

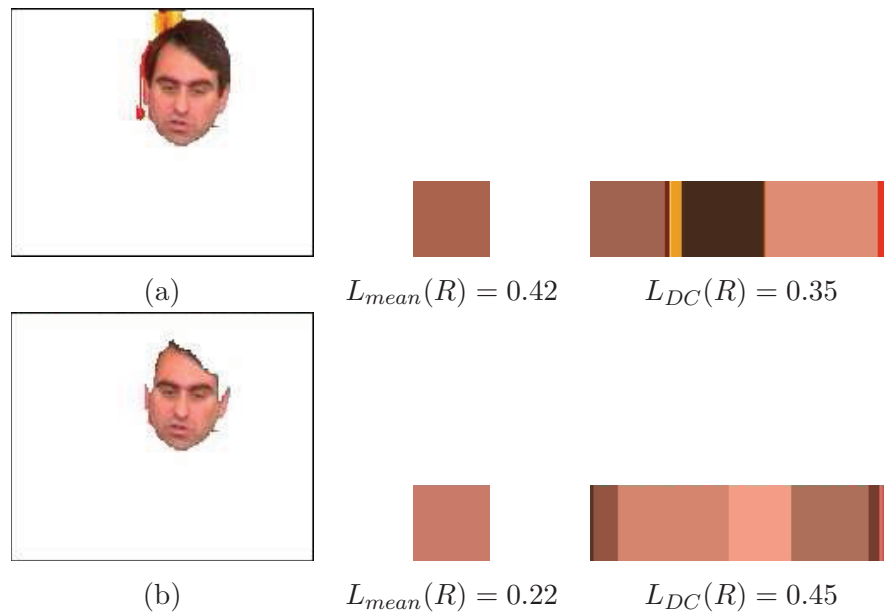


Figure 5.7: Mean color, dominant colors their skin likelihoods L_{mean} and L_{DC} , respectively, for regions (a) and (b).

distributed around the skin colors, we can use the likelihood $p(\mathbf{c}|S)$ instead of $p(S|\mathbf{c})$ to classify regions (see Section 2.2.3).

When a region R is represented by its mean color value $mean(R)$, its skin-color likelihood is computed using expression 5.1 and statistics shown in Table 5.1, where $\mathbf{c} = mean(R)$. For the dominant colors descriptor, the skin-color likelihood of region R is obtained by a weighted average of the likelihoods of the N dominant colors, $\sum_{i=1}^N p(\mathbf{c}_i|S)p_i$, where p_i is the fraction of pixels in R with dominant color \mathbf{c}_i . Both descriptors are calculated for the three color components but only Cb and Cr values are used when computing the likelihoods.

Figure 5.8 presents the ROC curves for a selection of images from three datasets of different complexity, XM2VTS, Banca (controlled sessions) and MPEG7, showing target and outlier acceptance rates for varying thresholds. We see that the dominant color descriptor slightly outperforms the mean color descriptor for XM2VTS and Banca; in both cases the performance is high. The XM2VTS set contains many outlier regions formed by face and hair, which explains the larger number of false positives. For the MPEG7 set the performance of both descriptors is the same, and significantly lower than for the other datasets. This is a reasonable result since MPEG7 images have clutter background with many no-face skin colored regions.

The mean color descriptor is very useful for regions with low color variability. For regions formed by areas with very different colors, the average color is meaningless. In those cases the dominant color descriptor provides a richer description, as illustrated by the examples in

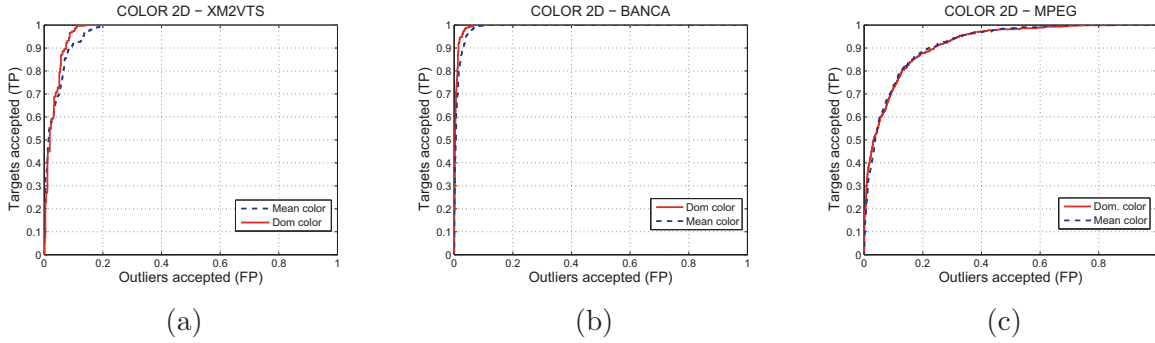


Figure 5.8: ROC curves for color classification using mean and dominant color descriptors, for XM2VTS (a), Banca (b) (515 target and 1424 outlier regions) and MPEG7 (c) (863 target and 3727 outlier regions) test sets.

Figure 5.7. The likelihoods for region (a), which includes face, hair and part of the background are 0.42 for the mean and 0.35 for the dominant colors, whereas for region (b), the correctly segmented face, likelihoods are 0.22 for the mean and 0.45 for dominant colors. The mean descriptor for the wrong region (a) averages colors and results in a higher likelihood than the correct face region (b).

Even though we are showing the ROC curves to assess the quality of color as a face attribute, the output of the color-based classifiers are continuous values in the range $[0, 1]$ that will be combined with the output of classifiers based on other features to produce the final face-likelihood associated with the region. The question of which of the two color classifiers will be used in the ensemble is studied in Section 5.4.

5.2 Texture

5.2.1 The texture cue

By texture we understand the spatial variation in pixel intensities and, therefore, it is an attribute measured on the luminance component of the image. Texture is an important cue for detecting objects that undergo shape deformation, pose changes or variations in illumination, and it is undoubtedly the attribute most commonly used for face detection, in particular by block-based techniques.

In the simplest approach, pixel intensities are directly input to the classifier. A feature vector is formed by a raster scan of pixel values within the subimage or region being analyzed, which is then classified using a neural network [137], an SVM [117] or other classifier [46, 96, 136].

A second group is formed by generative model techniques, which assume that face images

of the same size lie in a subspace of the overall image space. The subspace is represented using principal component analysis, linear discriminant analysis or factor analysis. A feature vector is extracted from the representation of the pattern in the subspace and used to classify the pattern. Among these techniques we find [111, 110, 166, 195].

A third group takes into account the statistical dependence between neighboring pixels. Patterns are represented in terms of a large set of local oriented intensity differences between adjacent areas in the image [120, 185] or by joint statistics of local appearance and position [157, 30]. Different classifiers can be used: SVM, naive Bayes, or cascades of classifiers.

In the following subsections we analyze the performance of different texture descriptors and classifiers. Four of them are one-class classifiers based on PCA features, and the last one is a two class classifier based on local features and trained by boosting. The objective of the study is to show their usefulness in discriminating faces from outliers when applied to the BPT nodes. Classifiers are analyzed individually, while their combination is studied in the last section of this chapter.

5.2.2 PCA-based classifiers

PCA based approaches have been traditionally applied to face detection and recognition problems. Principal Component Analysis is a classical technique for multivariate analysis, which provides a linear method to reduce the dimensionality of a data space.

Given a collection of n by m pixel training images represented as vectors of size $N = nm$ in an N -dimensional space, the PCA defines a transformation from R^N to a lower dimensional space R^M , where $M < N$, defined by $\mathbf{z} = W^t(\mathbf{x} - \bar{\mathbf{x}})$, where $\bar{\mathbf{x}}$ is the sample mean.

The column vectors of W , $\{w_i\}_{i=1\dots M}$, are the orthonormal axes that capture most of the variance present in the data. These column vectors are the M eigenvectors of the data covariance matrix with largest eigenvalues¹.

When the PCA is applied to face images, the eigenvectors are called *eigenfaces*. A face image \mathbf{x} can be approximately reconstructed using the coefficients of its projection onto the M -dimensional principal subspace $\hat{\mathbf{x}} = W\mathbf{z} + \bar{\mathbf{x}}$. The face image \mathbf{x} can thus be represented by the vector of coefficients \mathbf{z} (see Section 3.2.2).

Distance from the feature space

If all faces can be well approximated by their projection in a principal subspace of dimension M , it can be assumed that the face class lies in the subspace $F = \langle w_i \rangle_{i=1\dots M}$ spanned by the first M eigenvectors of a PCA computed on the training dataset.

¹In practice the data covariance matrix is often singular, in particular when the number of samples is lower than N . However, M ($< N$) principal eigenvectors can still be estimated using, for example, singular value decomposition.

If outlier objects do not satisfy the assumptions about the target distribution, they should be represented worse than true target objects. Then, their reconstruction error, that is the Euclidean distance between the candidate and its projection on the subspace F , should be high. The reconstruction error can therefore be used as a similarity measure to detect faces [176]. It is called DFFS or *distance from the feature space*:

$$DFFS(\mathbf{x}) = \epsilon^2 = \sum_{i=M+1}^N y_i^2 = \|\mathbf{x} - \bar{\mathbf{x}}\|^2 - \sum_{i=1}^M y_i^2 \quad (5.5)$$

where y_i is the projection of the mean normalized vector $\mathbf{x} - \bar{\mathbf{x}}$ on the i th-eigenvector.

Distance in the feature space

If the face class lies in the subspace F , a boundary method can be used to detect new faces. Assuming a Gaussian distribution of the data, a similarity measure between a candidate \mathbf{x} and the face class is the Mahalanobis distance in the subspace (the distance between \mathbf{x} and the sample mean $\bar{\mathbf{x}}$). It is called DIFS or *distance in the feature space*:

$$DIFS(\mathbf{x}) = \sum_{i=1}^M \frac{y_i^2}{\lambda_i} \quad (5.6)$$

where λ_i is the i th-eigenvalue.

DIFS defines, in the subspace F , concentric ellipses of points that are equidistant from the sample mean (in the sense of the Mahalanobis distance). Applying a threshold on this distance, an elliptical boundary of the face class is obtained. Therefore, DIFS can also be used as a similarity measure to detect faces.

While DIFS is a distance in F , DFFS is a distance in the orthogonal subspace F^\perp . DIFS and DFFS provide complementary information and may thus be combined.

High dimensional Mahalanobis distance

The two distances DIFS and DFFS are combined in the approach followed in [111], where the complete probability distribution of the face class is estimated using the eigenvector decomposition of the image space. The density is decomposed into two components, the density in the complete subspace F , and its orthogonal complement.

The class likelihood of a pattern \mathbf{x} is modeled as a unimodal Gaussian density function:

$$P(\mathbf{x}/\Omega) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \bar{\mathbf{x}})\right]}{(2\pi)^{N/2} |\boldsymbol{\Sigma}|^{1/2}} \quad (5.7)$$

where $\bar{\mathbf{x}}$ is the sample mean, and $\boldsymbol{\Sigma}$ is the data covariance matrix.

The *Mahalanobis distance* $d(\mathbf{x}) = (\mathbf{x} - \bar{\mathbf{x}})^T \Sigma^{-1} (\mathbf{x} - \bar{\mathbf{x}})$ is used to characterize the likelihood. A computationally tractable estimate of this distance based on the first M eigenvectors is (see Appendix C.1):

$$DMAH(\mathbf{x}) = \hat{d}(\mathbf{x}) = \sum_{i=1}^M \frac{y_i^2}{\lambda_i} + \frac{1}{\rho} \epsilon^2(\mathbf{x}) \quad (5.8)$$

The first term is the distance in the feature space DIFS (Eq. 5.6), and the second one is the reconstruction error DFES (Eq. 5.5), divided by ρ , which is the average of the $N - M$ remaining eigenvalues:

$$\rho = \frac{1}{N - M} \sum_{i=M+1}^N \lambda_i \quad (5.9)$$

In the following we test the three one-class classifiers, reconstruction with DFES, boundary with DIFS and density with DMAH, in the context of our approach to face detection. We also analyze the performance of a fourth classifier, the Support Vector Data Description (SVDD) based on PCA features, as a second boundary method alternative to the DIFS approach. The motivation is that the elliptical boundary description provided by DIFS is too simple to model the true class boundary. The SVDD is a classifier specifically developed by D. Tax [169] to solve one class classification problems. The technique, inspired in Support Vector Data learning theory, obtains a boundary around the target data set defined by some of the training points.

Support vector data description

In the basic formulation, given a set of training data $\{x_j\}, j = 1, \dots, N$, the SVDD defines an hypersphere around the data. The goal is to minimize the volume of the sphere while keeping the target objects inside its boundary. The boundary and center of the sphere are described in terms of a subset of the training samples, the *support vectors* x_i and a set of coefficients α_i . A new pattern \mathbf{x} is accepted as target object if it is inside the description:

$$\|\mathbf{x} - \mathbf{a}\|^2 \leq R^2 \quad (5.10)$$

where \mathbf{a} is the center and R the radius of the sphere.

The basic formulation is extended to obtain a more flexible description, by mapping the data non-linearly and implicitly, through a kernel function, into a high dimensional space. In our experiments we work with a Gaussian kernel

$$K(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{s^2}\right) \quad (5.11)$$

where s is a width parameter which controls how tight the description is around the data.

Using this kernel, a new object \mathbf{x} is accepted if

$$\sum_i \alpha_i \exp\left(\frac{-\|\mathbf{x} - x_i\|^2}{s^2}\right) \geq \frac{1}{2}(-R^2 + B) \quad (5.12)$$

where B depends only on the support vectors (see Appendix C.2).

Figure 5.9 shows the descriptions obtained for different values of s , using as feature space the first two PCA projections of a set of face images (see the experiments subsection). For small values of s , all the vectors tend to be support vectors. By increasing s , the number of support vectors decreases. For large values of s , the description approximates the original spherical solution.

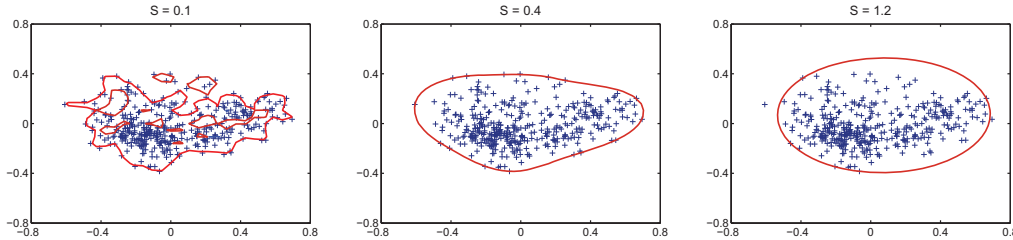


Figure 5.9: Scatterplots and data descriptions for the first two PCA projections of a training set of face images. Gaussian kernels with different widths ($s = 0.1, 0.4, 1.2$) are used. The solid lines show the description boundaries.

The original SVDD formulation corresponds to a hard-decision classifier (the output is 1 or 0). To be able to combine the SVDD with other classifiers we transform it into a soft-decision classifier. For an input pattern \mathbf{x} , the output of the SVDD is now:

$$SVDD(\mathbf{x}) = \sum_i \alpha_i \exp\left(\frac{-\|\mathbf{x} - x_i\|^2}{s^2}\right) \quad (5.13)$$

where x_i are the support vectors. Varying a threshold on this distance we obtain curves that are equidistant from the original description.

Experiments

To analyze the performance of PCA-based methods, we use the same set of eigenvectors and eigenvalues for all the classifiers. To train the PCA we work with a set of 1008 images. Frontal or near frontal face images of different people were obtained from three face datasets: the 400 ORL images, 400 images from XM2VTS (200 individuals, 2 samples per individual) and 208 images from Banca (52 individuals from each of the four country sets). Faces were manually cropped and rescaled to 40×48 pixels.

In order to reduce the influence of hair and background, an elliptical, binary mask is applied and only pixels in the mask are used. The effect of directional or global lighting changes is reduced by normalizing each image in mean and variance (mean 0, variance 1). The same preprocessing (masking and normalization) is applied to all the patterns to be classified (for the test set).

For the tests, we collected a set of target and outlier regions. For different images, BPTs were created, some of the nodes were discarded using size and aspect ratio criteria and the remaining nodes were manually annotated as target or outlier. We use a total of 3274 target regions (388 XM2VTS, 400 BioID, 2080 Banca Spanish, 256 Banca (other countries), 150 MPEG7), and 10559 outliers (696 XM2VTS, 5954 Banca and 3909 MPEG7). To train the SVDD classifier a second training set is needed, so half of the target data was used for training and the other half for testing.

To be able to compute the distances for a region associated with a BPT node, the region has to be scaled to the size of the training data, and missing information from outside the region has to be filled. We do this with the following steps: (i) find the region bounding box, (ii) scale the region (only luminance component) to an image with the size of the training images, preserving the aspect ratio of the region, (iii) fill the values outside the region with original image data, (iv) mask with the ellipse and (v) normalize in mean and variance.

Note that these preprocessing stages may be a bit costly, but they are only performed on a small number of nodes (between 5 and 10% of the original BPT nodes), those that remain active after the simplification step (see next chapter for details about the simplification stage).

To test DIFS, DFFS and DMAH we compute the distances for all target and outlier regions, for values of M , the dimension of the feature space, ranging from 1 to 200. The area under the curve (AUC) is used as a global measure of the classifier accuracy (the larger the AUC, the better the system). Figure 5.10 shows, for each classifier, the AUC plotted against M (left column) and the ROC with largest AUC (right column). We observe that DFFS and DMAH have a similar behavior, with slightly better results for DMAH. This is a reasonable result, since in the estimate of the Mahalanobis distance (eq. 5.8) the reconstruction error term or DFFS has a much greater weight than the DIFS term.

The best ROC is obtained with $M = 3$ features for DFFS (AUC = 0.948 and target acceptance rate TAR = 0.76 for a false positive rate FPR = 0.05) and with $M = 4$ features for DMAH (AUC = 0.955 and TAR = 0.79 for FPR = 0.05).

In turn, DIFS does not perform well for low values of M and, even though it improves for values of M larger than 40, the target acceptance rate is very low for acceptable values of false positives. These results suggest that the elliptical solution is a poor estimation of the target class boundary. The best ROC curve, with AUC = 0.90 is obtained for $M = 80$.

The results of the experiments performed with the SVDD are presented in Figure 5.11. Now there are two parameters, the dimension of the feature space M , which varies between

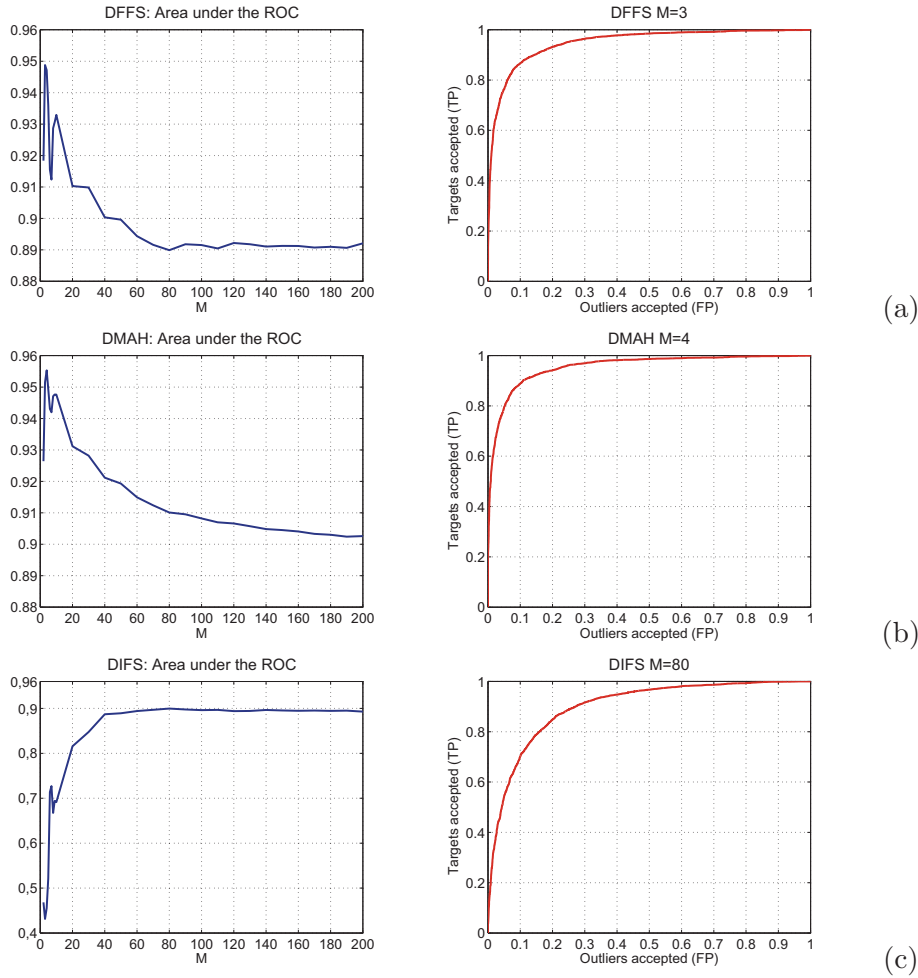


Figure 5.10: Areas under the curve and best ROCs for DFFS (a), DMAH (b) and DIFS (c) classifiers.

2 and 200, and the width of the Gaussian kernel, from 0 to 1. The experiments with SVDD were performed with Matlab PRTools4.1.3 [42] and dd.tools1.6.3 [170].

Figures 5.11(a) and (b) show, for varying M and s , the area under the curve and the number of vectors supporting each description, respectively. We observe that the AUC values are very high for values of s between 0.2 and 0.4, and increase with M . However, so does the number of support vectors.

The number of support vectors increases with the dimensionality of the feature space. In high dimensions (and for s lower than 0.3) most of the points become support vectors. This suggests that for such dimensions the number of training points is not enough, and more points are needed to find a reliable description of the class boundary [169, 183].

Also for very low values of s , most of the points become support vectors, since the de-

scription approximates a Parzen density estimation. By increasing s , the number of support vectors decreases as the boundary approximates an ellipse, but, at the same time, more outlier objects are accepted, so the AUC decreases.

In the next experiment, we fix an operation point for the classifiers so that the false negative rate is 0.05, and compute the target acceptance rates for varying values of M and s . The results, presented in Figure 5.11(c) show again that the best rates are obtained for tight descriptions with values of s between 0.2 and 0.4, and a feature space of dimension $M \geq 40$.

As a compromise between classifier accuracy and reliability of the description, we select the following parameters: $M = 40$ and $s = 0.4$, which gives a classifier with $AUC = 0.955$ and $TAR = 0.80$ for $FNR = 0.05$, where the 55% of the points are support vectors. This is the SVDD classifier that will be used in the ensemble. The ROC curve for this classifier is shown in Figure 5.11(d).

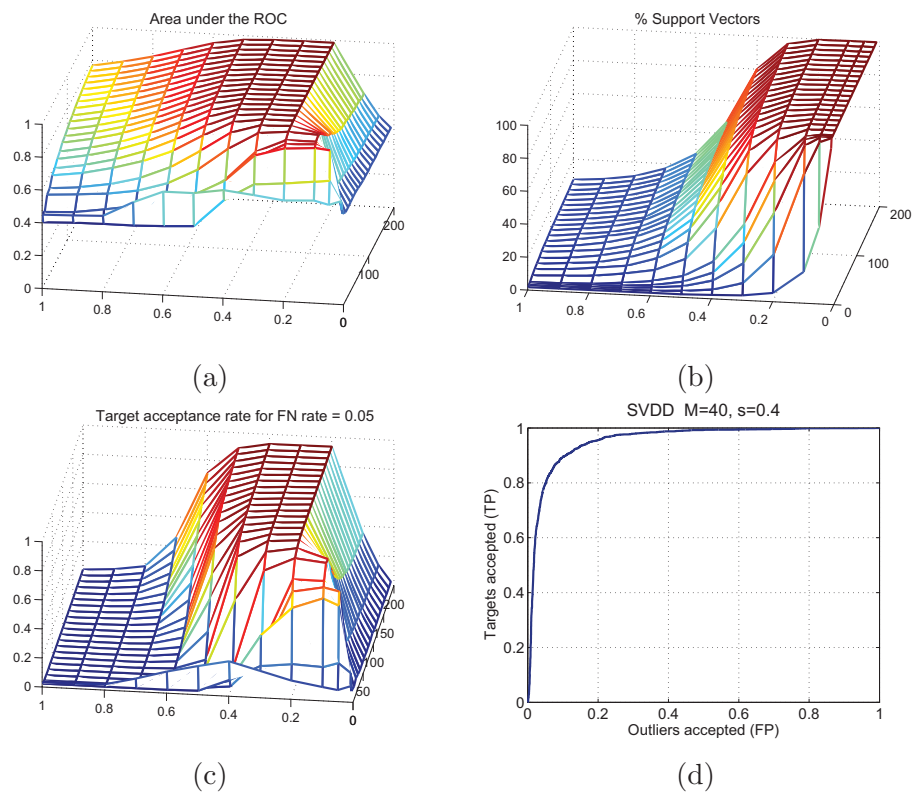


Figure 5.11: (a) Area under the ROC and (b) percentage of support vectors for SVDD classifiers trained with different values of M (from 1 to 200) and s (from 0.05 to 1). (c) Variations on the target acceptance rate for a fixed value of false negative rate 0.05, varying M and s , and (d) ROC curve of the selected classifier ($M = 40$, $s = 0.4$).

Besides the results presented in this section, we performed additional experiments with

DIFS and SVDD for different combinations of training and testing datasets. The results, published in [183] show that when the training set is a representative sample from the face class distribution, both classifiers perform well, and SVDD outperforms DIFS (same accuracy with half the features). When the training data distribution is very different from the test set distribution, the SVDD overtrains and its performance degrades. Also, as more variability is introduced in the training data (variations in pose expression and illumination), the complexity of the target class increases and more features (higher values of M) are needed to reach the same accuracy obtained for more homogeneous data. Since our goal is to detect faces in unconstrained scenarios, for the experiments presented in this section we have preferred to introduce some variability in the training set, and train the classifiers with images collected from different databases and with some variability in pose and facial expressions.

5.2.3 A two class classifier

The last texture-based classifier to be analyzed belongs to the group of techniques that use local features and a boosting method for feature selection and training.

Within this group we find state-of-the-art techniques like [185] (see Section 3.2.2). The two key points of their system are real time performance and detection accuracy. Fast performance is achieved by the use of an intermediate image representation to speed up the computation of features, and a cascade of classifiers with increasing complexity to rapidly reject outliers with the first classifiers. But in this chapter we are interested in detection accuracy, which in [185] is achieved by the use of AdaBoost to train weak classifiers (decision stumps) based on intensity differences between different rectangular configurations.

Boosting is a method of finding a highly accurate classification rule by combining many ‘weak’ hypothesis or classifiers, each of which is only moderately accurate [51, 152].

In the basic AdaBoost procedure, each base classifier m produces a classification rule $f_m(x) : \mathcal{X} \rightarrow \{-1, 1\}$, where \mathcal{X} is the feature domain. At each round of learning, one weak classifier is selected. Classifiers are trained on weighted versions of the training samples, giving higher weights to samples that are currently misclassified. The technique improves the performance of a single classifier by producing a ‘majority vote’ of similar classifiers. From a statistical perspective, AdaBoost fits an additive model $F(x) = \sum_{m=1}^M c_m f_m(x)$ in a forward stagewise manner (where the c_m are constants). The prediction is $sign(F(x))$ (see the basic AdaBoost algorithm in Appendix C.3).

A generalization of the basic procedure proposed in [52], the Real AdaBoost algorithm, uses real-valued predictions instead of binary outputs. In Real AdaBoost, the weak learners produce mappings $f_m(x) : \mathcal{X} \rightarrow \mathcal{R}$, where the sign of $f_m(x)$ gives the classification and the absolute value $|f_m(x)|$ gives a measure of the confidence in the prediction. Each weak learner returns a class probability estimate $p_m(x) = \hat{P}_\omega(y = 1|x) \in [0, 1]$. Its contribution to the final classifier is half the logit transform of this probability estimate $f_m(x) = \frac{1}{2} \log \frac{p_m(x)}{1-p_m(x)}$, which

is a real value (see the Real AdaBoost algorithm in Appendix C.3).

An improved version of this algorithm, called Gentle AdaBoost [52], uses a different strategy to fit the regression model, working with Newton stepping rather than exact optimization at each step. The main difference with Real AdaBoost is how it uses its estimates on the weighted class probabilities to update the functions. The update or contribution is now $f_m(x) = P_w(y = 1|x) - P_w(y = -1|x)$ rather than half the logit transform. This algorithm has a similar performance to Real AdaBoost on regular data but outperforms it on noisy data, and is much more resistant to outliers [95] (see the Gentle AdaBoost algorithm in Appendix C.3).

In the next subsection we study the performance of Real and Gentle AdaBoost algorithms using decision stumps and small decision trees (dimensionality 2 and 3) as weak classifiers. Instead of working with the large set of features computed from the integral image representation as in [185] we work with a much smaller set of features, the coefficient of a three level Haar wavelet decomposition of the region. As will be shown in the experiments, boosting these features in one strong classifier results in high accuracy with very low training cost since only one classifier is trained.

Experiments

Our feature set is formed by the coefficients of a three level Haar wavelet decomposition of a region. First, regions are preprocessed in the same way they were processed for the PCA-based classifiers, finding the region bounding box, scaling the region to an image with the size of the training data (40×48) but preserving the original aspect ratio, filling missing values with original image data, and normalizing in mean and variance. The rectangular subimage is then transformed, and masking is applied to transformed coefficients. The elliptical mask is scaled and used at each level of the decomposition, as shown in Figure 5.12. The total number of Haar features is 1676.



Figure 5.12: Elliptical mask.

For the experiments we work with the same sets of target and outlier regions used for

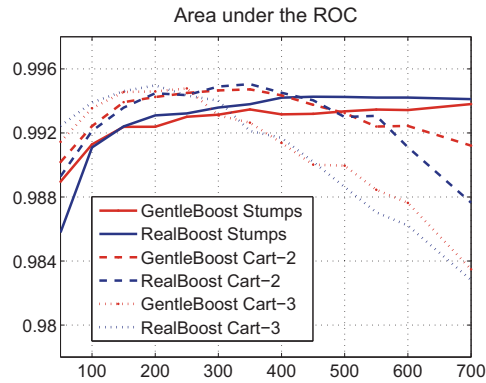


Figure 5.13: AUC for the six classifiers, varying the number of features from 50 to 700.

the PCA-based classifiers, plus an additional group of 4786 outliers. These outliers are the regions associated with the BPT nodes of 100 MPEG7 images that do not contain any face. The whole set is divided into two groups, one for training with 3159 target regions and 7720 outliers and the other one for testing, with 1123 targets and 6452 outliers. Experiments are performed using the GML Adaboost Matlab Toolbox [114].

We train several strong classifiers, with a number of features ranging from 50 to 700, using Real AdaBoost and Gentle AdaBoost training algorithms. As weak classifiers we try decision stumps and also decision trees (CART trees [41]) with 2 and 3 splits. Figure 5.13 shows the area under the ROC curve for the six different classifiers. As can be seen, stumps are outperformed by the weak tree classifiers with 2 or 3 split nodes when using less than 300 features. As more features are used, the performance of CART trees decreases, while for classifiers based on stumps increases and keeps stable.

The best tradeoff between complexity (in training and testing) and performance is obtained for decision stumps with 400 features. The Figures 5.14 (a) and (b) show the ROC curves and a zoom on their top-left hand corner for the two boosting algorithms. Both classifiers have a detection rate of about 0.94 for an error rate of 0.005. For lower false positive rates, Gentle AdaBoost outperforms Real Adaboost.

Figures 5.14 (c) and (d) show, for decision stumps with 400 rounds of boosting, the error rate as a function of the number of boosting iterations, for the training and test sets, respectively. We observe in (c) that the training error approaches zero exponentially in the number of rounds, for both algorithms.

5.2.4 Symmetry

Symmetry is an intrinsic characteristic of shapes and objects [201]. In its mathematical form symmetry is a binary feature of an object (an object is symmetric or not). However, when

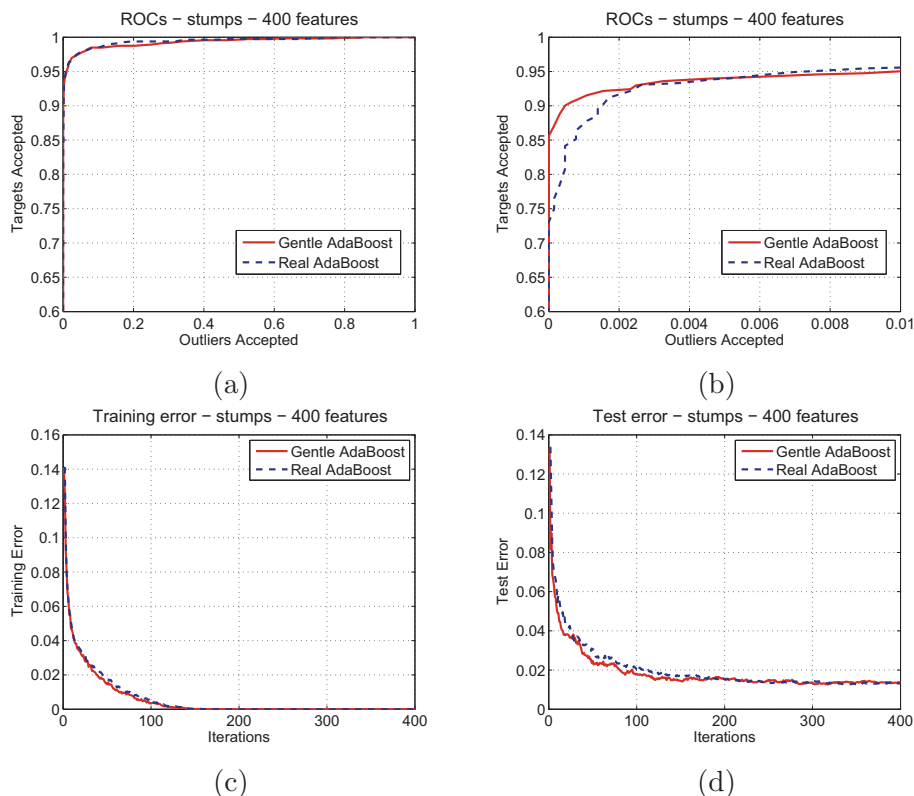


Figure 5.14: (a) ROC curves and (b) zoom for Gentle and Real Adaboost classifiers using decision stumps as weak learners and 400 features, (c) training error and (d) test error for Gentle and Real Adaboost classifiers with decision stumps and 400 features.

dealing with the projection of an object into the image plane additional deviation from exact symmetry occurs, due to perspective projection, digitalization, occlusion, etc. Therefore, a descriptor that measures the amount of symmetry of an object as a continuous value seems more appropriate to detect or retrieve symmetric or near symmetric objects. Since faces are almost symmetric (with reflectional symmetry), such a descriptor may be useful to discriminate faces from non-symmetric outliers.

In this work we define a descriptor of reflectional symmetry for image regions based on a local measure of symmetry in gray level images proposed in [87]. In [87], the measure is used to detect significant symmetry in images through a global optimization approach of a function parameterized by the location of the supporting region, its scale and the orientation of the axis of symmetry.

The 2D descriptor is based on the following symmetry measure for 1D functions. A 1D real function $y = f(x)$ defined on $[-L, L]$ is reflectionally symmetric with respect to the origin if $f(x) = f(-x) \forall x$, and antisymmetric if $f(x) = -f(-x) \forall x$. Any function

$f(x)$ can be uniquely represented as the sum of a symmetric and an antisymmetric function $f(x) = f_s(x) + f_a(x)$, where $f_s(x) = \frac{1}{2}(f(x) + f(-x))$ and $f_a(x) = \frac{1}{2}(f(x) - f(-x))$. A measure of symmetry of f with respect to the origin is

$$S\{f(x)\} = \frac{\|f_s(x)\|^2}{\|f(x)\|^2} = \frac{\|f_s(x)\|^2}{\|f_s(x)\|^2 + \|f_a(x)\|^2} \quad (5.14)$$

This measure verifies: $S\{f(x)\} \in [0, 1]$, $S\{f(x)\} = 1$ if $f(x)$ is symmetric, and $S\{f(x)\} = 0$ if $f(x)$ is antisymmetric.

To extend it to 2D functions, it is assumed that $f(x, y)$ is a 2D real function that is zero everywhere except within a circle of radius L centered at the origin. The reflectional symmetry is measured with respect to an axis t through the origin that makes an angle θ with respect to the x axis (see Figure 5.15).

$$\begin{aligned} t &= x \cos \theta + y \sin \theta \\ s &= x \sin \theta - y \cos \theta \end{aligned} \quad (5.15)$$

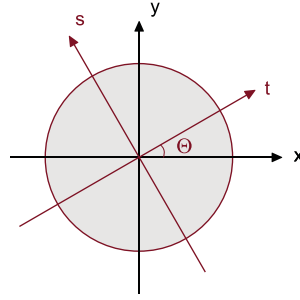


Figure 5.15: Coordinate system for 2D symmetry measurement.

In the new coordinate system, the function is reflectionally symmetric if $f(t, s) = f(s, t) \forall s, \forall t$. The function is 2D symmetric with respect to t axis if it is 1D-symmetric on line segments parallel to the s axis, for all values of t . Therefore, a 2D measure of symmetry with respect to axis t is defined in [87] as follows:

$$S_\theta\{f\} = \frac{\int \|f_s(t, s)\|^2 dt}{\int \|f(t, s)\|^2 dt} \quad (5.16)$$

where, for each t , $\|f(t, s)\|^2 = \int_{-L}^L f^2(t, s) ds$ is the norm of a 1D function of s .

The measure can be expressed in terms of the 1D symmetry values $S\{f(t, s)\}$:

$$S_\theta\{f\} = \frac{\int S\{f(t, s)\} \|f(t, s)\|^2 dt}{\int \|f(t, s)\|^2 dt} \quad (5.17)$$

Finally, the expression can be rewritten as follows:

$$S_{\theta}\{f\} = \frac{1}{2} \frac{\int \int f(t, s) f(t, -s) ds dt}{\int \int f^2(t, s) ds dt} + \frac{1}{2} \quad (5.18)$$

In the original formulation, the measure is applied locally to square $2L \times 2L$ image windows convolved with a Gaussian function which defines a circular support. A global optimization algorithm is then proposed to find the area with dominant symmetry in the image (varying scale, center and orientation).

In our implementation, Eq. 5.18 is discretized and applied to the luminance component of a region R , that is, $f(x, y) = Y(x, y)$ for $(x, y) \in R$ and 0 elsewhere. The function f is previously normalized by subtracting the average region luminance.

The origin of coordinates is the mass center of the region (x_0, y_0) . The axis of symmetry is defined by this center and the orientation of the region θ . The center and orientation can be computed with the descriptors proposed in Section 6.1.

As the regions are usually not symmetrically shaped (with respect to axis t), two minor changes are introduced in Eq. 5.18. First, the integral is computed only where both $f(t, -s)$ and $f(t, s)$ are points in R . An example is presented in Figure 5.16. The region shown in (a) is not symmetric (only one of the ears is present), so only points painted in blue (b) are used to compute the measure. We denote this area by S_R , the *symmetric support* of R . The black line shows the axis of symmetry.

A second modification is introduced to take into account the number of pixels which are really involved in the computation of the symmetry:

$$S_{\theta, x_0, y_0}\{R\} = S_{\theta, x_0, y_0}\{f(x)\} \cdot \frac{N_{S_R}}{N_R} \quad (5.19)$$

where S_R is the symmetric support of R , (x_0, y_0) the center, θ the orientation, N_{S_R} and N_R the number of pixels in S_R and R , respectively.

Up to now, we have assumed that the orientation of the region coincides with the direction of the axis of symmetry. However, as can be seen in the example of Figure 5.16(c) this assumption is not always verified. The dashed line shows the region orientation, and the solid line is the axis of symmetry.

Instead of using a fixed center and orientation, we search for the center and orientation that maximize the symmetry measure. The search space consists of a small number of center positions and orientations around the original mass center and orientation of the region. The result of the search is presented in Figure 5.16(d), where the solid line shows the best axis and the blue pixels are the pixels used to compute the symmetry (S_R).

$$SYM(R) = \arg \max_{\theta, x_0, y_0} S_{\theta, x_0, y_0}\{R\} \quad (5.20)$$

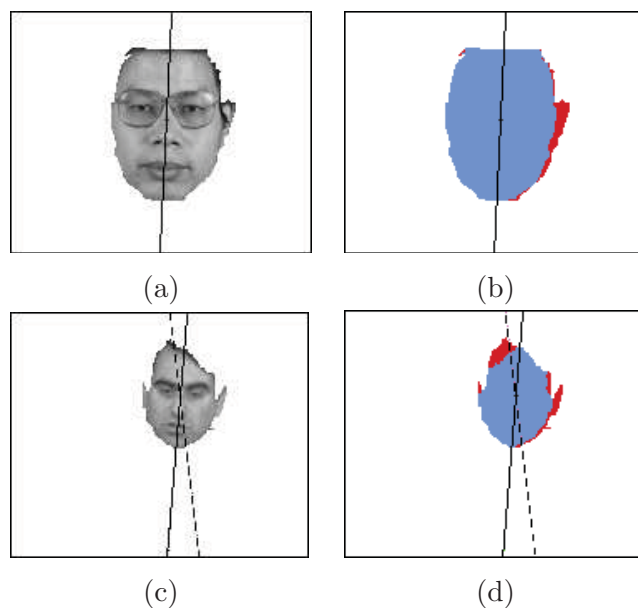


Figure 5.16: Image regions (a) and (c), their symmetric support and axis of symmetry in (b) and (d). In (c) and (d) the dotted line is the region orientation.

Experiments

To analyze the usefulness of the SYM descriptor, we test it on a set of target and outlier regions (450 target regions of frontal or near frontal faces from the Banca French and XM2VT sets, and 10550 outlier regions). The empirical distribution for the two classes and the ROC curve obtained varying a threshold on the symmetry value are shown in Figure 5.17.

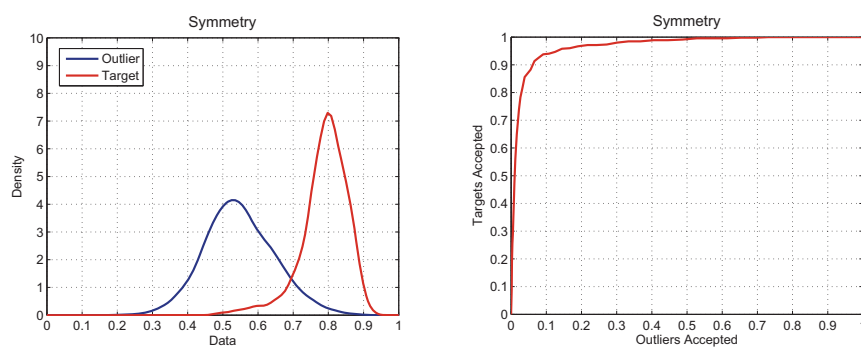


Figure 5.17: Empirical distribution of the symmetry measure for target and outlier regions (a) and ROC curve (b).

5.3 Shape

Object shape features provide a powerful clue to object detection since in many cases the shape of an object is strongly linked to the object functionality and identity. The goal of shape descriptors is to capture the shape of a region in a concise manner, without the requirement that the original shape can be reconstructed from the descriptor. A large number of shape descriptors have been proposed in the literature [103, 14], based on the spatial distribution of pixels (region-based descriptors) or on the outline contours (contour-based descriptors).

Although face shape varies with person and hairstyle, it is frequently modeled with an ellipse. This approximate model is not enough by itself to detect faces, but can be very useful for discriminating faces from many other non-elliptical objects. In this work, we model the face shape with an ellipse, and use a shape descriptor based on the Hausdorff distance that measures the resemblance between the contour of a region and the contour of the ellipse.

The Hausdorff distance is a metric between two point sets. Given two finite point sets $A = \{\mathbf{a}_1, \dots, \mathbf{a}_p\}$ and $B = \{\mathbf{b}_1, \dots, \mathbf{b}_q\}$, the Hausdorff distance is defined as

$$H(A, B) = \max(h(A, B), h(B, A)) \quad (5.21)$$

where

$$h(A, B) = \max_{\mathbf{a} \in A} \min_{\mathbf{b} \in B} d(\mathbf{a} - \mathbf{b}) \quad (5.22)$$

and $d(\cdot)$ is a distance between the two points (usually the L_2 distance).

The function $h(A, B)$ is called the *directed Hausdorff distance* from A to B . It identifies the point $\mathbf{a} \in A$ that is farthest from any point of B and measures the distance from \mathbf{a} to its nearest neighbor in B . Intuitively, if $h(A, B) = d$, then each point of A must be within distance d from some point of B , and there is also some point of A that is exactly at distance d from the nearest point of B (the most mismatched point). It is not a distance but a directed distance, since it is not symmetric.

The Hausdorff distance $H(A, B)$ measures the degree of mismatch between two sets by measuring the distance of the point of A that is farthest from any point of B and viceversa. Thus, the notion of resemblance encoded by this distance is that each member of A is near some member of B and viceversa. Unlike most methods of comparing shapes, there is no explicit pairing of points of A with points of B . The Hausdorff distance is a metric over the set of all closed, bounded sets [73].

If A are the contour points of a region and B the contour points of the ellipse, the Hausdorff distance can be used to determine the similarity between the region and the shape model.

This distance is very sensitive to noise, since a single outlier contour pixel in A or in B may cause $h(B, A)$ or $h(A, B)$ to be large.

A more robust measure (but not a distance) that decreases the impact of outliers is proposed in [39], where the *modified directed Hausdorff distance* is defined as the average of

single point distances:

$$h_{mod}(A, B) = \frac{1}{|A|} \sum_{\mathbf{a} \in A} \min_{\mathbf{b} \in B} \|\mathbf{a} - \mathbf{b}\| \quad (5.23)$$

The modified symmetric ‘distance’ is computed by taking the maximum of the two directed measures. The Hausdorff distance descriptor for a region R and shape model B is then:

$$HAUS(R) = \max(h_{mod}(R, B), h_{mod}(B, R)) \quad (5.24)$$

Experiments

To test the performance of the descriptor, the Hausdorff distance between the elliptical shape model and a set of target and outlier regions (the same set used for the symmetry descriptor) is computed. Previously, the region mask is scaled to the size of the shape model (40×48) and the contour points are extracted.

The empirical distributions of the face and outlier classes as well as the ROC curve obtained by thresholding this distance are shown in Figure 5.18

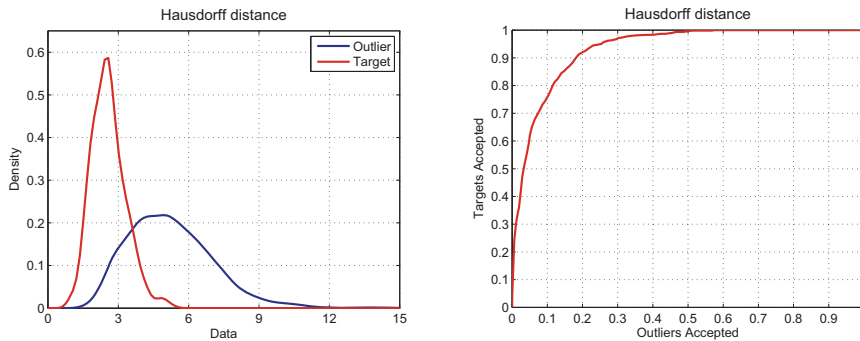


Figure 5.18: Empirical distribution of the Hausdorff distance for target and outlier regions (a) and ROC curve (b).

5.4 Combination of classifiers

In the previous sections we have analyzed color, texture and shape descriptors which are useful to characterize the class of faces, and we have proposed and tested several classifiers based on individual descriptors. The goal of this section is to study how to merge information provided by these attributes, through the combination of their associated classifiers.

Each single classifier outputs a score indicating the support for the target class. The objective of classifier fusion is to combine the outputs in order to improve the overall performance of the system, that is, to improve the accuracy of the best single classifier.

In this section three aspects of classifier fusion are addressed. First, the concept of classifier *diversity* is reviewed and the most diverse classifiers are selected from the set of available classifiers. Next, to ensure a meaningful combination, *score normalization* is performed by transforming the classifier outputs into a common domain. Then, normalized scores are combined through different *combination rules*, and finally the performance of the different combinations of normalization techniques and fusion rules is studied.

The data for the experiments is the same data used to train and test individual classifiers in previous sections; half the samples are used to analyze diversity and to train the different parameters that appear in normalization and fusion rules, while the rest of the samples are used in the evaluation of performance.

5.4.1 Diversity

Common intuition suggests that the classifiers in the ensemble should be as accurate as possible and should not make the same errors. That is, they should be diverse. Diversity, sometimes called negative dependence, independence or complementary, is recognized as a key issue when combining classifiers [159].

The diversity of classifiers is then a fundamental requirement for the success of the ensemble. Ensemble members should exhibit some level of accuracy and, in turn, ensemble members should not be identical, exhibiting some level of diversity. However, quantifying these statements has proved challenging [88].

Many measures of diversity have been proposed, and there are some studies that try to relate these measures to the accuracy of the ensemble [89, 88]. However, the results are only valid under some assumptions (concerning, for example, classifier output, optimization routine or sufficient training data), which are not always valid in practical situations.

In this work we use diversity as a tool to select the members of the ensemble. The goal is to select, from the set of proposed classifiers, a subset of diverse classifiers such that their combination improves the accuracy of the best single classifier. We work with pairwise measures, that consider a pair of classifiers at a time.

Since diversity measures are defined for classifiers with label outputs, we have to fix an operation point for each classifier to convert soft outputs into label outputs (1 for target, 0 for outlier).

An ensemble of L classifiers produces $L(L - 1)/2$ pairwise measures. To avoid having to calculate all these measures, we first find linear dependencies between classifiers, computing the Pearson correlation coefficient between all pairs using their soft labels (the original continue output values). Then, for the pairs with high correlation values, we fix an operation point to obtain class labels and analyze the diversity using other measures that take into account the classifiers' joint errors.

	MEANC	DOMC	HAUS	SYM	DFFS	DIFS	DMAH	GENT	REAL	SVDD
MEANC	1	0.898	-0.179	0.271	-0.202	-0.147	-0.220	0.397	0.398	0.303
DOMC	0.898	1	-0.150	0.218	-0.171	-0.122	-0.187	0.330	0.328	0.246
HAUS	-0.179	-0.150	1	-0.387	0.078	0.064	0.093	-0.286	-0.283	-0.157
SYM	0.271	0.218	-0.387	1	-0.133	-0.101	-0.150	0.328	0.342	0.223
DFFS	-0.202	-0.171	0.078	-0.133	1	0.705	0.969	-0.223	-0.227	-0.525
DIFS	-0.147	-0.122	0.064	-0.101	0.705	1	0.702	-0.185	-0.187	-0.397
DMAH	-0.220	-0.187	0.093	-0.150	0.969	0.702	1	-0.245	-0.249	-0.558
GENT	0.397	0.330	-0.286	0.328	-0.223	-0.185	-0.245	1	0.936	0.364
REAL	0.398	0.328	-0.283	0.342	-0.227	-0.187	-0.249	0.936	1	0.370
SVDD	0.303	0.246	-0.157	0.223	-0.525	-0.397	-0.558	0.364	0.370	1

Table 5.2: Correlation coefficient between classifiers.

The correlation coefficients are shown in Table 5.2. The table shows that there is a high correlation between color classifiers MEANC and DOMC, as well as between DIFS, DFFS and DMAH, and between the two Adaboost classifiers. The correlation between SVDD and the other three PCA base classifiers, even when they are based on the same set of PCA-coefficients is not very high but still considerable. The results are reasonable, giving high correlation values for classifiers which are based on the same attributes. We keep Hausdorff and Symmetry descriptors out of the analysis, since their correlation with the other classifiers is low, and study diversity within two subgroups: the two color classifiers and the six texture classifiers.

To obtain label outputs, we fix for all classifiers a false acceptance rate of 0.05 and select, for each classifier D_i with $i = 1, \dots, L$, a threshold Θ_i so that the FAR is 0.05%.

Then, given a set of patterns Z with known labels, we classify the elements in Z with each classifier D_i and compute the vector y_i such that

$$y_{i,j} = \begin{cases} 1, & \text{if } D_i \text{ classifies } z_j \text{ correctly,} \\ 0, & \text{otherwise.} \end{cases} \quad (5.25)$$

Next, for a pair of classifiers D_i and D_j , we count the number of elements in Z that are correctly or wrongly classified by the classifiers and estimate the probabilities dividing by the number of samples, as shown in Table 5.3. Note that $a + b + c + d = 1$.

Four pairwise diversity measures have been defined [159] for data in Table 5.3:

	D_j correct (1)	D_j wrong (0)
D_i correct (1)	a	b
D_i wrong (0)	c	d

Table 5.3: Relationship table with probabilities; $a + b + c + d = 1$.

- *Correlation coefficient:*

$$\rho_{i,j} = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+d)(b+d)}} \quad (5.26)$$

- *Q statistic:*

$$Q_{i,j} = \frac{ad - bc}{ad + bc} \quad (5.27)$$

This measure varies between -1 and 1 . For statistically independent classifiers, $Q_{i,j} = 0$. Classifiers that tend to correctly classify the same objects have positive values of Q . Q and ρ have the same sign, and $\|\rho\| \leq \|Q\|$.

- *Disagreement measure:*

$$Dis_{i,j} = b + c \quad (5.28)$$

This is an intuitive measure which is equal to the probability that the two classifiers disagree in their decisions.

- *Double fault measure:*

$$DF_{i,j} = d \quad (5.29)$$

This measure gives the probability of both classifiers being wrong.

The set of joint agreement and error rates and the four measures for the pairs of classifiers are given in Table 5.4. Any of these measures can be used to choose a subset of classifiers. We work with the double fault measure (DFault) because it is related, by definition, to the ensemble performance. It is based on the concept that it is more important to know when simultaneous errors are committed than when both classifiers are correct [88]. The strategy is the following: starting with an ensemble formed by all the classifiers, we proceed iteratively by selecting the least diverse pair of classifiers, and eliminating the less accurate classifier of the pair, only if the performance of the new (reduced) ensemble is improved by the deletion. The process continues until no more classifiers can be eliminated without decreasing the global performance. Since we use the double fault measure as diversity measure, at each stage the pair that makes more simultaneous errors is selected, and the classifier with higher detection rate is preserved. Note that in order to measure the global performance previously we need

Pair	a	b	c	d	Corr	Qtest	Disag	DFault
MEANC-DOMC	0.917	0.019	0.011	0.053	0.7636	0.9914	0.0300	0.0526
DFFS-DIFS	0.892	0.036	0.027	0.045	0.5545	0.9520	0.0637	0.0454
DFFS-DMAH	0.917	0.010	0.014	0.059	0.8194	0.9949	0.0238	0.0589
DFFS-GENT	0.882	0.045	0.069	0.004	0.0017	0.0146	0.1147	0.0037
DFFS-REAL	0.882	0.045	0.069	0.004	0.0031	0.0273	0.1146	0.0038
DFFS-SVDD	0.884	0.423	0.059	0.014	0.1604	0.6550	0.1021	0.0138
DIFS-DMAH	0.894	0.024	0.038	0.044	0.5589	0.9551	0.0617	0.0443
DIFS-GENT	0.874	0.045	0.077	0.004	0.0019	0.0162	0.1224	0.0041
DIFS-REAL	0.874	0.045	0.077	0.004	0.0018	0.0152	0.1225	0.0041
DIFS-SVDD	0.879	0.039	0.064	0.018	0.2025	0.7178	0.1035	0.0175
DMAH-GENT	0.885	0.046	0.066	0.003	-0.0061	-0.0579	0.1118	0.0030
DMAH-REAL	0.885	0.046	0.065	0.004	0.0020	0.0182	0.1110	0.0035
DMAH-SVDD	0.888	0.043	0.055	0.014	0.1657	0.6708	0.0984	0.0136
GENT-REAL	0.931	0.020	0.020	0.029	0.5657	0.9698	0.0406	0.0288
GENT-SVDD	0.897	0.046	0.054	0.003	0.0016	0.0156	0.1001	0.0029
REAL-SVDD	0.897	0.046	0.054	0.003	0.0033	0.0318	0.1001	0.0030

Table 5.4: Pairwise diversity measures.

to normalize data and chose a fusion rule. We have tried the four best combinations of normalization and fusion rules (see next section) with the same result.

The order in which classifiers are eliminated is the following: DFFS, Mean Color, DIFS and Real Adaboost, so the final set is formed by 6 classifiers: Dominant Color likelihood, Symetry, Hausdorff distance, Mahalanobis distance, Gentle Adaboost and Svdd. Another reasonable result, giving one classifier for each ‘family’ of classifiers.

We also tested the other diversity measures, obtaining the same final set of classifiers although the order of elimination was different.

5.4.2 Normalization

Ideally we want the normalized values to be estimates of the probabilities of the target class, given the input. Some classifiers output soft labels which are estimates of the probability that the pattern belongs to the target class, or estimates of the likelihood of a pattern given the target class. However, other classifiers produce distances or dissimilarity values, or even degrees of support which are not likelihoods or probabilities. Furthermore, the outputs of individual classifiers may not be on the same numerical range or they may follow different

statistical distributions. Due to these reasons, normalization is an essential first step to transform outputs into a common domain prior to combining them [78].

The problem of data normalization has been extensively studied, particularly in the field of multimodal biometric systems, where the matching scores of different biometric modalities (e.g. face, fingerprint, hand geometry, etc.) are combined to produce a final score for recognition. In this context, several solutions have been proposed. A review of the most popular techniques of score normalization in the domain of biometric systems is presented in [78], describing methods like the min-max normalization, decimal scaling, Z-score, median and median absolute deviation (MAD), double sigmoid function, or tanh estimator. However, not all these techniques are useful for our purposes, since some of them do not scale data to a common range, but normalize the values with respect to their mean and variance. Ideally, data should be related to the degree of support for the target class, within a common range $[0, 1]$, with 0 meaning no-support and 1 meaning full support.

We work with the min-max and the double sigmoid rules (both used by biometric systems) and with another two techniques, robust min-max and logistic regression, to try to overcome some limitations of the first ones. In the following definitions, x and \tilde{x} are the original and normalized values, respectively.

- *Min-max:*

$$\tilde{x} = \frac{x - \min}{\max - \min} \quad (5.30)$$

where \max and \min are, respectively, the maximum and minimum values of the classifier outputs (for a training data set). Using this expression, the minimum score is shifted to 0 and the maximum score is shifted to 1. This method retains the distribution of the original data, except for a scaling factor, and transforms all values into the $[0, 1]$ range. However, it has the disadvantage of being highly sensitive to outliers in the data used for the min and max estimation. If the classifier outputs are distances instead of likelihoods or similarity scores, the normalization is $\tilde{x} = 1 - \frac{x - \min}{\max - \min}$

- *Robust min-max:* to make the min-max normalization robust to outliers, we omit the 5% of the values at the extreme tails of the distributions to compute the \min and \max values from the training data, and proceed as before. Values below \min and above \max are clipped to 0 and 1, respectively.
- *Double sigmoid:* this normalization rule, proposed in [22] transforms the output values into the $[0, 1]$ interval with a double sigmoid function:

$$\tilde{x} = \begin{cases} \frac{1}{1 + e^{-(x-t)/r_1}} & \text{if } x < t_0 \\ \frac{1}{1 + e^{-(x-t)/r_2}} & \text{otherwise} \end{cases} \quad (5.31)$$

where t_0 is a reference operating point, and r_1 and r_2 denote the left and right edges of the interval where the function is linear (the double sigmoid function is linear in the

interval $(t_0 - r_1, t_0 + r_2)$. The main drawback of the technique is that it requires careful selection of parameters t_0 , r_1 and r_2 to obtain good efficiency.

We select the values for each classifier as follows: t_0 is set as the threshold value that produces the minimum error rate (both target and outlier errors are considered), that is, t_0 is the threshold that produces the minimum number of target rejections and outlier acceptances. Then, left and right edges are calculated. $r_1 = t_0 - t_1$, where t_1 is the 5 percentile of the classifier output values for a given training set of target samples. Finally, $r_2 = t_2 - t_0$, where t_2 is the 95 percentile of the classifier output values for a training set of outlier samples.

Figure 5.19(a) shows an example of the double sigmoid normalization for the outputs of the Symmetry classifier. The scores are mapped using $t_0 = 0.6838$, $t_1 = 0.57$ and $t_2 = 0.72$.

- *Logistic*: instead of working with two sigmoid functions, we use logistic regression [67] to fit a logistic function to the data, transforming outputs x into the $[0, 1]$ interval:

$$\tilde{x} = \frac{1}{1 + e^{-b_0 - b_1 x}} \quad (5.32)$$

where parameters b_0 and b_1 are found from a set of training data by maximum likelihood estimation.

Figure 5.19(b) shows an example of the logistic normalization curve for the outputs of the Symmetry classifier.

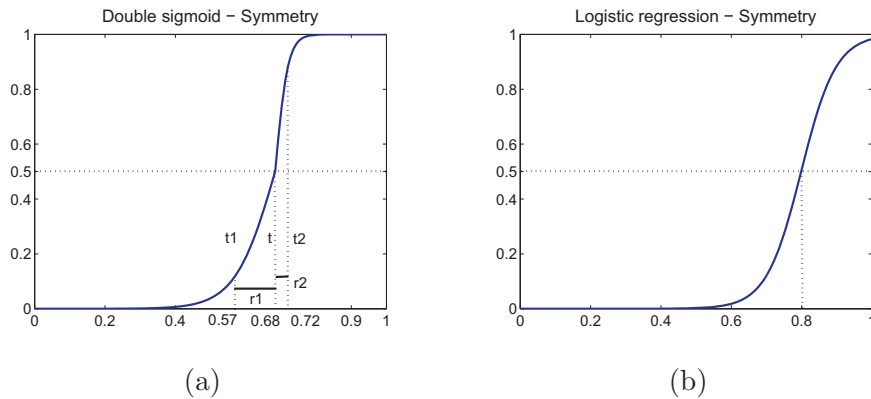


Figure 5.19: (a) Double sigmoid ($t = 0.6838$, $t_1 = 0.57$, $t_2 = 0.72$) and (b) Logistic regression normalization for the Symmetry classifier.

In the next subsection we discuss several rules to combine soft outputs and test the performance of these rules in combination with the proposed normalization approaches.

5.4.3 Combiners

In one-class classification the output of a classifier is, for each sample $\mathbf{x} \in \mathbb{R}^d$ (a feature vector), a real value which is the support for the hypothesis that the sample comes from the target class ω_T .

Let $\mathbb{D} = \{D_1, \dots, D_L\}$ be the set of L classifiers. After normalization the output values are in the $[0, 1]$ interval. Therefore, for each classifier D_i , $d_i : \mathbb{R}^d \rightarrow [0, 1]$, where $d_i(\mathbf{x})$ is the support for the target class. Let $\mu(\mathbf{x})$ denote the overall degree of support for the target class ω_T given by the ensemble.

In this section we test the performance of several combination rules. The first rules are simple, non-trainable combiners commonly used in the literature, which do not have extra parameters to train. Then, we test another three combiners. First, a weighted average combination, where weights are estimated by minimizing the average approximation errors of the classifiers using two different solutions for finding the weights. Finally, a different approach, similar to the logistic regression criterion used to normalize outputs, where a logistic function is fitted to data, using the classifier outputs as predictor variables.

- *Simple average* (μ_{ave}):

$$\mu_{ave}(\mathbf{x}) = \frac{1}{L} \sum_{i=1}^L d_i(\mathbf{x}) \quad (5.33)$$

- *Maximum, minimum, median* ($\mu_{max}, \mu_{min}, \mu_{med}$): which are defined as the maximum, minimum and median of the classifier outputs, respectively. For example, the *max* rule is

$$\mu_{max}(\mathbf{x}) = \max_i \{d_i(\mathbf{x})\} \quad (5.34)$$

- *Trimmed mean* (μ_{trim}): For a K percent trimmed mean, the L degrees of support are sorted and the K percent of the values are dropped on each side. The remaining values are averaged.

- *Product* (μ_{prod}):

$$\mu_{prod}(\mathbf{x}) = \prod_{i=1}^L d_i(\mathbf{x}) \quad (5.35)$$

- *Weighted average* (μ_{wave}):

$$\mu_{wave}(\mathbf{x}) = \sum_{i=1}^L w_i d_i(\mathbf{x}) \quad (5.36)$$

where $\sum_{i=1}^L w_i = 1$

The weights w_i are found by minimizing the variance of μ_{wave} , understood as a non-biased, minimum variance estimate of $P(\omega_T | \mathbf{x})$. One solution for finding the weights that

minimize the variance is derived by assuming that the classifier's errors in approximating the posterior probability $P(\omega_T|\mathbf{x}) - d_i(\mathbf{x})$ are normally distributed with zero mean [88]. If σ_{ij} is the covariance between the approximation errors by classifiers D_i and D_j , the weights are found by minimizing the following function:

$$J = \sum_{i=1}^L \sum_{j=1}^L w_i w_j \sigma_{ij} - \lambda \left(\sum_{i=1}^L w_i - 1 \right) \quad (5.37)$$

The solution minimizing J is

$$\mathbf{w} = \Sigma^{-1} \mathbf{I} (\mathbf{I}^T \Sigma^{-1} \mathbf{I})^{-1} \quad (5.38)$$

where $\mathbf{w} = [w_1, \dots, w_L]^T$ are the weights, Σ is the covariance matrix for the approximation errors and \mathbf{I} is an L -element vector with ones.

If it is assumed that the classifier outputs are independent, then Σ is diagonal with the variances of each classifier along the diagonal, and in that case the weights are proportional to the inverse of the variances, so the equation 5.38 reduces to:

$$w_i = \frac{\frac{1}{\sigma_i^2}}{\sum_{j=1}^L \frac{1}{\sigma_j^2}} \quad (5.39)$$

- *Logistic regression* (μ_{lreg}):

$$\mu_{lreg}(\mathbf{x}) = \frac{1}{1 + e^{-\sum_{i=1}^L \beta_i d_i(\mathbf{x})}} \quad (5.40)$$

where logistic regression is used to predict the class probability of \mathbf{x} , by fitting the data to a logistic curve, using the classifiers' outputs as predictor variables. Note that in this case it is not necessary to normalize the data previously, and we can use the original distances or likelihoods.

The performance has been studied for all combinations of normalization and fusion criteria. For each fusion rule, the four normalization strategies, namely min-max (*Minmax*), robust min max (*Rminmax*), double sigmoid (*Bisig*) and logistic (*LogReg*) normalization are used. Figure 5.20 presents the ROC curves for the four fusion rules that produce the best results: simple average (*Average*), weighted average for non-independent (*W1Average*) and for independent (*W2Average*) outputs, and logistic regression (*LogReg*). The results for the other fusion strategies (min, max, median, trimmed mean and product) are shown in Appendix D.

To summarize the results, Table 5.5 shows the area under the ROC (AUC), the equal error rate (EER) and the target acceptance rate (TAR) for various false acceptance rates (FAR) for the four best fusion methods. The best results for each fusion rule are in boldface, and the best global performances are marked with asterisks. The global best performance (the

highest AUC) is for logistic regression without normalization, which also shows the best target acceptance rates for very low false acceptance rates ($FAR = 0.001$). When larger values of FAR are accepted, weighted average combination (W1) and the simple average outperform logistic regression in terms of TAR. The best normalization rule for the two weighted average combinations is robust min max, while the double and single sigmoid functions are the best for the simple average.

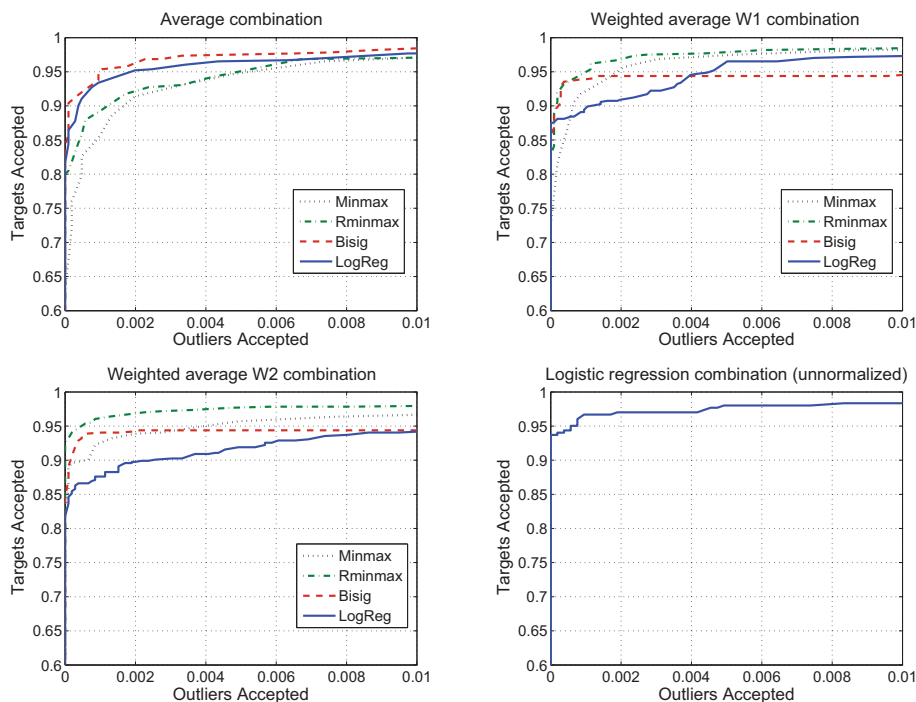


Figure 5.20: ROCs of fusion methods for different normalization rules: average, W1 weighted average (non-independent outputs, eq. 5.38), W2 weighted average (independent outputs, eq. 5.39) and logistic regression.

5.5 Summary

The proposed face detector is formed by two classifiers, a cascade of hard classifiers based on generic descriptors, and an ensemble of soft classifiers based on specific descriptors. In this chapter we have focused on the second classifier, which is the core part of the system. We have analyzed color, shape and texture attributes of faces and have proposed a set of region-based descriptors to measure these attributes. We have proposed some new descriptors or new implementations that take advantage of the region-based image model. Some of the descriptors rely on analogous descriptors proposed for block-based approaches and have been adapted to work with regions.

Fusion	Norm.	AUC	EER	TAR			
				FAR 0.01	FAR 0.005	FAR 0.001	FAR 0.0005
Average	Minmax	99.68	1.98	97.19	95.21	85.79	83.47
	Rminmax	99.75	1.81	97.19	95.54	90.08	87.77
	Bisigm	99.81	* 1.32	* 98.51	97.52	95.04	92.40
	Logit	99.70	1.87	97.69	96.69	93.39	91.24
W1 Average	Minmax	99.76	1.49	98.3	97.36	92.56	88.10
	Rminmax	99.82	1.49	* 98.51	* 98.18	94.55	93.55
	Bisigm	99.78	1.96	94.55	94.38	93.88	93.55
	Logit	99.72	1.95	97.19	96.53	89.59	88.43
W2 Average	Minmax	99.43	2.64	96.69	95.70	92.56	89.92
	Rminmax	99.79	1.98	98.02	97.85	96.20	* 95.37
	Bisigm	99.74	1.98	94.38	94.38	94.05	93.55
	Logit	99.36	2.85	94.21	91.90	87.60	86.61
LogReg	-	* 99.84	1.33	98.34	98.01	* 96.69	94.37

Table 5.5: Performance of the best combination rules with different normalization criteria.

Then, we have associated each descriptor with a one-class classifier (and a two class classifier for Haar coefficients) and have studied their individual performance. We have selected the most diverse classifiers and have studied the performance of several combinations of normalization and fusion criteria. The selected classifiers are Dominant color likelihood, Symmetry, Hausdorff distance, Mahalanobis distance on PCA coefficients, Gentle Adaboost on Haar coefficients and SVDD on PCA coefficients. The combinations of normalization and fusion rules that produce the best results in terms of the AUC are: logistic regression (without data normalization), simple average with double sigmoid normalization, and weighted average (W1 and W2) with robust min-max normalization.

Part of the work performed in this chapter is published in the following papers and book chapter:

- V. Vilaplana, F. Marqués, *Support vector data description based on PCA features for face detection*, Proceedings of EUSIPCO 2005, European Signal Processing Conference, Antalya, Turkey, September 2005.
- V. Vilaplana, F. Marqués, *Face detection and segmentation on a hierarchical image representation*, Proceedings of EUSIPCO 2007, 15th European Signal Processing Con-

ference, pp. 1955-1959, Poznan, Polonia, September 2007.

- X. Giro, V. Vilaplana, F. Marqués, P. Salembier, *Automatic extraction of visual objects information*, in *Multimedia content and Semantic Web*, edited by G. Stamou and S. Kollias, Wiley, 2005, ISBN: 0-470-85753-6.

Chapter 6

Face detection on BPTs

In Chapter 5 we studied color, shape and texture attributes of human faces and analyzed different strategies to build a strong classifier by combining simpler classifiers based on these attributes. In this chapter we complete the description of the face detection system proposed in Section 3.9.

The face class is described at two levels. First, at a very general level, with a set of low level features (which we call *generic descriptors*) associated with very simple classifiers, each classifier based on one single feature. Second, at a more specific level, using more complex features and classifiers (*specific descriptors*). The first description is used to simplify the search, eliminating many of the candidate nodes, while the second, more costly, is applied only to the remaining nodes.

In the first section of this chapter, a set of generic descriptors is proposed. They are associated with low level visual attributes, which are common to any object. They are simple and relatively easy to measure, and can be pre-computed for all the nodes when creating the tree. In the second section, each generic descriptor is associated with a simple threshold classifier which is trained on sample data and the classifiers are combined in a cascade, which is used to simplify the search, by rejecting many of the BPT nodes. Next, the node extension stage is detailed. Here, the area of support of each remaining node is modified to conform to a reference face shape model. Finally, a decision stage is applied to select, among several candidate nodes, only one node for each face that is present in the scene. We close the chapter with a large set of experiments that show the performance of the complete system in different scenarios.

6.1 Generic Descriptors

In this section we describe a set of region-based descriptors related to simple low-level visual attributes of image regions. The descriptors are computed for all the nodes in the search

space of the BPT (see Section 4.2). They are used in the simplification stage (see Section 6.2) or in other stages during the detection process.

6.1.1 Geometry

The following are descriptors that measure general properties of regions related to their position and orientation in the image.

Mass Center: The center of mass of region R is

$$mc(R) = (\bar{x}, \bar{y}) \quad (6.1)$$

where $\bar{x} = \frac{1}{N} \sum_{(x,y) \in R} x$, and $\bar{y} = \frac{1}{N} \sum_{(x,y) \in R} y$ and N is the number of pixels in R .

Orientation: The orientation of a region $o(R)$ is defined as the angle between the axis of the least moment of inertia and the x image axis (see Figure 6.1(a)). It is obtained by maximizing, with respect to ϕ , the sum

$$I(\phi) = \sum_{(x,y) \in R} [(x - \bar{x})\cos\phi - (y - \bar{y})\sin\phi]^2 \quad (6.2)$$

The result is [76]

$$o(R) = \phi = \arctan \left[\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}} \right] \quad (6.3)$$

where $\mu_{p,q}$ are the central moments $\mu_{p,q} = \sum_{(x,y) \in R} (x - \bar{x})^p (y - \bar{y})^q$.

Bounding box: The bounding box $bbox(R)$ of a region R is the smallest rectangle around the region that is aligned with the image axis. It can be characterized, for instance, with two points, the top-left P_1 and bottom-right P_3 points as shown in the example of Figure 6.1(a).

Oriented bounding box: The oriented bounding box $obbox(R)$ of a region R is the smallest rectangle around the region that is aligned with its orientation. Once the orientation of the region $o(R) = \phi$ is known, the transformation

$$\begin{aligned} \alpha &= x\cos\phi + y\sin\phi \\ \beta &= -x\sin\phi + y\cos\phi \end{aligned} \quad (6.4)$$

is used to find the values of α and β which define the coordinates of points P_1, \dots, P_4 . These four points characterize the oriented bounding box (see Figure 6.1(b)).

Position in the image: This feature describes how much of the image is occupied by the region, and where it is located. The image is divided into $n_r \times n_c$ rectangular subimages,

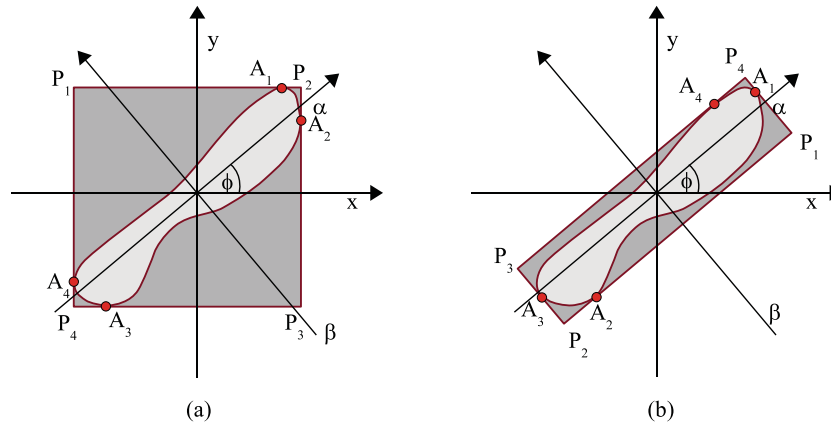


Figure 6.1: Orientation ϕ , bounding box (a) and oriented bounding box of a region.

and subimages are scanned by rows. For each subimage i , p_i measures the proportion of the region that is included in i , and N is the number of pixels in R .

$$\begin{aligned} pos(R) &= (n_r, n_c, \{p_i\}_{i=1, \dots, n_r n_c}) \\ p_i &= \sum_{(x,y) \in R} I((x,y) \in i) / N \end{aligned} \quad (6.5)$$

6.1.2 Color

Two color descriptors are used: mean and variance.

Color mean: The mean value of the pixels within the region in each color component (YCbCr space).

$$mean(R) = \left(\frac{1}{N} \sum_{(x,y) \in R} Y(x,y), \frac{1}{N} \sum_{(x,y) \in R} Cb(x,y), \frac{1}{N} \sum_{(x,y) \in R} Cr(x,y) \right) \quad (6.6)$$

where N is the number of pixels in region R .

Color variance: The variance $var(R)$ of color values of the region, in each component.

6.1.3 Texture

We work with a set of descriptors that measure the texture homogeneity of a region.

Homogeneity: The 2D discrete wavelet transform (DWT) decomposes an image into different frequency bands and different resolutions using a successive application of low-pass and high-pass filters (i.e., basis functions) resulting in smooth approximations as well as detailed information. We work with Haar basis functions and compute the energies

of the coefficients associated with the region pixels for the Low-High (LHi), High-Low (HLi) and High-High (HHi) subbands, in each level i of the decomposition ($i \in \{1, 2\}$ since we use two levels).

6.1.4 Shape

The proposed shape descriptors are simple global 2D shape descriptors:

Perimeter: The perimeter $per(R)$ measures the length of the region contour.

Area: We measure the area $area(R)$ of region R as the number of pixels in the region.

Circularity: The circularity of region R

$$circ(R) = \frac{(per(R))^2}{4\pi area(R)} \quad (6.7)$$

describes how much the shape resembles a circle. When R is a circle its value is 1.

Aspect Ratio and Oriented Aspect Ratio: The aspect ratio of region R is

$$ar(R) = \frac{w(R)}{h(R)} \quad (6.8)$$

where $w(R)$ and $h(R)$ are, respectively, the width and height of the region bounding box. Similarly, the oriented aspect ratio is computed, taking into account the width and height of the oriented bounding box.

Compactness: This descriptor measures how compact a shape is, or how much it fills its bounding box

$$comp(R) = \frac{area(R)}{w(R) * h(R)} \quad (6.9)$$

6.2 Simplification of the search space

The first stage of the detector is a simplification stage, that aims to discard as many tree nodes as possible by a fast analysis of the regions associated with them. Each node passes through a cascade of very simple one-class classifiers. Each classifier is trained on one simple descriptor and produces a binary output. If a node is rejected by one classifier it is not analyzed by the next classifiers in the cascade.

Among the region descriptors proposed in 6.1 the ones that proved to be more useful for this first simplification are: color mean, aspect ratio, compactness, circularity and texture homogeneity. In a particular scenario, other descriptors like area, orientation, mass center or position can help to reduce even more the set of candidates if further knowledge about the faces in the scene is available.

Each of the proposed descriptors, if we work separately with the three color components, produces a one dimensional feature. The most straightforward method to build one binary classifier for each feature is to use density models. That means, for each feature, to estimate the density of the training data and to set a threshold on this density. Density models can be trained using a parametric approach (e.g. a Gaussian model or a mixture of Gaussian distributions) or finding the empirical distribution from the training data.

In our work, thresholds are found using the empirical target distributions since, as can be seen in Figure 6.2, the obtained densities do not conform very well to parametric models. Figures 6.2 show the normalized histograms of the proposed descriptors for a set of training face regions: mean color in Y, Cb and Cr components, aspect ratio, compactness, circularity and texture homogeneity using LH, HL and HH bands in two levels. We observe that for faces all distributions, except for color features, are unimodal and convex.

For the experiments, the training set consists of 156 images from the Banca French dataset (controlled sessions, group 1), 150 images from XM2VTS set and 150 images from the MPEG7 set. For each image, the search area of the BPT was generated, the smallest regions were discarded (size smaller than 200 pixels) and the remaining regions were manually annotated, giving a total number of 1320 target regions which correspond to complete or almost complete face representations.

The goal is to find the optimal threshold for each feature; a threshold that produces a given acceptance rate and minimizes the volume of the data description. For one dimensional distributions, that means to find the intervals or regions with a given coverage probability, while minimizing the size of the intervals. This criterion leads to regions called highest density regions or HDR's [74].

Let $f(x)$ be a density function of a random variable X . Then the $100(1 - \alpha)\%$ HDR is the subset $R(f_\alpha)$ of the sample space of X such that

$$R(f_\alpha) = \{x : f(x) \geq f_\alpha\} \quad (6.10)$$

where $R(f_\alpha)$ is the largest constant such that $Pr(X \in R(f_\alpha)) \geq 1 - \alpha$.

From the definition it follows that of all regions of probability coverage $1 - \alpha$, the HDR has the smallest possible volume in the sample space X .

For unimodal distributions, the goal is to find the interval $[c1, c2]$ with minimum length that covers the sample space with probability $1 - \alpha$. If the density has a single maximum, the interval has minimum length $c2 - c1$, if $f(c1) = f(c2)$. A simpler but suboptimal solution is found if we determine $c1$ and $c2$ such that they are the $\frac{\alpha}{2}$ and $1 - \frac{\alpha}{2}$ quantiles. This solution is optimum if $f(x)$ is symmetric about its mean because in that case $f(c1) = f(c2)$. For multimodal densities, the HDR usually consists of several disjoint subregions. A general algorithm to compute the HDR from any given bounded density (not necessarily unimodal) is proposed in [74].

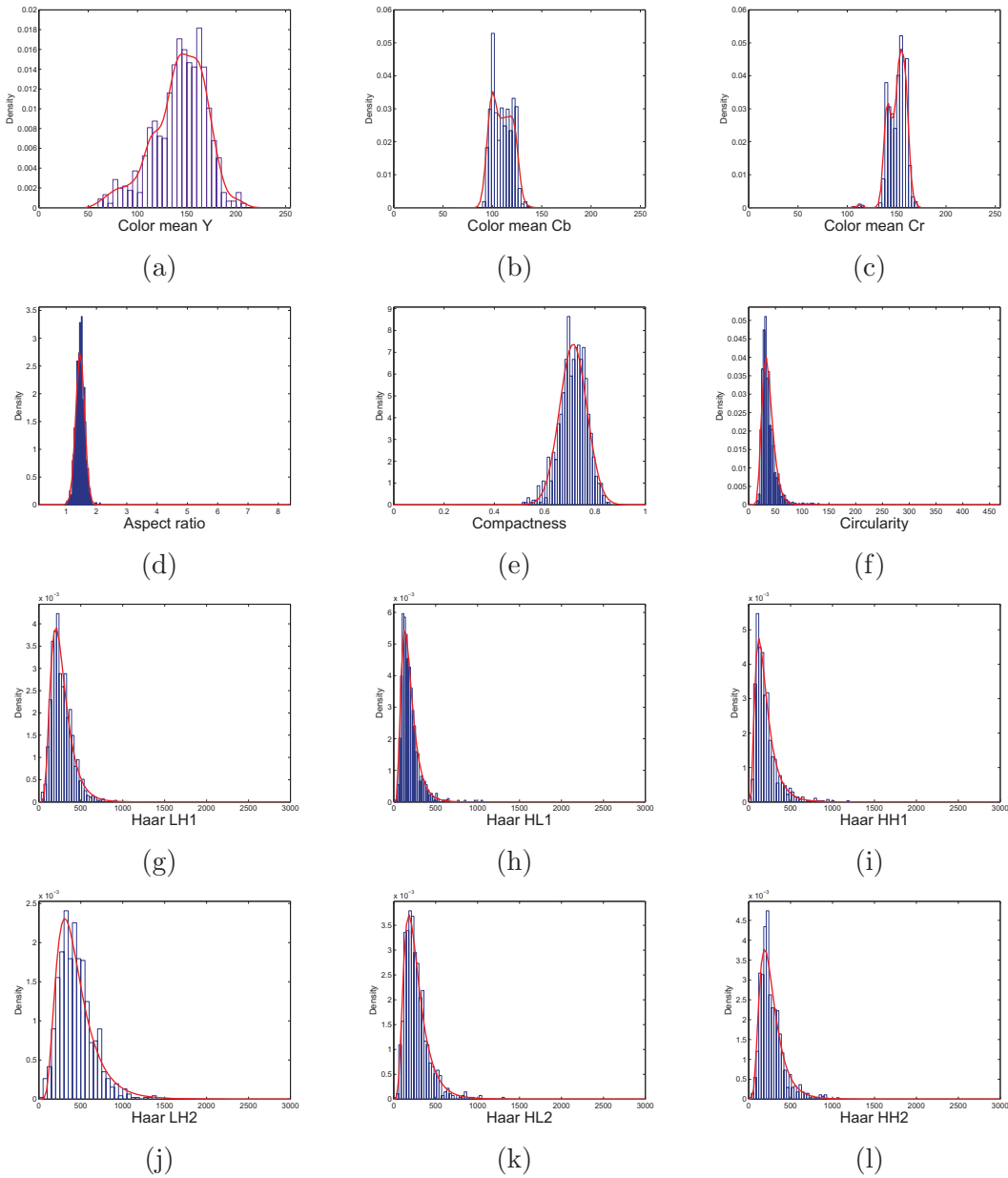


Figure 6.2: Distribution of basic descriptors for target regions (a) color Y, (b) color Cb, (c) color Cr, (d) aspect ratio, (e) compactness, (f) circularity, (g) texture LH1, (h) texture HL1, (i) texture HH1, (j) texture LH2, (k) texture HL2, (l) texture HH2.

We work with the suboptimal solution (quantiles) even for the the multimodal color descriptors, since, as will be shown, the obtained thresholds lead to very good simplification results. We choose $\alpha = 0.002$ and set as threshold the 0.001 and 0.999% quantiles. Table 6.1 shows the thresholds for the descriptors, as well as their estimated mean, variance, min and max values.

	Thr1	Thr2	Mean	Var	Min	Max
C. Mean Y	60	208	142.58	704.11	60	209
C. Mean Cb	92	136	109.42	98.17	92	137
C. Mean Cr	108	169	150.54	70.51	107	169
Aspect ratio	1.01	2.06	1.46	0.02	1	2.10
Compactness	0.52	0.85	0.71	0.002	0.52	0.85
Circularity	17.17	127	36.78	169.25	17.01	130.38
Haar LH1	46.10	858	272.89	14593	39.29	922.75
Haar HL1	20.50	1015.1	188.83	11033	7.21	1041
Haar HH1	24.1	1102	203.19	18963	23.64	1176.60
Haar LH2	53.3	1364	440.77	40669	47.45	1373.40
Haar HL2	25	1193	269.44	21979	5.90	1300.30
Haar HH2	29.5	1003.8	272.49	20420	21.81	1057.20

Table 6.1: Selected thresholds, mean, variance, min and max values of generic descriptors.

To test the classifiers, we work with another set of 156 images from the French Banca dataset, 50 images from XM2VTS and 110 from MPEG7. Again, for each image, the search area of the BPT was generated and the smallest regions were discarded. The remaining regions were manually annotated into target and outlier regions, leading to a total of 914 targets and 10197 outliers.

Within the simple classifiers we also consider the Area criterion, with bounds [150, 15000], to avoid the use of too small or too large regions. The results of the simplification are presented in Table 6.2. This table shows the percentage of BPT nodes rejected by the use of each basic descriptor, for the three datasets separately, and the global result in the last column. This percentage takes into account both outliers and targets rejected. The table also presents the number of false negatives caused by each type of filter. Note that a face may be represented by several nodes, with different degrees of precision. Many of these nodes may be discarded by the classifiers, but as long as one representative node remains active, the face can still be detected. Therefore, we count a false negative when all the nodes associated with a given face are rejected by the classifiers.

When we use all the classifiers the percentage of nodes discarded is very high: 96.01%. The use of generic descriptors largely reduces the amount of nodes to be further analyzed to




	XM2VTS		BANCA		MPEG7		ALL
							
	# nodes 3308		#nodes 19384		# nodes 15278		# nodes 37970
CLASSIFIER	% rej.	FN	% rej.	FN	% rej.	FN	% rej.
C. Mean (all)	53.57	-	43.94	-	37.66	1	42.25
Area	12.52	-	14.82	-	17.81	1	15.82
Aspect ratio	56.59	-	58.49	5	62.75	8	60.04
Compactness	73.58	2	65.80	13	65.44	8	66.33
Circularity	36.43	-	24.72	2	26.36	1	26.40
Haar LH1	53.87	-	41.89	-	39.51	1	41.97
Haar HL1	32.41	-	28.34	-	21.47	-	25.93
Haar HH1	41.20	-	35.53	-	29.85	-	33.74
Haar LH2	62.54	-	52.43	-	53.94	1	53.92
Haar HL2	43.77	-	42.15	-	39.25	-	41.13
Haar HH2	52.84	-	49.99	-	46.30	-	48.76
ALL FILTERS	93.29	2	96.09	20	96.49	22	96.01

Table 6.2: Simplification results.

only the 4% of the BPT nodes.

The criteria which produce the largest number of rejected nodes are **compactness** and **aspect ratio**. However, at the same time, they are responsible for a large number of false negatives (see Table 6.2). A deeper analysis shows that these misses occur in images where the complete face is not represented as a single node in the tree. There are several nodes that can be considered good face markers, and which, after the extension step, modify their area of support to represent the complete face. However, if these marker nodes are rejected by the compactness or aspect ratio criteria, the face is lost. The thresholds used in the experiment have been obtained using complete face representations. Therefore, one solution to reduce the false negative rate is to relax the upper and lower bounds of shape based classifiers (aspect ratio, compactness) so that incomplete but potentially good markers are not discarded. For example, if we repeat the estimation of thresholds for regions that, after

Classifier	CMean	Area	AR	Comp	Circ	LH1	HL1	HH1	LH2	HL2	HH2
% Nodes rejected	2.00	0.29	3.50	2.25	0.95	0.29	0.18	0.45	0.77	0.22	0.27

Table 6.3: Percentage of nodes rejected by only one classifier.

the extension, represent complete faces, the ranges change to $[0.9, 2.6]$ for aspect ratio criterion and $[0.4, 1]$ for compactness. Using these new thresholds, the overall percentage of rejected nodes reduces to 43.26 and 38.47, respectively, with no false negatives.

Something similar happens for the **circularity** descriptor, although in this case the number of false negatives is low (26.40% of rejected nodes with 3 false positives).

The **color mean** classifier also produces very good results for the three sets, removing 42.25% of the nodes with only 1 miss in MPEG7 images, a data set with large variability in skin color since images have been acquired under different illumination conditions.

The **area** criterion is the one that gives the lowest number of rejected nodes (16%) with one false negative, a very small face in MPEG7 set. However, it has to be mentioned that the bounds used for this experiment were very loose (a minimum size of 150 pixels and a maximum size of 15000 pixels) and they can surely be adapted to further reduce the search space if we can estimate the average face size in a particular context.

The **texture** criteria performed very well, particularly for the XM2VTS set. A detailed analysis of the nodes rejected by these descriptors shows that texture descriptors are very useful to discriminate between high, medium and low textured regions. Therefore, homogeneous, round skin colored regions, which appear frequently as BPT nodes and cannot be discarded using color or shape descriptors can be removed using the texture descriptors. In turn, highly textured areas can also be discarded.

Next, we analyze the diversity of the classifiers as we did in Section 5.4. However, at this stage the problem is different. First, because classifiers are trained to accept all targets; therefore, the useful information is now the outlier rejection rate (the performance depends only on this rate). Second, because classifiers have a binary output and are combined in a cascade. We can remove one classifier without decreasing the ensemble performance only if the classifier is redundant; if the reduced set rejects the same number of outliers that are rejected by the complete set. One possible strategy to remove redundant classifiers is the following. First, for each classifier, we count the number of outliers that are rejected only by the classifier. If one of this values is zero, we can remove the classifier without decreasing the performance of the ensemble. We repeat the process until no more classifiers can be eliminated.

Table 6.3 shows, the percentage of nodes rejected by only one of the classifiers. None of

the classifiers is completely redundant, the lowest value is 0.18 for HL1. We could remove this classifier and continue the analysis. However, the computational cost of this descriptor (and the other homogeneous texture descriptors) is negligible once the Haar transform has been computed. Moreover, the other region descriptors (size, color, etc.) are used in other stages of the detector. Therefore, in the following experiments we work with the full set of classifiers.

To show the relationship between classifiers, we compute the correlation coefficient for each pair. The results, presented in Table 6.4, show that the correlation is moderately high (values between 0.5 and 0.6) between HH2 and LH2 and HL2, and between similar bands in the two levels (LH1 and LH2; HL1 and HL2). For the other pairs, as expected, values are very low (lower than 0.06) between color, size, aspect ratio, compactness and circularity, and relatively low between these ones and the texture classifiers.

	CMean	Size	AR	Comp.	Circ	LH1	HL1	HH1	LH2	HL2	HH2
CMean	1	-0.041	0.059	0.033	0.018	0.247	0.218	0.213	0.260	0.203	0.220
Size	-0.041	1	0.150	-0.023	-0.025	-0.048	-0.029	-0.022	0.154	0.190	0.188
AR	0.059	0.150	1	-0.016	0.002	0.049	0.037	0.058	0.072	0.084	0.106
Comp.	0.033	-0.023	-0.016	1	0.255	0.024	0.003	-0.007	0.089	0.074	0.057
Circ.	0.018	-0.025	0.002	0.255	1	-0.013	-0.002	-0.003	0.020	0.027	0.010
Haar LH1	0.247	-0.048	0.049	0.024	-0.013	1	0.378	0.477	0.546	0.356	0.478
Haar HL1	0.218	-0.029	0.037	0.003	-0.002	0.378	1	0.527	0.281	0.536	0.441
Haar HH1	0.213	-0.022	0.058	-0.007	-0.003	0.477	0.527	1	0.310	0.374	0.519
Haar LH2	0.260	0.154	0.072	0.089	0.020	0.546	0.281	0.310	1	0.474	0.549
Haar HL2	0.203	0.190	0.084	0.074	0.027	0.356	0.536	0.374	0.474	1	0.592
Haar HH2	0.220	0.188	0.106	0.057	0.010	0.478	0.441	0.519	0.549	0.592	1

Table 6.4: Correlation coefficient between classifiers.

6.3 Accurate object representation

The search space of the BPT was created combining color and contour complexity criteria, with the purpose of getting nodes that represent the objects in the scene. However, since these criteria are generic, in some cases nodes are not complete representations of the objects.

Nodes can be good estimates of the objects, giving information about their scale (region size) and location (region position), but the accurate analysis of the image requires a precise object definition.

We propose to improve the object representation making use of the accuracy partition (see Section 4.2). The accuracy partition is finer than the search partition, that is, all the contours in the search partition are present in the accuracy partition as well. This characteristic facilitates the combination of information from both partitions.

Our approach, useful when an object has a known, rather rigid shape (like a human face), uses shape information to modify the area of support associated with each node. This process, that we call *node extension*, fits a reference shape model of the face to the node, and uses this shape to modify the area of support of the node, by adding or removing small regions from the set of regions that are initially associated with the node. The shape fitting is performed with a shape matching technique using distance transforms.

6.3.1 Shape matching based on templates

Shape matching based on templates deals with transforming a shape and measuring its resemblance with another shape, using some similarity measure [179]. Shape matching has numerous applications including object localization, image retrieval, model registration and tracking. In these applications, the shape can be represented by different features, typically by its contour points. A common approach to shape matching, then, relies on the detection of the most relevant contours in the image and the comparison with the contour of the prototype shape (the template). The goal is to search for the best matching while permitting the deformation of the template, so that a similarity measure or generalized distance between the matched features is minimized. To compute the matching we use distance transforms.

Let I be a binary image where zero-valued pixels represent the contour points of a node. A *distance transform* on I , DT_I is an image where each pixel value denotes the distance from the pixel to the nearest contour point in I [135] [34].

$$DT_I(\mathbf{x}) = \min\{d(\mathbf{x}, \mathbf{y}) | I(\mathbf{y}) = 0\} \quad (6.11)$$

Typically, $d(\mathbf{x}, \mathbf{y})$ is the Euclidean distance or a discrete approximation.

The template is transformed by a set of parametric transformations that describe how the shapes can be geometrically distorted in relation to one another. Matching proceeds by correlating the transformed template against the DT_I image.

The measure of similarity is the average distance to the nearest feature:

$$D(S, I) = \frac{1}{N_S} \sum_{\mathbf{x} \in S} DT_I(\mathbf{x}) \quad (6.12)$$

where S is the contour of the transformed template \bar{S} and N_S is the number of contour points in S . Other measures can be used, such as the median, the maximum or the mean square average [15].

The goal is to search for the best matching; that is, the parameters of the transformation that minimize the similarity measure $D(S, I)$. In our work, the allowed transformations for the reference shape are translation, rotation and scaling. For each node, the set of possible parameter values is bounded and quantized taking into account the node width, height and position in the image and an exhaustive search in the parameter space is performed.

The advantage of matching with distance transforms is their ability to handle noisy or imperfect data [149]. Areas with low gradient values in the original image which may lead to errors in definition of the contours (absence of some contours) have distance values related to the closest contour. Such distance values are usually low and, therefore, gaps in the contour definition are correctly filled.

6.3.2 Binary and chamfer distances

Two distance transforms are used in this work: chamfer distances [15] and binary distances. *Chamfer distance transforms* are a class of discrete algorithms that offer a good approximation to the Euclidean distance at a lower computational cost. They can also give integer valued distances that are more suitable for several image processing tasks. The term ‘chamfer’ originally referred to a sequential two-pass distance transform algorithm developed by Rosenfeld and Pfaltz [135] and later improved and generalized by Borgefors in [15]. The chamfer distance transform approximates the global distance computation with repeated propagation of local distances within a small neighborhood mask. The approximation error depends on the neighborhood size and the selection of the local distances. A commonly used distance is the chamfer 3-4 distance, where a 3x3 mask is used, and the local distances are 3 and 4 for horizontal/vertical and diagonal distances, respectively. *Binary distances* assign a distance value equal to 1 to a ring around the contours (in our work, typically, of 5 pixels width), whereas the remaining points in the space receive a distance value equal to 0.

In our experiments (see Appendix E), binary distances achieve better results when the reference shape is a good model of the objects being searched, allowing partial matchings when the node is an incomplete representation of the object. Chamfer distances perform better than binary distances when there is more variability between the shape model and the objects in the image. However, Chamfer distances may lead to local minima when the node is not a correct estimate of the complete object. In the case of face detection, since ellipses are good shape models, binary distances are used.

The shape matching process is illustrated in Figure 6.3 with an example of node extension for face detection. The reference contour which is used as model for the human face shape is an ellipse. The best representation of the face in the BPT is the node shown in Figure 6.3

(b). Note that there is a region in the node (see the initial partition in Figure 6.3 (c)) that spans part of the face and the hair, and there is a missing region (the right eye) in the face representation. In spite of that, the reference contour is correctly located around the face. Figure 6.3 (d) shows the result of the shape matching using a binary distance transform.

6.3.3 The extension

The next step in the node extension process is the redefinition of the support area of the node, which can be performed in several ways. Here we propose a simple, generic solution that uses only geometric information. We analyze the degree of overlap between the regions from the search partition and the fitted shape model \bar{S} . Regions in the search partition which are completely included in \bar{S} , or which are partially included but that do not extend too far from the shape \bar{S} (typically, if they extend less than 10% of the minor axis of the ellipse) are included in the extended node. Regions that do not overlap with \bar{S} are not included. Figure 6.3 (e) presents the set of regions from the search partition associated with the node in the previous example that are totally included in the shape \bar{S} .

For the remaining regions, that is, regions partially included but extending far from the shape \bar{S} , the analysis is performed in terms of the accuracy partition, to improve the precision of the representation. Fine partition regions that overlap with the shape are included in the extended node, while fine partition regions that do not overlap with the shape are removed. Figure 6.3 (f) presents the regions in the previous example that are analyzed using the accuracy partition information. In turn, Figure 6.3 (g) shows the set of regions from the accuracy partition that are included in the extended node and Figure 6.3 (h) shows the final area of support of the extended node.

6.3.4 Experiments

More examples of node extension are presented in Figure 6.4 using a binary distance and modeling the shape of a human face with an ellipse. In the first case, the shape model is correctly matched despite the fact that the area of support of the node spans the face and the neck. A missing region (the right eye) is included and a very common leakage in human face segmentation (the neck area) is removed. In the second case, the shape model helps to correctly define the region of support of the face, even though the view is not completely frontal (and, therefore, the ellipse model does not perfectly fit the object).

In order to assess the usefulness of the proposed extension stage, even when using a simple shape model, Table 6.5 presents the statistics of the asymmetric and symmetric distances, for the accuracy and search partitions, and for the original and extended BPT nodes, respectively (see Chapter 4) between the 116 human faces manually segmented from the MPEG-7 database subset and the selected (i) union of regions in the accuracy partition, (ii) union of regions in

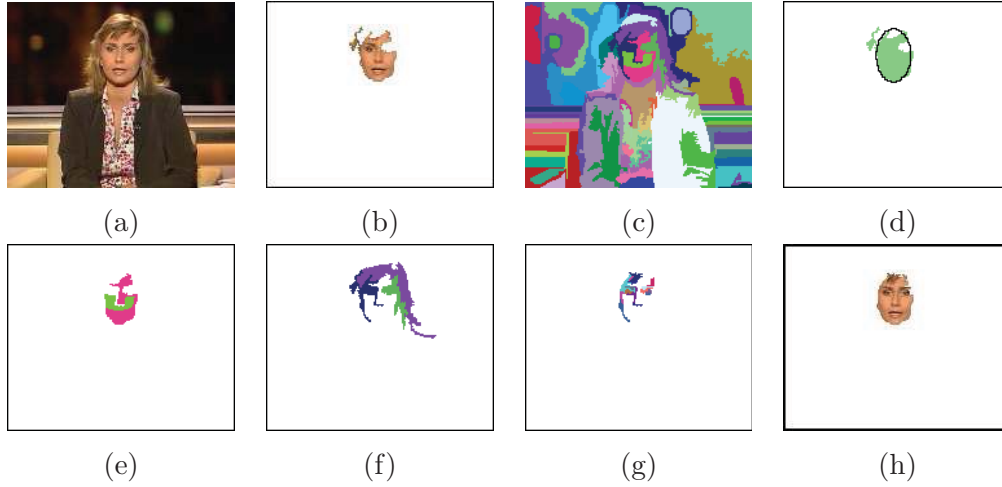


Figure 6.3: Node extension: (a) Original image, (b) Best representation of the face in the BPT, (c) Search partition, (d) Shape matching with an elliptical shape model, (e) Set of initial included regions from the search partition, (f) Set of analyzed regions from the search partition, (g) Set of included regions from the accuracy partition and (h) Final extended node.

the search partition, (iii) node in the search space of the BPT and (iv) extended node in the BPT. As it can be observed, in this database containing mainly images of high complexity, the set of nodes proposed in the search space of the BPT present lower quality than the best union of regions that can be selected from the search partition. The reason for this effect is mainly twofold: first, images in this database present illumination problems and, second, human faces are complex objects (non completely homogeneous in color). Therefore, some nodes in the BPT are not perfect markers of the faces in the scene.

The usefulness of the extension of nodes using shape matching with distance transforms for objects with different shapes is illustrated in Appendix E.

	Asymmetric Distance		Symmetric Distance	
	Accuracy Part.	Search Part.	BPT Node	Ext. Node
Mean	5.99	16.48	28.18	20.08
σ^2	0.07	0.89	3.05	1.52

Table 6.5: Asymmetric distance for the selected union of regions from the accuracy partition and from the search partition as well as symmetric distance for the selected BPT node and for the selected extended node with respect to the MPEG-7 subset. Mean and variance values are multiplied by 10^2 .

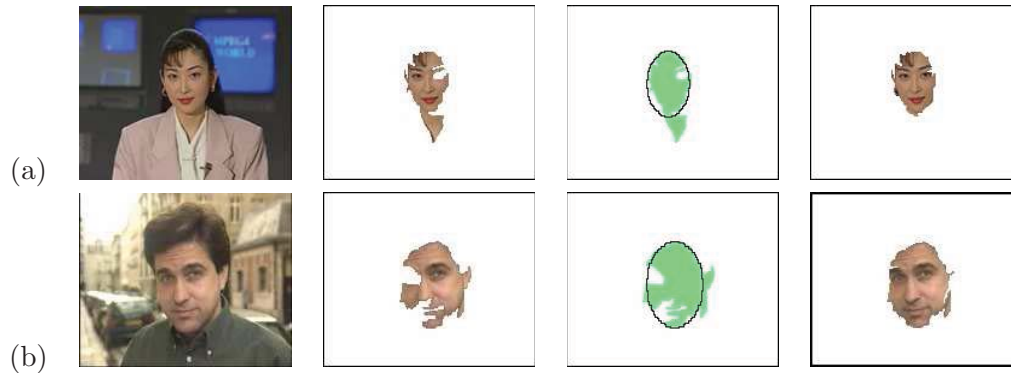


Figure 6.4: Improvement in the representation of faces achieved by the extension of the nodes. First column: original images. Second column: object nodes. Third column: shape fitting. Fourth column: extended nodes.

6.4 Best nodes selection

The face detector classifies a given region as face if its likelihood is above a predefined threshold. However, when running the detector on the BPT of an image, several overlapping nodes may represent the same face. Therefore, more than one node (those with likelihood larger than the threshold) can be selected for the same face. Further analysis is needed to find the best node for each face.

We need to find the nodes that represent the same face and to decide which, among these overlapping nodes, is the node that best represents each face. To solve these problems we make use of the tree structure. Note that nodes have been extended. The extension stage adds or removes small regions from the original tree nodes. Therefore, the relationship between an extended node and its ascendant or descendant nodes in the BPT is no longer a strict inclusion relationship. However, we can still consider the inclusion in a loose way, since the overlap between an extended node and the extension of its descendant/ascendant nodes is still very similar to that of the original nodes.

Let N be the set of extended nodes with likelihood greater than a threshold T_{face} . The procedure to select one node for each face is the following:

- 1: Find the node in N with maximum likelihood n_{max} .
- 2: Remove n_{max} from N .
- 3: Remove from N all the tree nodes that represent the same face as n_{max} . This is performed in two stages: first, all the ascendant nodes of n_{max} are removed from N , since their corresponding regions are included in the region associated with n_{max} . In the same way, all the descendant nodes of n_{max} are removed from N , because their regions contain the region associated with n_{max} .
- 4: Repeat steps 1 to 3 until N is empty.

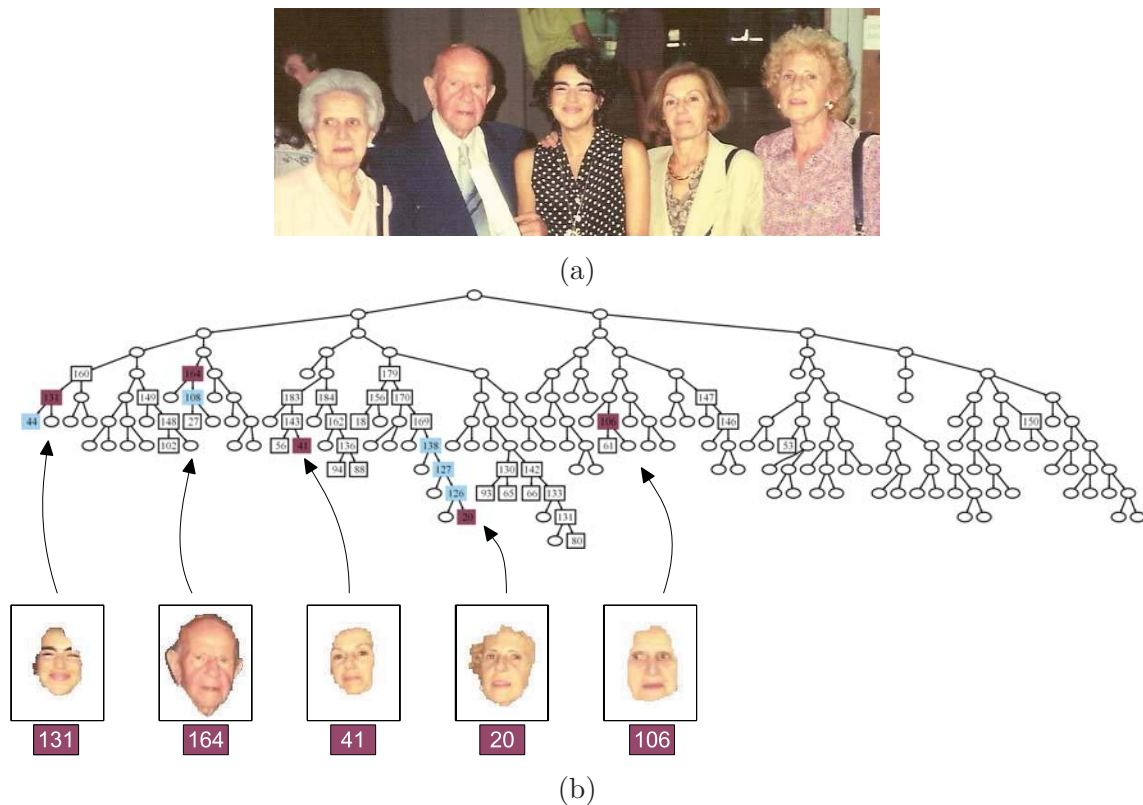


Figure 6.5: Original image (a) and its BPT with the result of the final decision and segmented faces (b).

Figure 6.5 illustrates the decision stage for an image (a) with five faces. Uncolored, circular nodes in the BPT (b) represent nodes that were rejected during the simplification stage. Uncolored, squared nodes represent extended nodes with low values of likelihood (with values lower than the threshold T_{face}). Colored, squared nodes, are nodes with high values of likelihood that were included in the N set. Within this group, the red ones represent the nodes with highest likelihoods.

6.5 Evaluation

In this section we evaluate the complete system in terms of false alarms, missed detection rates, precision, recall and segmentation accuracy. Experiments are presented for different test sets, from simple ones like XM2VTS and Banca controlled to challenging ones such as Banca adverse and MPEG7 sets. We also analyze the performance of the gray-level version of the detector with the BioId face set, and show the results obtained when using the detector as a first stage in a face recognition application.

For each dataset we present the Receiving Operating Characteristic curve, the Precision-Recall curve, two measures of segmentation accuracy, some examples showing successful detections and examples that illustrate problematic cases.

The ROC curves typically plot the rate of false positives (x -axis) versus the rate of true positives (y -axis). The false positive rate is measured with respect to the total number of patterns analyzed. To make this information more explicit, we prefer to plot the number of false positive patterns instead of the false positive rate.

Precision-Recall (PR) curves plot *precision*, a measure of exactness or fidelity, versus *recall*, a measure of completeness:

$$Precision = \frac{\#TruePositives}{\#TruePositives + \#FalsePositives} \quad (6.13)$$

$$Recall = \frac{\#TruePositives}{\#TruePositives + \#FalseNegatives} \quad (6.14)$$

To evaluate the results, first we need to define what a successful or correct detection is.

We use the spatial overlap information between a ground truth region R_{GT} and a region output by the system R_D to compute the *overlap ratio*:

$$OR(R_D) = \frac{|R_D \cap R_{GT}|}{|R_D \cup R_{GT}|} \quad (6.15)$$

The overlap ratio is an accuracy metric which produces a real number value between zero (worst possible performance) and one (best possible performance).

An output region is considered a correct detection (a true positive) if the overlap ratio between the region and the corresponding ground truth face is larger than 0.40. In all the experiments performed to calculate the ROC and PR curves, the overlap ratio is estimated by visual inspection, by comparing each detected region with the facial region (the ground truth). For each face in the image, at most one output region can be a correct detection; any remaining region counts as false positive. A ground truth face for which there is not an output region with overlap ratio that exceeds 0.40 counts as missing or false negative.

We also evaluate the overall segmentation accuracy numerically. We have manually segmented the facial areas for a subset of images selected from each dataset. For these subsets, we numerically calculate the accuracy of each detection using Eq. 6.5.

The global accuracy is measured as the average of the overlap ratio, over all the N_{mapped} mapped faces in the set (output regions for which there is a corresponding ground truth region). The values are presented in Table 6.7.

$$AvOR = \frac{1}{N_{mapped}} \sum_{i=1}^{N_{mapped}} \frac{|R_{Di} \cap R_{GTi}|}{|R_{Di} \cup R_{GTi}|} \quad (6.16)$$

We also measure the accuracy in terms of the symmetric distance used in Chapter 4 to compare partitions. Here, the partitions have two regions (face and background) and the symmetric distance between them can be computed with the following expression:

$$SD(R_D) = \frac{|R_D \cup R_{GT}| - |R_D \cap R_{GT}|}{|R_{GT}|} \quad (6.17)$$

This distance is averaged over all the mapped faces. The values for all the datasets are also shown in Table 6.7.

We compute the ROC and PR curves using the four combinations of normalization and fusion strategies proposed in Sec. 5.4: unnormalized data combined with logistic regression (*LogReg*), normalization with the double sigmoid function and simple average (*AveBisig*), and robust min-max normalization with the two weighted average fusion methods (*W1Rminmax* and *W2Rminmax*).

In all cases the BPTs are created with the same criteria: WEDM merging criterion to create the accuracy and the search partition (see Eq. 4.5), a fixed number of regions (500) in the accuracy partition and the AMC stopping criterion (see Eq. 4.3.4, with $T_{AMC} = 0.12$) for the search partition. The merging criterion used to build the BPT is NWMC (see Eq. 4.8), the criterion that takes into account both normalized color and contour complexity.

The face model is almost the same in all the experiments (the model obtained in Chapter 5). The only classifiers that are trained specifically for some of the datasets are the color classifiers (mean and dominant colors) since, as was shown in Section 5.1, skin color distribution depends strongly on the camera characteristics and on the scene illumination. There are also variations in the size thresholds used in the simplification stage in some of the sets.

Finally, we compare the detection performance of our system with the OpenCV implementation of a cascade Haar-based face detector. This system is an improvement [95] of the system initially proposed by Viola and Jones in [185]. In this detector, candidate rectangles, that is, rectangles that pass the cascade, are grouped and accepted as faces if there is a large enough number of overlapping candidates. It also uses some heuristics to reduce the number of candidates: it applies a Canny edge detector and rejects patterns that contain too few or too much edges. The number of neighbor rectangles that make up an object is the only parameter that can be used to vary the performance of the detector. Therefore, we cannot compute a complete ROC for this system, but only obtain a few operation points. The scale factor by which the search window is scaled between subsequent scans is set to 1.1. The minimum window size is 20x20.

6.5.1 XM2VTS

The XM2VTS test set consists of 1180 images. This set is relatively simple for detection, since it contains only frontal head and shoulder images with a uniform background.

The performance for this set is very good for all the fusion criteria, with a detection rate of 96% for LogReg and W2Rminmax fusion rules, and 94% for AveBisig and W1Rminmax, without any false positive. The detection rates for LogReg and W2Rminmax grow to 99.6% accepting only four false positives, which means a false positive rate of 5.10^{-5} , since the total number of regions analyzed is 80.622 (the BPT nodes). The same detection rate (99.6%) is reached by the Viola and Jones based detector, with 29 false positives. For a detection rate of 99.4%, we obtain 5 false positives. The average number of regions of the search partitions in this set is 68. The ROC and PR curves are presented in Figure 6.6.

Figure 6.12 shows some results, where faces with different skin color and hairstyles, with or without glasses and beards are correctly segmented. The last row presents two examples of segmentations errors. In the left image, the chin region is missing while in the other example the selected node contains part of the hair around the face. The average segmentation accuracy for this set is 0.86.

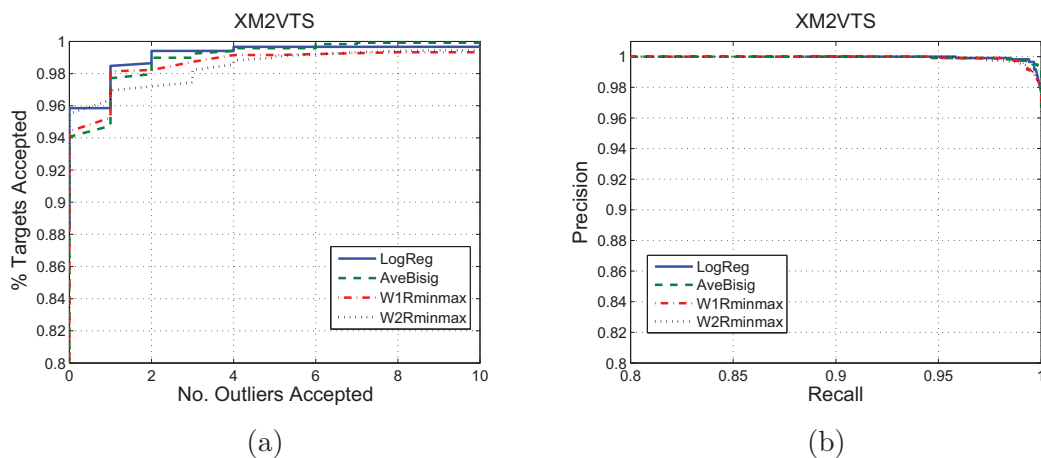


Figure 6.6: ROC and Precision-Recall curves for different combinations of normalization and fusion rules, XM2VTS set.

6.5.2 MPEG7

The MPEG set consists of 1020 faces in 980 images. The images are much more complex than those in the previous set, with large variability in skin color, face pose, background and illumination conditions. ROC and PR curves are presented in Figure 6.7. The performance is similar for the four fusion rules, with slightly better results for AveBisig and W2Rminmax. Here, we reach a detection rate of 95.4% with 100 false positives, where the total number of regions is 160.000, with an average of 133 regions in each search partition. The Viola and Jones based detector has a slightly lower performance. It achieves a maximum detection rate of 93.4%, with 98 false positives, and 88.2% with 15 false positives.

Some segmentation examples are presented in Figure 6.13. The last two rows show some errors in the segmented faces, similar to the problems observed for the other datasets: missing regions or parts of the background or the hair that are merged with the face region. The last row shows two images of the same person; the face on the left is correctly segmented, but for the other image, the selected region includes a bald patch.

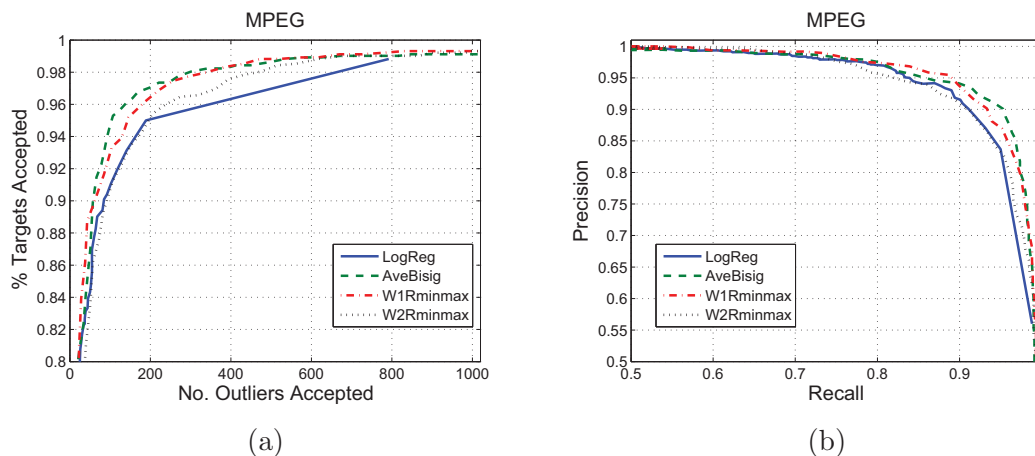


Figure 6.7: ROC and Precision-Recall curves for different combinations of normalization and fusion rules, MPEG set.

6.5.3 BANCA

The BANCA subset used for testing consists of 520 images in each of three different scenarios (controlled, degraded and adverse) selected from the French subset (sessions 3 and 4, group g2).

A high quality digital camera and uniform background and illumination are used in the *controlled scenario*, where the results are very good for all the fusion strategies. The W2Rminmax criterion outperforms the others for very low false positive rates: 98% without false positives, and 99,5% with only 7 false positives. In this set, the total number of nodes for the 520 images is 48.272 and the average number of regions in the search partitions is 93. The ROC and PR curves are presented in Figure 6.8 and examples of the results are shown in Figure 6.15. The last row shows some difficult examples: on the left, the best node is an incomplete representation of the face, since the beard region is merged with the hair region instead of merging with the rest of the face; on the right, the best node is an overcomplete representation, since the neck region and the lower part of the face are merged in an early stage during the BPT creation.

The second scenario is the *degraded scenario*, where a cheap analogue web cam was used to capture the images. Here, the detection performance significantly decreases, as shown by

the curves in Figure 6.9. The main problem is the side illumination, which causes that the partition contours do not correspond with the face contours. In many cases, the face is not represented by any node in the BPT, and the extension stage cannot fill the missing parts.

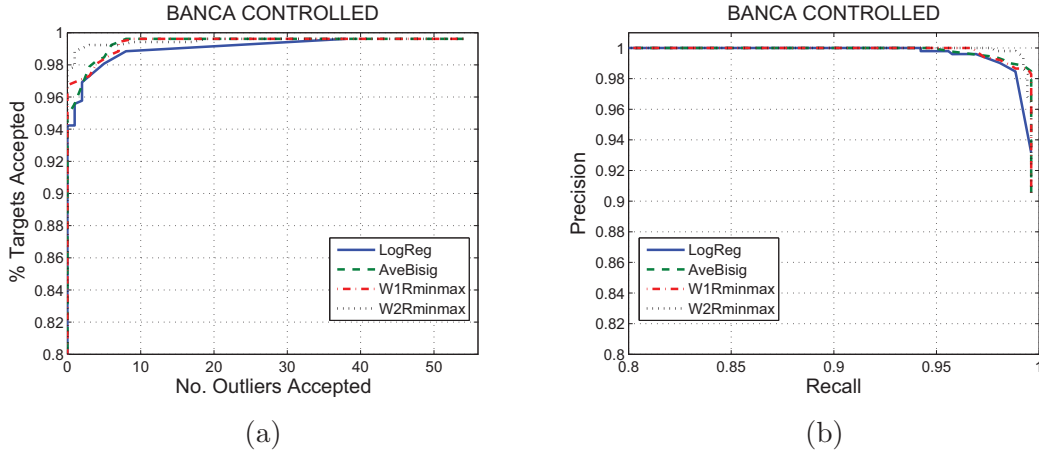


Figure 6.8: ROC and Precision-Recall curves for different combinations of normalization and fusion rules, BANCA controlled set.

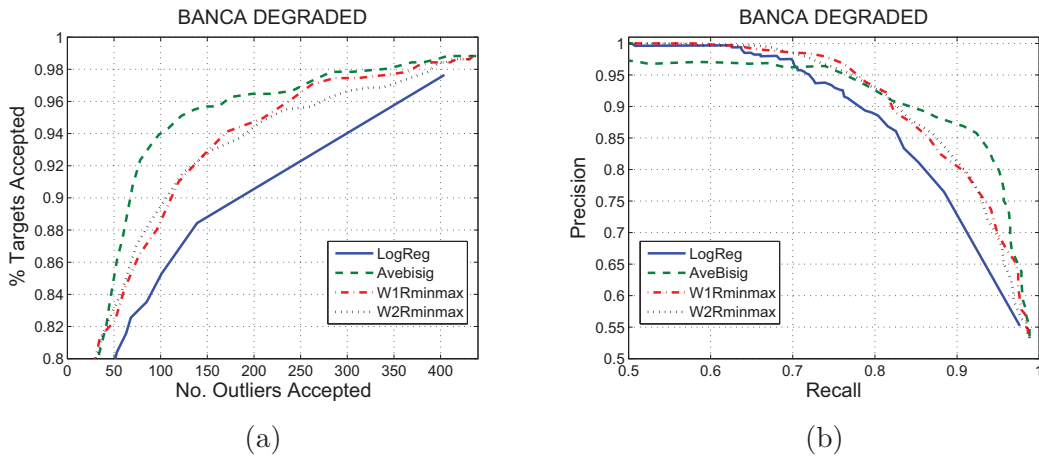


Figure 6.9: ROC and Precision-Recall curves for different combinations of normalization and fusion rules, BANCA degraded set.

The best results are for the AveBisig method, with a target acceptance rate of 94% with 100 false positives out of 51858 nodes. The average number of regions in the search partition is 100 (larger than the number of regions in the controlled scenario, since images are much more complex). The ROC and PR curves in Figure 6.9 show that the LogReg criterion performs much worse than the other criteria. LogReg normalization tends to increase the separation

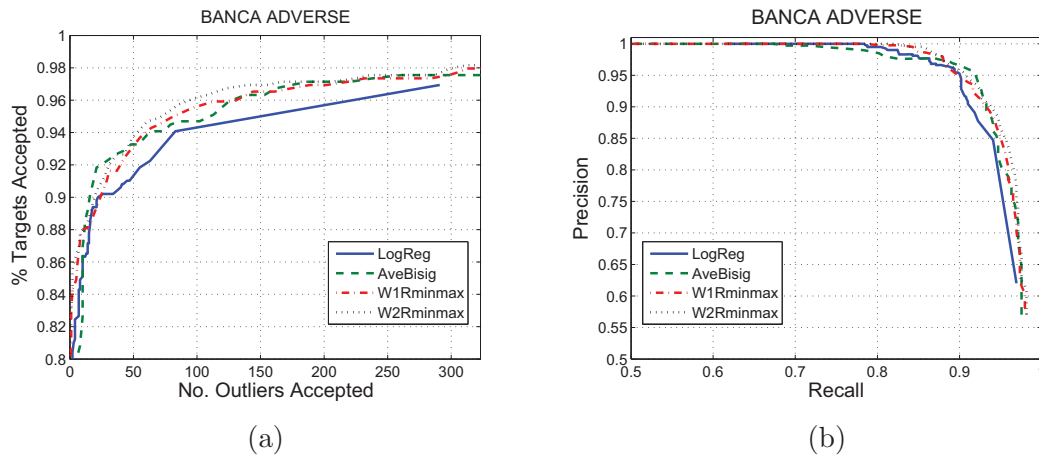


Figure 6.10: ROC and Precision-Recall curves for different combinations of normalization and fusion rules, BANCA adverse set.

between target and outlier classes, and works very well in controlled situations, when target regions are complete face representations and have high likelihoods. But it has an unstable behavior. A region that represents a complete face gets a very high normalized likelihood value, but an incomplete representation (even if a small part is missing) leads to a normalized value near to zero.

Some examples of correct and wrong segmentations are shown in Figure 6.16, where we observe that errors are mainly due to the side lighting or to the color similarity between the face and the back wall.

Finally, in the *adverse scenario*, a high quality camera was also used, but images have a cluttered background and changing illumination.

The results for this set are better than those obtained for the degraded scenario (see Figure 6.10): for a detection rate of 94% the number of false positives is 50, out of 91.622 regions analyzed. The average number of regions in the search partitions is 176. In this case, the performance of the four strategies is similar, with slightly worse results for the LogReg criterion. Some segmentation results are shown in Figure 6.17.

In the controlled scenario, our detector and the Viola and Jones based system have a similar performance. The Viola and Jones based system has a detection rate of 98.5% without false positives, and 99.42% with only 5 false positives. However the Viola and Jones based detector outperforms our system in terms of detection rate in the last two scenarios. For many images, due to the extreme lighting conditions, contour partitions do not correspond to the face contours and faces are not well represented in the tree nodes. However, in the block-based approach faces are correctly found by the sliding window at the correct position and scale. The detection rates are 94% with 15 false positives for the degraded set, and 97%

with 18 false positives for the adverse set.

6.5.4 BioID

The BioID test set contains 1000 gray-scale images. For this set, the criteria used to build the tree are slightly different than those used for the color images. The accuracy partition has 500 regions and is obtained with the WEDM criterion (using only luminance), but the merging criterion used to build the search partition uses both luminance and contour information. In the absence of color information, the best search partitions are obtained when we introduce also the contour criterion at this stage, to avoid the merging of part of the face with the background.

Since the face model does not use color classifiers, only 5 classifiers are combined to produce the final face likelihood. The best ROC curve is obtained for the W1Rminmax rule, with a detection rate of 90% with 250 false positives (3.10^{-3}), and 96% with 800 false positives, while the total number of nodes is 79000, and the average number of regions in each search partition is 79. ROC and PR curves and some results are shown in Figures 6.11 and 6.14, respectively. The performance is worse than in previous scenarios, mainly because several images cannot be correctly segmented. Here, the Viola and Jones based detector outperforms our system, with a detection rate of 95% with 35 false positives.

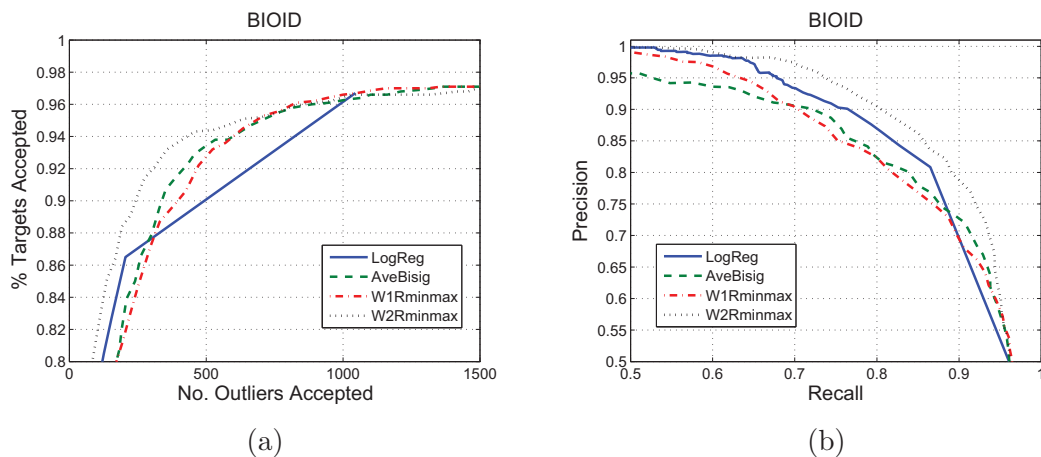


Figure 6.11: ROC and Precision-Recall curves for different combinations of normalization and fusion rules, BioID set.

6.5.5 Summary of results

To summarize the results, Table 6.6 shows the false acceptance rates obtained for different values of target acceptance rates, for the six data sets, using in each case, the normalization

and fusion strategies that produced the best results (the larger area under the ROC curve).

Table 6.7 shows the segmentation accuracy in terms of the average overlap ratio and the symmetric distance between the segmented face and the ground truth face. The table shows the mean and variance of these distances computed for a set of manually segmented faces in each dataset.

Note that the segmentation accuracy for the Banca adverse set is quite large even when the system performance in terms of detection rate is not good. Here, the extension of nodes is an important stage that obtains a complete representation of faces that are not represented as single nodes in the tree.

DataSet	Best Fusion	# Faces	# Nodes	TAR			
				FAR 10^{-3}	FAR 10^{-4}	FAR 10^{-5}	FAR 10^{-6}
XM2VTS	LogReg	1180	80622	100.00	99.66	95.85	95.85
MPEG-7	AveBisig	1020	160000	96.37	76.67	42.35	39.51
BioId	W2Rminmax	1000	79000	78.90	58.90	31.00	31.00
Banca Cont.	W2Rminmax	520	48272	99.62	99.23	97.69	97.69
Banca Deg.	AveBisig	520	51858	83.33	45.29	34.51	34.51
Banca Adv.	W2Rminmax	520	91622	95.10	92.93	87.14	81.43

Table 6.6: Performance of the best combination rules with different normalization criteria.

DataSet	# GT Faces	AvOR		Symm. Dist	
		Mean	Var	Mean	Var
XM2VTS	370	0.8622	0.0085	0.1503	0.0225
MPEG-7	100	0.7906	0.0139	0.2512	0.0315
BioId	100	0.8354	0.0106	0.1869	0.0180
Banca Cont.	100	0.8508	0.0083	0.1646	0.0151
Banca Deg.	100	0.8281	0.0116	0.2023	0.0228
Banca Adv.	100	0.8693	0.0087	0.1375	0.0100

Table 6.7: Segmentation accuracy.

6.5.6 Application to face recognition: i3Media

In this section we show the results obtained when the detector is used as a first step in a face recognition application. Our face detector is being used in the i3media project [75], where one of the tasks is the recognition of the news presenter on a set of key-frames captured from TV news videos.

Here, the test set is formed by 800 images, of which 708 are images of a TV studio where there is a news presenter, and 92 images do not contain any face. For the studio images, we can assume that there is only one person in each scene, which is located approximately in

the middle part of the image. Therefore, we set the system parameters so that the number of output faces is fixed to one, and the node with largest likelihood is output, only if its likelihood is above a predefined acceptance threshold. This threshold is set so that there are very few -or not- miss-detections, at the expense, if necessary, of more false alarms, because these false alarms can be eliminated by the recognition step. Following the second assumption (person in the middle of the scene), we add a position classifier (see Section 6.1 to the cascade, to increase the number of outliers rejected in the simplification stage.

The results obtained for the four combinations of normalization and fusion strategies are very similar, with slightly better results (in terms of segmentation accuracy and false positive rate) for the W2Rminmax. Results are summarized in Tab. 6.8, where we show the number of true and false positives, false negatives, and the detection rate. In this set, the Viola and Jones based detector reaches the same detection rate, 99.58%, with 32 false positives.

# Images	# Faces	True pos.	False neg.	False pos.	Det. rate
800	708	705	2	6	99.58%

Table 6.8: Results for i3media set.

6.6 Summary

This chapter has completed the study of the proposed face detector, focusing on three main aspects: the initial classifier, the extension of the nodes and the exhaustive evaluation of the complete face detector.

First, we have proposed a set of generic descriptors related to low-level features, which are used to obtain a simple description of the face class. Descriptors are associated with simple threshold classifiers which are combined in a cascade. We have shown that, regardless their simplicity, they are very useful in the initial simplification of the search space. Their usefulness in describing in a simple manner other semantic objects will be illustrated in Chapter 8.

Next, the concept of node extension has been proposed in order to improve the representation of the objects, by creating regions that are not originally present in the BPT. The area of support of these new regions is defined using available shape information of the object, ellipses in the case of faces. In Chapter 8 and Appendix E other examples of node extension are proposed, using both well defined shapes (e.g. triangles for traffic sign detection) or geometric information when objects do not have a characteristic shape (e.g. sky detection).

Finally, the complete system has been evaluated for XM2VTS, BANCA, BioID, MPEG7 and i3media datasets.

Some of the results presented in this chapter appear in the following papers:

- V. Vilaplana, F. Marqués, P. Salembier, *Binary Partition Trees for Object Detection*, IEEE Transactions on Image Processing, vol. 17, n.11, pp. 2201-2216, 2008. ISSN: 1057-7149.
- V. Vilaplana, F. Marqués, *Face detection and segmentation on a hierarchical image representation*, Proceedings of EUSIPCO 2007, 15th European Signal Processing Conference, pp. 1955-1959, Poznan, Polonia, September 2007.
- O. Salerno, M. Pardàs, V. Vilaplana, F. Marqués, *Object recognition based on Binary Partition Trees*, Proceedings of ICIP 2004, IEEE International Conference on Image Processing, Singapore, 2004

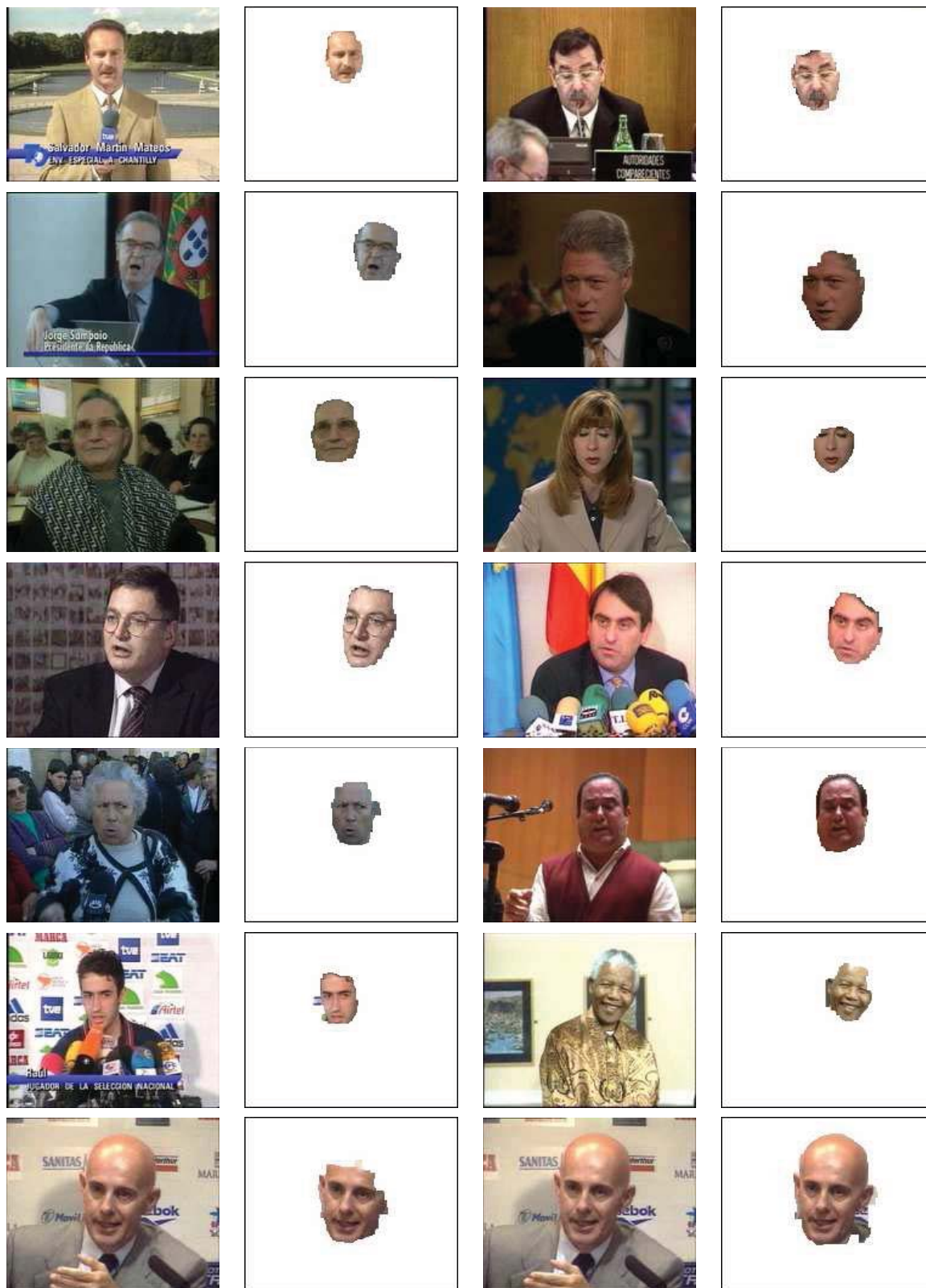


Figure 6.13: Results for MPEG. Correct segmentations and some examples of segmentation errors.

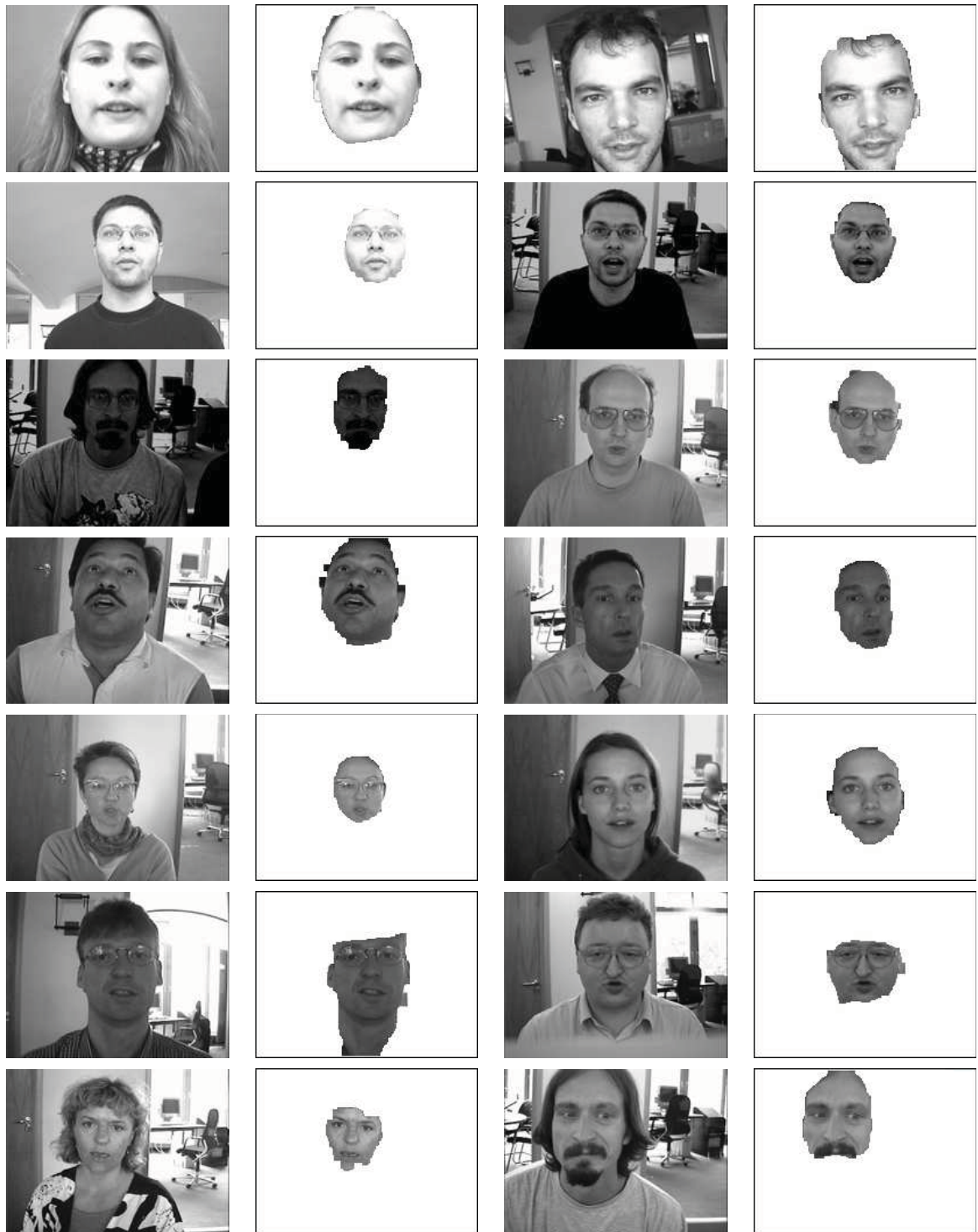


Figure 6.14: Results for BioID. Correct segmentations and two examples of segmentation errors (last row).



Figure 6.15: Results for Banca Controlled. Correct segmentations and some examples of segmentation errors.



Figure 6.16: Results for Banca Degraded. Correct segmentations and some examples of segmentation errors.

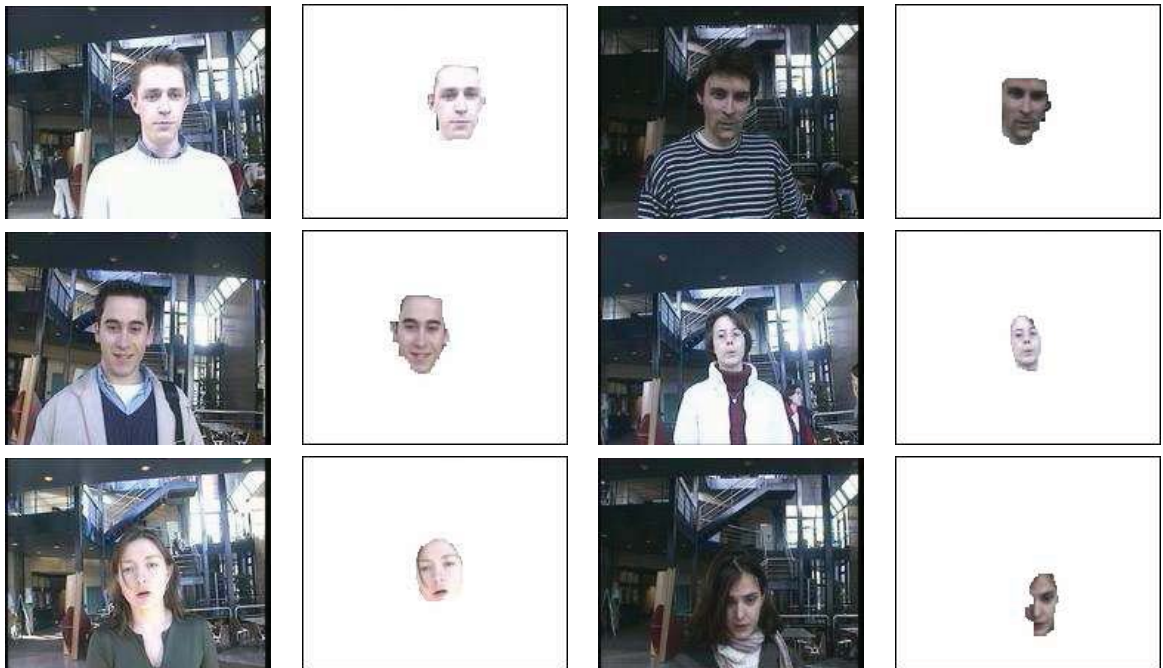


Figure 6.17: Results for Banca Adverse. Correct segmentations and some examples of segmentation errors.

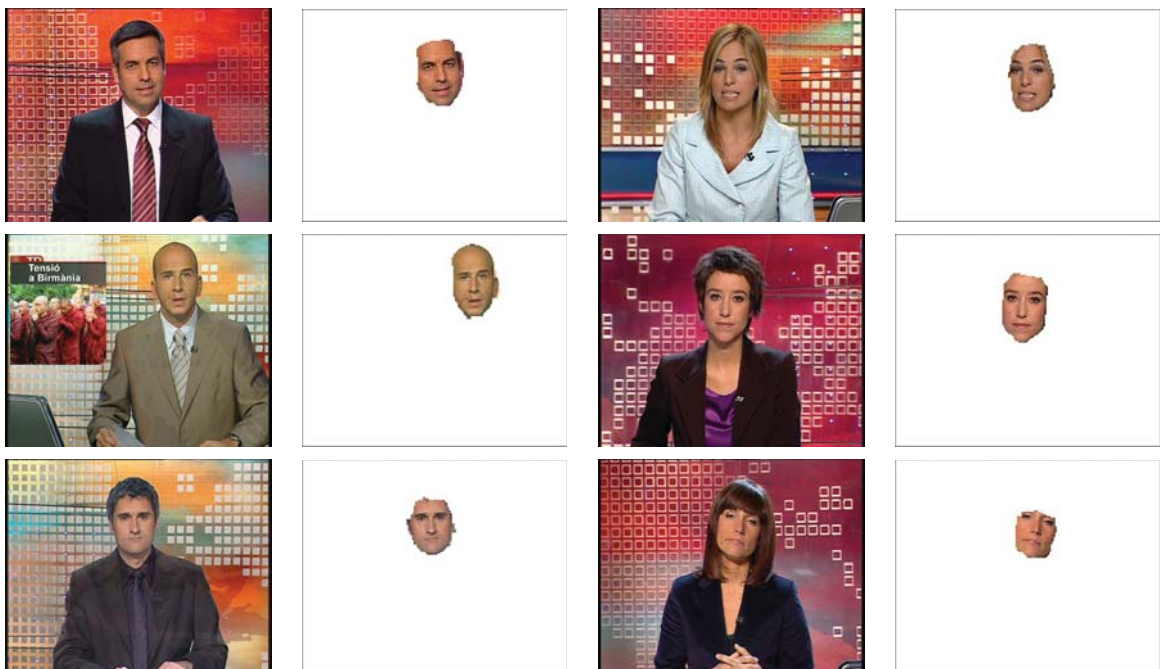


Figure 6.18: Results for i3media set.

Chapter 7

Face tracking

In this chapter we turn our attention to the problem of face tracking. Two different strategies are proposed. The first one is a unified detection and tracking mechanism based on the use of the Binary Partition Tree. The need to reduce the computational cost of the tracker and, at the same time, to obtain segmented faces motivates the second approach. Here, the tracking is performed with the mean shift algorithm, and the segmentation is achieved by the use of a region-based image model. This second system is compared with another two face trackers based on mean shift, Camshift and ABCShift, on a various challenging sequences.

7.1 Object tracking

Face tracking plays an important role in many applications such as video indexing, visual surveillance, human-computer interaction, or facial expression recognition. In these applications it is necessary to detect the faces, track them from frame to frame and analyze the tracks, for instance, to recognize or understand their behavior.

There are three main approaches to the problem [180]: frame based detection, where the face is detected in each frame without taking into account the temporal information; separated detection and tracking, in which the face is detected in the first frame and then is tracked in the following frames; and integrated approaches, where temporal relationships between frames are used to detect the face through the sequence. While the first approach can rely on any of the face detection strategies discussed in Section 3.2, the other two approaches typically apply techniques specifically developed for the more general problem of object tracking. In the following we will concentrate on the last two approaches, presenting a very brief summary of the different groups of techniques used in the area of object tracking with references to the most significant works on face tracking within each group. The reader is referred to [197] for a comprehensive survey of the problem.

In its simplest form, an object tracker generates the trajectory of an object over the time

by locating its position in every frame of a video sequence [197]. While this information may be sufficient for some applications (e.g. detecting the presence of an intruder), in other applications the tracker should provide additional data, since it may be necessary to know the orientation, extension or the precise shape of the object at every frame (e.g. facial expression recognition).

Object tracking is a complex problem due to several factors: the loss of information caused by the projection from a 3D real scene to a 2D image, image noise, the nonrigid or articulated nature of objects, complex object motion, partial or full occlusions, illumination changes and real time processing requirements. A large amount of research has been performed in the area, and well-performing trackers that work in real time have been developed. Their usefulness in realistic scenarios is, however, limited since they often impose constraints on the motion, amount of occlusion or number or size of objects or object appearance. Object tracking in unconstrained scenarios is still an open problem for which new solutions are continuously being proposed. The different solutions differ mainly on the type of representation used for the objects, which, in turn, depends on the end use of the tracking information.

In a recent survey Yilmaz et al [197] organize tracking methods (for both separated detection and tracking and integrated approaches) in three main groups, taking into account the model selected to represent the object shape:

Point tracking: Objects are represented by points (e.g. the object centroid or a set of characteristic points). The tracker establishes a correspondence between points in consecutive frames. Point correspondence methods are grouped in two main categories: deterministic methods and statistical methods. In deterministic methods, a correspondence cost of associating points in frame $t-1$ and t is defined taking into account motion, proximity or velocity constraints, and the cost is minimized, formulating the problem as a combinatorial optimization problem. Statistical methods, in turn, take into account measurement and model uncertainties, and use a state space approach to model object properties like position, velocity or acceleration. Kalman filters, particle filters [5] and joint probabilistic data association filters [132] belong to this class. Face tracking systems based on this type of trackers are [7] (deterministic) and [60, 115, 204, 180] (statistical).

Kernel tracking: The term kernel here refers to the object shape and appearance (e.g. a rectangular or elliptical template with an appearance model [44] inside the shape). The objects are tracked by estimating the motion, typically a parametric transformation, of the kernel in consecutive frames. The object motion is estimated by maximizing the object appearance similarity between the previous and the current frame. The estimation process can be in the form of a brute force search (e.g. template matching) or by using a gradient ascent-based maximization process (e.g. mean shift tracking). Some face trackers based on these strategies are proposed in [20, 32, 38].

Silhouette tracking: These methods are employed in applications that require to define with precision the area covered by the object. The object is represented by a region (an image mask) or by contours (a set of control points or level sets, for instance). The tracking is performed by estimating the area of support of the object in each frame, using for instance shape matching techniques for the case of region representations and methods that evolve an initial contour in the previous frame to its new position in the current frame, for contour representations. Face tracking systems in this group are [121, 100].

In the next sections we describe two different strategies that we have developed for face tracking. The first one belongs to the third group of techniques, silhouette tracking methods, and its structure is an extension of the face detection system presented in previous chapters. Detections, however, are not performed independently in each frame; detection and tracking are integrated in a unified framework.

This approach is appealing because it is based on the same concepts used for the intra-detection: the image represented with a BPT, a face model which is updated to incorporate specific characteristics of the face being tracked, decision based on the same sets of one-class classifiers, etc.

However, it has a major weakness: its computational complexity. Applications that require face tracking usually need faster techniques (though not necessarily real-time).

The need to reduce the computational cost and, at the same time, to obtain a segmentation of the faces motivates our second approach, where we still work with regions (now from an accuracy partition) but the tracking is performed with a fast kernel method. It is based on the mean shift algorithm and on the segmentation of the images into small regions which are homogeneous in color.

We assume that faces have been detected in the first frame and then are tracked through the sequence. To detect and track new incoming faces, an intra frame detection stage has to be performed after a certain number of tracking iterations.

For the sake of completeness, we provide an overview of the BPT based tracker in the next section. We concentrate on the second approach in Section 7.2, where we detail the technique, including its theoretical bases and results, which are compared with results obtained for another two systems based on the mean shift algorithm.

7.2 BPT based tracking

The main steps of the BPT-based approach are the following:

- 1: Create the BPT for the current frame
- 2: **for** each face detected in the previous frame **do**

- 3: Search for an instance of the face in the current frame
- 4: Constrain the BPT (deactivate some nodes)
- 5: **end for**
- 6: Search for new faces in the current frame, on the constrained BPT
- 7: Return to step 1 for the next frame

In the following, each step is explained and illustrated using two frames from the ‘Minister’ sequence. Non consecutive frames have been selected to show the case of a new face entering the scene.

Creation of BPT

First, the accuracy partition, search partition, and the BPT search space for the current frame are created with the same parameters used for the intra-detection (see Figure 7.1).

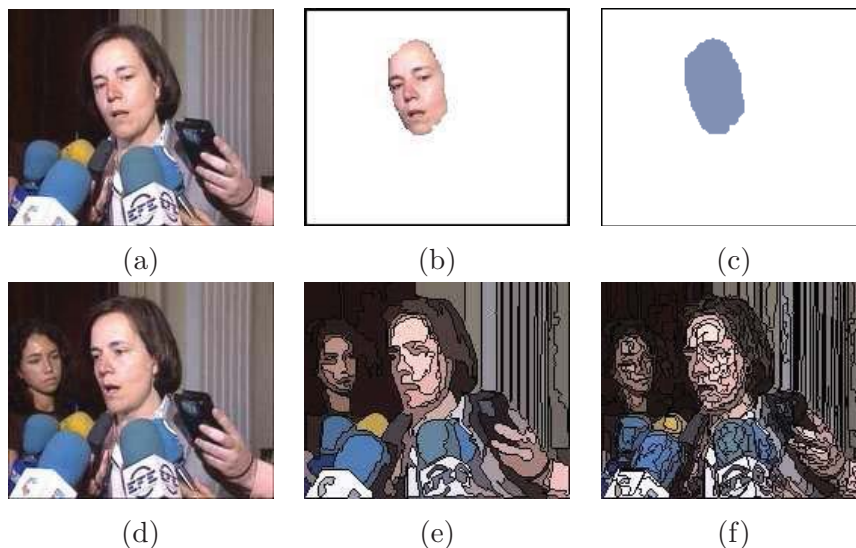


Figure 7.1: Frame 70 (a), detected face (b) and face mask (c) in frame 70, frame 90 (d), search partition (e) and accuracy partition (f) for frame 90.

Search for the face in the current frame

The goal is to detect, in the current frame, an instance of a face found in the previous frame. Some features of the previous face are used to simplify and guide the search. They are the following generic descriptors (see Section 6.1): the face mass center, size, bounding box, mean color, aspect ratio, compactness, simplicity, homogeneous texture, and also a mask defined by the face area (the *face mask*, shown in Figure 7.1(c)).

We assume that face areas in the two frames overlap, and use the face mask as a marker to select some of the BPT nodes. We mark the leaves in the tree that intersect the face mask, and propagate these markers up to the root. Only these marked nodes are used in next steps; we call them *active nodes*. The remaining (*non-active*) nodes are not taken into account.

For fast moving faces more complex approaches could be used, like estimating the face trajectory using information from a set of previous frames [104]. Note that if the position prediction fails and the tracker loses the face in a frame, it can still be found by the last tracking step (detection of new faces) in the following frame.

Next, a simplification stage is performed using, as in the general case, a cascade of binary one class classifiers. However, the classifier bounds are now adapted to the values of the generic descriptors calculated for the previous face. Then, the remaining nodes are extended, using the shape of the previous face instead of using an ellipse. Finally, nodes are classified using the same classifier trained and used for the intra-detection. The candidate with highest likelihood (above a threshold) is selected.

Figure 7.2 shows the BPT for frame 90 after the simplification stage, with active nodes depicted with rectangles; the segmented face is shown in Figure 7.4(a).

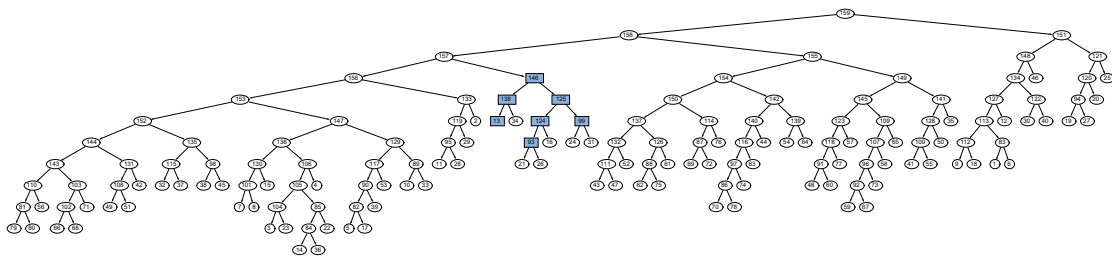


Figure 7.2: Inter-frame detection. Active nodes after the simplification stage

Search for new faces

To find new faces, an intra-detection stage is performed on a constrained BPT. We mark all the tree leaves that are associated with regions which are included in a previously detected face, and propagate the markers up to the root. These marked nodes are deactivated, since they cannot be part of another face. The search is performed on the remaining nodes, the active ones. Finally, the complete detection procedure is performed on the constrained BPT, using the generic face model.

The constrained BPT after the simplification stage (active nodes depicted with rectangles), the tracked face and the new face are shown in Figures 7.3 and 7.4.

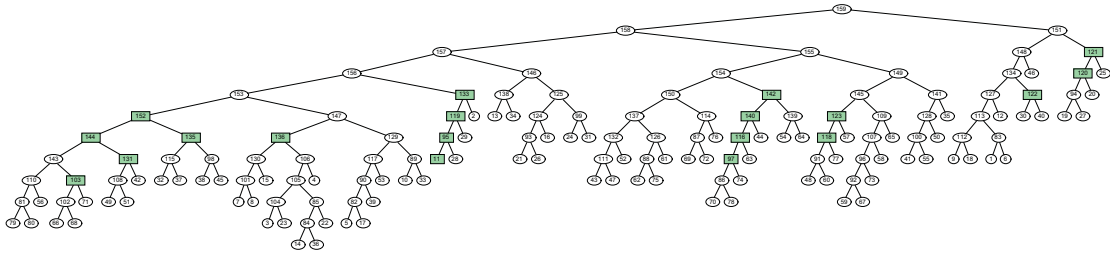


Figure 7.3: Constrained BPT used to detect new faces.

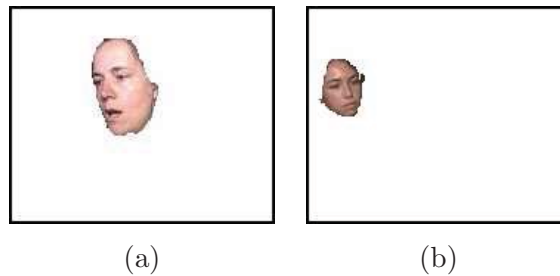


Figure 7.4: Tracked face (a) and new detected face (b).

7.3 Mean shift tracking

The second technique combines the mean shift algorithm for tracking with a representation of the images in terms of regions homogeneous in color. As will be discussed in the following, mean shift can be used as a robust and flexible algorithm for tracking, requiring minimal training and computational resources. In turn, the use of regions permits a robust estimation of object and background models, as well as the precise definition of the face shape. We begin with a review of mean shift and its use for object tracking.

7.3.1 Mean shift

Mean shift is a non parametric, iterative procedure introduced by Fukunaga and Hostetler [55] for seeking the mode of a density distribution represented by a set of samples. The procedure was revisited by Cheng [27] who developed a more general formulation and demonstrated the use of mean shift for clustering and global optimization. In the last years the technique has been applied to different areas of image processing, such as image segmentation, appearance-based clustering and tracking.

In this section we briefly review the method following the notation used in [27], introducing first the concepts of kernel and profile which are required to define the mean shift.

Let X be an n -dimensional Euclidean space. A function $K : X \rightarrow R$ is a *kernel* if there is a function $k : [0, \infty] \rightarrow R$, called its *profile*, which is nonnegative, nonincreasing, piecewise continuous and with $\int_0^\infty k(r)dr < \infty$, such that

$$K(x) = k(\|x\|^2) \quad (7.1)$$

Let $S \subset X$ be a finite set, the *sample data*, K a kernel and $w : S \rightarrow [0, \infty)$ a weight function. The *sample mean* with kernel K at a point $x \in X$ is defined as

$$m(x) = \frac{\sum_{s \in S} K(s-x)w(s)s}{\sum_{s \in S} K(s-x)w(s)} \quad (7.2)$$

The difference $m(x) - x$ is called *mean shift*. The repeated movement of data points to the sample mean is called mean shift algorithm. The idea is to compute the sample means for a reduced set of points $T \subset X$ and move the points in T towards their mean, until convergence. That is, if $m(T) = \{m(t) : t \in T\}$, the mean shift procedure iterates and evolves T until it finds a fixed point $T = m(T)$.

The weights $w(s)$ can be fixed through the process or re-evaluated after each iteration, and may also be a function of the current T . Although in the original formulation of mean shift [55] $T = S$ (in what is called a *blurring* process), T and S are usually different sets, with S a fixed set of samples. For example, in a clustering application T consists of the set of cluster centers whereas in object tracking it is the object centroid.

The kernel K defines an influence zone for each point x in T . Two kernels typically used are the *unit flat* kernel (see Figure 7.5 (a) and (b))

$$K(x) = \begin{cases} 1 & \text{if } \|x\| \leq 1 \\ 0 & \text{if } \|x\| > 1 \end{cases} \quad (7.3)$$

and the *unit Gaussian* kernel

$$K(x) = e^{-\|x\|^2} \quad (7.4)$$

Kernels can be scaled to modify their spatial extent, defining $K_\alpha(x) = K(\frac{x}{\alpha})$ for any $\alpha > 0$; this α is called the kernel *size* or *bandwidth*. As pointed in [27], any kernel K can be replaced by the kernel $(\alpha K)(x) = \alpha K(x)$ for any $\alpha > 0$. Therefore, there is no need to impose the condition $\int_X K(x)dx = 1$ or to normalize the weights so that $\sum_{s \in S} w(s) = 1$; we can disregard these factors and work with the simplest possible expressions for the kernel and the weights.

Cheng showed in [27] that the mean shift with fixed S seeks the modes of the density estimate $q(x)$ computed with another kernel H which is called the shadow kernel of K :

$$q(x) = \sum_{s \in S} H(s-x)w(s) \quad (7.5)$$

The two kernels must satisfy the relationship $h'(r) = -ck(r)$, where h and k are the profiles of H and K , respectively, $r = \|s-x\|$ and $c > 0$ is some constant.

This relationship guarantees that the mean shift vector $m(x) - x$ is in the gradient direction of the density estimate $q(x)$. Moreover, the magnitude of the mean shift vector is proportional to the ratio of the gradient and the local density using kernel K , having therefore an adaptive step size, that is, it moves fast when it is far from the mode and in short steps when it is near the mode (see [45] for a deep analysis on the step size of the shifts).

The shadow of the unit flat kernel is the *Epanechnikov* kernel (see Figure 7.5 (c) and (d))

$$H(x) = \begin{cases} 1 - \|x\| & \text{if } \|x\| \leq 1 \\ 0 & \text{if } \|x\| > 1 \end{cases} \quad (7.6)$$

while the shadow of a Gaussian kernel is the same Gaussian kernel.

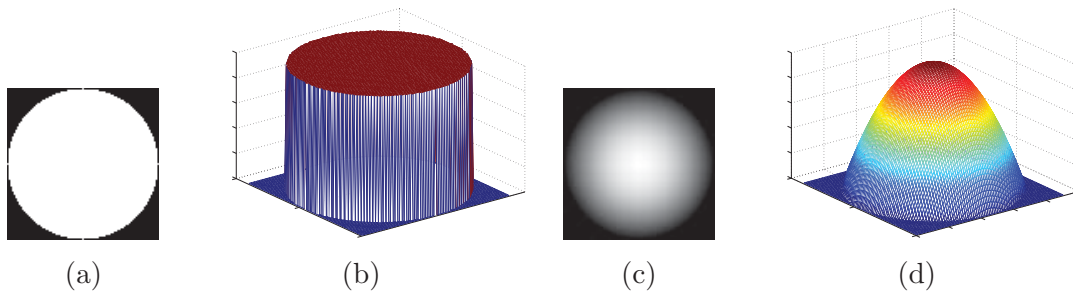


Figure 7.5: (a,b) Flat kernel and (c,d) Epanechnikov kernel.

7.3.2 Mean shift for tracking

In the last years mean shift has become a popular method for object tracking due to its ease of implementation, real time response and robust tracking performance. Despite its promising performance, the traditional mean shift procedure has some limitations that will be discussed in this section.

The main idea is to track the location of the object at each frame in the sequence. The evolving set T , therefore, consists of just one point, the object centroid. In the context of tracking, a sample corresponds to the spatial coordinates of a pixel x , and has an associated sample weight $w(x)$, which defines how likely the pixel with color $I(x)$ belongs to an object model. The mean shifts then seeks the mode of the kernel density computed with these weights.

To illustrate the weight and kernel concepts in the context of tracking, Figure 7.6 shows, for the image (a), the weights associated to each point x , or the *weight image* (b) (later on we will see how this image is defined), the kernel density $q(x)$ estimated with the Epanechnikov kernel (c), and the result of the mean shift iterations for a set T with only one point. The coordinates of the fixed point are drawn in red on the original image (d).

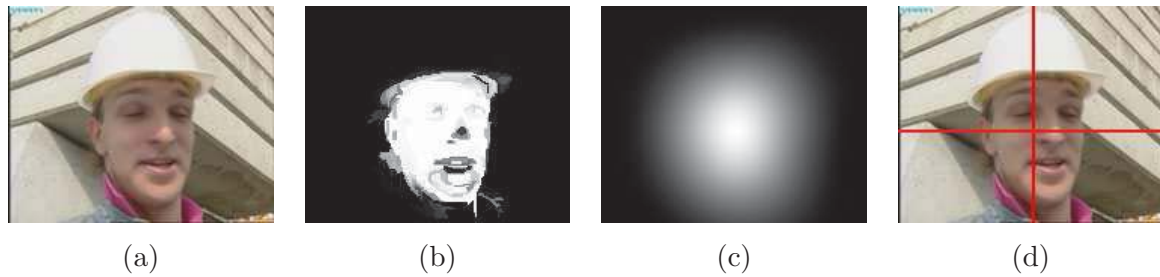


Figure 7.6: (a) Original image, (b) weight image, (c) kernel density and (d) coordinates of the fixed point.

The main steps of the basic mean shift approach to object tracking are the following:

- 1: Compute a model for the object
- 2: Estimate the initial position x_0 and size of the object in the current frame, and scale the kernel according to this size
- 3: Compute the weight function
- 4: Iterate mean shift $x_{i+1} = m(x_i)$ until convergence
- 5: Define the output, that is, the shape and extent of the tracked object
- 6: Return to step 2 for the next frame

A particular implementation of the mean shift algorithm requires the definition of the kernel (scale and shape), an object model, the weight function and the shape and extension of the final tracked object. Next, we analyze different solutions that provide different degrees of robustness to changes in the object and the background.

Kernel selection

The basic mean shift method requires isotropic kernels, such as flat or Gaussian kernels, for which the convergence of the algorithm has been proved, and assumes the constancy of the object scale and orientation during the course of the tracking [196]. However, in a tracking scenario objects may have complex shapes whose scale and orientation constantly change due to camera or object motion. This leads, in practice, to the use of generalized kernels or support areas that do not necessarily follow the definition given in Exp. 7.1.

Scale: The scale of the mean shift kernel directly determines the size of the window within which sample weights are examined. This size should be proportional to the expected area of the object being tracked.

As commented in [28], the kernel scale is a critical parameter in the performance of the mean shift algorithm. If the scale is too large, the tracking window may contain many

background points that resemble the object model, leading to an overestimation of the object size. A too large window may even make the tracker converge to an area between multiple modes instead of converging to one of them. On the other side, if the scale is too small, the shifts may move within a flat zone of likelihood around the mode, leading to poor object localization.

Changes in the object scale require an adjustment of the kernel bandwidth in order to consistently track the object. However, the basic algorithm does not provide a natural mechanism for choosing or adapting the kernel scale when tracking objects that are changing in size. The problem has been approached with different strategies: [20] uses moments of the weight image to estimate the object size and orientation, [31] performs the iterations several times in each frame, changing the kernel scale at each iteration and selecting the scale that maximizes a similarity measure between target and model distributions. In [28], after the object center is estimated, a mean shift procedure computes the bandwidth of the kernel in the scale space which is formed by convolving the image with a set of difference of Gaussians (DoG) kernels at various scales.

Shape: In the basic formulation radially symmetric kernels which are isotropic in shape are used. However, objects often have anisotropic structure. Using, for instance, a circular kernel to track an elongated object may bias the position estimation due to non-object pixels residing inside the kernel. Therefore, to minimize this effect, anisotropic symmetric kernels like rectangles or ellipses are frequently used. Different solutions have been proposed: [20] works with a rectangular kernel, whose size at each frame is estimated using second order moments of the kernel density in the previous frame, while [196] introduces the use of asymmetric kernels in the density estimation process through level set kernels that implicitly represent shapes that do not have an analytical form, and introduces scale and orientation as additional dimensions in the mean shift estimation process. These methods have a robust performance when tracking objects that change in scale and orientation, but may fail for substantial changes in the object shape.

Object model and weight image

The tracked object is modeled as a class conditional color distribution $P(I(x)/\mathcal{O})$, that estimates, for each pixel with color $I(x)$, the probability of the color of the pixel, given that the pixel belongs to the tracked object \mathcal{O} . This object distribution is learned offline from training images or during the initialization (e.g. the result of an object detection module applied to the first frame of the sequence). The model is typically built with histograms in a particular color space (1D hue, 2D CbCr, 3D normalized RGB, etc.).

The weight function measures, for each pixel in the image, some feature related to its similarity to the object model. Typically, weights are determined using a color-based object

appearance model [28]. In [20], the object is modeled with a histogram $h_{\mathcal{O}}$ and histogram backprojection is used to assign to each pixel the frequency associated with its value in the object histogram. In [31], the object histogram is compared with a histogram of colors observed within the current mean shift target window (pixels within the kernel support), $h_{\mathcal{W}}$. The sample weight at each pixel with color $I(x)$ is set to $\sqrt{h_{\mathcal{O}}(I(x))/h_{\mathcal{W}}(I(x))}$, a value related to the Bhattacharyya coefficient of histogram similarity. [163] proposes a similar measure within a Bayesian framework and assigns to each pixel the probability that the pixel represents the tracked object given its color. This measure will be further explained in the following section.

In [20] a static background is assumed, whereas in [31, 163], the background model is continuously relearned, since the histogram within the current window is recomputed at each iteration. Therefore, these methods adapt to moving cameras or background changes. However, they still work with a fixed object model, which may lead to poor localization when the object changes its appearance due to, for instance, lighting variation.

To achieve robust tracking performance, an adaptive object model should be taken into consideration. However, model update is a challenging problem [123]: over updated models can make the tracker sensitive to noise and occlusion, while under updated models can make the tracker loose object appearance changes.

Object model adaptation is seldom performed, since the object contours are not precisely defined and therefore it is difficult to decide which area of the window in the actual frame should be used in the adaptation. Adaptive schemes are proposed in [123], which works with a Kalman filter and hypothesis testing on the object kernel histogram or in [63], for a face tracking application, where a linear combination of the new and previous object model histograms is computed, the new histogram being estimated using pixels inside an ellipse fitted to the weight image.

Output definition

For the tracking of general objects, the output in each frame is usually the object centroid and a rectangle which has the size of the window used in the last iteration. This rectangle is used as an estimate of the object extent in approaches that vary the kernel bandwidth following changes in the object scale. However, they are typically too large and include parts of the background. The kernel size in the next frame is defined as a function of the previous object size. The object size is estimated as the zero order moment of the weight image [163].

Other techniques output the length, width and orientation of the object estimated with second order moments of the weight image, and the bounding box defined by these values [20].

7.3.3 Region-based mean shift

Our approach to face tracking combines mean shift with the use of regions. Regions are useful to compute in a robust way the weight image and to define precisely the contours of the tracked faces, and therefore provide a natural mechanism to initialize the search in the next frame.

The key points of the approach, which are detailed below, are the following:

- Partition of each image into regions homogeneous in color
- Kernel defined by fitting the image partition to a rectangular window, the search window, which is variable in size and shape
- Weight function based on a region-based estimation of the probability of each pixel of being part of the face
- Face and background models built with histograms, using quantized colors in the YCbCr color space
- Region-based update of the face and background models
- Definition of the final shape in terms of regions through a shape matching and partition fitting stage

Kernel selection

The algorithm works with pixels that lie within a subimage defined by a rectangular search window \mathcal{W} and a partition \mathcal{P} of the image into regions homogeneous in color.

Size: At each frame, the width and height of the search window are defined as the width and height of the bounding box of the object O found in the previous frame, scaled by a fixed factor (which is constant through the process). The window size is the same for all iterations within a frame.

Shape: The image partition \mathcal{P} is next fitted to the search window to define the kernel shape. The kernel is defined by all the regions R in partition \mathcal{P} that are completely included in \mathcal{W} :

$$K(x) = \begin{cases} 1 & \text{if } x \in \{R \in \mathcal{P} / R \subset \mathcal{W}\} \\ 0 & \text{otherwise} \end{cases} \quad (7.7)$$

The kernel scale, therefore, changes according to the size of the tracked object and its shape takes into account the color homogeneity observed in the image since it is defined by the regions in the partition. The fitting is performed at each iteration of the mean shift process.

An example is presented in Figure 7.7, that shows (a) the original image, (b) a partition, (c) the fitted partition, that is, the regions completely included in the rectangular search window, and (d) the kernel. In this example, the factor used to scale the previous object size is set to 1.7, and the partition has 500 regions. The partition has been created using the merging algorithm described in Chapter 4, with the WEDM merging criterion and a fixed number of regions (500).

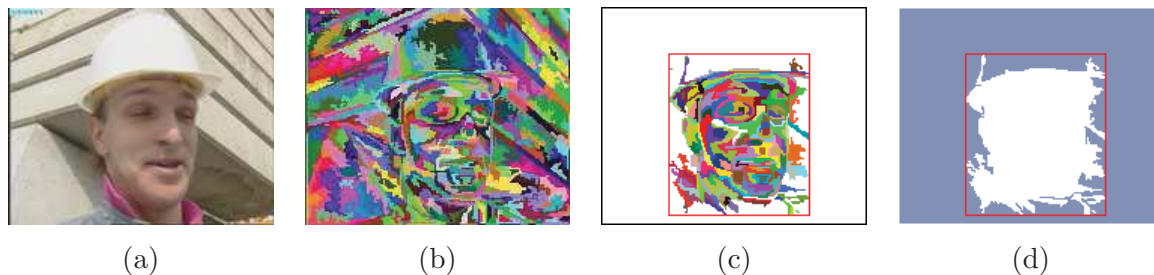


Figure 7.7: Original image (a), partition (b), fitted partition (c) and fitted partition mask (d).

Object and background color models

The object color is modeled as a class conditional color distribution computed with a histogram in the YCbCr color space, with 32 bins per component. Therefore, given a pixel x with color $I(x)$, the likelihood of the pixel given that it belongs to the object is $p(I(x)/\mathcal{O}) = h_{\mathcal{O}}(I(x))$, where $h_{\mathcal{O}}$ is the object histogram. This histogram is first learned from the object segmented in the first frame of the sequence, which is obtained with the detection system presented in the previous chapters.

The background color model is $p(I(x)/\mathcal{B}) = h_{\mathcal{B}}(I(x))$, where $h_{\mathcal{B}}$ is the background histogram. This model is first learned using a neighborhood around the object \mathcal{O} segmented in the first frame.

During the tracking, the object and background models obtained at frame $t - 1$ are used to segment the object at frame t . The object probability of a pixel x with color $I(x)$ at frame t can be computed using the Bayes' formula:

$$p(\mathcal{O}_t/I(x)) = \frac{p(I(x)/\mathcal{O}_{t-1})p(\mathcal{O}_{t-1})}{p(I(x))} \quad (7.8)$$

where

$$p(I(x)) = p(I(x)/\mathcal{O}_{t-1})p(\mathcal{O}_{t-1}) + p(I(x)/\mathcal{B}_{t-1})p(\mathcal{B}_{t-1}) \quad (7.9)$$

$p(I(x)/\mathcal{O}_{t-1})$ is the object model and $p(\mathcal{O}_{t-1})$ is the object probability (at $t - 1$). In turn, $p(\mathcal{B}_{t-1})$ is the background probability and $p(I(x)/\mathcal{B}_{t-1})$ the background color model.

These probabilities are used (i) to define the weight image and also (ii) to classify each region into object or background and define the shape of the tracked object in the current frame.

Object and background color models are computed at each frame, using the object segmented in the previous frame. To compute these models, both $p(I(x)/\mathcal{O})$ and $p(I(x)/\mathcal{B})$ should be continuously relearned. However, instead of computing explicitly a histogram for the background, the whole expression $p(I(x))$ is estimated, building a histogram $h_{\mathcal{W}}$ of the pixels that are within the last search window (in the previous frame $t - 1$). We do not work with all the pixels but only with those pixels that are inside the kernel. The continuous relearning of the background (through $p(I(x))$) avoids tracking failure when the background scene changes. The value of $p(\mathcal{O})$ is estimated as the ratio between the size in pixels of the object detected in the previous frame, and the kernel size in pixels.

Weight function

The weight image is computed in a region-based manner as opposed to the typical pixel-based approach. Each region R_i in the fitted partition is assigned a weight value which is the average:

$$w(R_i) = \frac{1}{|R|} \sum_{x \in R} p(\mathcal{O}_t/I(x)) \quad (7.10)$$

where $|R|$ is the region size in pixels. Therefore, the weight image is constant within each region in the partition

Following the example presented in Figure 7.7, the weight image for the last iteration of the mean shift is shown in Figure 7.8(c). For display purposes the values of $p(\mathcal{O}/I(x))$ have been scaled to $[0, 255]$.

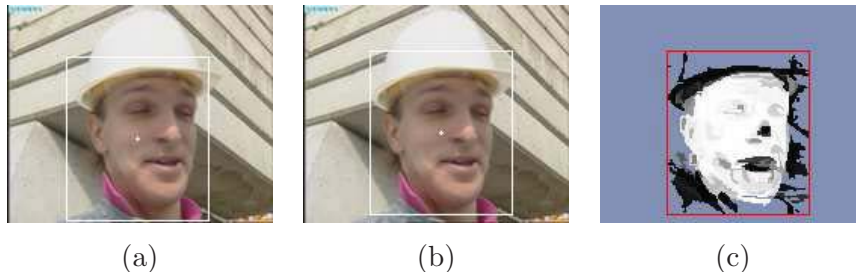


Figure 7.8: Search window and object position in the first (a) and last (b) iteration, (c) weight image in the last iteration.

Mean shift iterations

The evolving set T is initialized with the position of the object detected in the first frame and the search window is centered on this location in the second frame. The sample mean is estimated for the weight image and the search window is moved to the new position. The process is iterated until convergence. Figures 7.8 (a) and (b) show the search window at the first and last (second) iteration of the algorithm.

The final position of the object is used to initialize the search for the next frame. This approach assumes a smooth object motion and a continuous tracking of the object. A more complex approach could be used, like estimating the trajectory of the object and predicting its position in the following frame. However, our experiments show that the simple initialization suffices since the algorithm converges in very few iterations (2.36 on average).

Output definition

Once the mean shift converges, the fitted partition in the last search window is used to define the final shape of the face. The procedure is performed in three steps:

Initial object mask: First, pixels are classified into object or background; we choose for each pixel the class with highest probability. Then a region R within the fitted partition is said to be an object region if the majority of its pixels are classified as object pixels, that is, if

$$\sum_{x \in R} \mathcal{I}(c(I(x)) = \mathcal{O}) > \frac{1}{2}|R| \quad (7.11)$$

where $c(I(x))$ is the pixel class. A mask is created including all regions classified as object regions (see the example in Fig. 7.9(a)).

For sequences where the object color is very different from the background color, this simple procedure provides a very good estimate of the object shape. However the mask may contain background areas which present colors similar to object colors and may also lack some object areas due, for instance, to an illumination change.

At this point we introduce shape information to correct these problems.

Shape matching: A shape fitting stage is performed to define more precisely the face contour. For this step we could use the matching strategy based on distance transforms described in Section 6.3 for the extension of the BPT nodes, working with a fixed shape model of the face or using the shape of the face detected in the previous frame.

However, since ellipses are good face shape models, we follow a simpler and faster approach. We assume that the face area is one connected component without holes and that its shape is approximately elliptical and find the best-fit ellipse on the base of

second order moments [76] computed on the weight image. The orientation θ , and the length of major and minor axis of the ellipse, l_{max} and l_{min} , are:

$$\theta = \frac{1}{2} \arctan\left(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}\right) \quad (7.12)$$

$$l_{max} = \left(\frac{4}{\pi}\right)^{1/4} \left[\frac{I_{max}^3}{I_{min}}\right]^{1/8} \quad l_{min} = \left(\frac{4}{\pi}\right)^{1/4} \left[\frac{I_{min}^3}{I_{max}}\right]^{1/8} \quad (7.13)$$

where $\mu_{i,j}$ are central moments and $I_{min} = \sum_{x/K(x)=1} [(x - \bar{x})\cos\theta - (y - \bar{y})\sin\theta]^2$, $I_{max} = \sum_{x/K(x)=1} [(x - \bar{x})\sin\theta - (y - \bar{y})\cos\theta]^2$. An example of ellipse fitting is presented in Fig. 7.9(b).

Final object mask: Finally, we combine information from the fitted ellipse and the initial object mask. The final object is formed by the set of regions that are: (i) completely included in the mask and in the fitted shape, (ii) completely included in the mask, and partially included (more than 50% in our experiments) in the scaled shape, and (iii) regions that are completely inside the mask but have likelihood values lower than the threshold (the holes in the initial mask). The three kinds of regions are shown in red, blue and green, respectively, in Fig.7.9(c), and (d) shows the final face contour smoothed and superimposed on the original image.

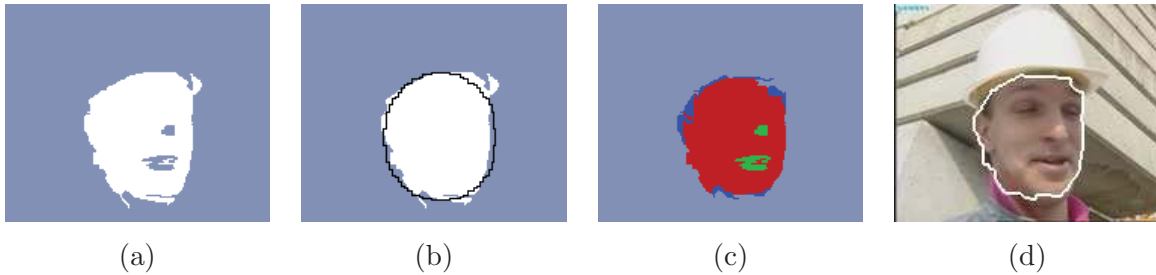


Figure 7.9: (a) Initial object mask, (b) ellipse fitting result, (c) final object mask and (d) final face contour.

7.3.4 Experimental results

To check the effectiveness of the proposed method, we have tested it on a wide variety of challenging image sequences. For comparison purposes, results are also presented for another two systems based on the mean shift algorithm. One is the Camshift (for Continuously Adaptive Mean shift) system [20] from the OpenCV library, and the other one is an improvement of Camshift proposed in [163] and named ABCshift (for Adaptive Background Camshift). Camshift combines the basic mean shift algorithm with an adaptive step that resizes the

search window at each frame, but assumes a static background and works with a fixed object color model. ABCshift also works with a fixed object model but updates the background model and has a mechanism for window shrink control. At each iteration a similarity measure between object and background distributions is computed, and the search window is resized when this measure drops below a preset threshold.

The key differences between these two systems and our algorithm are: (i) our technique not only tracks but also segments the objects through the use of regions, (ii) the use of regions for the centroid computation and (iii) the update of both the object and background models.

In all the examples, object and window histograms are computed in the YCbCr color space, with 32 bins for each component. The scale factor used to define the search window is 1.7.

Figures 7.11 to 7.18 show results obtained with the three systems. For our technique we show the face contour superimposed on the original image, for ABCShift the output is a rectangle surrounding the face and for Camshift, an ellipse.

The examples presented in Figure 7.11 illustrate the tracking of faces with changes in pose in two simple, static scenarios, where faces and background have quite different colors. The three techniques present very good performance. In the ‘Minister’ sequence Camshift underestimates the face size and loses the top part of the face. The ABCshift algorithm underestimates the face size in some frames of the ‘Asturias’ sequence due to an incorrect scaling of the search window or an incorrect estimation of the face size. However, thanks to the comparison between object and background distributions, the tracker recovers in a few frames. Camshift adds part of the flag that is behind the man’s head in the ‘Asturias’ sequence in frame 591, but in the next frames the face moves and hides the flag, and the tracker recovers the correct face shape.

The performance of the three techniques in the sequence ‘Foreman’ (see Figure 7.12) is very similar until the moment in which the person’s hand partially occludes his face (around frame 154). The ABCshift algorithm includes the hand in the tracked area, given the similarity with the initial face model, leading to an overestimation of the object area and a bias in the centroid position. The presence of the hand is correctly managed by the region-based approach due to the fitting process and the correct update of the face model. It has to be noticed that in this case the ABCshift technique recovers very fast from the problem once the hand disappears. In the last part of the sequence, the area of support of the face is underestimated, the same problem that was discussed for the previous examples. Results with the region-based approach are much more robust due to the correct model update. Finally, for the proposed approach, the accuracy of the object definition has to be noticed, even in profile images, where the elliptical face model is not so adequate.

The sequence ‘Surveillance’ (see Figure 7.13) is used to illustrate the relevance of the model update. As the person gets into the room, the illumination changes and only a portion of

the face matches the initial distribution. In this situation, the ABCshift tracking is restricted to this portion of the face and, eventually, the tracked area collapses. On the contrary, the updating capability of the region-based approach allows the correct tracking of the face. In this sequence, even though object and background models are fixed, Camshift can correctly track the face.

The ‘Jamie’ sequence (see Figure 7.14) was captured with a static camera, and shows changes in face pose and scene illumination (although when the illumination changes, in frame 31, the face is not moving). The region-based mean shift and the Camshift approaches correctly track the face, but in ABCshift the size of the search window oscillates; in some frames the output includes a large part of the background and in other frames it misses part of the face.

The sequence ‘Jamie1r’ (see Figure 7.15) presents a fast moving face with large variations in scale. As the face approaches the camera, the region-based approach correctly tracks and segments the face (although the segmentation is not very accurate), and when it moves backwards it recovers the correct shape in a few frames. The Camshift algorithm fails when the face moves across part of the background with a similar color (the door), and a large region of the background is mistaken for the object. ABCshift correctly adapts to background changes, but for the fast movements around frame 22 it cannot resize the search window correctly.

The ‘Ms’ sequence (see Figure 7.16), shows variations in face pose and an arm approaching the face. The presence of the hand is correctly managed by the proposed algorithm due to the fitting process. The ABCshift and Camshift trackers correctly deal with pose changes but around frame 205 the search windows start growing, including the arm and part of the background and the algorithms converges to a point between the two modes.

In ‘Antonio’ sequence, the face moves ahead and backward very fast, on a background of a very similar color. The region-based mean shift and ABCshift can correctly track the face (although the face is underestimated in some frames), but Camshift adds part of the back door and misses the face.

The last example, the ‘Jm’ sequence, the proposed algorithm works correctly during the first frames. But around frame 53 the face approaches the camera and then goes back in a very fast movement. The segmentation fails and part of the face is incorrectly used to update the background model. ABCshift and Camshift also underestimate the face size, but after some frames they can recover the face.

Figure 7.10 presents, for a set of 20 sequences, the average number of mean shift iterations per frame. The global average is 2.36 for the region-based algorithm, 2.42 for ABCshift and 2.68 for Camshift.

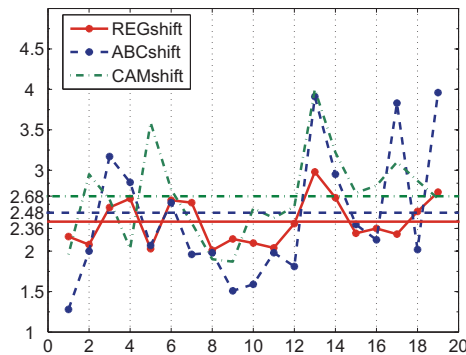


Figure 7.10: Average number of iterations per frame.

7.4 Summary

In this chapter we have mainly concentrated on the proposal of a tracking system, an extension of the basic mean shift tracking algorithm, that relies on the use of a color-homogeneous image partition. Objects are represented by sets of regions, which are used to build explicit models of object and background. These models permit the Bayesian estimation of pixel probabilities, which are used first to define the weight images during the tracking iterations and at the end, together with a face shape model, to segment the final face. The precise definition of face contours provides a mechanism for adapting the kernel size while tracking faces through changes in scale, and for updating the object and background models.

The results obtained with the region-based mean shift algorithm outperform Camshift and ABCShift algorithms and are robust to camera motion, background changes, face changes in pose, orientation and scale, and illumination variations.

The contributions of this chapter appear in the following papers:

- V. Vilaplana, D. Varas, *Face tracking using a region-based mean-shift algorithm with adaptive object and background models*, Proceedings of WIAMIS 2009, 10th International Workshop on Image Analysis for Multimedia Application Services, pp 17-20, London, 2009.
- V. Vilaplana, F. Marqués, *Region-based mean shift tracking: application to face tracking*, Proceedings of ICIP 2008, IEEE International Conference on Image Processing, San Diego, USA, October 2008.

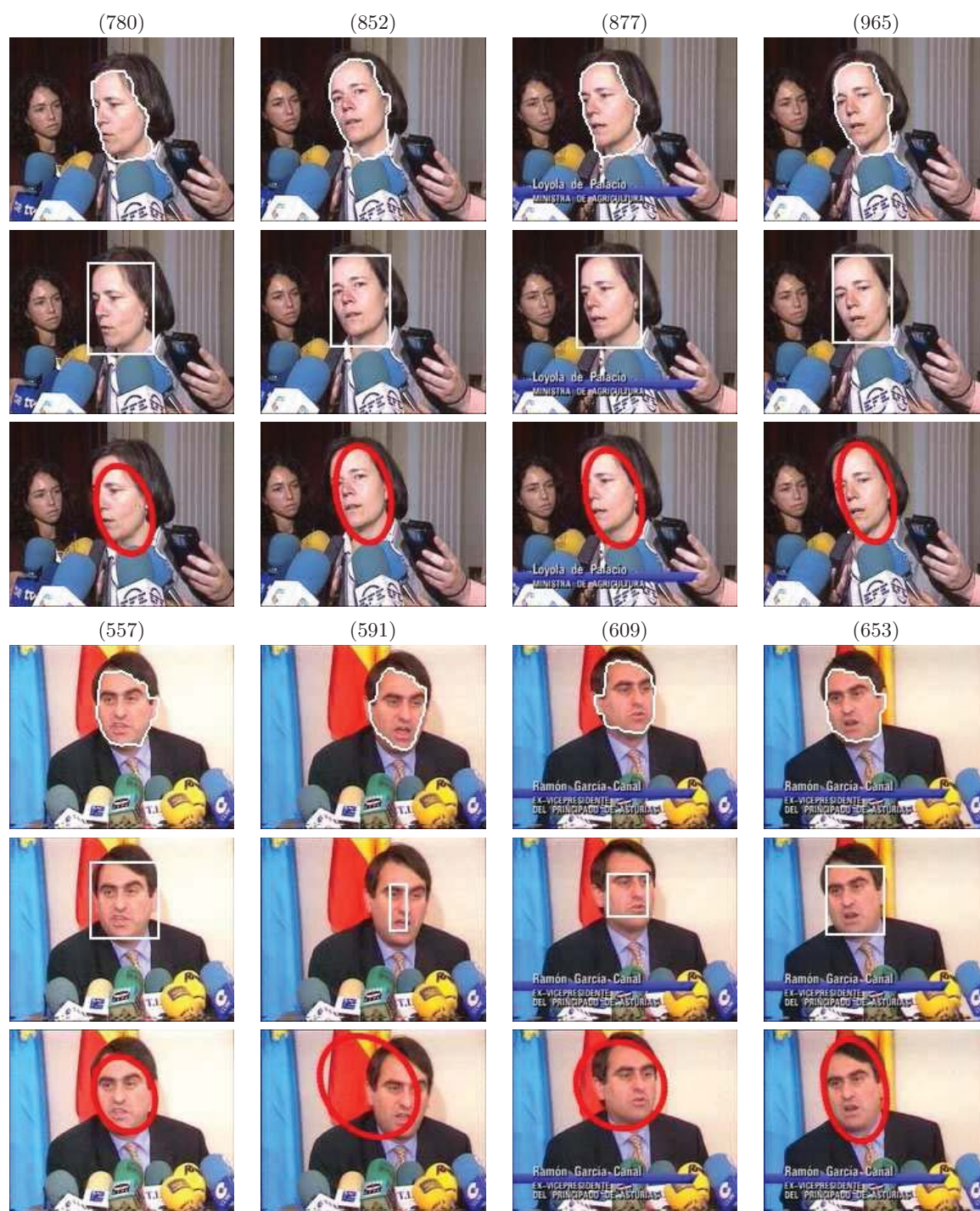


Figure 7.11: Results for sequences 'Minister' and 'Asturias' sequences using region-based mean shift, ABCShift (rectangles) and Camshift (ellipses) algorithms.

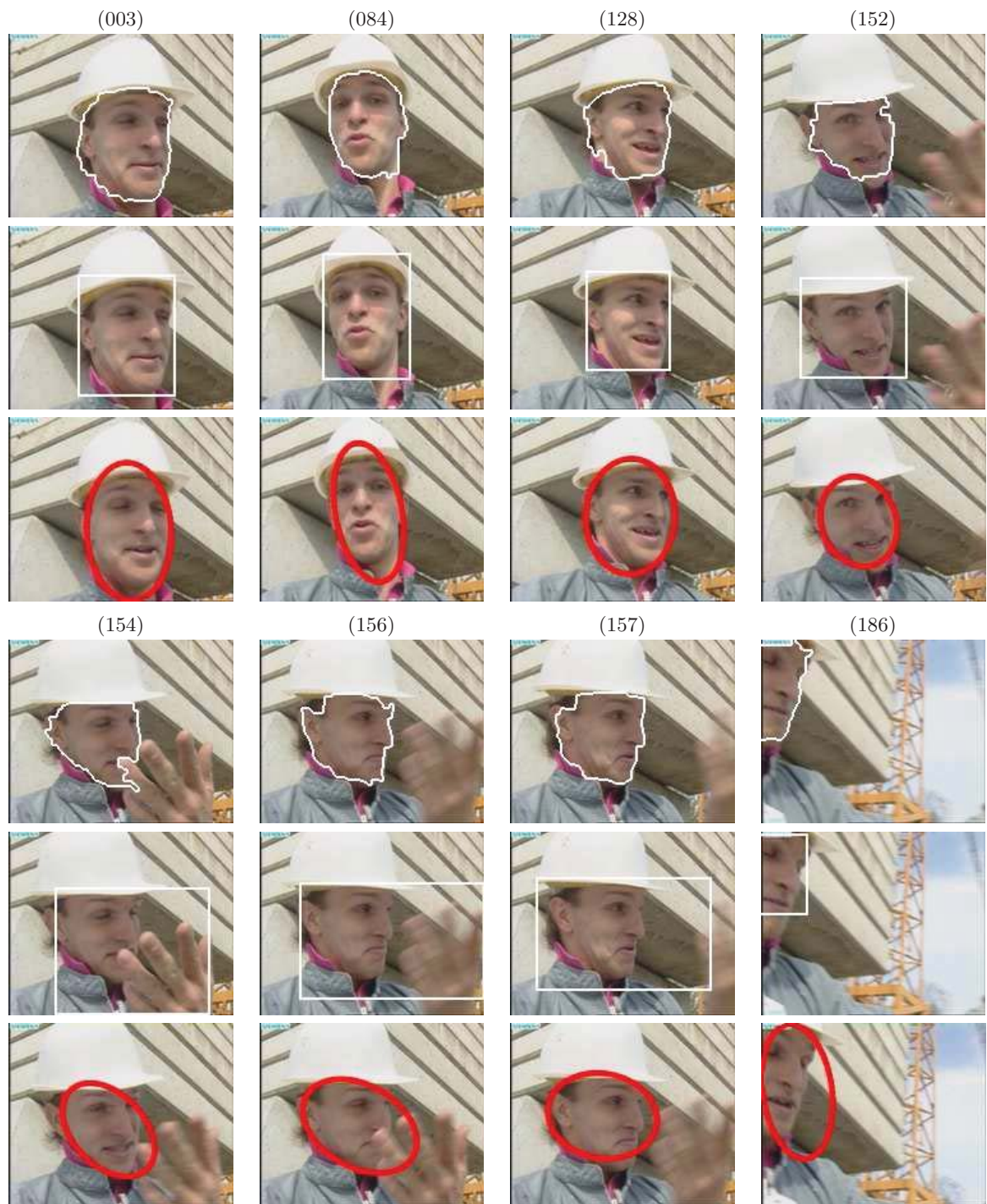


Figure 7.12: Results for 'Foreman' sequence using region-based mean shift, ABCShift (rectangles) and Camshift (ellipses) algorithms.

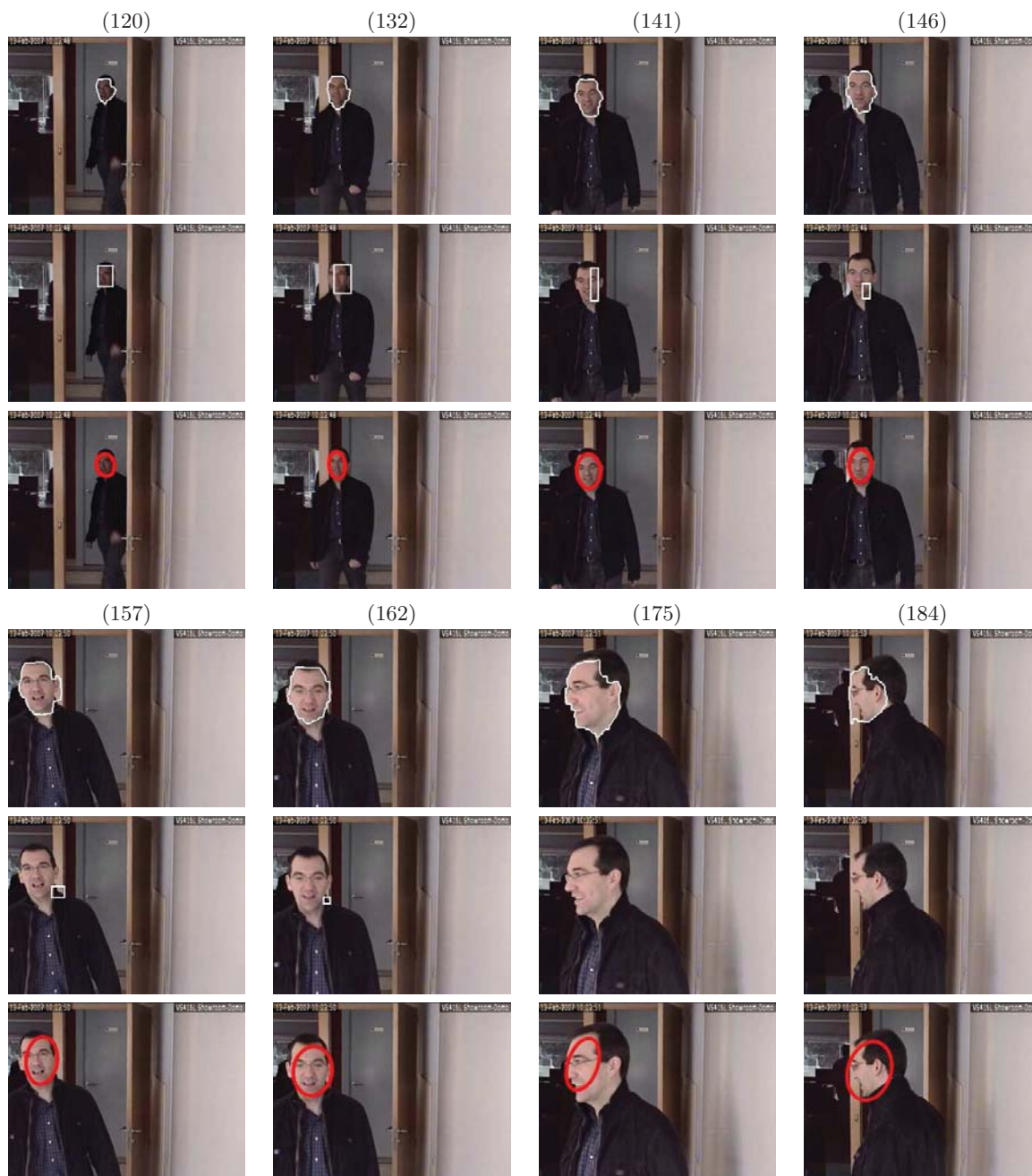


Figure 7.13: Results for ‘Surveillance’ sequence using region-based mean shift, ABCShift (rectangles) and Camshift (ellipses) algorithms.

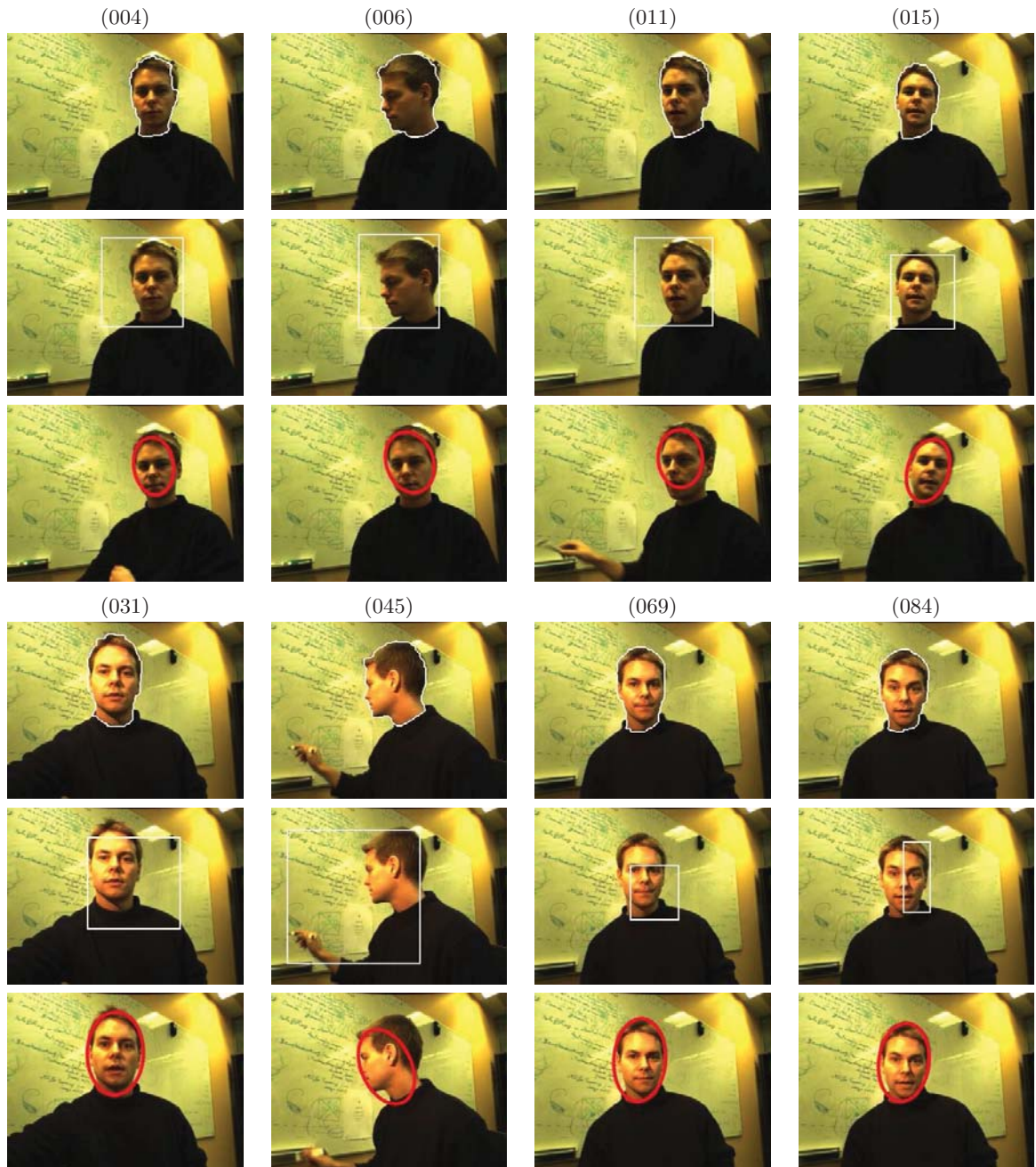


Figure 7.14: Results for 'Jamie' sequence using region-based mean shift, ABCShift (rectangles) and Camshift (ellipses) algorithms.

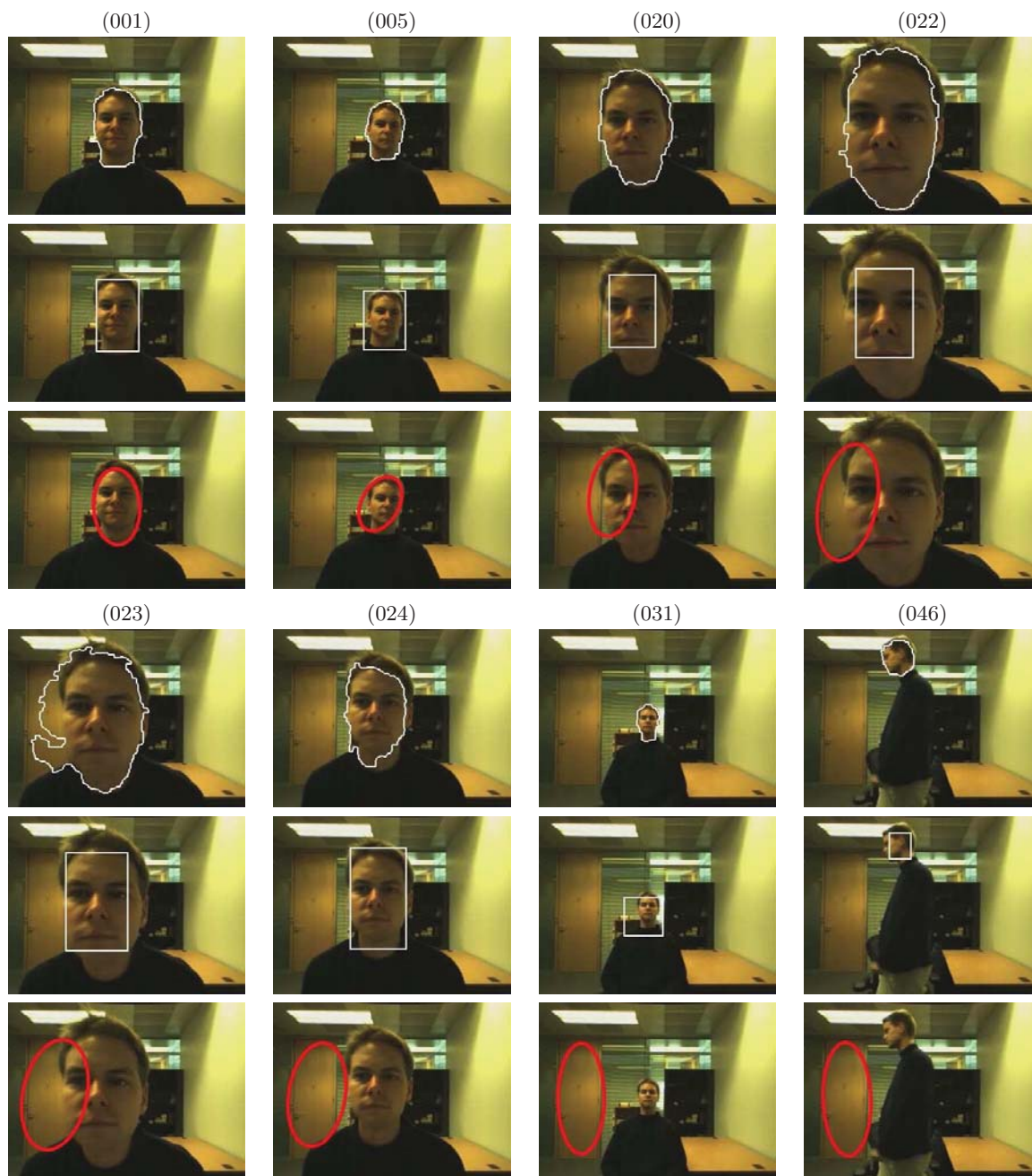


Figure 7.15: Results for ‘Jamie1r’ sequence using region-based mean shift, ABCShift (rectangles) and Camshift (ellipses) algorithms.



Figure 7.16: Results for 'Ms' sequence using region-based mean shift, ABCShift (rectangles) and Camshift (ellipses) algorithms.

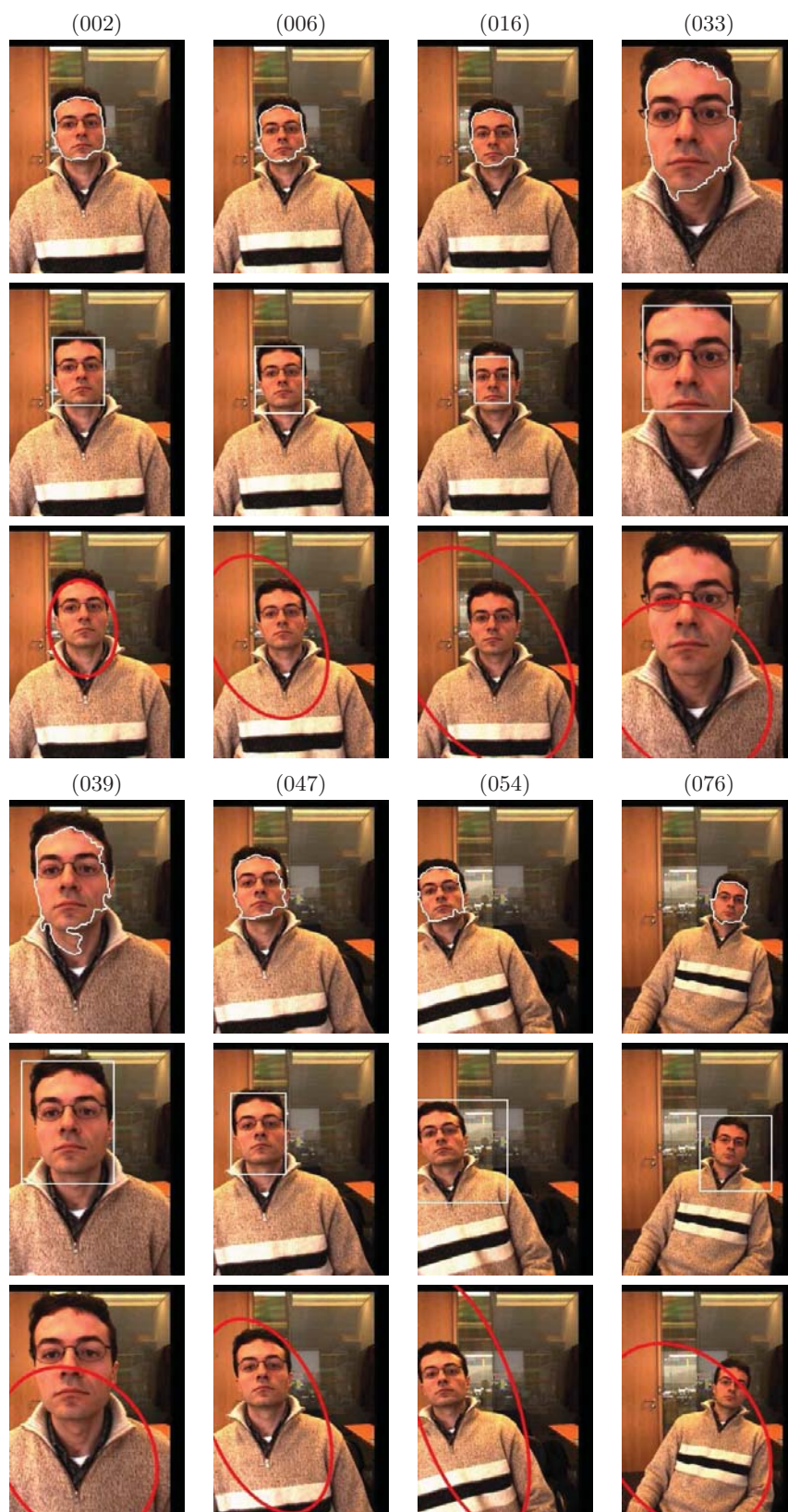


Figure 7.17: Results for 'Antonio' sequence using region-based mean shift, ABCShift (rectangles) and Camshift (ellipses) algorithms.



Figure 7.18: Results for 'Jm' sequence using region-based mean shift, ABCShift (rectangles) and Camshift (ellipses) algorithms.

Chapter 8

Beyond faces

The goal of this chapter is to show that the detection and tracking strategies proposed for faces in the previous chapters can be extended to handle other semantic classes.

In the first part of the chapter, some parts of the detection framework are adapted to cover a new set of semantic objects. Following the approach used for faces, we work with the Binary Partition Tree, which is created with the goal that nodes in the tree represent objects in the scene or a very good approximation of these objects. Next, object class knowledge is used to select and combine regions from the hierarchy, in order to define the exact object shape. We illustrate the approach with three different classes: traffic signs, sky areas and licence plates.

In the second part of the chapter, we extend the tracking technique proposed for faces to track objects with arbitrary, non rigid shapes. The tracking mechanism is the region-based mean shift algorithm. However, the final segmentation through shape matching and partition fitting is replaced by an energy formulation problem solved with a graph-cut algorithm.

8.1 Object detection and segmentation

In this section we show that the detection framework proposed for faces in Chapters 4, 5 and 6 is also valid to detect and segment other object classes.

It has to be said, however, that the characteristics of the proposed approach make it suitable only for certain types of objects, while for other objects different strategies may be more appropriate.

Our approach is aimed at detecting object classes where the main part of the object can be represented by a single node in the BPT. Then, an extension stage can be used to ‘break’ the tree structure and complete the object representation. Therefore, objects to be detected should be formed by only one connected component, and, according to the criteria used to build the tree, objects should be compact or fairly homogeneous in color. Nevertheless, we

will show a simple modification of the basic approach that, for the particular case of sky detection, permits the extraction of several components that form the sky.

The detection of other types of objects may require a different approach. For example, working with the same image model based on the BPT, we could use a search strategy of greater complexity, and describe an object in terms of parts and relationships between the parts, assuming that the parts are nodes in the tree and searching for these relationships [59].

A review of the published works on object detection shows, as in the face detection case (see Section 3.2), that the first and most common approaches to object detection are block-based: the image is scanned at multiple scales with a sliding window of fixed size and shape -typically rectangular-, and the contents of the window are input to a classifier. One limitation of these approaches, due to the large amount of candidates to evaluate, is their computational cost. Different speed up strategies have been proposed, like the use of heuristics, cascades of classifiers [83] or branch and bound schemes [90]. But their most important limitation is that the output of these systems is typically a bounding box surrounding the object. This is unsatisfactory as a final result because a bounding box does not capture the true shape of the object. The box may contain many non-object pixels, or may lack some object parts. The accurate segmentation of the object requires a post-processing stage.

In the last years, alternative methods have been proposed that incorporate unsupervised image segmentation into an object detection and segmentation framework, assuming that objects are defined by one or several regions in a segmentation of the image.

Ideally, region boundaries should correspond to object boundaries. However, as was discussed in Chapter 4, segmenting semantic objects is a very challenging problem. It is extremely difficult to partition an image into semantically meaningful elements, not just blobs of similar color or texture. Objects may be over-segmented or under-segmented. Moreover, regions in a partition may be produced by illumination discontinuities or may be artifacts introduced by the segmentation algorithm.

To overcome these limitations, some recent works use multiple segmentations of the same image (varying parameters or even the segmentation algorithm), assuming that the object to be detected is correctly segmented in at least one of them [139], or integrate the information from multiple segmentations by classifying each region (in each partition) and combining the results into an object mask [119]. Regions are then classified into one of a fixed number of classes. There are several works in this line, for example, those published in the context of the PASCAL Visual Object Challenges [116], where objects to be detected belong to a visual world formed by 20 object classes (person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor).

8.1.1 System overview

Our approach uses the same tools developed for faces to design models for other object classes, by selecting, training and combining an appropriate set of one-class classifiers.

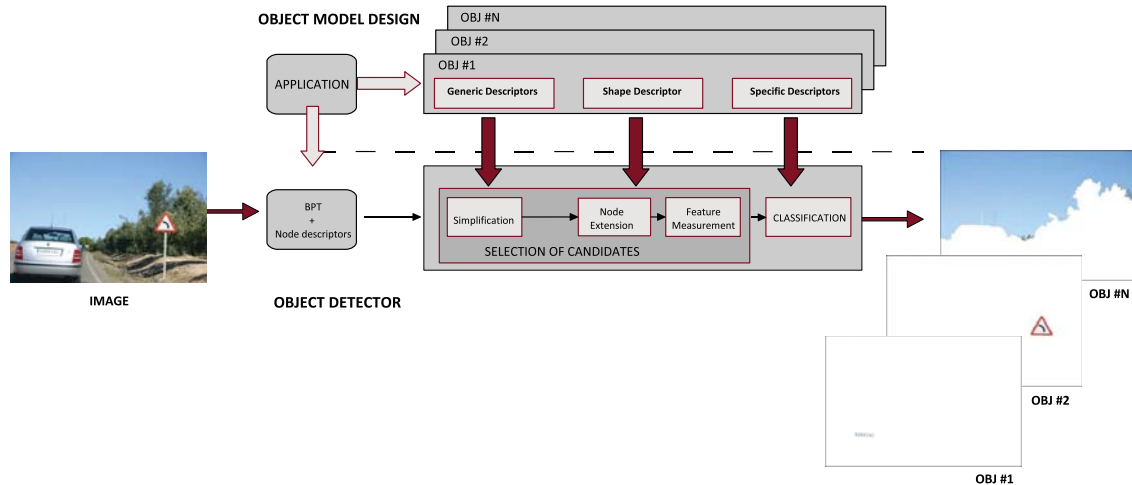


Figure 8.1: Object detection system

Figure 8.1 shows a general scheme of the system. Each input image is represented with a Binary Partition Tree (*image model*). We would like the nodes to represent complete or nearly complete objects. As shown in Appendix E, this is possible for many kinds of objects (compact or fairly homogeneous) if the initial partition is fine enough and a similarity measure that takes into account both color similarity and contour complexity is used for the merging process. While color is useful to define homogeneous regions, contour information favors the merging of regions with partial or total inclusions (unless they are very different in color), leading to regions with simple contours. The tree nodes are described by a set of simple geometric, color and texture features. These features are computed for all the nodes when creating the BPT and stored to be used later, in the detection of different classes of objects.

In the training mode, *object models* that characterize the different classes in terms of several region-based descriptors are learned. Each class is described at two levels. First, at a very general level, with a set of low-level features (the generic descriptors) associated with very simple classifiers, each based on one single feature. Second, at a more specific level, using more complex features and classifiers (specific descriptors). The first description is used to simplify the search, eliminating many of the candidate nodes, while the second, more costly, is applied only to the remaining nodes.

In the detection mode, for a particular object model, the tree is analyzed. Initially, some nodes are rejected (*Simplification*) based on the information provided by the generic descriptors. For the remaining nodes, a second set of descriptors (specific descriptors) is

computed for the final *Classification and Decision*. Between these two steps, there is the optional *Extension* stage. This step modifies the area of support of the node to conform to a reference object shape (shape descriptor) or to the geometry of the object.

In the following subsections we illustrate the usefulness of this approach for the detection and segmentation of three different classes: traffic signs, sky and licence plates. For each class, we briefly explain the interest in the detection task, analyze the attributes describing the class, list the descriptors used to train the object model and present some results.

Note that the proposed framework allows one to define new object models in an easy and flexible way. New object models can be designed by selecting, training and combining an appropriate subset from the set of available descriptors and classifiers.

8.1.2 Traffic sign detection

Traffic sign detection and recognition is a key point on the development of automatically driven vehicles and safety systems in human-driven vehicles.

The problem has a number of good characteristics [8]: the design of traffic signs is unique, thus object variations are small, and sign colors usually contrast well against the environment. Moreover, traffic signs are often set up in clear sight to the driver. However, the problem also involves several challenges: weather and lighting conditions vary significantly in traffic environments, as the camera moves motion blur or abrupt contrast changes may occur, and the signs may change over time resulting in rotated signs or degenerated colors. Some publications on traffic sign detection and recognition are [102, 8, 35].

Attributes

To illustrate the case of a perfectly defined object, in this work we focus on European warning signs, which have the shape of an equilateral triangle with a thick red border and a white background, and a black picture inside. We also consider within the group some priority signs which also have a triangular shape with the same red, white and black colors (or without black structures). For other shapes or colors a similar analysis can be done.

Object model

Given the previous attributes, the class of warning traffic signals is represented using the generic descriptors *Aspect ratio*, *Compactness*, *Homogeneity*, *Size* and *Mean Color*. The mean color descriptor is used to rapidly discard nodes with mean colors very different from the average sign color (e.g. blue, green). The shape descriptor is a triangle whereas the following specific descriptors are used: *Color histogram* and *Hausdorff distance*.

The simplification stage rejects as many nodes as possible using size, color, compactness,

aspect ratio and homogeneity descriptors. Shape is an important characteristic of traffic signs. Since warning signs can appear with different sizes and orientations (e.g. in the yield sign the triangle is upside down), the extension stage plays a critical role. Here, the allowed transformations for the reference shape (an equilateral triangle) are translation, rotation and scaling, and the matching is performed with binary distances. Even though traffic signs may appear rotated in the space, this set of transformations is enough to approximate the various views. Small regions from accuracy partition are added or removed from each node to fit the reference shape. Examples of the improvement in the representation of traffic signs achieved by the extension stage are shown in Figure E.1, Appendix E.

After the extension stage, two descriptors are computed for each candidate node: color histogram and the Hausdorff distance between the node and the reference shape. The color of the signs is modeled with a histogram using the three image components and 16 bins per component. The distance between histograms of a candidate node and the model histogram is computed using the Bhattacharyya distance. Before computing the Hausdorff distance, the reference shape model is transformed using the parameters found in the shape fitting stage. The distances obtained for the two descriptors are then normalized (robust min-max criterion) and combined by simple averaging.

Results

For the experiments, a set of 100 images with warning signs were collected from datasets [102, 66]. 30 of these images were used for training and the remaining ones for testing. Another 30 images without traffic signs were used to compute recall and precision values, which are 0.94 and 0.99, respectively. The BPTs were created with the same parameters used in the face experiments (Section 6.5).

The segmentation accuracy, the overlap between a ground truth region and a detected region, averaged over 70 images with ground truth is 0.98.

Figure 8.2 shows some segmentation results. Signs are correctly detected and accurately segmented in spite of clutter background and variations in scale and orientation.

8.1.3 Sky detection

Sky detection can contribute to image understanding by indoor/outdoor classification. Moreover, accurate sky segmentation can be used for content-based manipulation, like picture quality improvement by color enhancement, or background detection for 3D depth-map generation [202, 99].

Sky can have different appearances, such as clear, cloudy or overcast, and sky regions can significantly vary in color within an image (e.g. a clear sky image tends to be more saturated at the top and less saturated near the horizon). An additional challenge is the partial occlusion

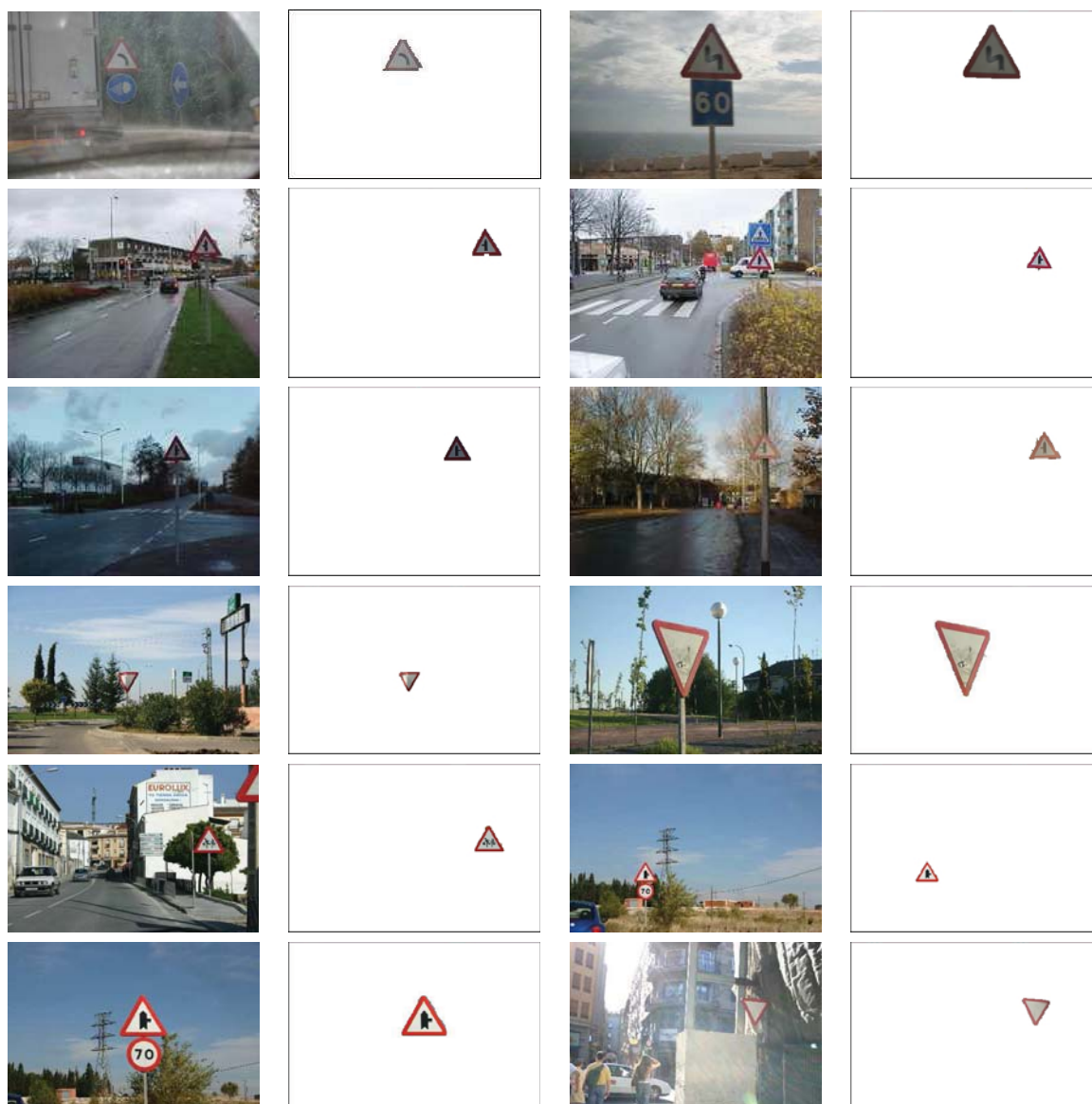


Figure 8.2: Traffic sign detection results.

of sky by foreground objects which separate sky into disconnected components. Furthermore, distinguishing between sky and objects that look similar to sky (e.g. areas or water, reflections of sky, other objects with similar color and texture as sky) is a non trivial task [202]. Finally, the shape of the sky region in an image is defined by the other objects in the scene. Different approaches to sky detection can be found in [153, 99, 202].

Attributes

Sky may appear as one or several connected components which are more likely to be found on the top of the image, and to present a smooth texture.

Besides the position in the image, color is the second characteristic property of sky. The color can range from pure white over different kinds of gray inside clouds to shades of blue of different saturation and brightness in cloudless regions.

Object model

We model sky color with a Gaussian distribution, which in spite of its simplicity produces very good results. Mixtures of Gaussians or the empirical distribution can also be used.

The Simplification stage works with *Position*, *Size*, texture *Homogeneity* and color features. However, instead of working with the mean color descriptor as we did for faces and traffic signs, we use the *Dominant colors* descriptor.

The motivation is that in many cases, the sky occupies a large part of the image. Large sky regions tend to appear as nodes in the upper part of the BPT. While the mean color is useful to characterize the color of homogeneous or small regions, the upper nodes in the tree are created by merging regions that may have different colors. In those cases, averaging colors may create new, false colors that still have a high sky-color likelihood. The dominant color descriptor can be used to detect and reject these nodes, by the individual analysis of each dominant color.

The binary classifier associated with the dominant descriptor rejects a region if one of its dominant colors is representative enough and has a low sky-color likelihood. The output for a region R with dominant colors $\{\{(\mathbf{c}_i, p_i, v_i)\}_{i=1, \dots, N}, s\}$ is:

$$\prod_{i=1}^N (1 - \mathcal{I}(l(\mathbf{c}_i) < th_c, p_i > th_p)) \quad (8.1)$$

where N is the number of dominant colors, \mathcal{I} is the indicator function, th_c and th_p are thresholds for the color likelihood and for fraction of the region R with color \mathbf{c}_i .

The example in Figure 8.3 illustrates the usefulness of this descriptor to characterize sky colors. The region shown in (c) (which is the merging of part of the sky with a region with trees) is rejected by the use of the classifier based on the dominant color descriptors. If we used a classifier based on the global mean color, the node would be accepted.

As previously commented, sky does not have a characteristic shape. The extension stage, in this case, does not fit the region to a shape model, but modifies the active nodes to adapt to the geometry of the sky. The idea is to break the tree structure, allowing mergings which are not represented in the tree.

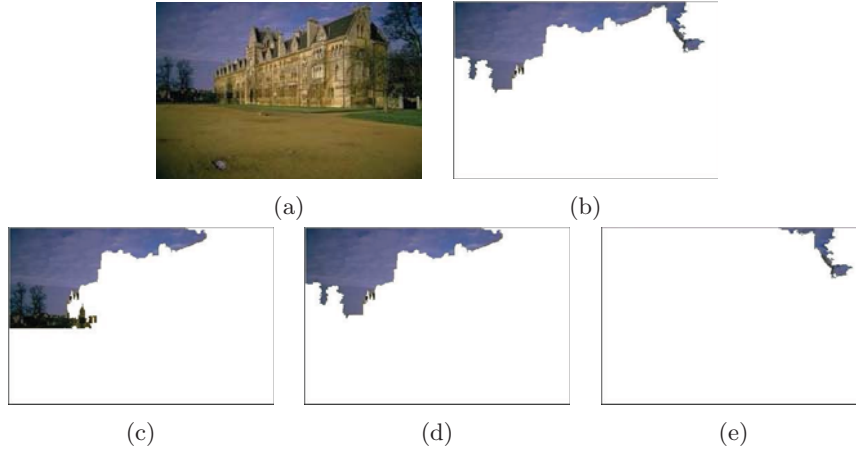


Figure 8.3: Original image (a), segmented sky (b), node rejected by the dominant colors classifier (c) and candidate nodes (d) and (e)

The extension stage merges active nodes with their neighboring regions if they are similar in color. The process is performed using the tree structure. First, we find all disjoint subtrees where the subtree root is a sky candidate (the node has been accepted by all the simple classifiers). For each subtree, its root is selected as the only sky representative for the subtree, ignoring all the other candidate nodes.

Then, each active node is merged with its active neighbors if they have a similar color. To measure the similarity between the two dominant color descriptors of the two regions, we use the distance proposed in [103]:

$$d(DC_1, DC_2) = \sum_{i=1}^N Np_{1i} + \sum_{i=1}^N Np_{2i} - \sum_{i=1}^N \sum_{j=1}^N a_{1i,2j} p_{1i} p_{2j} \quad (8.2)$$

where $a_{1i,2j}$ is a similarity coefficient between two colors \mathbf{c}_i and \mathbf{c}_j ,

$$a_{1i,2j} = \begin{cases} 1 - d_{i,j}/d_{max}, & d_{i,j} \leq T_d \\ 0, & d_{i,j} > T_d \end{cases} \quad (8.3)$$

and T_d is the maximum distance between two colors to be considered similar and $d_{max} = \alpha T_d$. The result of this process is a set of ‘extended’ candidate nodes.

For sky we do not use specific classifiers, but a final decision stage, where the final sky region is built by selecting all the candidate nodes which are situated in the upper part of the image. We assume that all sky regions have a vertical connection to the upper border of the image, either directly or through other regions classified as sky. All candidates that are connected with the upper border of the image form the final sky region. The final sky region may, therefore, be formed by several connected components.

This decision stage is illustrated in Figure 8.4. There are four candidates, nodes (d) to (g), which are shown together in image (b). The only two neighboring regions, nodes (d) and (e), are not merged because they differ in color. The only candidate that intersects the image border is (d), which defines the final sky region, which is shown in image (c).

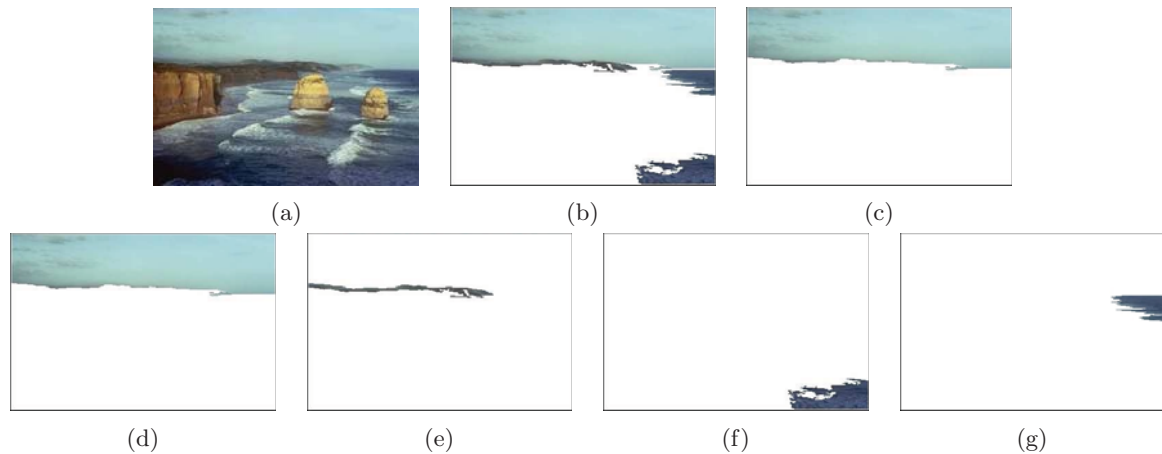


Figure 8.4: Original image (a), sky region (b), node rejected (c) by the dominant colors classifier, candidate nodes (d) and (e)

Results

For evaluation, we use 75 sky images for training and 500 for testing (400 with and 100 without sky), from the COREL dataset. The selected images present a large variety of sky appearances and challenging situations. The BPTs are created with the same parameters used in the face experiments (Section 6.5). Some results are shown in Figure 8.5.

In the first nine examples sky is correctly segmented, while the last three images show some segmentation errors which are commented below. In the second example of row 5, since only sky components that are on the top of the image are selected in the decision stage, the sky that can be seen through the construction is missing. In the first example, last row, part of the sky is merged with the opera house. In the second example, last row, the mountains and sea regions are merged with the sky.

We obtain very good results with recall and precision values of 0.97 and 0.93, respectively. The segmentation accuracy, measured on 200 manually segmented sky images is 0.92 (see Table 8.1).

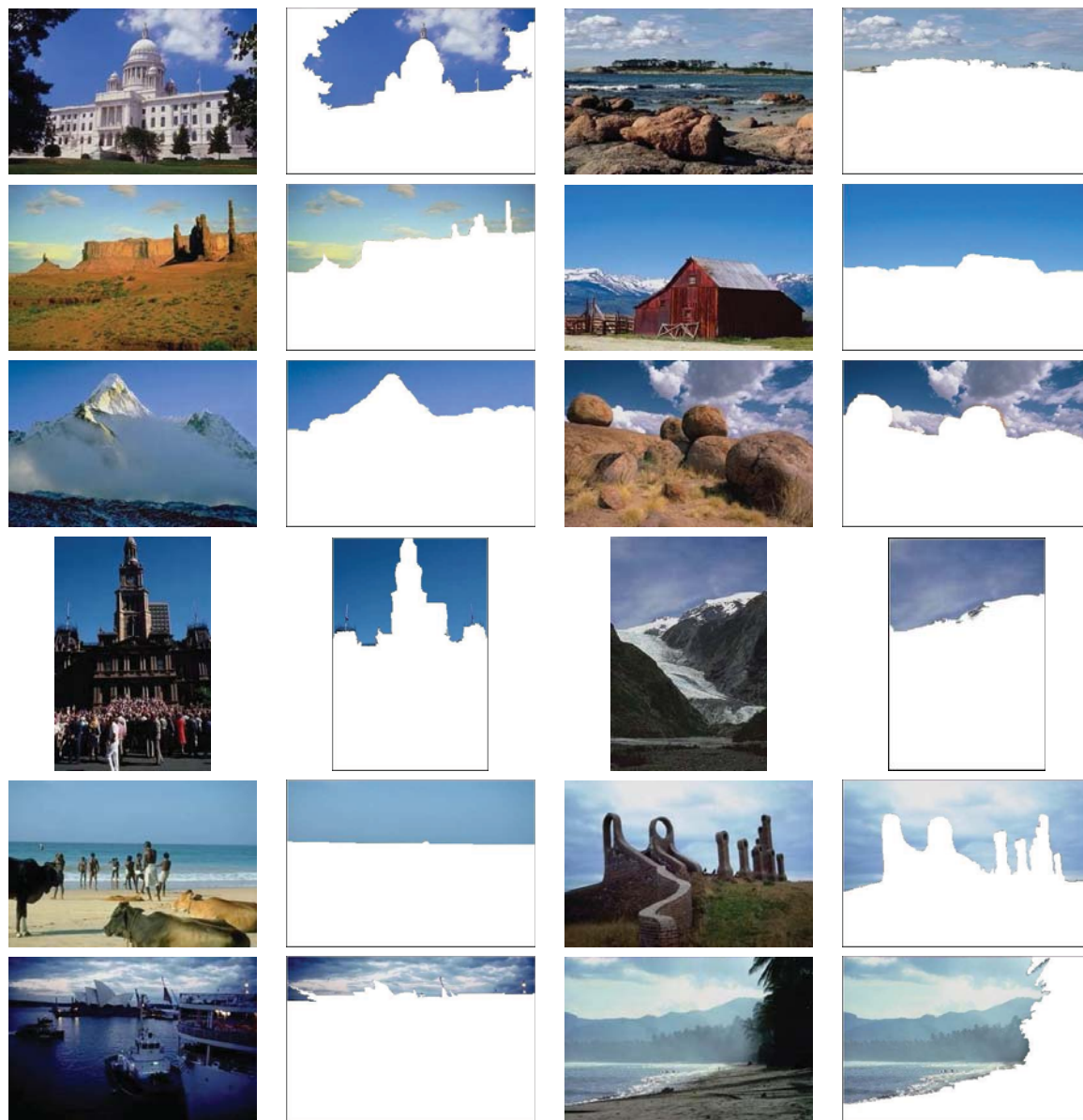


Figure 8.5: Sky detection results

8.1.4 Licence plate detection

Licence plate detection and recognition algorithms are used as core modules for intelligent infrastructure systems (ITS) like electronic payment systems (e.g. toll payment), parking lots access control, or freeway and arterial management systems for traffic surveillance [4].

These algorithms should operate fast enough to fulfill the needs of ITS. Even though our approach is not fast enough for some of these applications, it may be useful in other contexts

(e.g. image annotation). Moreover, a similar strategy can be used to detect other classes like caption text, which can be characterized with the same descriptors and classifiers used for licence plates [91].

The problem may have some practical difficulties such as poor image quality due to uneven lighting conditions, various observation angles from the vehicles and cameras. A survey of different strategies for licence plate detection and recognition on still images and video sequences is presented in [4].

Attributes

Licence plates are typically identified as rectangular areas in the image, containing rich edge and texture information.

Object Model

The previous attributes can be translated into constraints on the following generic descriptors: *Size*, *Aspect ratio*, *Compactness* and *Circularity*. After simplification, nodes are extended by fitting their shape to a rectangle, and classified using the *Homogeneous Texture* descriptor and the *Hausdorff distance* to the rectangle. The outputs of the classifiers are normalized with robust min-max and combined by simple average.

The Homogeneous Texture descriptor is used to characterize the texture of plates. We could also use Haar coefficients. However, we do not have a large training set of licence plate images, which would be necessary to train the Adaboost-based classifier associated with this descriptor. Therefore, we work with the MPEG7 Homogeneous Texture descriptor [103], that characterizes the region texture using the mean energy and energy deviation from a set of frequency channels. The 2D frequency plane is partitioned into 30 channels. The mean energy and deviation are computed in each of the channels. The extraction is done as follows: the image is first filtered with a bank of orientation and scale tuned filters using Gabor filters. The first and the second moments of the energy in the frequency domain in the corresponding sub-bands are then used as the components of the texture descriptor. The number of filters used is $5 \times 6 = 30$ where 5 is the number of scales and 6 is the number of directions used in the multi-resolution decomposition using Gabor functions. The Homogeneous Texture descriptor provides a precise quantitative description of a texture that can be used for accurate search and retrieval [103].

We classify the texture features with a nearest neighbor classifier. We choose this classifier because it works well for low-sample, high dimensional data [169]. Note that we could work with other classifiers, like the SVDD. However, we have preferred to add a new classifier to the set of available classifiers.

Results

For the experiments, we work with a set of 250 images with licence plates, showing large variations in scale, angle of vision, and illumination and with another set of 50 images without plates. The first group belong to a data set of Greek licence plates [4]. 50 of the plate images are used for training and the remaining ones for testing. The BPTs are created with the same parameters used for the other object classes.

The recall and precision values are 0.92 and 0.95, respectively, and the segmentation accuracy is 0.94. Some examples of correct segmentations are shown in Figure 8.6, while Figure 8.7 presents two false positive results.

8.1.5 Summary of results

Table 8.1 summarizes the results for the three classes in terms of recall, precision and segmentation accuracy. The segmentation accuracy between a ground truth region R_{GT} and a detected region R_D measures the area of overlap: $Acc = \frac{|R_{GT} \cap R_D|}{|R_{GT} \cup R_D|}$. The Acc is averaged over all segmented objects for which there is a ground truth region.

	SIGN	SKY	PLATE
Objects for training	30	75	50
Img. with+without objects	70 + 30	400 + 100	200 + 50
Objects for detection	70	400	200
Recall	0.94	0.97	0.92
Precision	0.99	0.93	0.95
Objects with ground truth	70	200	100
Segmentation Accuracy	0.98	0.92	0.94

Table 8.1: For each class: number of training objects, number of images with and without objects, number of objects in the images with objects, recall, precision, number of objects for which the ground truth segmentation is available and average segmentation accuracy.

8.2 Object tracking

In this section we extend the tracking technique proposed for faces to track objects with arbitrary, non rigid shapes. The goal is to track and segment an object in scenarios where objects may change in shape and scale and camera may be static or moving. The only

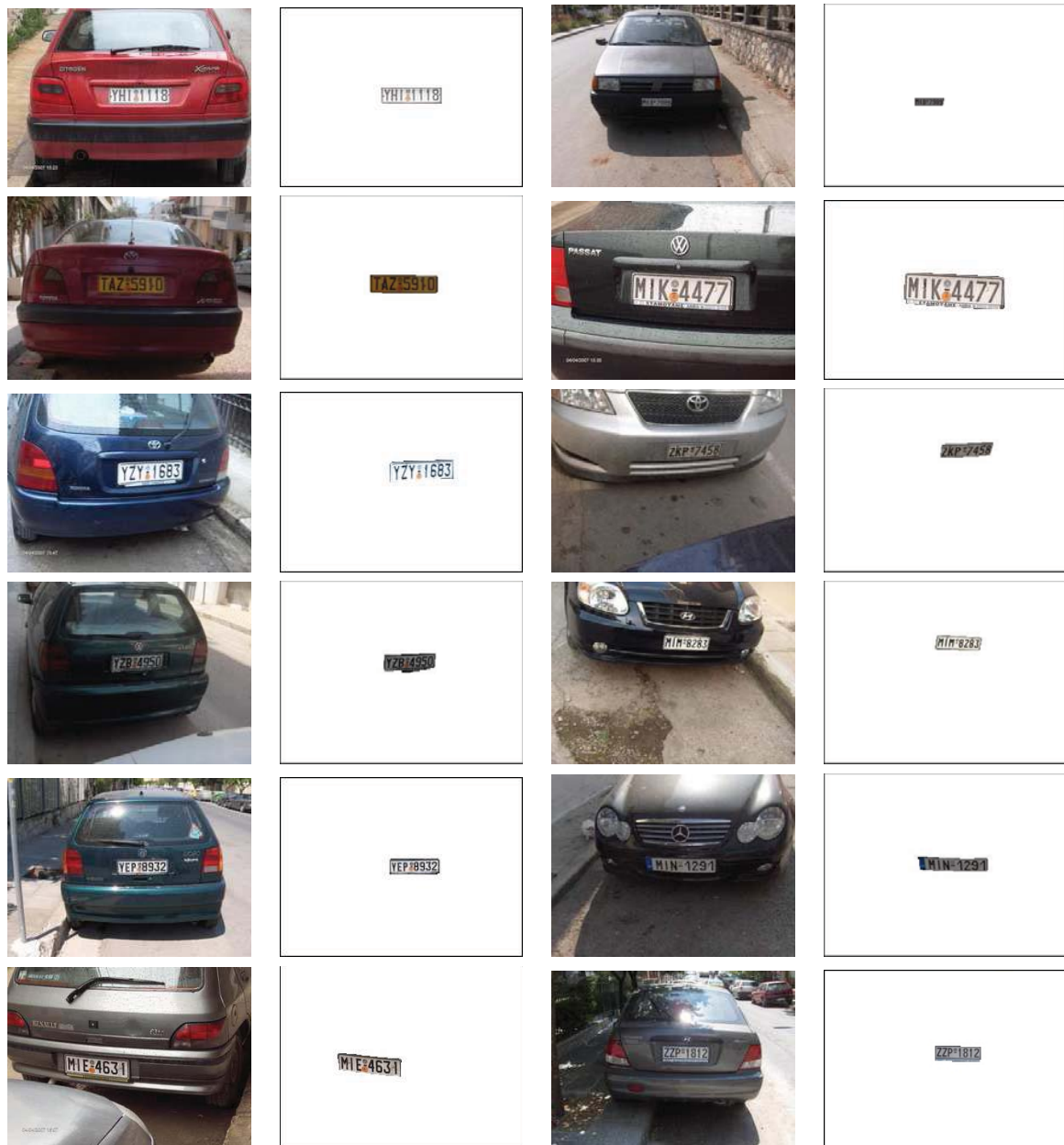


Figure 8.6: Licence plate detection results.

constraint is that the objects should be represented with only one connected component, without holes.

The main challenges in such kind of sequences are that parts of the object and background may share similar colors, the objects can have non-rigid appearance and may have complex and rapid motions, and that other moving background objects may cause errors in the segmentation if object and background are not correctly modeled.



Figure 8.7: Licence plate detection: false positive results.

We use the same tracking mechanism proposed in Section 7.3, that is, the region-based mean shift. The novelty is the final segmentation, where we replace the shape matching and partition fitting stages by an energy minimization process for object/background segmentation.

For faces, an ad-hoc procedure was used, assuming an elliptical shape model. Regions were individually analyzed and classified as face or background taking into account the best fit ellipse (see Subsection 7.3.3).

The new approach formulates the object segmentation as an energy minimization problem, for which a global solution is found through graph cut segmentation, without imposing shape constraints.

An important difference between the classic graph cut formulation typically used for segmentation and our approach is that the nodes in the graph are regions instead of pixels. The use of regions implies an important decrease in memory and time requirements. We will show that by combining the mean shift tracking with the graph cut segmentation, both based on regions, we obtain a fast and robust technique to track and accurately segment non-rigid objects in unconstrained scenarios.

In the following we describe the object/background segmentation as an energy minimization problem and review the solution through graph cuts. Next, we detail our approach in the context of object tracking, and present some experiments.

8.2.1 MRF energy formulation

The final tracking stage (Subsection 7.3.3) can be viewed as a problem of classification of the regions in the search window into one of two possible classes, object or background. This problem can be formulated in terms of energy minimization. A classical use of energy minimization is to solve the pixel-labeling problem, where the input is a set of pixels \mathcal{P} and a set of labels \mathcal{L} . The goal is to find a labeling (i.e., a mapping from \mathcal{P} to \mathcal{L}) that minimizes some energy function.

A standard form of the energy function is

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \alpha \sum_{p, q \in \mathcal{N}} V_{p, q}(f_p, f_q) \quad (8.4)$$

where $\mathcal{N} \subset \mathcal{P} \times \mathcal{P}$ is a neighborhood system on pixels. $D_p(f_p)$ is a function derived from the observed data and measures the cost of assigning the label f_p to the pixel p . $V_{p, q}(f_p, f_q)$ measures the cost of assigning the labels f_p, f_q to neighboring pixels p and q , and is used to impose spatial (boundary) smoothness. $\alpha \geq 0$ is a parameter that specifies a relative importance of the data properties terms (D_p) versus the boundary properties terms ($V_{p, q}$).

Energy functions of this form can be justified on Bayesian grounds using the Markov Random Field formulation (MRF) [167], which we briefly review below.

According to Bayes' Rule, the posterior distribution for a given set of measurements \mathbf{y} , $p(\mathbf{y}|\mathbf{x})$ combined with a priori $p(\mathbf{x})$ over the unknowns \mathbf{x} is given by

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \quad (8.5)$$

where $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) d\mathbf{x}$ can be considered a normalizing constant. Taking the negative logarithm of both sides,

$$-\log p(\mathbf{x}|\mathbf{y}) = -\log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{x}) + C \quad (8.6)$$

The maximum a posteriori (MAP) solution \mathbf{x} given the measurements \mathbf{y} is found by minimizing this negative log-likelihood, which can also be thought as an energy (dropping the C constant):

$$E(\mathbf{x}, \mathbf{y}) = E_d(\mathbf{x}, \mathbf{y}) + E_v(\mathbf{x}) \quad (8.7)$$

The term $E_d(\mathbf{x}, \mathbf{y})$ is the *data energy* or *data penalty*, and measures the negative log-likelihood that the data were observed given \mathbf{x} , and the term $E_v(\mathbf{x})$ is the *prior energy*.

In the pixel-labeling problem, the inputs are a set of pixels \mathcal{P} (the measurements \mathbf{y}) and the labels \mathcal{L} . The goal is to find a labeling f which minimizes the energy function in Eq. 8.7. The unknowns \mathbf{x} are the pixel labels.

For a MRF, the probability $p(\mathbf{x})$ is a Gibbs distribution whose negative log-likelihood can be written as a sum of pairwise interaction potentials

$$E_v(\mathbf{x}) = E_v(f) = \sum_{p, q \in \mathcal{N}} V_{p, q}(f_p, f_q) \quad (8.8)$$

where $V_{p, q}(f_p, f_q)$ measures the cost of assigning the labels f_p, f_q to neighbor pixels p and q . This smoothness energy is named the *boundary term* in [16].

The data energy or *region term*, measures the cost of assigning the label f_p to the pixel p ,

$$E_d(\mathbf{x}) = E_d(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \quad (8.9)$$

and we obtain the energy function given in Eq. 8.4.

Originally, energy-based segmentation problems were optimized using general iterative gradient descent techniques, which are extremely slow in practice [168, 167]. Over the last years, efficient optimization algorithms have been developed that find the exact minimum or local solutions in a strong sense. One of these techniques, which has demonstrated a great potential for solving many problems in vision, is graph cuts, which is based on combinatorial optimization [65, 19].

The idea is to construct a specialized graph for the energy function to be minimized, such that the minimum cut in the graph also minimizes this energy. The minimum cut, in turn, can be efficiently found using max-flow algorithms [50]. Graph cuts are successfully used for a wide variety of vision problems, including image restoration, stereo and motion, image segmentation, multi-camera scene reconstruction and medical imaging (see [168] and the references cited therein). The next section reviews the terminology used with graph cuts in the context of binary segmentation, as proposed by Boykov and Jolly in [17].

8.2.2 Graph cuts for segmentation

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph that consists of a set of nodes \mathcal{V} and a set of edges \mathcal{E} that connect them. Each edge is assigned a nonnegative cost. There are two special nodes called *terminals*, the *source* s and the *sink* t .

An s - t -cut, $C = \{S, T\}$ is a partition of the nodes in \mathcal{V} into two disjoint sets S and T , such that $s \in S$ and $t \in T$. The cost of the cut is the sum of the costs of all the edges that go from S to T .

$$c(S, T) = \sum_{u \in S, v \in T, (u, v) \in \mathcal{E}} c(u, v) \quad (8.10)$$

where $c(u, v)$ is the cost associated to the edge between u and v .

The minimum s - t -cut problem is to find a cut C with the smallest cost. This is equivalent to computing the maximum flow from the source to the sink [50].

A minimal cut can be computed in polynomial time with algorithms based on network flow such as the one proposed by Ford and Fulkerson [50] or algorithms proposed more recently in [65, 18].

In the context of segmentation, usually nodes represent pixels, and terminals correspond to the labels that can be assigned to the pixels. In [17], where graph cut is used for interactive object segmentation, the terminal nodes s and t represent ‘object’ and ‘background’ labels, respectively, and the non-terminal nodes in the graph represent image pixels $p \in \mathcal{P}$.

There are two kinds of edges; n -links that connect neighbor pixels $\{p, q\}$ (using 8 connectivity) and t -links that connect pixels with terminals. Each pixel in p has two t -links: $\{p, s\}$ and $\{p, t\}$. Each edge in \mathcal{E} is assigned some non-negative cost.

A cut can be understood as a binary partition of the graph, or a labeling f from the set of nodes $\mathcal{V} - \{s, t\}$ to $\{0, 1\}$, where $f(v) = 0$ means that $v \in S$ and $f(v) = 1$ means that $v \in T$. A cut in the graph separates the nodes in two groups, and corresponds to some segmentation of the underlying image.

A simple example of a graph for a 3×3 image is shown in Figure 8.8. The minimum cost cut determines the object boundary.

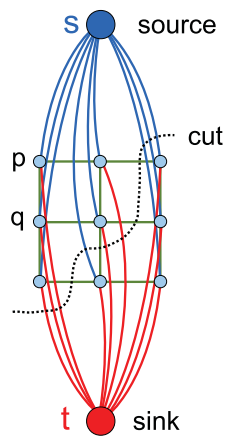


Figure 8.8: s - t graph for a 3×3 image and s - t -cut.

A minimum cost cut generates a segmentation that is optimal in terms of the properties that are built into the edge costs. The costs assigned to the edges in [17] follow the MRF formulation proposed by Greig in [65]. The data or regional term assumes that the individual penalties for assigning a pixel p to object \mathcal{O} or background \mathcal{B} reflect how the color of the pixel p fits into a given color model (e.g. histogram) of the object and background, respectively:

$$D_p(\mathcal{O}) = -\log(p(I(p)|\mathcal{O})); \quad (8.11)$$

$$D_p(\mathcal{B}) = -\log(p(I(p)|\mathcal{B})); \quad (8.12)$$

In the boundary or smoothness term, $V_{p,q}$ is interpreted as a penalty for a discontinuity between p and q . $V_{p,q}$ is large when p and q are similar and close to zero when they are very different:

$$V_{p,q} = \exp(-\beta\|I(p) - I(q)\|^2) \quad (8.13)$$

where $I(p)$ is the pixel color, and $\beta = (2\langle\|I(p) - I(q)\|^2\rangle)^{-1}$, where $\langle.\rangle$ denotes expectation over the image. This factor ensures that the exponential term switches appropriately between low and high contrast.

The edge costs reflect the data in the regional and boundary terms of the energy function (Eq. 8.4). The costs of the t -links are given by Equations 8.11 and 8.12, and the costs of the

n-links by Equation 8.13. Furthermore, the costs of t-links can also reflect any known position of some of the pixels. This idea is used in [17, 94] for interactive segmentation, where the user hard-constrains some pixels to be on the foreground or in the background. The definition of the edge weights is, in this case, different. Nodes (pixels) marked as object or background have, respectively, s-links and t-links of ‘infinite’ capacity so that these edges cannot be in the minimal cut.

8.2.3 Graph cuts in object tracking

In this section we describe how to use graph cuts in the final segmentation stage of the region-based mean shift tracking system presented in Chapter 7.

The key point is that the graph cut segmentation is region-based. That means that nodes in the graph are not pixels as in the usual formulation but regions in the partition that was built for the tracking stage. The edges connect neighboring regions (n-links) and regions with terminals (t-links). Since the number of regions is much lower than the number of pixels, this approach speeds up the classical graph cut segmentation, and also reduces its memory requirements.

Graph cuts are also used to merge partition regions in other contexts. For example, [94] presents an interactive image cutout tool, where images are pre-processed with a watershed segmentation. The user marks the boundary of the object of interest and a graph cut is performed on the small regions. The result is refined by boundary editing, this time using pixel-based graph cuts. [162] also uses an over-segmentation of the image given by a watershed, and then graph cut optimization for interactive object segmentation.

At each frame t , we work with the partition created for the tracking (see Figure 7.7.b), taking into account only regions within the search window (see Figure 7.7.c). The object and background likelihoods $p(I(p)|\mathcal{O})$ and $p(I(p)|\mathcal{B})$ are also the likelihoods computed for the tracking, where object and background histograms were obtained from the object segmented at frame $t - 1$.

The graph \mathcal{G} is the region adjacency graph (RAG) of the regions within the search window, plus the two terminal nodes and the links between terminals and regions, as illustrated in Figure 8.9.

Edges between adjacent regions should be defined so that a cut merges regions of similar characteristics. Therefore, the n-links costs should be related to the dissimilarity between regions. We first compute the mean color m_i for each region R_i , and define the cost of an n-link between regions R_i and R_j as

$$V_{R_i, R_j} = \exp\left(-\frac{\|m_i - m_j\|^2}{2\sigma^2}\right) \quad (8.14)$$

where σ^2 is the average L^2 distance between mean colors of all neighboring regions.

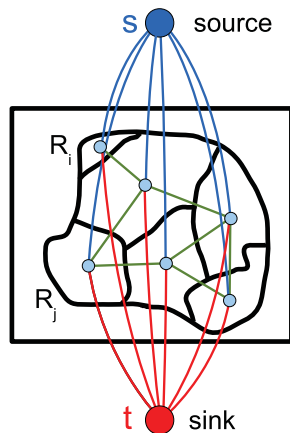


Figure 8.9: s - t region-based graph.

The costs for s -links and t -links should reflect how a region fits into one of the two classes, object or background, a measure related to the likelihood of being part of the object or the background.

We first defined the likelihoods modeling each region R_i with its mean color m_i , $p(m_i|\mathcal{O}) = h_{\mathcal{O}}(m_i)$ and $p(m_i|\mathcal{B}) = h_{\mathcal{B}}(m_i)$. However, our experiments show that much more accurate segmentations are obtained if the likelihoods are estimated for each pixel and then averaged over all the pixels in the region:

$$p(R_i|l) = \frac{1}{|R_i|} \sum_{p \in R_i} p(I(p)|l) \quad (8.15)$$

where $l \in \{\mathcal{O}, \mathcal{B}\}$. Using these likelihoods, the data terms for a region R_i are:

$$D_{R_i}(\mathcal{O}) = -\log(p(R_i|\mathcal{B})) \quad (8.16)$$

$$D_{R_i}(\mathcal{B}) = -\log(p(R_i|\mathcal{O})) \quad (8.17)$$

The edge costs reflect the data in the regional and boundary terms of the energy function, and can also reflect any known position of some of the regions. Since the object and camera may be moving, the search window may contain new colors that were not present in the object or background models in the previous frame.

We need to decide how to manage regions for which one or both likelihoods $p(R_i|\mathcal{O})$ and $p(R_i|\mathcal{B})$ are 0, and how to assign the cost of t -links in those cases. We assume that a region R_i such that $p(R_i|\mathcal{O}) = 0$ (for any value of $p(R_i|\mathcal{B})$) is a new background region, so we set the t -links accordingly. Analogously, a region such that $p(R_i|\mathcal{B}) = 0$ and $p(R_i|\mathcal{O}) > 0$ is an object region. The cost of the t -link to the source for an object region and the cost of the t -link to the sink for a background region are set to a very high value K (the ‘infinite’ capacity) to prevent those links from being part of the minimum cut. Table 8.2 gives the cost of all edges.

We use the Edmonds Karp algorithm [43], an implementation of the Ford Fulkerson method for computing the maximum flow in $\mathcal{O}(nm^2)$, where n is the number of nodes and m the number of edges.

Edge	Weight	for
$\{R_i, R_j\}$	V_{R_i, R_j}	R_i and R_j neighbor regions
	K	$R_i \in \mathcal{O}$
$\{R_i, s\}$	0	$R_i \in \mathcal{B}$ and $p(R_i \mathcal{B}) = 0$
	$\lambda D_{R_i}(\mathcal{B})$	otherwise
$\{R_i, t\}$	K	$R_i \in \mathcal{B}$
	$\lambda D_{R_i}(\mathcal{O})$	otherwise

Table 8.2: Edge costs for graph cut segmentation.

8.2.4 Experiments

The proposed algorithm has been applied to a variety of sequences to verify its performance.

The first two examples (Figures 8.10 and 8.11) correspond to Formula 1 races, in two difficult sequences where both the object and the camera move, and there are large variations in object scale and view. In both sequences the car is correctly tracked and accurately segmented. In the first example, during a few frames (around frame 87) the caption text superimposed on the image is merged with the car.

In the next example, a football match (Figure 8.12), a soccer player is segmented. In some of the frames the player legs or arms are missing mainly because the object being tracked is too small and the number of regions used in the partition is too low (in all the examples we run the tracker with the same parameters).

The fourth example is a sequence of a skier (Figure 8.13). There are some segmentation errors when the skier passes in front of an area of trees, where the man's head and the tree trunks have a similar color. However these background regions are not used to update the object model and after a few frames the silhouette of the man is correctly retrieved.

In the next sequence (Figure 8.14), the goal is to track and segment one fish that is swimming alongside another three, and they are all the same shape and color. When the fish being tracked is too close to another one, both regions are classified as objects because they agree with the object color model. Then, the system only keeps the largest connected component, and only one of the components is preserved. The result is that the tracker may jump from one fish to another.

Finally, the last examples in Figures 8.15 and 8.16, correspond to some of the sequences

used in Chapter 7 to illustrate the performance of the face tracker. We note that for sequences where object and background are quite different, both trackers perform well. However, when object and background colors are similar, the shape constraint used in the previous chapter prevents the algorithm from merging the face with the background. With the graph-cut approach, the segmentation is not so accurate in those cases.

8.3 Summary

This chapter has presented the extension of the previous face detection and tracking techniques to handle other semantic objects.

In the first part, the flexibility and potential of the region-based face detection framework detailed in Chapters 4 to 6 has been assessed. The approach has been adopted to detect objects with different characteristics: sky areas, traffic signals, and car licence plates. The same image model has been used for all the classes (see Chapter 4), whereas the object model (see Chapter 5 and Chapter 6) has been adapted to the specificities of the different classes. It has been shown that the main notions, such as the usefulness of the division on generic descriptors and specific descriptors or the concept of node extension to include the object shape information, can be successfully extended to the design of new object detectors.

In the second part of the chapter, the region-based mean shift face tracker has been extended to track other objects. In a generic case, objects may present non-rigid appearances and deformable motions that lead to complex shape evolutions. To solve this problem, a new approach has been proposed to obtain the final shape of the object. This new approach does not rely on shape priors, as it was the case of ellipses in face tracking. Here, the final shape is obtained through an energy minimization process based on a graph-cut segmentation algorithm. Experiments show that the tracker produces accurate segmentations even in difficult scenarios.

Part of the contributions presented in this chapter appear in the following papers:

- V. Vilaplana, F. Marqués, M. León, A. Gasull, *Object detection and segmentation on a hierarchical region-based image representation*, Proceedings of ICIP 2010, IEEE International Conference on Image Processing, Hong Kong, China, September 2010.
- M. León, V. Vilaplana, A. Gasull, F. Marqués, *Region-based caption text extraction*, Proceedings of WIAMIS 2010, 11th International Workshop on Image Analysis for Multimedia Application Services, Desenzano del Garda, Italy, 2010.
- M. León, V. Vilaplana, A. Gasull, F. Marqués, *Caption text extraction for indexing purposes using a hierarchical region-based image model*, Proceedings of ICIP 2009, IEEE International Conference on Image Processing, Cairo, Egypt, November 2009.

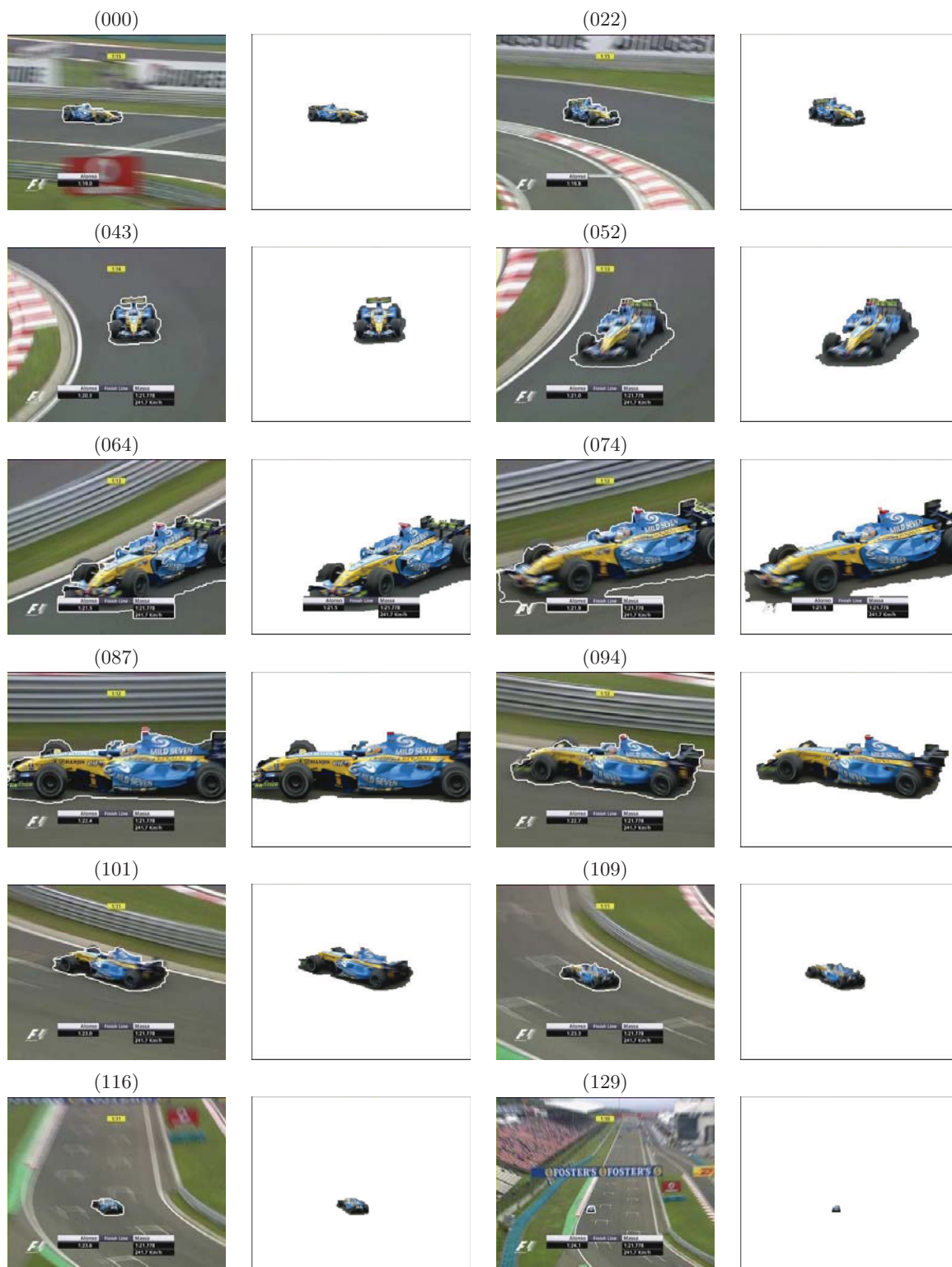


Figure 8.10: Tracking of a Formula 1 car.

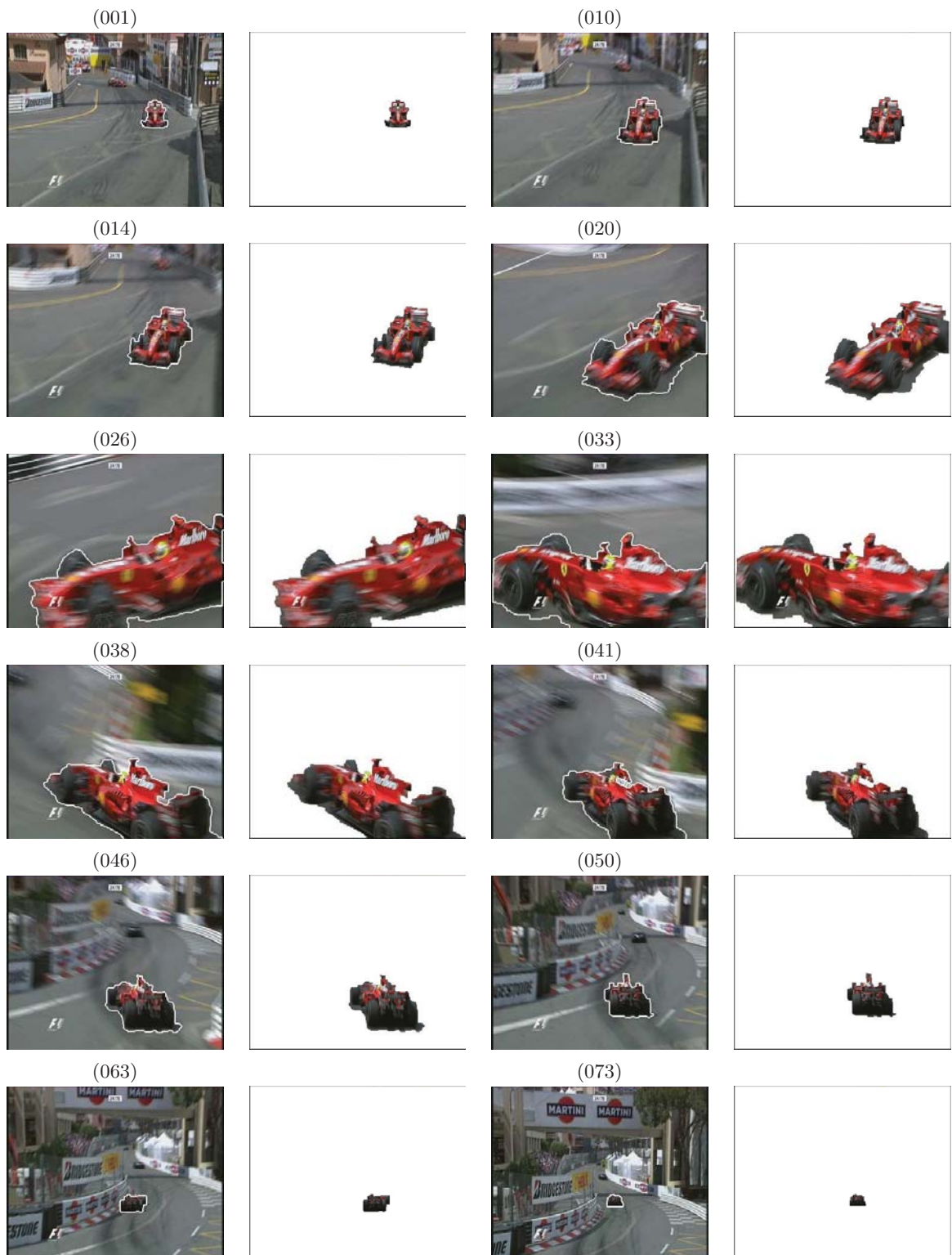


Figure 8.11: Tracking of a Formula 1 car.



Figure 8.12: Tracking of a football player.

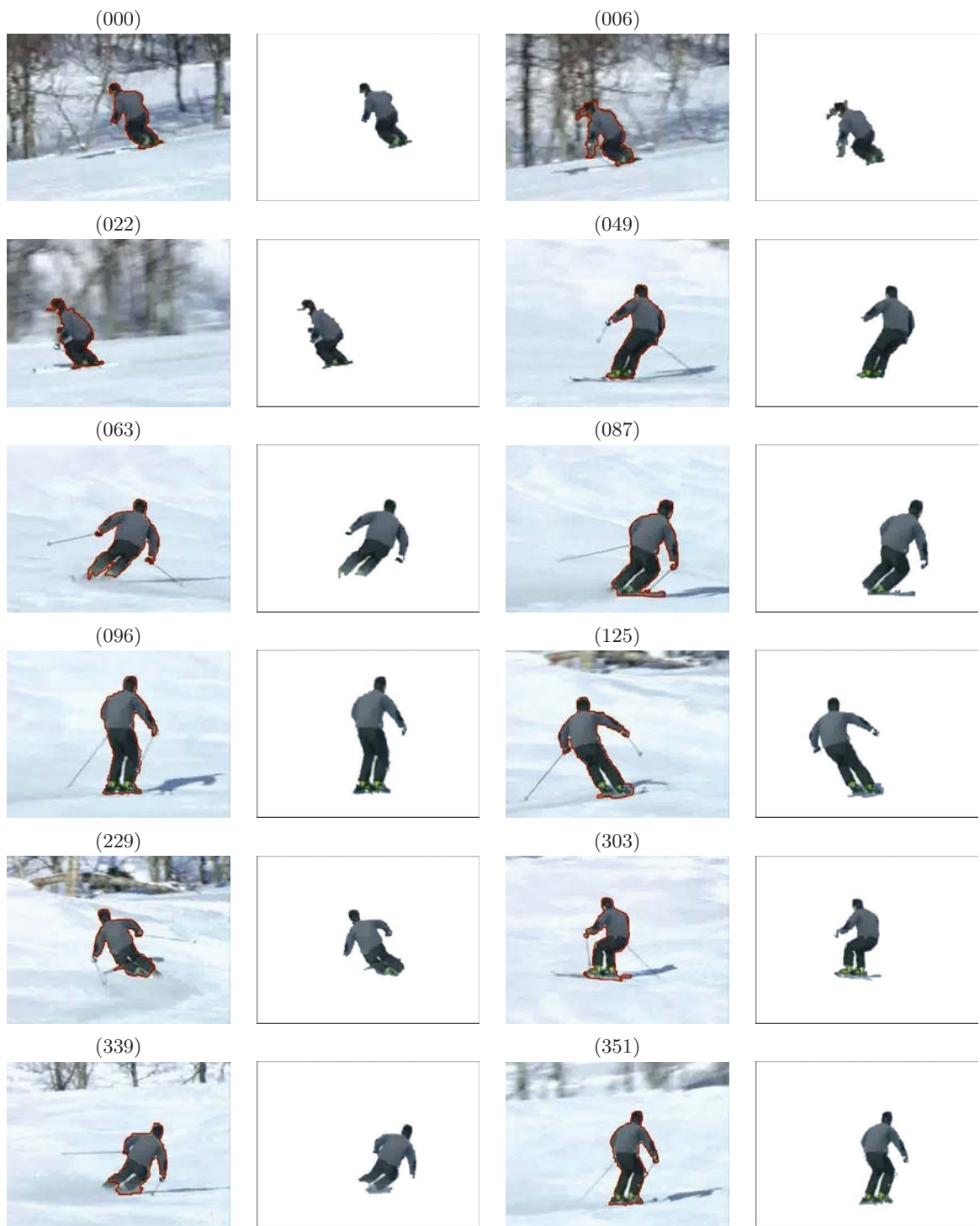


Figure 8.13: Tracking of a skier.



Figure 8.14: Tracking of a fish.

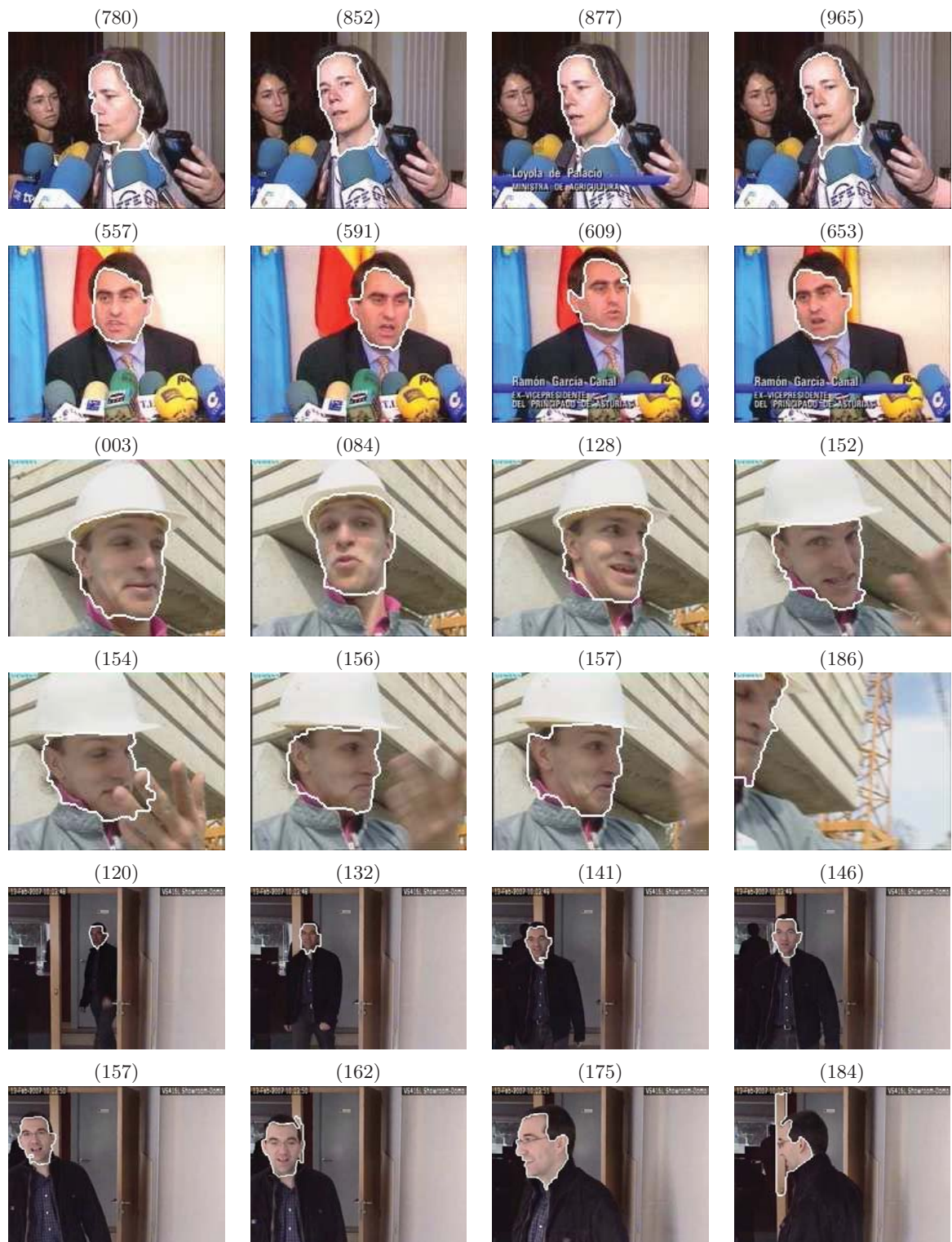


Figure 8.15: Tracking of faces. Results for ‘Minister’, ‘Foreman’ and ‘Surveillance’ sequences.

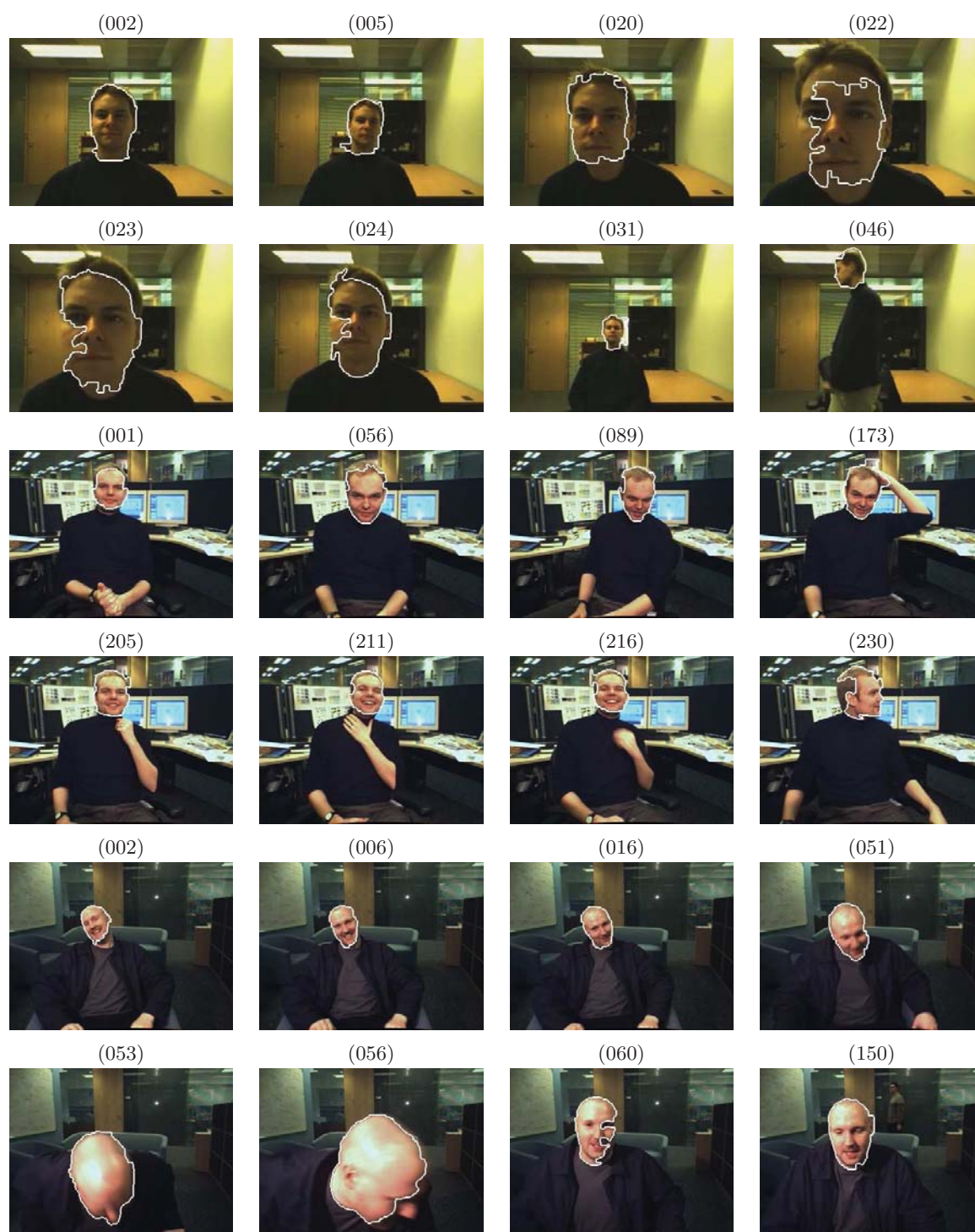


Figure 8.16: Tracking of faces. Results for ‘Jamie1r’, ‘Ms’ and ‘Jm’ sequences.

Chapter 9

Conclusions and future work

The primary objective of this thesis was to adopt a region-based approach to address the problems of face detection and tracking. In finding a solution we have tried to create a general framework, as independent as possible of the face class. This last chapter summarizes our work towards these goals. It discusses its main contributions, results and limitations, and suggests some directions for future research.

9.1 Conclusions

We have developed a technique for joint detection and segmentation of frontal or nearly frontal human faces, in color or grayscale images. The system has been extensively tested on different datasets, producing high detection rates and accurate segmentations, and proving to be quite robust to variations in face scale, position, orientation, lighting conditions and background changes. Clearly, the performance varies greatly with the quality of the sets. For high quality images like those in XM2VTS set, the detection rate reaches the 99.6% with a false positive rate of $5.10^{-5}\%$. For images with side illumination or strong shadows, like images in Banca degraded scenario, the true positive rate drops to 94% with a false positive rate of 0.19%. However, the success in the detection strongly depends on the quality of the image representation. We cannot detect any face that is not represented, at least partially, by a node in the tree.

The technique relies on a hierarchical region-based image model, the Binary Partition Tree. In this work we have studied and optimized its construction for object detection. As a compromise between computational complexity reduction and accuracy, we have defined two parts in the tree, the accuracy and the search space, which are created using a stopping criterion based on the accumulation of the merging costs. We have compared different similarity measures for the tree creation, and we have concluded that the WEDM criterion (based on color similarity) may be used to define the accuracy space, while the NWMC criterion, which

takes into account color similarity and contour complexity, is the best to create the BPT search space.

The variability of the face class is managed within a learning approach. In the operation mode, the detector is formed by three blocks: Simplification, Extension and Classification. Through a large set of experiments we have shown that each of these blocks is a useful and well performing part of the system. In the Simplification block, a cascade of binary classifiers based on generic descriptors rejects most of the tree nodes (on average, more than 95% of the nodes), greatly reducing the search space. Note that, as discussed in the text, the descriptors used in this step can be computed before the face detection process, which drastically reduces the computational load of the process. The Extension stage modifies the area of support of the node to conform to a reference shape model, improving the node representation. Finally, in the Classification stage, nodes are classified with an ensemble of diverse classifiers based on specific color, shape and texture descriptors.

We have shown that the technique proposed for faces can be easily adapted to deal with other object classes. New object models can be built by selecting and training an appropriate set of descriptors and classifiers. The usefulness of the approach has been illustrated with three different classes: sky, traffic signs and licence plates. In these cases the analysis was not so detailed as it was for faces, partially due to the lack of suitable datasets. Few descriptors were used in the object models, which were trained and tested on small datasets, with excellent results. For some of these objects and also for faces in the case of difficult images (like images in the Banca adverse set), the node extension is a fundamental stage to obtain good object representations. The proposed approach, however, has its limitations. It is suitable only for object classes where the core part of the object can be represented by a region, specifically by a node in the BPT. The detection of other types of objects may require a different approach.

Finally, we have addressed the face tracking problem with two different strategies. The first one is a unified detection and tracking mechanism based on the use of the BPT. The second one combines the efficiency of the mean shift algorithm for tracking with the use of regions to segment the faces through the sequence. Here, the final segmented face is obtained through a shape matching and partition fitting stage, assuming an elliptical shape model. This second system has been adapted to deal with other deformable objects, replacing the last stage by a region-based graph-cut method for object/background segmentation. Experiments show that both mean-shift based trackers produce accurate segmentations in difficult scenarios.

9.2 Future work

In this section we provide a brief summary of open issues and future research directions.

The face detector has a good performance and is robust to different sources of variability. However it is limited to detect frontal or nearly frontal faces. We have observed that the

performance of the Haar classifier, which is one of the best performing single classifiers, decreases rapidly as the face moves away from the frontal position. In order to deal with those faces, we could train a new version of this classifier using sets of faces in half profile poses (left and right). We could go further and extend the system to detect faces in different views. This could be done training versions of the specific texture-based classifiers (SVDD, DMAH, and Gentle AdaBoost) separately for profile, half profile and frontal views (both sides), building different ensembles and implementing a mechanism to arbitrate between the multiple ensembles.

The image model is a crucial part of the system. If the region boundaries do not correspond to object boundaries (the search partition creation fails), or if the main part of the object is not represented by a node in the BPT (the search space creation fails), the detection fails. Therefore, any improvement in the image representation will translate into an improvement of the detector.

As commented in Chapter 8, the proposed technique is suitable only for certain type of object classes, mainly compact or fairly homogeneous in color, since the search is constrained to the tree nodes. But the object model could be improved with new descriptors and classifiers. In particular, we think that the inclusion of descriptors for salient points on regions could be useful to deal with other types of objects.

Concerning the region-based mean shift trackers, there are several interesting issues related to the use of regions that could be analyzed. First, we could study the influence of the number and size of regions on the convergence, accuracy and time performance of the tracking algorithm (the mean shift). Second, we could analyze the impact of the number of regions in the performance of the graph-cut based segmentation. Regarding this second tracker, we have observed that its performance on some of the face sequences (the most difficult ones, those with similar face and background colors and fast movements) is worse than the performance of the first version, which used an ellipse to segment the final face. An interesting solution could be the combination of both strategies, by the inclusion of shape priors on some of the edge weights (a shape term in the energy function) in the graph.

Finally, as said in the introduction, face detection and segmentation are necessary steps in a large set of applications and services. Therefore, the use of the face detector for applications requiring the accurate segmentation of the face (such as face identification or videogames) or its precise location (such as applications based on human interaction) can be also studied.

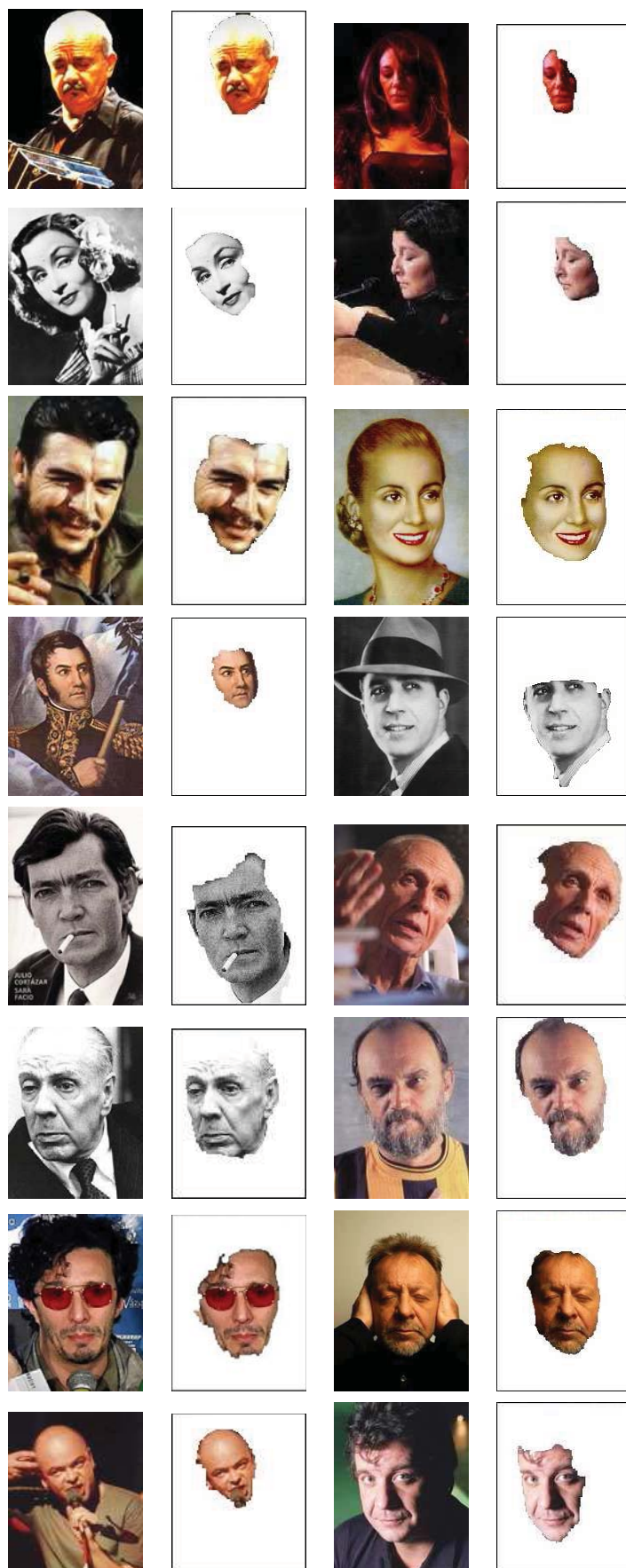


Figure 9.1: Segmentation results for faces presented in Figure 1.4.

Appendix A

Data sets

This appendix describes the face data sets used for all the experiments presented in this work, from the controlled sets used mainly for training, to the more challenging ones used to evaluate the performance of the different classifiers and the complete system.

A.1 ORL

The ORL database [6] from AT&T Cambridge Laboratories (formerly known as the Olivetti Research Laboratory database) consists of 10 different gray-level face images from 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). We use this set to train the texture-based classifiers.

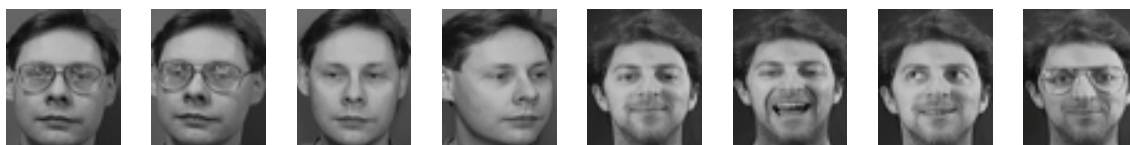


Figure A.1: Image samples from ORL.

A.2 BioID

The BioID dataset [81] consists of 1521 gray level images. Each one shows the frontal view of a face of one out of 23 different persons. Images have been acquired under a large variety of

illumination, background and face sizes. Some typical sample images are shown in Figure A.2. This set is used to test the gray-level version of the face detector.



Figure A.2: Image samples from BioID.

A.3 XM2VTS

XM2VTS frontal face database [109] was acquired within the Extended M2VTS project (Multi Modal Verification for Tele-services and Security applications). It contains 1180 color images of frontal faces taken from the video recordings made during the acquisition of the database. A total of 295 subjects were recorded in 4 different sessions at approximately one month intervals. Images have been acquired under quite controlled conditions, having almost uniform background, with slight variations in illumination, no occlusions, faces with a 'neutral' expression. We use this set both for training and test.



Figure A.3: Image samples from XM2VTS.

A.4 Banca

BANCA [9] is a multi-modal database intended for training and testing multi-modal identity verification systems. The database was captured in four European languages (Spanish, French, Italian and English) in two modalities (face and voice). For each language, video and speech data were collected for 52 subjects (26 females and 26 males), i.e. a total of different 208 subjects. Each subject recorded 12 sessions separated into three different scenarios: controlled, adverse and degraded.

A cheap analogue web cam was used in the degraded scenario, while a high quality digital camera was used in the controlled and adverse scenarios. In the controlled scenario images have a uniform background and illumination is controlled and in the adverse scenario images have a cluttered background and different lighting conditions. Lighting conditions are also challenging in the degraded scenario. Figure A.4 shows some examples from the three scenarios.

The face data consists of 10 images taken for each subject in each session, so there is a total of $120 \times 52 = 6240$ images in each language set. Banca evaluation protocol (intended mainly for identity verification) divides each language and gender specific population into 2 groups of 13 subjects, denoted as g1 and g2, to facilitate the definition of development and test sets in evaluation experiments. We work with g1 set for training and evaluating classifiers and images from the g2 set for testing.

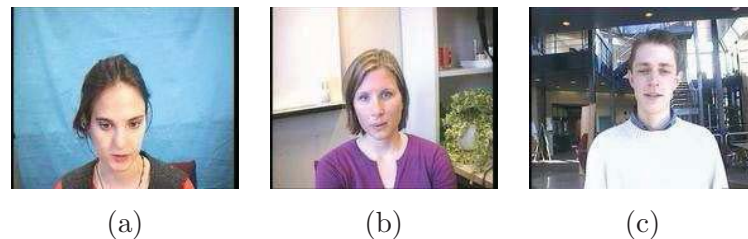


Figure A.4: Image samples from Banca controlled (a), degraded (b) and adverse (c) scenarios.

A.5 MPEG-7

A set of images containing one or more faces was selected from different videos from the MPEG-7 test set [103]. The data set contains 1000 images with cluttered background, faces with different expressions, pose and lighting effects and was used to test individual classifiers and the complete system. Some example images are shown in Figure A.5.



Figure A.5: Image samples from MPEG-7 set.

Appendix B

The image model: Similarity measures and stopping criteria

This appendix completes the analysis of the merging and stopping criteria proposed in Chapter 4 for the creation of the image model. For the sake of completeness, we repeat the description of some of the experiments.

B.1 Definition of the merging criterion for the accuracy area

Experiment 1: To quantify the performance of the similarity measures, different merging criteria are compared on the set of 100 images from the Corel database (see Table B.1). In this experiment, to decouple the effects of the merging and stopping criteria (that is, of the creation of the accuracy area and the search partition), a simple stopping criterion is used: merge up to 50 regions. The comparison is performed in terms of mean values of the final PSNR (between the image created by filling each region with its model and the original image) and of the variance of the region sizes.

	MSE	SE	WSDM	WEDM	NWMC
PSNR	22,99	24,72	25,33	24,87	20,16
σ_{reg}^2 on region size	16,90	2,86	4,74	1,51	2,18

Table B.1: Merging criteria comparison on the Corel subset. PSNR values are given in dBs. σ_{reg}^2 values are divided by 10^5 .

In terms of PSNR, the WEDM criterion improves all criteria except the WSDM which is only slightly better. However, these results for the WEDM are obtained while largely outperforming the other criteria in terms of variance of the region sizes. As previously com-

mented, this is a relevant feature since it ensures that regions obtained with the WEDM will be adequate for a subsequent robust feature estimation: large enough and homogenous in size while presenting similar PSNR values than previous measures and leading to visually good representation. Regarding the NWMC criterion, note that it yields the worst results in terms of PSNR. This is due to the fact that this criterion tends to create regions with smooth contours which are not common in the lower scales of the image representation.

Experiment 2: In this experiment, the quality of the partitions obtained under the conditions of Experiment 1 is assessed in terms of their accuracy. We use the 160 objects manually segmented from the Corel database subset. The asymmetric distances between the partitions obtained with each of the five merging criteria (using as stopping criterion $N_{reg} = 50$ regions) and the object partition (ground truth) are computed for each object. Figure B.1 shows the difference between the asymmetric distance values obtained using the four previous merging criteria and the WEDM merging criterion for each one of the 160 objects of the Corel database subset. Statistics of each merging criterion are presented in table B.2. Note that, in this case, the global behaviors of the SE and WEDM criteria are very similar, outperforming those of MSE, WSDM and NWMC criteria.

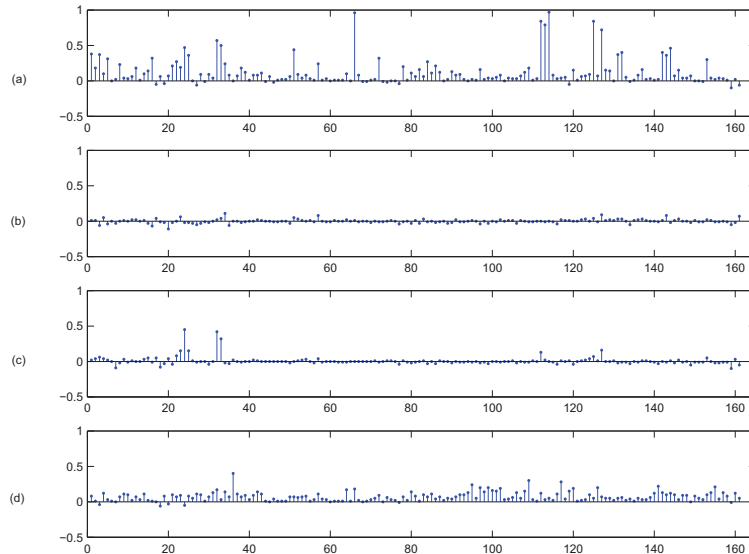


Figure B.1: Difference between the asymmetric distance values computed for different merging criteria on the COREL subset. (a) MSE - WEDM. (b) SE - WEDM. (c) WSDM - WEDM. (d) NWMC - WEDM.

B.2 Definition of the search partition: Stopping criterion

A new stopping criterion that takes into account the scene complexity is proposed. The criterion is defined as a fraction $T_{AMC} \in [0, 1]$ of the total accumulated merging cost AMC.

Asymmetric distance	MSE	SE	WSDM	WEDM	NWMC
Mean	22.57	10.52	11.46	10.53	17.73
σ^2	5.549	1.012	1.880	1.038	1.610

Table B.2: Merging criteria comparison on the Corel subset. Asymmetric distance mean and variance values are multiplied by 10^2 .

This criterion stops the merging process at iteration $\bar{m} = \min\{m/AMC(m) > AMC(N - 1)T_{AMC}\}$, where the AMC is $AMC(m) = \sum_{k=1}^m O(k)$ and $O(k)$ is the cost of the merging at iteration k ; that is, the similarity measure between the regions merged at iteration k .

The influence of the T_{AMC} in terms of mean values of the asymmetric distance, PSNR and number of regions of the partitions obtained with the Corel subset is presented in Figure B.2. Based on these plots, a value of $T_{AMC} = 0.12$ is selected for the experiments performed in the remainder of this thesis, since it leads to a low asymmetric distance and high enough PSNR values, while producing a reasonably small number of regions.

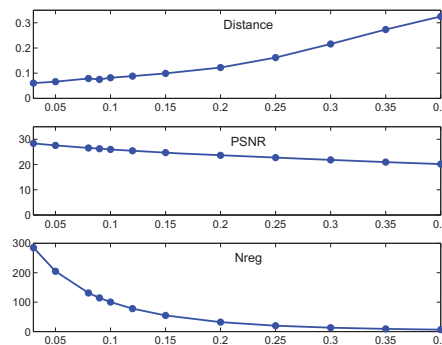


Figure B.2: Analysis of the T_{AMC} impact in terms of mean values of the asymmetric distance, PSNR and number of regions of the partitions obtained with the Corel database subset.

Experiment 3: The proposed AMC stopping criterion is quantitatively assessed, as in subsection B.1, using the Corel subset and the asymmetric distance. In this case, the similarity measure is the WEDM and we compare as stopping criteria a fixed N_{reg} , a fixed PSNR and the proposed AMC. The used N_{reg} and PSNR values are the mean values obtained by the AMC criteria over the Corel database subset ($N_{reg} = 77$ and PSNR = 25.54 dB).

Figure B.3 shows the difference between the asymmetric distance values obtained using the two previous stopping criteria with respect to the AMC stopping criterion for each one of the Corel 160 objects. Notice that, for complex images, the AMC criterion outperforms the N_{reg} criterion. This is the case of the classes *tigers* and *horses* where the amount of

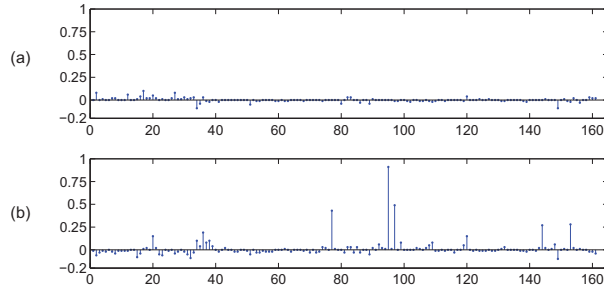


Figure B.3: Difference between the asymmetric distance values. (a) N_{reg} -AMC. (b) PSNR-AMC.

regions is too low producing undersegmented results that lead to higher asymmetric distances (see examples in Figures 4.6.a and 4.6.d). In turn, for simple images, the AMC criterion outperforms the PSNR criterion. This is the case of the classes *eagles* and *jets* where the selected PSNR is too low, producing undersegmented results that lead to higher asymmetric distances (see examples in Figures 4.6.c and 4.6.e).

The statistics of each stopping criterion are presented in Table B.3. It can be concluded that the AMC stopping criterion (computed over the WEDM merging criterion) outperforms the PSNR criterion, while slightly improving the results obtained with the N_{reg} criterion.

As it has been shown, the behavior of the AMC criterion can be, in several cases, approximated by the N_{reg} criterion. This is the reason why we have proposed to define the accuracy partition by a fixed (and rather large) number of regions. In turn, the ACM criterion is computed starting from this accuracy partition. The accuracy partition represents the highest resolution in the hierarchy and, therefore, should ensure a sufficient definition in the scene representation. We analyze this point in the next experiment.

	Nreg	PSNR	AMC
Mean	8.89	10.37	8.81
σ^2	0.66	1.70	0.57

Table B.3: Stopping criteria comparison on the Corel subset. Asymmetric distance mean and variance values are multiplied by 10^2 .

Experiment 4: The accuracy partition is assessed by segmenting the Corel database subset, using the WEDM merging criterion and the N_{reg} stopping criterion. Figure B.4 presents, for different values of the final number of regions, the mean values of the PSNR corresponding to the accuracy partitions and of their asymmetric distance with respect to the Corel database subset. As it can be seen, the evolution of both parameters is smooth and

steady. For the selected number of regions (500), the obtained values are 31dB and 0.057, respectively, which, as it is shown through the various experiments in this paper, represent a good compromise.

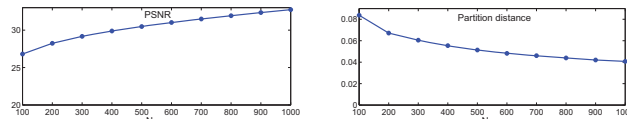


Figure B.4: Analysis of the impact of the number of regions in the accuracy partition.

B.3 Definition of the search space of the BPT

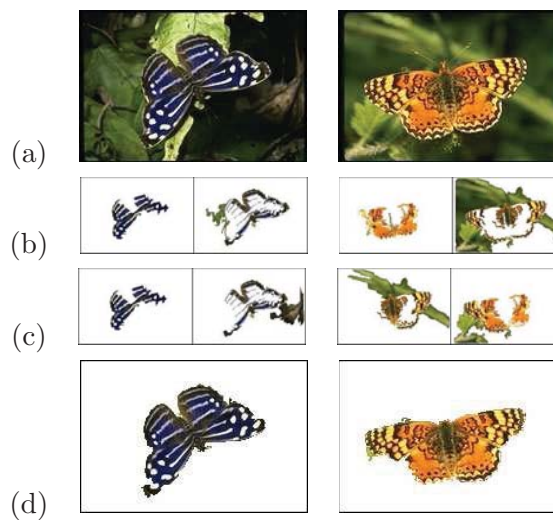


Figure B.5: Face nodes obtained with a criterion based on equally weighted components: row (a) Original images, row (b) WEDM, row (c) a criterion based on normalized components and row (d) a combination of color and contour complexity: NWMC.

More examples of the improvement achieved with the NWMC criterion are presented in Figure B.5. Row (a) in Figure B.5 shows the original images. In turn, row (b) presents the nodes that represent object regions for the original WEDM criterion (eq.4.5), row (c) those nodes obtained by the criterion based on normalized color but without the contour complexity term (eq.4.6), and row (d) those nodes obtained by the proposed NWMC criterion (eq.4.8), respectively. As it can be seen, for this type of semantic objects, the rationale behind the definition of the new criterion works perfectly: the use of a normalized color term already improves the quality of the created nodes (row (c)) which is further improved by the inclusion of the contour complexity term (row (d)).

Symmetric distance	WSDM	WEDM	NWMC
Mean	24.62	25.65	17.77
σ^2	4.45	4.30	2.55

Table B.4: Symmetric distance over the BPT on the Corel subset. Mean and variance values are multiplied by 10^2 .

Experiment 5: The behavior illustrated in the previous figures is further analyzed in this experiment by using the Corel database subset. For the whole subset, the node in the BPT leading to the smallest symmetric distance with respect to each manually segmented object has been selected. Statistics of the symmetric distances achieved with the nodes selected from BPTs created using the WSDM, WEDM and NWMC merging criteria are presented in table B.4. Note that, in the case of more generic objects as those represented in the Corel subset (not all of them homogeneous in color), the NWMC also outperforms both the WSDM and the WEDM criteria.

Symmetric distance	WSDM	WEDM	NWMC
Mean	44.28	41.87	27.16
σ^2	3.55	3.56	2.75

Table B.5: Symmetric distance over the BPT on the MPEG-7 subset. Mean and variance values are multiplied by 10^2 .

Experiment 6: The next experiment compares the use of the WSDM, WEDM and NWMC for the creation of the search space of the BPT with the MPEG-7 face database subset. As previously, for the whole subset, the node in the BPT leading to the smallest symmetric distance with respect to each manually segmented human face has been selected. Statistics of the symmetric distances achieved with the nodes selected from BPTs created using the WSDM, WEDM and NWMC merging criteria are presented in table B.5. As it can be seen, results confirm the improvements introduced by the NWMC criterion. Nevertheless, it has to be noticed that the symmetric distances achieved with the MPEG-7 database are larger (almost the double) than those obtained with the Corel database. This is due to the further difficulty of the MPEG-7 database scenarios, in which all images are complex and several ones presenting illumination problems.

Experiment 7: In this experiment, the sensitivity of NWMC with respect to the value of the parameter α is analyzed. We use the 160 objects manually segmented from the Corel database subset. The asymmetric distances between the partitions obtained with each

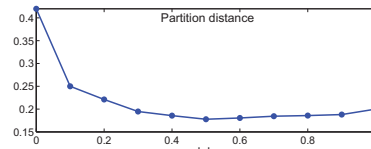


Figure B.6: Mean value of the asymmetric distance computed on the Corel database subset for different α values.

of the different α values (using as stopping criterion $N_{reg} = 50$ regions) and the object partition (ground truth) are computed for each object. Figure B.6 shows the mean value of the asymmetric distance computed over the object database. As it was previously commented, the sensitivity of the segmentation process is very low with respect to variations of the alpha parameter around $\alpha = 0.5$ and, therefore, this value is used.

Appendix C

Texture classifiers

C.1 Probabilistic Visual Learning

The visual learning method proposed in [111] estimates the complete probability distribution of the object's appearance using an eigenvector decomposition of the image space.

The class membership of an image i_x is modeled as a unimodal Gaussian density

$$P(\mathbf{x}/\Omega) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \bar{\mathbf{x}})\right]}{(2\pi)^{N/2} |\boldsymbol{\Sigma}|^{1/2}} \quad (\text{C.1})$$

where \mathbf{x} is the N-dimensional vector made up from the raw by raw reading of image i_x . The mean $\bar{\mathbf{x}}$ and the covariance matrix $\boldsymbol{\Sigma}$ are estimated using a training data set of face images.

The *Mahalanobis* distance $d(\mathbf{x}) = \tilde{\mathbf{x}}^T \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{x}}$ is a sufficient statistic for characterizing this likelihood, where $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ is the mean normalized image. Using the eigenvector and eigenvalue decomposition of $\boldsymbol{\Sigma}$, this distance can be written as

$$d(\mathbf{x}) = \tilde{\mathbf{x}}^T \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^T [\boldsymbol{\Phi} \boldsymbol{\Lambda}^{-1} \boldsymbol{\Phi}^T]^{-1} \tilde{\mathbf{x}} = \mathbf{y}^T \boldsymbol{\Lambda}^{-1} \mathbf{y} = \sum_{i=1}^N \frac{y_i^2}{\lambda_i} \quad (\text{C.2})$$

where $\mathbf{y} = \boldsymbol{\Phi}^T \tilde{\mathbf{x}}$ are the new variables obtained by the change of coordinates in a KLT, and λ_i are the eigenvalues. In the KLT basis the Mahalanobis distance is decoupled into a weighted sum of uncorrelated component energies. Evaluation of (C.2) is still computationally expensive due to the high dimensionality of the data. However, an estimate of $d(\mathbf{x})$ can be obtained based on the first M principal components

$$\hat{d}(\mathbf{x}) = \sum_{i=1}^M \frac{y_i^2}{\lambda_i} + \frac{1}{\rho} \epsilon^2(\mathbf{x}) \quad (\text{C.3})$$

where $\epsilon^2(\mathbf{x})$, the residual reconstruction error, can be computed from the first M principal components, the M principal eigenvalues and the norm of the mean normalized image $\tilde{\mathbf{x}}$

$$\epsilon^2(\mathbf{x}) = \sum_{i=M+1}^N y_i^2 = \|\tilde{\mathbf{x}}\|^2 - \sum_{i=1}^M y_i^2 \quad (\text{C.4})$$

In this way, the likelihood estimate based on $\hat{d}(\mathbf{x})$ may be written as the product of two marginal and independent Gaussian densities,

$$\begin{aligned}\hat{P}(\mathbf{x}/\Omega) &= \left[\frac{\exp\left[-\frac{1}{2}\sum_{i=1}^M \frac{y_i^2}{\lambda_i}\right]}{(2\pi)^{M/2} \prod_{i=1}^M \lambda_i^{1/2}} \right] \left[\frac{\exp\left[-\frac{\epsilon^2(\mathbf{x})}{2\rho}\right]}{(2\pi\rho)^{(N-M)/2}} \right] \\ &= P_F(\mathbf{x}/\Omega)\hat{P}_{\bar{F}}(\mathbf{x}/\Omega)\end{aligned}\quad (\text{C.5})$$

where $P_F(\mathbf{x}/\Omega)$ is the true marginal density in the principal subspace F generated by the M first eigenvectors and $\hat{P}_{\bar{F}}(\mathbf{x}/\Omega)$ is the estimated marginal density in the orthogonal complement \bar{F} .

The optimal value of ρ is determined by minimizing a cost function, the Kullback-Leibler divergence between the true density $P(\mathbf{x}/\Omega)$ and its estimate $\hat{P}(\mathbf{x}/\Omega)$, which yields

$$\rho = \frac{1}{N-M} \sum_{i=M+1}^N \lambda_i \quad (\text{C.6})$$

and therefore ρ is the average of the $N-M$ remaining eigenvalues.

C.2 Support Vector Data Description

Support Vector Data Description was developed by Tax and Duin [172] to solve one-class classification problems.

Inspired by the Support Vector Machine learning theory, SVDD obtains a boundary around the target data set; this boundary is used to decide whether new objects are target objects or outliers.

Given a set of training target data $\{x_i\}, i = 1, \dots, N$, the simplest form of SVDD defines a hypersphere around the data. The sphere is characterized by a center a and a radius R . The goal is to minimize the volume of the sphere -minimize R^2 - keeping all the training objects inside its boundary.

The structural error to minimize is:

$$F(R, a) = R^2 \quad (\text{C.7})$$

with the constraints:

$$\|x_i - a\|^2 \leq R^2, \quad \forall i \quad (\text{C.8})$$

To allow outliers in the training set, the constraints are relaxed by introducing slack variables ξ_i ; the minimization problem turns into:

$$F(R, a) = R^2 + C \sum_i \xi_i \quad (\text{C.9})$$

with the constraints:

$$\|x_i - a\|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0 \quad \forall i \quad (\text{C.10})$$

The parameter C controls the tradeoff between errors and the volume of the description.

By introducing Lagrange multipliers α_i and γ_i , the following Lagrangian is obtained:

$$\begin{aligned} L(R, a, \xi, \alpha, \gamma) = & R^2 + C \sum_i \xi_i - \\ & - \sum_i \alpha_i (R^2 + \xi - (x_i \cdot x_i - 2a \cdot x_i + a \cdot a)) - \sum_i \gamma_i \xi_i \end{aligned} \quad (\text{C.11})$$

L has to be minimized with respect to R , a and ξ_i and maximized with respect to α_i and γ_i . Solving the partial derivatives of L the following constraints are found:

$$\sum_i \alpha_i = 1 \quad (\text{C.12})$$

$$a = \sum_i \alpha_i x_i \quad (\text{C.13})$$

$$0 \leq \alpha_i \leq C \quad (\text{C.14})$$

Replacing equations (C.12-C.14) into equation (C.11), the Lagrangian is:

$$L = \sum_i \alpha_i (x_i \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \quad (\text{C.15})$$

The maximization of (C.15) gives a set $\{\alpha_i\}$. Objects x_i with $\alpha_i > 0$ are called the *support vectors* (SV) of the description. Support vectors lie on the boundary (if $0 < \alpha_i < C$) or outside the boundary (if $\alpha_i = C$) of the sphere that contains the data. Equation (C.13) shows that the center of the sphere is a linear combination of the support vectors.

A new object z is accepted as a target object if it is inside the description. Hence, the following condition has to be verified:

$$\|z - a\|^2 = (z \cdot z) - 2 \sum_i \alpha_i (z \cdot x_i) + \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \leq R^2 \quad (\text{C.16})$$

This formulation of SVDD can be extended to obtain a more flexible description. Data is mapped nonlinearly into a higher dimensional space where a hyperspherical description can be found. The mapping is performed implicitly, replacing the inner products in (C.15) by a kernel function:

$$K(x_i, x_j) = (\Phi(x_i) \cdot \Phi(x_j)) \quad (\text{C.17})$$

Several kernel functions have been proposed in [172]. In our experiments we use a Gaussian kernel of the form:

$$K(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{s^2}\right) \quad (\text{C.18})$$

This kernel is independent of the position of the data with respect to the origin since only uses distances between objects. The parameter s is a width parameter that controls how tight the description is around the data.

Using this kernel, a new object z is accepted if:

$$\sum_i \alpha_i \exp\left(\frac{-\|z - x_i\|^2}{s^2}\right) \geq \frac{1}{2}(-R^2 + B) \quad (\text{C.19})$$

where $B = 1 + \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j)$ depends only on the support vectors x_i . This function is a threshold on a weighted sum of Gaussians, so the boundary description is strongly influenced by the width parameter s . For small values of s all the objects tend to be support vectors. The description approximates a Parzen density estimation with a small width parameter. For large values of s , the SVDD description approximates the original spherical solution. For intermediate values of s , a weighted Parzen density is obtained.

C.3 Boosting Algorithms

Boosting is a procedure that combines or boosts the performance of many ‘weak’ classifiers to produce a powerful classifier [52].

Basic AdaBoost

This is the most commonly used version of the AdaBoost procedure. The training data is $(x_1, y_1), \dots, (x_N, y_N)$, with x_i a vector value feature and $y_i = -1$ or 1 . Defining $F(x) = \sum_1^M c_m f_m(x)$, where each $f_m(x)$ is a classifier producing values ± 1 , and c_m are constants, the corresponding prediction is $\text{sign}(F(x))$. The Adaboost procedure trains the classifiers $f_m(x)$ on weighted versions of the training samples, giving higher weights to cases that are currently misclassified. This is done for a sequence of weighted samples, and then the final classifier is defined to be a linear combination of the classifiers for each stage. E_w is the expectation with respect to the weights $w = (w_1, w_2, \dots, w_n)$. At each iteration AdaBoost increases the weights of the observations misclassified by $f_m(x)$ by a factor that depends on the weighted training error.

1. Start with weights $w_i = 1/N$, $i = 1, 2, \dots, N$.
2. Repeat for $m = 1, 2, \dots, M$:
 - (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data
 - (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$
 - (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(y \neq f_m(x))}]$, $i = 1, 2, \dots, N$, and renormalize so that $\sum_i w_i = 1$
3. Output the classifier $\text{sign}[\sum_{m=1}^M c_m f_m(x)]$

Real AdaBoost

A generalized version of AdaBoost is Real AdaBoost, in which the weak learner returns a class probability estimate $p_m(x) = \hat{P}_\omega(y = 1|x) \in [0, 1]$. The contribution to the final classifier is half the logit-transform of this probability estimate.

1. Start with weights $w_i = 1/N$, $i = 1, 2, \dots, N$.
2. Repeat for $m = 1, 2, \dots, M$:
 - (a) Fit the classifiers to obtain a class probability estimate $p_m(x) = \hat{P}_\omega(y = 1|x) \in [0, 1]$, using weights w_i on the training data
 - (b) Set $f_m(x) \leftarrow \frac{1}{2} \log \frac{p_m(x)}{1-p_m(x)} \in (R)$
 - (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, $i = 1, 2, \dots, N$, and renormalize so that $\sum_i w_i = 1$
3. Output the classifier $\text{sign}[\sum_{m=1}^M f_m(x)]$

Gentle AdaBoost

Gentle AdaBoost is a ‘gentler’ version, that works with adapting Newton steps instead of the exact optimization used by Real AdaBoost to fit the regression model:

1. Start with weights $w_i = 1/N$, $i = 1, 2, \dots, N$, $F(x) = 0$.
2. Repeat for $m = 1, 2, \dots, M$:
 - (a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i , with weights w_i .
 - (b) Update $F(x) \leftarrow F(x) + f_m(x)$
 - (c) Update $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$, $i = 1, 2, \dots, N$, and renormalize so that $\sum_i w_i = 1$
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^M f_m(x)]$

Appendix D

Fusion of classifiers

Figure D.1 shows the ROC curves obtained for the nine combination rules proposed in Section 5.4.3, using four normalization criteria: min-max (*Minmax*), robust min-max (*Rminmax*), double sigmoid (*Bisig*) and logistic regression (*LogReg*). The logistic regression combiner is used with original data, without normalization.

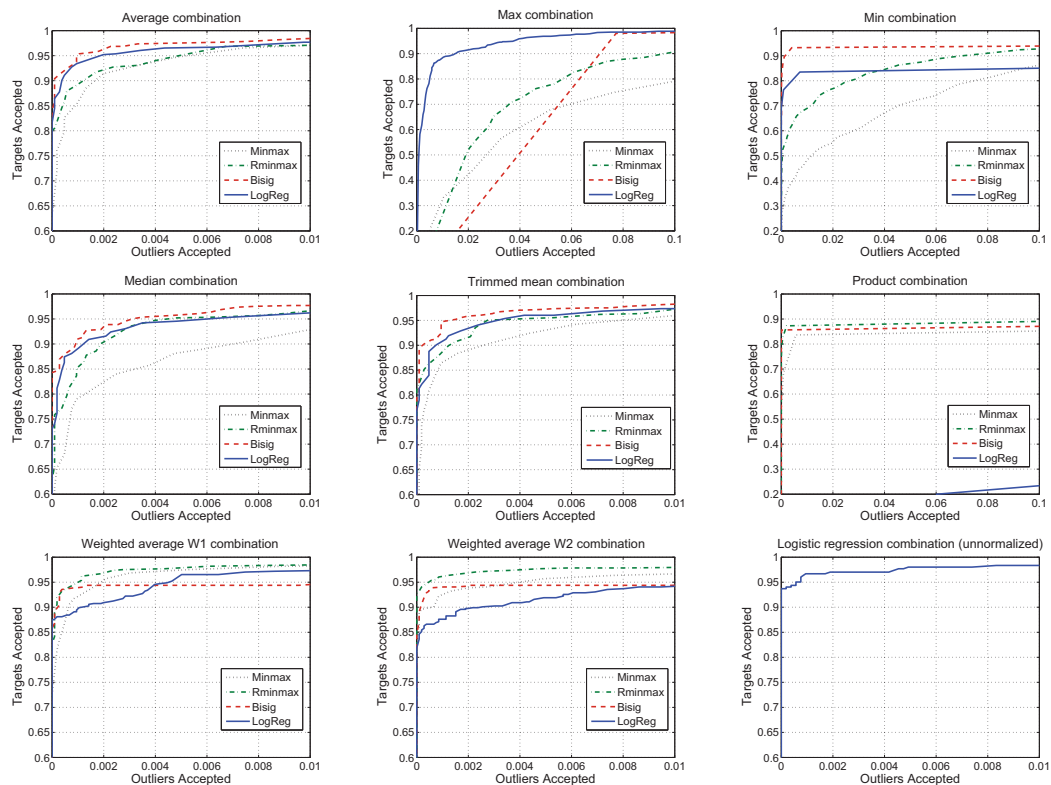


Figure D.1: ROCs of fusion methods: average, max, min, median, trimmed mean, product, weighted average W1, weighted average W2 and logistic regression.

Appendix E

Node extension for object detection

In this appendix we illustrate the usefulness of the node extension stage for the detection of objects with different characteristics: traffic signals, butterflies and cars.

Two sets of examples are presented in Figure E.1 and in Figure E.2. Figure E.1 has three examples of traffic signs, objects whose shapes can be characterized by simple geometric figures. Figure E.1 has six examples of objects that require more complex shape models: the first four cases analyze the detection of butterflies and the following two cases the detection of cars.

In all the examples the search partitions were created using the WEDM merging criterion and the AMC stopping criterion with $T_{ACM} = 0.12$, starting from an accuracy partition with 500 regions. Nodes in the search space of the BPTs were created with the NWMC merging criterion proposed in Section 4.3.1 ($\alpha = 0.5$) which encourages the creation of compact and color - homogeneous nodes. Therefore, in many cases, the objects are correctly represented by a single node in the tree. We only show examples where objects are not correctly represented and the extension step is required. In the examples of Figure E.1 the shape matching process uses a binary distance whereas in those in Figure E.2, a Chamfer 3-4 distance transform [15] is applied.

Examples in Figures E.1 show pictures from a traffic sign image database [66]. For the bike sign (a), the model is a circle; for the pedestrian crossing sign (b), a square, and for the junction sign (c), a triangle. In these cases the image representation correctly represents (as nodes in the tree) regions which are very good markers of small objects (the traffic signs), even in cluttered backgrounds. The node extension removes leakages that are due to the presence of other objects similar in color to the traffic signs.

Another examples are presented in Figure E.2, illustrating the use of more complex shape descriptions. The first four examples belong to the *butterfly* class and the last two to the *car* class of the Corel subset. The butterfly and car masks have been manually created using as example, in each case, a different Corel image of the same class.



Figure E.1: Improvement in the representation of objects achieved by the extension of the nodes, for a traffic sign detection application. First column: original images. Second column: object nodes. Third column: shape fitting. Fourth column: extended nodes.

Figure E.2 (a) presents the example of a wrong representation of the object in the search space (two different objects are merged in a node whose shape is very different from the shape of the original object) and the successful extension of the node. In this case, the shape is correctly matched to a part of the region represented by the node and the correct matching requires the re-orientation of the model shape. In turn, Figure E.2 (b) presents the correct extension of a node in an image in which the background and the object present very similar colors and where shadows are present.

Figure E.2 (c) shows an example of the unsuccessful extension of a node. In this case, the shape model used for the matching does not correspond to the type of object being sought.

	Asymmetric Distance		Symmetric Distance	
	Accuracy Part.	Search Part.	BPT Node	Ext. Node
Mean	6.31	22.25	22.60	16.29
σ^2	0.97	3.52	3.58	1.71

Table E.1: Asymmetric distance for the selected union of regions from the accuracy partition and from the search partition as well as symmetric distance for the selected BPT node and for the selected extended node with respect to the MPEG-7 subset. Mean and variance values are multiplied by 10^2 .

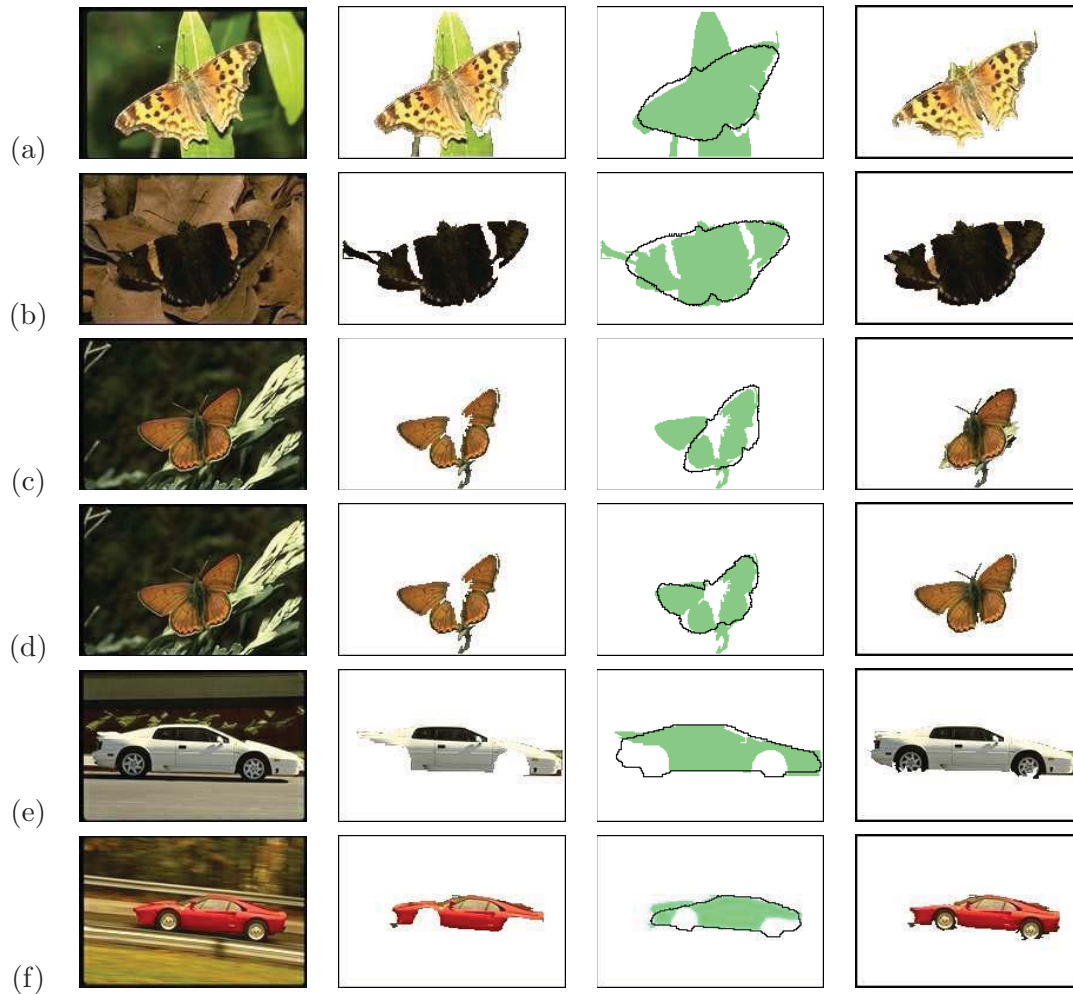


Figure E.2: Improvement in the representation of objects achieved by the extension of the nodes, for the butterfly and car detection applications. First column: original images. Second column: object nodes. Third column: shape fitting. Fourth column: extended nodes.

The shape of the butterfly present in the image is better represented by the shape model used in figure E.2 (d) and, in this case, the node is correctly extended.

For the last two sets of images, Figure E.2 (e) and (f), the node extension correctly fits the car shapes and adds the missing regions, even if they are very different in color as it is the case of the wheels.

Table E.1 presents the statistics of the asymmetric (for the accuracy and search partitions) or symmetric (for the BPT and extended BPT nodes) distances between the 45 traffic signals manually segmented from the dataset [66] and the selected (i) union of regions in the accuracy partition, (ii) union of regions in the search partition, (iii) node in the search space of the BPT and (iv) extended node in the BPT. In this database which contains images of medium/high complexity (see columns (c), (d) and (e) in Figure E.1), the asymmetric distances to the

accuracy partition and the distances to best node in the BPT are very similar (on average), which means that the objects are well represented by the tree nodes. Moreover, the use of the extended node leads to a decrease on the symmetric distance; that is, the extension improves the object representation. Finally, the statistical behavior is very similar for the three types of signals in the database (slightly better for the triangular signal).

Bibliography

- [1] T. ADAMEK, *Using Contour Information and Segmentation for Object Registration, Modeling and Retrieval*, PhD thesis, School of Electronic Engineering, Dublin City University, June 2006.
- [2] N. AHUJA, *A transform for multiscale image segmentation by integrated edge and region detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 18 (1996), pp. 1211–1235.
- [3] A. ALBIOL, L. TORRES, AND E. J. DELP, *Optimum color spaces for skin detection*, in IEEE Int. Conf. on Image Processing (ICIP), Tessaaloniki, Greece, 2001.
- [4] C. ANAGNOSTOPOULOS, I. ANAGNOSTOPOULOS, I. PSOROULAS, V. LOUMOS, AND E. KAYAFAS, *Licence plate recognigion from still images and video sequences: a survey*, IEEE Trans. on Intelligent Transportation Systems, 9 (2008), pp. 377–390.
- [5] M. ARULAMPALAM, S. MASKELL, N. GORDON, AND T. CLAPP, *A tutorial on particle filters for online Nonlinear/Non-gaussian bayesian tracking*, IEEE Transactions on Signal Processing, 50 (2002), pp. 174–188.
- [6] AT&T, *Laboratories cambridge*. <http://sww.cl.cam.ac.uk/research/DTG/attarchive/facedatabase.html>.
- [7] A. BAGDANOV, A. D. BIMBO, AND W. NUNZIATI, *Improving evidential quality of surveillance imagery through active face tracking*, in IEEE Int. Conf. on Pattern Recognition (ICPR), Hong Kong, August 2006.
- [8] C. BAHLMANN, Y. ZHU, V. RAMESH, M. PELLKOFER, AND T. KOEHLER, *A system for traffic sign detection, tracking, and recognition using color, shape and motion information*, in Proc. of the IEEE Symposium on Intelligent Vehicles, 2005, pp. 255–260.
- [9] E. BAILLY-BAILLIRE, S. BENGIO, F. BIMBOT, M. HAMOUZ, J. KITTLER, J. M. THOZ, J. MATAS, K. MESSER, V. POPOVICI, F. PORE, B. RUIZ, AND J.-P. THIRAN, *The BANCA database and evaluation protocol*, in Proc. Audio and Video Based

- Biometric Person Authentication - AVBPA 2003, Guildford, U.K., 2003, pp. 625–638.
<http://www.ee.surrey.ac.uk/banca/>.
- [10] C. BALLESTER, V. CASELLES, AND P. MONASSE, *The tree of shapes of an image*, ESAIM: COCV, 9 (2003), pp. 1–18.
- [11] C. BISHOP, *Novelty detection and neural network validation*, in IEE Proc. on Vision, Image and Signal Processing. Special Issue on Applications of Neural Networks, vol. 141, 1994, pp. 217–222.
- [12] ———, *Neural Networks for Pattern Recognition*, Oxford University Press, Walton Street, Oxford OX2 6DP, 1995.
- [13] D. BLACK, *The theory of committees and elections.*, London: Cambridge University Press, 1963.
- [14] M. BOBER, *MPEG-7 visual shape descriptors*, IEEE Transactions on Circuits and Systems for Video Technology, 11 (2001), pp. 716–719.
- [15] G. BORGEFORS, *Hierarchical chamfer matching: A parametric edge matching algorithm*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 10 (1988), pp. 425–429.
- [16] Y. BOYKOV AND G. FUNKA-LEA, *Graph cuts and efficient n-d image segmentation*, Int. Journal of Computer Vision, 70 (2006), pp. 109–131.
- [17] Y. BOYKOV AND M. JOLLY, *Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images*, in Int. Conf on Computer Vision (ICCV), Vancouver, Canada, 2001.
- [18] Y. BOYKOV AND V. KOLMOGOROV, *An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004), pp. 1124–1137.
- [19] Y. BOYKOV, O. VEKSLER, AND R. ZABIH, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 1222–1239.
- [20] G. BRADSKY, *Computer vision face tracking for use in a perceptual user interface*, in IEEE Workshop on Applications of Computer Vision, Princeton, N.J., 1998, pp. 214–219.
- [21] P. BURT AND E. ADELSON, *The laplacian pyramid as a compact image code*, IEEE Transactions on Communications, 31 (1983), pp. 532–540.

- [22] R. CAPPELLI, D. MAIO, AND D. MALTONI, *Combining fingerprint classifiers*, in First Int. Workshop on Multiple Classifier Systems, 2000.
- [23] J. CARDOSO AND L. CORTE-REAL, *Toward a generic evaluation of image segmentation*, IEEE Transactions on Image Processing, 14 (2005), pp. 1773 – 1782.
- [24] D. CHAI AND K. NGAN, *Locating facial region of a head-and-shoulders color image*, in 3rd. Int. Conf. on Automatic Face and Gesture Recognition, Nara, Japan, April 1998, IEEE, pp. 124–129.
- [25] D. CHAI AND K. NGAN, *Face segmentation using skin-color map in videophone applications*, IEEE Transactions on Circuits and Systems for Video Technology, 9 (1999), pp. 551–564.
- [26] R. CHELLAPPA, C. WILSON, AND S. SIROHEY, *Human and machine recognition of faces: A survey*, Proceedings of the IEEE, 83 (1995), pp. 705–740.
- [27] Y. CHENG, *Mean shift, mode seeking and clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17 (1995), pp. 790–799.
- [28] R. COLLINS, *Mean-shift blob tracking through scale space*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Madison, WI, June 2003.
- [29] A. COLMENAREZ AND T. S. HUANG, *Face detection with information-based maximum discrimination*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 1997, pp. 782–787.
- [30] ———, *Pattern detection with information-based maximum discrimination and error bootstrapping*, in Int. Conf. on Pattern Recognition (ICPR), 1998.
- [31] D. COMANICIU, V. RAMESH, AND P. MEER, *Kernel-based object tracking*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 564–577.
- [32] T. COOTES, G. EDWARDS, AND C. TAYLOR, *Active appearance models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 681 – 685.
- [33] P. CORREIA AND F. PEREIRA, *Stand-alone objective segmentation quality evaluation*, EURASIP Journal on Applied Signal Processing, 2002 (2002), pp. 389–400.
- [34] O. CUISENAIRE AND B. MACK, *Fast euclidean distance transformation by propagation using multiple neighborhoods*, Computer Vision and Image Understanding, 76 (1999), pp. 163–172.
- [35] A. DE LA ESCALERA, J. M. ARMINGOL, J. PASTOR, AND F. RODRIGUEZ, *Visual sign information extraction and identification by deformable models for intelligent vehicles*, Trans. Intelligent Transportation Systems, 15 (2004), pp. 57–68.

- [36] Y. DENG, B. MANJUNATH, AND M. MOORE, *An efficient color representation for image retrieval*, IEEE Transactions on Image Processing, 10 (2001), pp. 140–147.
- [37] P. DOMINGOS AND M. PAZZANI, *On the optimality fo the simple bayesian classifier under zero-one loss.*, Machine Learning, 29 (1997), pp. 103–130.
- [38] F. DORNAIKA AND J. AHLBERG, *Fast and reliable active appearance model search for 3-d face tracking*, IEEE Transactions on Systems, Man and Cybernetics, part B, 34 (2004), pp. 1838 – 1853.
- [39] M. DUBUISSON AND A. JAIN, *A modified haussdorff distance for object matching*, in Int. Conf. on Pattern Recognition (ICPR), Jerusalem, Israel, 1994, pp. A:566–568.
- [40] R. DUDA AND P. HART, *Pattern Classification and Scene Analysis*, John Wiley & Sons, N.Y., 1973.
- [41] R. DUDA, P. HART, AND D. STORK, *Pattern Classification, Second Edition*, Wiley Interscience, 2001.
- [42] R. DUIN, P. JUSZCZAK, P. PACLIK, E. PEKALSKA, D. DE RIDDER, D. TAX, AND S. VERZAKOV, *Prtools4.1. A Matlab Toolbox for Pattern Recognition*, 2007.
- [43] J. EDMONDS AND R. M. KARP, *Theoretical improvements in algorithmic efficiency for network flow problems*, Journal of the ACM, 19 (1972), pp. 248–264.
- [44] G. J. EDWARDS, C. J. TAYLOR, AND T. F. COOTES, *Learning to identify and track faces in image sequences*, in 3rd. Int. Conf. on Automatic Face and Gesture Recognition, Nara, Japan, April 1998, IEEE, pp. 260–265.
- [45] M. FASHING AND C. TOMASI, *Mean shift as a bound optimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 27 (2005), pp. 471–474.
- [46] R. FÉRAUD, O. BERNIER, J. VIALLET, AND M. COLLOBERT, *A fast and accurate face detector based on neural networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 42–53.
- [47] C. FERRAN AND J. CASAS, *Object representation using color, shape and structure criteria in a binary partition tree*, in IEEE Int. Conf. on Image Processing (ICIP), vol. 3, September 2005, pp. 1144–1147.
- [48] F. FLEURET AND D. GEMAN, *Coarse-to-fine face detection*, International Journal of Computer Vision, 41 (2001), pp. 85–107.
- [49] P. FONSECA AND J. NESVADBA, *Face detection in the compressed domain*, in IEEE Int. Conf. on Image Processing (ICIP), Singapore, October 2004, IEEE, pp. 2015–2018.

- [50] L. FORD AND D. FULKERSON, *Flows in Networks*, Princeton University Press, 1962.
- [51] Y. FREUND AND R. SCHAPIRE, *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of Computer and System Sciences, 55 (1997), pp. 119–139.
- [52] J. FRIEDMAN, T. HASTIE, AND R. TIBSHIRANI, *Additive logistic regression: A statistical view of boosting*, The Annals of Mathematical Statistics, 38 (2000), pp. 337–374.
- [53] K. FU, *Syntactic Pattern Recognition and Applications*, Englewood Cliffs, N.J., 1982.
- [54] K. FUKUNAGA, *Statistical Pattern Recognition*, Academic Press, 1990.
- [55] K. FUKUNAGA AND L. HOSTETLER, *The estimation of the gradient of a density function with applications in pattern recognition*, IEEE Transactions on Information Theory, 21 (1975), pp. 32–40.
- [56] C. GARCIA AND G. TZIRITAS, *Face detection using quantized skin color regions merging and wavelet packet analysis*, IEEE Transactions on Multimedia, 1 (1999), pp. 264–277.
- [57] L. GARRIDO, *Hierarchical Region Based Processing of Images and Video Sequences: Application to Filtering, Segmentation and Information Retrieval*, PhD thesis, Department of Signal Theory and Communications, Technical University of Catalonia, April 2002.
- [58] S. GEMAN, E. BIENENSTOCK, AND R. DOURSAT, *Neural networks and the bias/variance dilemma*, Neural Computation, 4 (1992), pp. 1–58.
- [59] X. GIRO, V. VILAPLANA, F. MARQUES, AND P. SALEMBIER, *Multimedia Content and Semantic Web*, Wiley, 2005, ch. Automatic Extraction of Visual Objects Information, pp. 203–221.
- [60] S. GONG, S. MCKENNA, AND A. PSARROU, *Dynamic Vision. From Images to Face Recognition*, Imperial College Press, London, UK, 2000.
- [61] R. GONZALEZ AND R. WOODS, *Digital Image Processing*, Prentice-Hall, Inc., New Jersey, 2002.
- [62] V. GOVINDARAJU, *Locating human faces in photographs*, International Journal on Computer Vision, 19 (1996), pp. 129–146.
- [63] J. GRACIA-ROCHE, C. ORRITE, E. BERNUES, AND J. HERRERO, *Color distribution tracking for facial analysis*, Lecture Notes in Computer Science, 3522 (2005), pp. 484–491.

- [64] H. GRAF, T. CHEN, E. PETAJAN, AND E. COSATTO, *Locating faces and facial parts*, in Int. Workshop on Automatic Face and Gesture Recognition, Zurich, Switzerland, June 26-28 1995, pp. 41–46.
- [65] D. GREIG, B. PORTEOUS, AND A. SEHEULT, *Exact maximum a posteriori estimation for binary images*, Journal of the Royal Statistical Society. Series B (Methodological), 51 (1989), pp. 271–279.
- [66] C. GRIGORESCU AND N. PETKOV, *Distance sets for shape filters and shape recognition*, IEEE Transactions on Image Processing, 12 (2003), pp. 1274–1286.
- [67] T. HASTIE, R. TIBSHIRANI, AND J. FRIEDMAN, *The elements of statistical learning*, Springer, 2009.
- [68] N. HERODOTOU, K. PLATANIOTIS, AND A. VENETSANOPOULOS, *Automatic location and tracking of the facial region in color video sequences*, Signal Processing: Image Communication, 14 (1999), pp. 359–388.
- [69] E. HJELMAS AND B. LOW, *Face detection: A survey*, Computer Vision and Image Understanding, 83 (2001), pp. 236–274.
- [70] T. HO, J. HULL, AND S. SRIHARI, *Decision combination in multiple classifier systems*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 16 (1994), pp. 66–75.
- [71] R. HSU, M. A. MOTTALEB, AND A. JAIN, *Face detection in color images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (2002), pp. 696–706.
- [72] Y. HUANG AND C. SUEN, *A method of combining multiple experts for the recognition of unconstrained handwritten numerals.*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17 (1995), pp. 90–93.
- [73] D. HUTTERNLOCHER, G. KLANDERMAN, AND W. RUCKLIDGE, *Comparing images using the hausdorff distance*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 15 (1993), pp. 850–863.
- [74] R. HYNDMAN, *Computing and graphing highest density regions*, The American Statistician, 50 (1996), pp. 120–126.
- [75] I3MEDIA PROJECT. <http://www.i3media.org/>.
- [76] A. JAIN, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989.
- [77] A. JAIN, R. DUIN, AND J. MAO, *Statistical pattern recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 4–36.

- [78] A. JAIN, K. NANDAKUMAR, AND A. ROSS, *Score normalization in multimodal biometric systems*, Pattern Recognition, 38 (2005), pp. 2270–2285.
- [79] A. K. JAIN, Y. ZHONG, AND M. DUBUISSON-JOLLY, *Deformable template models: A review*, Signal Processing, 71 (1998), pp. 109–129.
- [80] N. JAPKOWICZ, *Concept-Learning in the Absence of Counter-Examples: An Autoassociation-Based Approach to Classification*, PhD thesis, The State University of New Jersey, New Brunswick Rutgers., 1999.
- [81] O. JESORSKY, K. KIRCHBERG, AND R. FRISCHHOLZ, *Robust face detection using the hausdorff distance*, in Int. Conf. on Audio and Video Based Person Authentication - AVBPA 2001, 2001.
- [82] I. JOLLIFFE, *Principal Component Analysis*, Springer - Verlag, New York, 1986.
- [83] M. JONES AND P. VIOLA, *Fast multi-view face detection*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), June 2003.
- [84] P. KAKUMANU, S. MAKROGIANNIS, AND N. BOURBAKIS, *A survey of skin-color modeling and detection methods*, Pattern Recognition, 40 (2007), pp. 1106–1122.
- [85] D. KEREN, M. OSADCHY, AND C. GOTSMAN, *Antifaces: A novel, fast method for image detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 747–761.
- [86] C. KERVRANN, F. DAVOINE, P. PÉREZ, H. LI, R. FORCHHEIEMER, AND C. LABIT, *Generalized likelihood ratio-based face detection and extraction of mouth features*, in Third Int. Conf. on Automatic Face and Gesture Recognition, 1998, pp. 103 – 108.
- [87] N. KIRYATI AND Y. GOFMAN, *Detecting symmetry in grey level images: The global optimization approach*, International Journal on Computer Vision, 29 (1998), pp. 29–45.
- [88] L. KUNCHEVA, *Combining Pattern Classifiers. Methods and Algorithms.*, Wiley Interscience, USA, 2004.
- [89] L. KUNCHEVA AND C. WHITAKER, *Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy*, Machine Learning, 51 (2003), pp. 181–207.
- [90] C. LAMPERT, M. BLASCHKO, AND T. HOFMANN, *Beyond sliding windows: object localization by efficient subwindow search*, in IEEE Computer Vision and Pattern Recognition (CVPR), 2008.

- [91] M. LEON, V. VILAPLANA, A. GASULL, AND F. MARQUES, *Caption text extraction for indexing purposes using a hierarchical region-based image model*, in IEEE Int. Conf. on Image Processing (ICIP), Cairo, Egypt, November 2009.
- [92] T. LEUNG, M. BURL, AND P. PERONA, *Probabilistic affine invariants for recognition*, in IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 1998, pp. 678–684.
- [93] Y. LI, S. GONG, J. SHERRAH, AND H. LIDDELL, *Support vector machine based multi-view face detection and recognition*, Image and Vision Computing, 22 (2004), pp. 413–427.
- [94] Y. LI, J. SUN, C. TANG, AND H. SHUM, *Lazy snapping*, ACM Transactions on Graphics, 23 (2004), pp. 309–314.
- [95] R. LIENHART, E. KURANOV, AND V. PISAREVSKY, *Empirical analysis of detection cascades of boosted classifiers for rapid object detection*, in DAGM 25th Pattern Recognition Symposium, 2003, pp. 297–304.
- [96] S. LIN, S. Y. KUNG, AND L. LIN, *Face recognition/detection by probabilistic decision-based neural network*, IEEE Transactions on Neural Networks, 8 (1997), pp. 114–132.
- [97] Z. LIU, J. YANG, AND N. PENG, *An efficient face segmentation algorithm based on binary partition tree*, Signal Processing: Image Communication, 20 (2005), pp. 295–314.
- [98] H. LUO AND A. ELEFThERIADIS, *On face detection in the compressed domain*, in ACM Multimedia Conference, 2000, pp. 285–294.
- [99] J. LUO AND S. ETZ, *A physical model-based approach to detecting sky in photographic images*, IEEE Trans. on Image Processing, 11 (2002), pp. 201–212.
- [100] D. MAGEE AND B. LEIBE, *On-line face tracking using a feature-driven level set*, in British Machine Vision Conference (BMVC), Norwich, UK, September 2003.
- [101] D. MAIO AND D. MALTONI, *Real-time face location on gray-scale static images*, Pattern Recognition, 33 (2000), pp. 1525–1539.
- [102] S. MALDONADO-BASCN, S. LAFUENTE-ARROYO, P. GIL-JIMNEZ, H. GMEZ-MORENO, AND F. LPEZ-FERRERAS, *Road-sign detection and recognition based on support vector machines*, IEEE Trans. on Intelligent Transportation Systems, 8 (2007), pp. 264–278.
- [103] B. MANJUNATH, P. SALEMBIER, AND T. S. (EDS.), *Introductionn to MPEG-7. Multimedia Content Description Interface*, John Wiley & Sons, Inc., New York, USA, 2002.
- [104] F. MARQUÉS AND V. VILAPLANA, *Face segmentation and tracking based on connected operators and partition projection*, Pattern Recognition, 35 (2002), pp. 601–614.

- [105] E. MARSZALEC, B. MARTINKAUPPI, M. SORIANO, AND M. PIETIKÄINEN, *A physics-based face database for color research*, Journal of Electronic Imaging, 9 (2000), pp. 32–38.
- [106] A. MARTINEZ AND R. BENAVENTE, *The AR face database*, Tech. Report CVC 24, Purdue University, 1998.
- [107] B. MARTINKAUPPI, *Face Colour Under Varying Illumination. Analysis and Applications*, PhD thesis, Department of Electrical and Information Engineering, Faculty of Technology, University of Oulu., 2002.
- [108] L. MENG AND T. NGUYEN, *Frontal face detection using multi-segment and wavelet*, in Conference on Information Science and Systems, 1999.
- [109] K. MESSER, J. MATAS, J. KITTLER, J. LUETTIN, AND G. MAITRE, *XM2VTSDB: The extended M2VTS database*, in 2nd Int. Conf. on Audio and Video Based Biometric Person Authentication - AVBPA 1999, New York, 1999, Springer Verlag, pp. 72 – 77.
- [110] B. MOGHADDAM, *Probabilistic Visual Learning for Detection and Recognition*, PhD thesis, Department of Electrical Engineering and Computer Science, MIT, August 1997.
- [111] B. MOGHADDAM AND A. PENTLAND, *Probabilistic visual learning for object representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19 (1997), pp. 696–710.
- [112] P. MONASSE AND F. GUICHARD, *Fast computation of a contrast-invariant image representation*, IEEE Transactions on Image Processing, 5 (2000), pp. 860–872.
- [113] MPEG, *Description of MPEG-7 content set*, ISO/IEC JTC1/SC92/WG11/N2467, October 1998.
- [114] C. V. G. MSU GRAPHICS & MEDIA LAB. GML AdaBoost Matlab Toolbox, <http://graphics.cs.msu.ru>.
- [115] K. NUMMIARO, E. KOLLER-MEIER, AND L. V. GOOL, *Object tracking with an adaptive color-based particle filter*, in Symposium for Pattern Recognition of the DAGM, 2002, pp. 355 – 360.
- [116] P. N. OF EXCELLENCE. <http://www.pascal-network.org/>.
- [117] E. OSUNA, R. FREUND, AND F. GIROSI, *Training support vector machines: An application to face detection*, in IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Puerto Rico, June 1997.
- [118] I. OZER AND W. WOLF, *Human detection in compressed domain*, in IEEE Int. Conf. on Image Processing (ICIP), October 2001, pp. 274–277.

- [119] C. PANTOFARU, C. SCHMID, AND M. HEBERT, *Object recognition by integrating multiple image segmentations*, in European Conf. on Computer Vision (ECCV), 2008.
- [120] C. PAPAGEORGIOU AND T. POGGIO, *A trainable system for object detection*, International Journal of Computer Vision, 38 (2000), pp. 15–33.
- [121] M. PARDÀS AND E. SAYROL, *A new approach to tracking with active contours*, in IEEE Int. Conf. on Image Processing (ICIP), Vancouver, Canada, September 2000.
- [122] E. PARZEN, *On estimation of a probability density function and mode*, Annals of Mathematical Statistics, 33 (1962), pp. 1065–1076.
- [123] N. PENG, J. YANG, AND Z. LIU, *Mean shift blob tracking with kernel histogram filtering and hypothesis testing*, Pattern Recognition Letters, 26 (2005), pp. 605–614.
- [124] P. J. PHILLIPS, H. MOON, P. J. RAUSS, AND S. RIZVI, *The FERET evaluation methodology for face recognition algorithms*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000).
- [125] M. PIETIKAINEN AND A. ROSENFELD, *Image segmentation by texture using pyramid node linking*, IEEE Transactions on Systems, Machines and Cybernetics, SMC-11 (1981), p. 822..825.
- [126] S. PIGEON AND L. VANDENDROPE, *The M2VTS multimodal face database*, in 1st. Int. Conf. Audio and Video Based Biometric Person Authentication - AVBPA 1997, 1997.
- [127] V. POPOVICI AND J. THIRAN, *Face detection using SVM trained in eigenfaces space*, in 4th Int. Conf. on Audio and Video-Based Biometric Person Authentication (AVBPA), vol. 2688 of Lecture Notes in Computer Science, Berlin, 2003, Springer-Verlag, pp. 190–198.
- [128] V. POPOVICI, J. THIRAN, Y. RODRIGUEZ, AND S. MARCEL, *On performance evaluation of face detection and localization algorithms*, in Int. Conf. on Pattern Recognition (ICPR), 2004.
- [129] C. POYNTON, *A Technical Introduction to Digital Video*, John Wiley and Sons, 1996.
- [130] C. PROJECT, tech. report, Computers in the Human Interaction Loop. Clear Evaluation Plan, 2006.
- [131] Y. RAJA, S. MCKENNA, AND S. GONG, *Tracking and segmenting people in varying lighting conditions using color*, in 3rd. Int. Conf. on Automatic Face and Gesture Recognition, Nara, Japan, April 1998, pp. 228–233.

- [132] C. RASMUSSEN AND G. HAGER, *Probabilistic data association methods for tracking complex visual objects*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 560–576.
- [133] G. RITTER AND M. GALLEGOS, *Outliers in statistical pattern recognition and an application to automatic chromosome classification*, Pattern Recognition Letters, 18 (1997), pp. 525–539.
- [134] S. ROMDHANI, P. TORR, B. SCHÖLKOPF, AND A. BLAKE, *Efficient face detection by a cascaded support vector machine expansion*, in Proceedings of the Royal Society, Series A (Accepted), 2004.
- [135] A. ROSENFELD AND J. PAFALTZ, *Distance functions in digital pictures*, Pattern Recognition, 1 (1968), pp. 33–61.
- [136] D. ROTH, M. YANG, AND N. AHUJA, *A SNoW-based face detector*, Advances in Neural Information Processing Systems 12 (NIPS12), (2000).
- [137] H. ROWLEY, S. BALUJA, AND T. KANADE, *Neural network-based face detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (1998), pp. 23–38.
- [138] ———, *Rotation invariant neural network-based face detection*, in IEEE Conf. Computer Vision and Pattern Recognition (CVPR), June 1998, pp. 38–44.
- [139] B. RUSSELL, A. A. EFROS, K. SIVIC, W. FREEMAN, AND A. ZISSERMAN, *Using multiple segmentations to discover objects and their extent in image collections*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2006.
- [140] D. RUTA AND B. GABRYS, *An overview on classifier fusion methods*, Computing and Information Systems, 7 (2000), pp. 1–10.
- [141] E. SABER AND A. M. TEKALP, *Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost functions*, Pattern Recognition Letters, 19 (1998), pp. 669–680.
- [142] H. SAHBI, *Coarse-to-Fine Support Vector Machines for Hierarchical Face Detection*, PhD thesis, Versailles University, 2003.
- [143] H. SAHBI AND N. BOUJEMAA, *Coarse to fine face detection based on skin color adaptation*, in Workshop on Biometric Authentication, ECCV’s 2002, Copenhagen, Denmark., 2002, Springer Verlag.
- [144] H. SAHBI, D. GEMAN, AND N. BOUJEMAA, *Face detection using coarse-to-fine support vector classifiers*, in IEEE Int. Conf. on Image Processing (ICIP), Rochester, New York, USA, 2002.

- [145] P. SALEMBIER AND L. GARRIDO, *Binary partition tree as an efficient representation for image processing, segmentation and information retrieval*, IEEE Transactions on Image Processing, 9 (2000), pp. 561–575.
- [146] P. SALEMBIER, L. GARRIDO, AND D. GARCIA, *Image sequence analysis and merging algorithm*, in Int. Workshop on Very Low Bit-Rate Video (VLBV), Linköping, Sweden, 1997, pp. 1–8.
- [147] P. SALEMBIER AND F. MARQUÉS, *Region-based representation of image and video: Segmentation tools for multimedia services*, IEEE Transactions on Circuits and Systems for Video Technology, 9 (1999), pp. 1147–1167.
- [148] P. SALEMBIER, A. OLIVERAS, AND L. GARRIDO, *Anti-extensive connected operators for image and sequence processing*, IEEE Transactions on Image Processing, 7 (1998), pp. 555–570.
- [149] O. SALERNO, M. PARDÀS, V. VILAPLANA, AND F. MARQUÉS, *Object recognition based on binary partition trees*, in IEEE Int. Conf. on Image Processing (ICIP), 2004, pp. 929–932.
- [150] F. SAMARIA, *Parameterisation of a stochastic model for human face identification*, in 2nd. IEEE Workshop on Applications of Computer Vision, Sarasota FL, 1994.
- [151] H. SAMET, *The Design and Analysis of Spatial Data Structures*, Addison Wesley, 1990.
- [152] R. SCHAPIRE AND Y. SINGER, *Improved boosting algorithms using confidence-rated predictions*, Machine Learning, 37 (1999), pp. 297–336.
- [153] F. SCHMITT AND L. PRIESE, *Sky detection in csc-segmented color images*, in Int. Conf. on Computer Vision Theory and Applications (VISAPP), Lisboa, Portugal, 2009.
- [154] H. SCHNEIDERMAN, *Learning statistical structure for object detection*, in Computer Analysis of Images and Patterns (CAIP), Springer Verlag, August 2003.
- [155] —, *Feature-centric evaluation for efficient cascaded object detection*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), IEEE, June 2004.
- [156] —, *Learning a restricted bayesian network for object detection*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), June 2004.
- [157] H. SCHNEIDERMAN AND T. KANADE, *Probabilistic modeling of local appearance and spatial relationships for object recognition*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), July 1998, pp. 45–51.
- [158] —, *Object detection using the statistics of parts*, International Journal of Computer Vision, 56 (2004), pp. 151–177.

- [159] C. SHIPP AND L. KUNCHEVA, *Relationships between combination methods and measures of diversity in combining classifiers*, Information Fusion, 3 (2002), pp. 135–148.
- [160] L. SIGAL, S. SCLAROFF, AND V. ATHITSOS, *Skin color-based video segmentation under time-varying illumination*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004), pp. 862–877.
- [161] K. SOBOTTKA AND I. PITAS, *A novel method for automatic face segmentation, facial feature extraction and tracking*, Signal Processing: Image Communication, 12 (1999), pp. 263–281.
- [162] J. STAWIASKI AND E. DECENCIERE, *Region merging via graph cuts*, Image Analysis and Stereology, 27 (2008), pp. 39–45.
- [163] R. STOLKIN, I. FLORESCU, AND G. KAMBEROV, *An adaptive background model for camshift tracking with a moving camera*, in 6th Int. Conf. on Advances in Pattern Recognition, Calcutta, India, January 2007, World Scientific Publishing.
- [164] M. STÖRRING, *Computer Vision and Human Skin Color*, PhD thesis, Computer Vision and Media Technology Laboratory, Faculty of Engineering and Science, Aalborg University, Denmark, 2004.
- [165] K. K. SUNG, *Learning and Example Selection for Object and Pattern Detection*, PhD thesis, MIT Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences, 1996.
- [166] K. K. SUNG AND T. POGGIO, *Example-based learning for view-based human face detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (1998), pp. 39–51.
- [167] R. SZELISKI, *Computer Vision. Algorithms and Applications*, Springer, 2010.
- [168] R. SZELISKI, R. ZABIH, D. SCHARSTEIN, O. VEKSLER, V. KOLMOGOROV, A. AGARWALA, M. TAPPEN, AND C. ROTHER, *A comparative study of energy minimization methods for markov random fields with smoothness-based priors*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 30 (2008), pp. 1068–1078.
- [169] D. TAX, *One-Class Classification: Concept Learning in the Absence of Counter-Examples*, PhD thesis, TU Delft, 2001.
- [170] D. TAX. DDtools, the Data Description Toolbox for Matlab, July 2007. version 1.6.1.
- [171] D. TAX, M. V. BREUKELLEN, R. DUIN, AND J. KITTLER, *Combining multiple classifiers by averaging or by multiplying?*, Pattern Recognition, 33 (2000), pp. 1475–1485.

- [172] D. TAX AND R. DUIN, *Support vector data description*, Machine Learning, (2004), pp. 45–66.
- [173] J. TERRILLON, M. SHIRAZI, AND M. SADEK, *Invariant face detection with support vector machines*, in Int. Conf. on Pattern Recognition (ICPR), Barcelona, Spain, 2000, pp. 210–217.
- [174] J. C. TERRILLON, M. DAVID, AND S. AKAMATSU, *Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments*, in 3rd. Int. Conf. on Automatic Face and Gesture Recognition, Nara, Japan, April 1998, IEEE, pp. 112–117.
- [175] M. E. TIPPING AND C. M. BISHOP, *Mixtures of probabilistic principal component analysis*, Neural Computation, 11 (1999), pp. 443–482.
- [176] M. TURK AND A. PENTLAND, *Eigenfaces for recognition*, Journal of Cognitive Neuroscience, 3 (1991), pp. 71–86.
- [177] VACE, *Performance evaluation protocol for face, person and vehicle detection and tracking in video analysis and content extraction*, tech. report, ARDA, 2006.
- [178] V. VAPNIK, *Statistical Learning Theory*, John Wiley & Sons, New York, 1998.
- [179] R. VELTKAMP AND M. HAGEDOORN, *State of the art in shape matching*, Springer-Verlag, 2001.
- [180] R. C. VERMA, C. SCHMID, AND K. MIKOLAJCZYK, *Face detection and tracking in a video by propagating detection probabilities*, IEEE. Transactions on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 1215–1228.
- [181] V. VEZHNEVETS, V. SAZONOV, AND A. ANDREEVA, *A survey on pixel-based skin color detection techniques*, in GraphiCon, Moscow, Russia, September 2003.
- [182] V. VILAPLANA AND F. MARQUÉS, *Face segmentation using connected operators*, in International Symposium on Mathematical Morphology, Amsterdam, 1998, pp. 207–214.
- [183] V. VILAPLANA AND F. MARQUES, *Support vector data description based on pca features for face detection*, in European Signal Processing Conference (EUSIPCO), Antalya, Turkey, September 2005.
- [184] P. VILLEGAS AND X. MARICHAL, *Perceptually-weighted evaluation criteria for segmentation masks in video sequences*, IEEE Transactions on Image Processing, 13 (2004), pp. 1092–1103.

- [185] P. VIOLA AND M. JONES, *Rapid object detection using a boosted cascade of simple features*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Kauai, HI, December 2001.
- [186] T. VLACHOS AND A. G. CONSTANTINIDES, *Graph-theoretic approach to color picture segmentation and contour classification*, in IEE Proceedings, vol. 130, February 1993, pp. 36–45.
- [187] H. WANG AND S. CHANG, *A highly efficient system for automatic face region detection in MPEG video*, IEEE Transactions on Circuits and Systems for Video Technology, 7 (1997), pp. 615–628.
- [188] K. WERNECKE, *A coupling procedure for discrimination of mixed data*, Biometrics, 48 (1992), pp. 497–506.
- [189] D. WOLPERT, *The Mathematics of Generalization*, Goehring D., SantaFe Institute, 1994.
- [190] L. XU, A. KRZYZAK, AND C. SUEN, *Methods for combining multiple classifiers and their applications in handwritten character recognition*, IEEE Transactions on Systems, Man and Cybernetics, 22 (1992), pp. 418–435.
- [191] J. YANG, W. LU, AND A. WAIBEL, *Skin-color modeling and adaptation*, in Asian Conference on Computer Vision, Hong Kong, 1998, pp. 687–694.
- [192] M. YANG, N. AHUJA, AND D. KRIEGMAN, *A survey on face detection methods*, March 1999.
- [193] M. YANG, D. KRIEGMAN, AND N. AHUJA, *Detecting faces in images: A survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (2002), pp. 34–58.
- [194] M. H. YANG AND N. AHUJA, *Detecting human faces in color images*, in IEEE Int. Conf. on Image Processing (ICIP), Chicago, USA, October 1998, pp. 127–130.
- [195] M. H. YANG, N. AHUJA, AND D. KRIEGMAN, *Mixtures of linear subspaces for face detection*, in 4th. Int. Conf. Automatic Face and Gesture Recognition, 2000, pp. 70–76.
- [196] A. YILMAZ, *Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection*, in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Minneapolis, June 2007.
- [197] A. YILMAZ, O. JAVED, AND M. SHAH, *Object tracking: A survey*, ACM Computing Surveys, 38 (2006).
- [198] K. YOW AND R. CIPOLLA, *Feature-based human face detection*, Image and Vision Computing, 15 (1997), pp. 713–735.

-
- [199] K. C. YOW AND R. CIPOLLA, *A probabilistic framework for perceptual grouping of features for human face detection*, in 2nd. Int. Conf. on Automatic Face and Gesture Recognition, Killington, Vermont, Oct 1996, IEEE, pp. 16–21.
- [200] A. YPMA AND R. DUIN, *Support objects for domain approximation*, in ICANN'98, Skovde, Sweden, 1998.
- [201] H. ZABRODSKY, *Computational Aspects of Pattern Characterization - Continuous Symmetry*, PhD thesis, Hebrew University in Jerusalem, 1993.
- [202] B. ZAFARIFAR AND P. H. N. DE WITH, *Blue sky detection for picture quality enhancement*, in Advanced Concepts for Intelligent Vision Systems, vol. 4179 of Lecture Notes in Computer Sciences, Springer Berlin / Heidelberg, 2006, pp. 522–532.
- [203] D. ZHANG, S. LI, AND D. GATICA-PEREZ, *Real-time face detection using boosting learning in hierarchical feature spaces*, Tech. Report IDIAP-RR 03-70, IDIAP, December 2003.
- [204] W. ZHENG AND S. BHANDARKAR, *A boosted adaptive particle filter for face detection and tracking*, in IEEE Int. Conf. on Image Processing (ICIP), Atlanta, USA, October 2006, pp. 2821 – 2823.