Gesture recognition from video sequences

Mohamed Kaâniche

Université de Nice - Sophia Antipolis – UFR Sciences
École Doctorale STIC

THÈSE

pour obtenir le titre de :

Docteur en Sciences de l'Université de Nice - Sophia Antipolis

Spécialité : AUTOMATIQUE, TRAITEMENT DE SIGNAL ET D'IMAGES

présentée et soutenue par

Mohamed Bécha KAÂNICHE

HUMAN GESTURE RECOGNITION

Thèse dirigée par François BRÉMOND

Équipe d'accueil : PULSAR – INRIA Sophia Antipolis - Méditerranée

Soutenue à l'INRIA le 28/10/2009, à 10h45 devant le jury composé de :

Président :	Bernard	MERIALDO	UNSA / EURECOM
Directeur :	François	BRÉMOND	UNSA / INRIA
Rapporteurs :	Kamel	HAMROUNI	UTM / ENIT
	Paolo	REMAGNINO	KUL / CISM
Examinateurs :	Jenny	BENOIS-PINEAU	UB1 / LABRI
	Philippe	JOLY	UPS / IRIT
Invité :	Lionel	MARTIN	ST Microelectronics

THÈSE DE DOCTORAT / PHD THESIS

RECONNAISSANCE DE GESTES
À PARTIR DE
SÉQUENCES VIDEOS

GESTURE RECOGNITION
FROM
VIDEO SEQUENCES

MOHAMED BÉCHA KAÂNICHE

October 2009

Université Nice SOPHIA ANTIPOLIS

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche SOPHIA ANTIPOLIS - MÉDITERRANÉE

Dedication

In memory of aunt Bahya (1945-2009)
Your amazing bravery, relentless resolution, and deep concern for knowledge seekers will inspire me always.

I will never forget you.

To whom it may concern !
Keep reaching for that rainbow !

Sara Paddison, The Hidden Power of the Heart :
"When the heart is enlivened again, it feels like the sun coming out after a week of rainy days. There is hope in the heart that chases the clouds away. Hope is a higher heart frequency and as you begin to reconnect with your heart, hope is waiting to show you new possibilities and arrest the downward spiral of grief and loneliness. It becomes a matter of how soon you want the sun to shine. Listening to the still, small voice in your heart will make hope into a reality."

Résumé

Dans cette thèse, nous voulons reconnaître les gestes (par ex. lever la main) et plus généralement les actions brèves (par ex. tomber, se baisser) effectués par un individu dans une séquence vidéo. De nombreux travaux ont été proposés afin de reconnaître des gestes dans un contexte précis (par ex. en laboratoire) à l'aide d'une multiplicité de capteurs (par ex. réseau de caméras ou individu observé muni de marqueurs). Malgré ces hypothèses simplificatrices, la reconnaissance de gestes reste souvent ambigüe en fonction de la position de l'individu par rapport aux caméras. Nous proposons de réduire ces hypothèses afin de concevoir un algorithme général permettant de reconnaître les gestes d'un individu évoluant dans un environnement quelconque et observé à l'aide d'un nombre réduit de caméras. Il s'agit aussi d'estimer la vraisemblance de la reconnaissance des gestes en fonction des conditions d'observation.

La méthode proposée consiste à classifier un ensemble de gestes à partir de l'apprentissage de descripteurs de mouvement. Les descripteurs de mouvement sont des signatures locales du mouvement de points d'intérêt (c.-à-d. coins) associés aux descriptions locales de la texture du voisinage des points considérés.

En effet, à chaque personne détectée dans une scène, nous associons un ensemble de points caractéristiques suffisamment distribués tout au long de son corps. A chaque coin, nous associons un descripteur HOG (Histogram of Oriented Gradients) qui est ensuite suivi à l'aide d'un filtre de Kalman. Ainsi, nous calculons un descripteur de mouvement pour chaque descripteur HOG suivi. Un geste est représenté par un ensemble de descripteurs de mouvements locaux.

Afin de reconnaître des gestes, nous proposons d'apprendre et de classifier les gestes en se basant sur l'algorithme de catégorisation k-means et le classificateur des k-plus proches voisins. Pour chaque vidéo dans la base de données d'apprentissage, nous générons tous les descripteurs de mouvements locaux correspondants et nous les annotons avec le label du geste correspondant à la vidéo. Ensuite, chaque vidéo d'apprentissage étant pris séparément, les descripteurs sont groupés en k catégories en utilisant l'algorithme k-means. Le paramètre k est fixé empiriquement. Chaque catégorie de descripteurs est associée aux gestes correspondants : des descripteurs similaires peuvent être annotés par des gestes différents. Finalement, étant données toutes les catégories de descripteurs annotées comme base de données, l'algorithme des k-plus proches voisins est utilisé pour classifier les gestes réalisés dans les vidéos de la base de données test. Une vidéo est classifiée en tenant compte du nombre de voisins ayant voté pour un geste donné et générant ainsi la vraisemblance de la reconnaissance.

L'approche a été validée sur les bases de données de gestes publiques KTH et IXMAS ; des résultats encourageants ont été obtenus.

Abstract

In this thesis, we aim at recognizing gestures (e.g. *hand raising) and more generally short actions* (e.g. *falling, bending) accomplished by an individual in a video sequence. Many techniques have already been proposed for gesture recognition in specific environment* (e.g. *laboratory) using the cooperation of several sensors* (e.g. *camera network, individual equipped with markers). Despite these strong hypotheses, gesture recognition is still brittle and often depends on the position of the individual relatively to the cameras. We propose to reduce these hypotheses in order to conceive a general algorithm enabling the recognition of the gesture of an* **individual** *acting in an unconstrained environment and observed through a limited number of cameras. The goal is to estimate the likelihood of gesture recognition in function of the observation conditions.*

We propose a gesture recognition method based on local motion learning. First, for a given individual in a scene, we track feature points over its whole body to extract the motion of the body parts. Hence, we expect that feature points are sufficiently distributed over the body to capture fine gestures. We have chosen corner points as feature points to improve the detection stage and HOG (Histogram of Oriented Gradients) as descriptor to increase the reliability of the tracking stage. Thus, we track the HOG descriptors in order to extract the local motion of feature points.

In order to recognize gestures, we propose to learn and classify gesture based on the k-means clustering algorithm and the k-nearest neighbors classifier. For each video in a training dataset, we generate all local motion descriptors and annotate them with the associate gesture. Then, for each training video taken separately, the descriptors are clustered into k clusters using the kmeans clustering algorithm. The k parameter is set up empirically. Each cluster is associated to its corresponding gesture, so similar clusters can be labeled with different gestures. Finally, with all generated clusters as a database, the k-nearest neighbor classifier is used to classify gestures occurring in the test dataset. A video is classified according to the amount of neighbors which have voted for a given gesture providing the likelihood of the recognition.

We demonstrate the effectiveness of our motion descriptors by recognizing the actions of KTH and IXMAS public databases.

Acknowledgments

First of all, I would like to thank deeply
my thesis advisor, Dr. François Brémond, for
his careful guidance during my Ph.D. internship.
He managed to protect my autonomy while support-
ing my work. From him, I have learned that scientific en-
deavor means much more than conceiving nice algorithms
and to have a much broader view at problems from different
perspectives. Second, my thanks go to Dr. Monique Thonnat for the
constructive discussions about my work and for the opportunity she
gave me to enroll as a Ph.D. student in the thrilling project-team "PULSAR".
I would also like to thank all the members of the team for their support and for
the funny shared time; Special thanks go to Marcos Zuniga, Bernard Boulay, Na-
dia Zouba, Valéry Valentin, Rym Romdhane, Guido Pusiol and Daniel Zullo
for their friendship. I am extremely grateful for the members of my reading
committee, Dr. Kamel Hamrouni and Dr. Paolo Remagnino, for their
constructive comments on this manuscript. Distinguished thanks
goes to Pr. Bernard Merialdo, Pr. Jenny Benois-Pineau, Pr.
Phillipe Joly and Mr. Lionel Martin for accepting been part
of the examining committee. I am thankful to Catherine
Martin who makes my PhD journey easier for her
unfailing assistance in all administrative stuff.
Last but not least, I wish to thank my friends
and my family for their constant support.

Mohamed Bécha Kaâniche
Sophia Antipolis
October 7, 2009

Contents

List of Figures

List of Tables

Preface

This manuscript is the final dissertation on my Ph.D. study at STIC (Sciences and Technologies of Information and Communication) Doctoral School, Science Formation and Research Unit (UFR - Sciences), University of Nice Sophia Antipolis (UNSA). It illustrates my work during the Ph.D. internship from May 2006 to September 2009. The internship has been fulfilled at the Project-Team PULSAR (Perception, Understanding and Learning for Activity Recognition) at the Sophia Antipolis - Mediterranean research center of the French National Institute of Computer Science and Automatic (INRIA). The Ph.D. has been leaded in cooperation with ST-Microelectronics Rousset under PS26/27 Smart Environment Project which is funded by the regional council of Provence-Alpes-Côte d'Azur (PACA).

The thesis manuscript consists of seven chapters: one introductory chapter, five main matter chapters and one conclusionary chapter. The main matter chapters cover the state of the art, the proposed method and the results. Each chapter is structured to be, as far as possible, self-explanatory. The statements and discussions contained in this manuscript are solely those of the author and do not necessarily state or reflect those of University of Nice Sophia Antipolis or the French National Institute of Computer Science and Automatic (INRIA).

Mohamed Bécha Kaâniche
Sophia Antipolis
October 7, 2009

Chapter 1

Introduction

An eye is not a camera. A brain is not a computer.

Paul F. Wehlan

As a major fulfillment of machine intelligence, gesture recognition has been a prominent domain of research since the last three decades. Gestures are an important form of human interaction and communication: hands are usually used to interact with things (pick up, move) and our body gesticulates to communicate with others (no, yes, stop). Thus, a wide range of gesture recognition applications has been experienced up to now thanks to a certain level of maturity reached by sub-fields of machine intelligence (Machine learning, Cognitive Vision, Multi-modal monitoring). For example, humans can interact with machine through a gesture recognition device (*c.f*. Wii-mote in (Schlömer et al. 2008), CyberGlove in (Kevin et al. 2004) and Multi-touch screen in (Webel et al. 2008)). Nonetheless, contact device based methods are intrusive and require the user cooperation to use correctly the device. Therefore, vision-based methods propose to overcome these limits and allow the recognition of gestures remotely with or without slight user cooperation (*e.g*. body markers, cloth conditions). Since it is preferable to avoid these constraints, vision based methods have to overcome several challenges such as illumination changes, low-contrasted or/and noisy videos. Nevertheless, methods based on cameras tend to be brittle and less precise than the ones based on contact devices.

In this thesis, we aim at developing a gesture recognition system from video sequences. After presenting the motivations of this Ph.D. thesis in section 1.1, the key concepts related to the thesis topic are over-viewed in section 1.2. The context of the study is described in section 1.3. Section 1.4 states the objectives, the hypotheses and the contributions of this work and section 1.5 concludes this chapter by exposing the manuscript structure.

1.1 Motivations

Gesture recognition from video sequences is one of the most important challenges in computer vision and behavior understanding since it offers to the machine the ability to identify, recognize and interpret the human gestures in order to control some devices, to interact with some human machine interfaces (HMI) or to monitor some human activities. Generally defined as any meaningful body motion, gestures play a central role in everyday communication and often convey emotional information about the gesticulating person. During the last decades, researchers have been interested to recognize automatically

human gestures for several applications: sign language recognition (Fang et al. 2007), socially assistive robotics (Baklouti et al. 2008), directional indication through pointing (Nickel & Stiefelhagen 2007), control through facial gestures (Baklouti et al. 2008), alternative computer interfaces, immersive game technology, virtual controllers, affective computing and remote control.

In this thesis, the goal is to propose a real-time gesture recognition approach for smart environments. A smart environment is a physical world in which a set of sensors (*e.g*. cameras, microphones, pressure sensors), actuators (*e.g*. magnetostriction/electrostriction actuators), interfaces and processing units is integrated (*i.e*. embedded) seamlessly in order to react to some events and/or provide some services to present people. In the context of this thesis, we are interested in recognizing human gestures from video sequences to monitor human activities for healthcare and surveillance services.

The main challenge of vision-based gesture recognition is to cope with the large variety of gestures. Recognizing gestures involves handling a considerable number of degrees of freedom (DoF), huge variability of the 2D appearance depending on the camera view point (even for the same gesture), different silhouette scales (*i.e*. spatial resolution) and many resolutions for the temporal dimension (*i.e*. variability of the gesture speed). Moreover, we need also to balance the accuracy-performance-usefulness trade-off according to the type of application, the cost of the solution and several criteria:

- The number of cameras is crucial in a vision system and especially for gesture recognition purposes. Indeed, gestures appear differently from different points of view and even a human being cannot recognize or can mistake certain gestures if they are seen from an unusual point of view. So, the number of cameras depends on the type of the approach and application. Generally, we expect that the trade-off between the number of cameras and the methods ensures a sufficient robustness for gesture recognition even in case of point view changes.

- Another important issue is the type of cameras and the precision of the obtained 3D information: using a stereo-camera can provide a true 3D information for most of the points in the scene which is not possible with a regular monocular camera. However, 3D world information is only important for 3D approaches and becomes superfluous in case of methods based only on image plan information (2D). A Pan-Tilt-Zoom (PTZ) camera enables to obtain high resolution image of gestures.

- Robustness to image noise (background and foreground) and occlusions (partial or full) are also challenging issues. In fact, to the best of our knowledge, there is no perfect low-level vision algorithms (*e.g*. segmentation, detection and tracking). So, gesture recognition methods need to take into account noise, missdetections and occlusions.

- Gesture recognition methods generally expect video sequences with a good quality (reasonable image resolution and frame rate). The distance to the observed person

from the camera must not be too big since the shape of the person body parts must be distinguishable. The speed of detected gestures is usually related to frame-rate: we cannot expect recognizing rapid gestures with an insufficient frame-rate (*e.g.* four frames per second is required for detecting the blink of an eye).

- Last but not least, gesture recognition methods need to cope with illumination conditions and contrast changes. The cloth of detected people and the illumination changes should not influence the recognition process.

Note that the quality of low-level video processing usually influences the quality of gesture recognition. Depending on the application, one can privilege one aspect of the trade-off. For instance, for surveillance we can tolerate a certain degree of imprecision in detection and a certain delay. However, for robots interacting with humans the detection needs to be rigorous and real-time. Monitoring human activities for healthcare and/or surveillance is the most challenging application for gesture recognition since we cannot expect any user cooperation and the cameras are relatively far from the targets.

1.2 Key Concepts

The main concepts related to the topic of gesture recognition from video sequences are: computer vision, behavior understanding, people and body part detection, people and body part tracking, and posture detection.

Computer Vision also called **Machine Vision** (when focusing on industrial applications) is the broader research field of gesture recognition. On the frontiers of artificial intelligence, Machine Learning (Cognitive Vision), and Image/Signal Processing, it aims at developing artificial systems that analyze and understand video streams (*i.e.* sequence of images) or static images which are not, generally, intended to emulate human vision. Computer vision is considered as the cross-road of several research fields: Mathematics (Geometry, Statistical Analysis, Optimization problems), Physics (Optics), Imaging (smart cameras), Robotics (robot vision), Neurobiology (biological vision). Biological inspired methods (*e.g.* attentional vision) are also part of computer vision. Algorithms in computer vision are generally categorized into three levels: (1) low-level vision algorithms (*i.e.* image processing related directly to pixels without deep analysis), (2) middle-level vision algorithms (*i.e.* pattern matching, object detection and tracking) and (3) high-level vision algorithms (*i.e.* interpretation and semantics extraction from images).

Behavior Understanding is the sub-field of computer vision that is interested in detecting events and activities of human beings. Activities and events can be primitive or complex. This includes both detection of events such as intrusion in safety zone (*e.g.* forbidden access) and probably reacting (*e.g.* firing an alarm, calling police). Gesture recognition is the branch of behavior understanding that focuses on any human body motion in order to recognize this gesture as a meaningful behavior for later analysis. Algorithms for behavior under-

standing belong to high-level vision.

People detection and body part detection is concerned with detection of people and/or their body parts. These middle-level algorithms require often low-level algorithms like background updating and foreground segmentation. The main challenge of people detection is to cope with different styles of clothing, various types of posture and occlusions (partial/total with static objects or with other people). The three main categories of people detectors are: (1) Holistic Detectors, (2) Part-based Detectors and (3) Hybrid Detectors using both global and local cues. In holistic detection, a global search in the whole frame is performed. People are detected when the features, considered around a local window, meet some criteria. Global features can be used such as edge template (Papageorgiou & Poggio 2000) or also local features like Histogram of oriented Gradient (HoG) (Dalal & Triggs 2005). Concerning part-based methods, people are considered as collections of parts. Part hypotheses are generated by learning local features such as edgelet features (Wu & Nevatia 2005), orientation features (Mikolajczyk et al. 2004). Then the part hypotheses are joined to form the best assembly of people hypotheses. Note that the task of part detection (*e.g*. face, arm, legs) is challenging and difficult. As for hybrid detectors, they combine both local descriptors and their distribution within a global descriptor. For instance, (Leibe et al. 2005) introduce the Implicit Shape Model (ISM) where the global descriptor is the silhouette composed of edges matching with a learned model. In the training process, a codebook of local appearance is learned. During the detection process, extracted local features are used to match against the codebook entries, and each match casts a vote for the pedestrian hypotheses which are refined to obtain the final detection results.

People and body part tracking consist of matching the people or their part during a lap of time (*i.e*. several frames). This is a wide domain of research in computer vision and is of paramount importance in gesture recognition. The motion can be local (*e.g*. motion of feature points, motion of body parts) or global (*e.g*. the whole body motion signature). The main goal is to extract people motion features in order to analyze them for gesture recognition. Once the movement of the body or its parts is detected, computations are made to identify the type of motion which are known as the motion analysis step. This analysis may be then used by different middle-level algorithms: object tracker (when we deal with object motion) and gesture recognition (when we deal with object and body part motion). More details about this key concept are described in the next chapter.

Posture detection can be viewed as a sub-field of gesture recognition since a posture is a "static gesture". In practice, posture recognition is usually at the crossroad between people detection and gesture recognition. Sometimes we are only interested in the posture at a given time which can be performed by a people detector (Zuniga 2008). In other cases, posture detection can be sometimes considered as a first step for gesture recognition, for instance, by associating postures to states of a Finite State Machine (FSM) (Boulay 2007). The challenges of posture recognition are seamlessly the same as gesture recognition except that

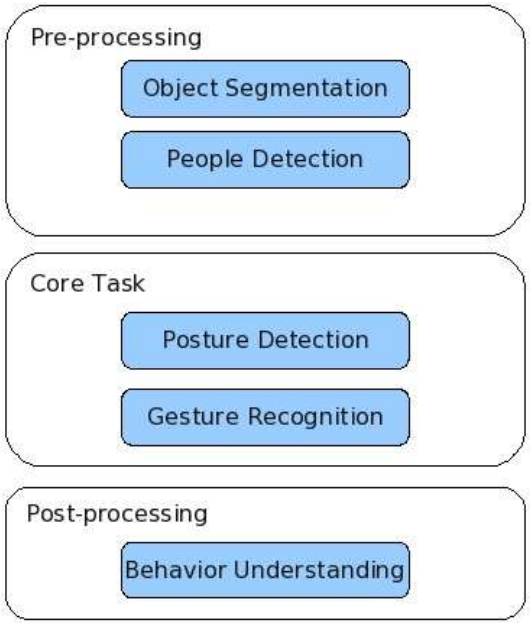


Figure 1.1: *Behavior Understanding*

the temporal aspect is not accounted. Like the equivalent trade-off in gesture recognition, an adequate balance between accuracy, precision and processing time is usually difficult to find. Yet again, we provide more details on this key concept in the next chapter.

1.3 Context of the Study

Understanding the context of the gesture recognition study is important in order to localize the gesture recognition task in the whole behavior understanding chain. Figure 1.1 illustrates the general structure of a behavior understating framework. Thus, gesture recognition is the kernel of any advanced behavior understanding framework unless only basic and simple behaviors have to be monitored. A video understanding system aims at automating the recognition of specific human behaviors from video sequences. During the last decades, several video understanding frameworks have been proposed:

- (Jaimes et al. 2003) propose a framework for video indexing using rules constructed from a priori knowledge and multimedia ontologies with a rule-based engine. The authors introduce the idea of representing perceptual concepts in terms of textual words. The presented method for scene understanding follows these steps: scene cut detection, feature extraction and visual detection.

- (Qureshi et al. 2004) introduce a cognitive vision system for space robotics. The proposed system is composed of two modules: (1) an object detector and tracker processing video streams from a calibrated passive camera-pair and (2) a cognitive vision

controller based on symbolic scene description and symbolic reasoning for scene interpretation and planning for the robot motion.

(Wang et al. 2003) overview approaches for human motion analysis. The authors emphasize three steps for a video understanding process: human detection, human tracking and activity understanding. They underline the challenges in object segmentation, occlusion handling, 3D-modeling, object tracking, multi-view processing and activity recognition.

This thesis work has been conducted in the PULSAR project-team (former Orion) at INRIA Sophia Antipolis - Mediterranean research center. The PULSAR team has developed a Scene Understanding Platform (called previously VSIP (Avanzi et al. 2005) and currently SUP) for activity recognition and more precisely for real-time semantic interpretation of dynamic scenes observed by sensors. Several algorithms of this platform (*e.g.* object detection and tracking) have been used as a preprocessing stage for the proposed gesture recognition method. However, this recognition method is independent of the preprocessing stage. Also, the current work can be used as a first step for the behavior understanding process of the SUP platform which integrates a behavior recognition module based on a priori knowledge (*e.g.* model of the empty scene, models of predefined scenarios) and a reasoning mechanism for real-time detection of temporal scenarios. However, the platform lacks a gesture recognition module which is independent from a specific application. Therefore, we aim at developing a general approach for gesture recognition independently from the type of application.

Additionally, the current work has been conducted in cooperation with ST Microelectronics under the "smart environment" project which aims at embedding real-time vision algorithms to video cameras. ST Microelectronis is interested to sell cheap smart cameras for general public specially for video surveillance and homecare applications (*i.e.* ambient intelligence).

1.4 Objectives and Contributions

Upon this thesis, our main objective is to develop a novel approach for recognizing gestures from video sequences. The goal is to recognize gestures but not their high level semantics because the meaning of a gesture depends on cultural environment and varies through different situations. The proposed approach should be suitable for healthcare and surveillance services and thus robust to noisy video sequences which are the case of real-world data. For instance, real-time processing is a desirable feature. More precisely, we require that the approach satisfies the following constraints:

- Mono-camera application: a monocular camera should be sufficient to recognize a gesture. Given the price of cameras and the space to be monitored in healthcare or surveillance applications, only one camera is observing the same scene to avoid redundancy (few overlapping cameras).
- Fixed camera: Surveillance and healthcare applications use mostly fixed cameras. Since we use the SUP platform which contains algorithms specific to fixed cameras, we limit

our work to this category of camera. However, the proposed approach does not require this constraint: a mobile camera can be used with the appropriate people detector.

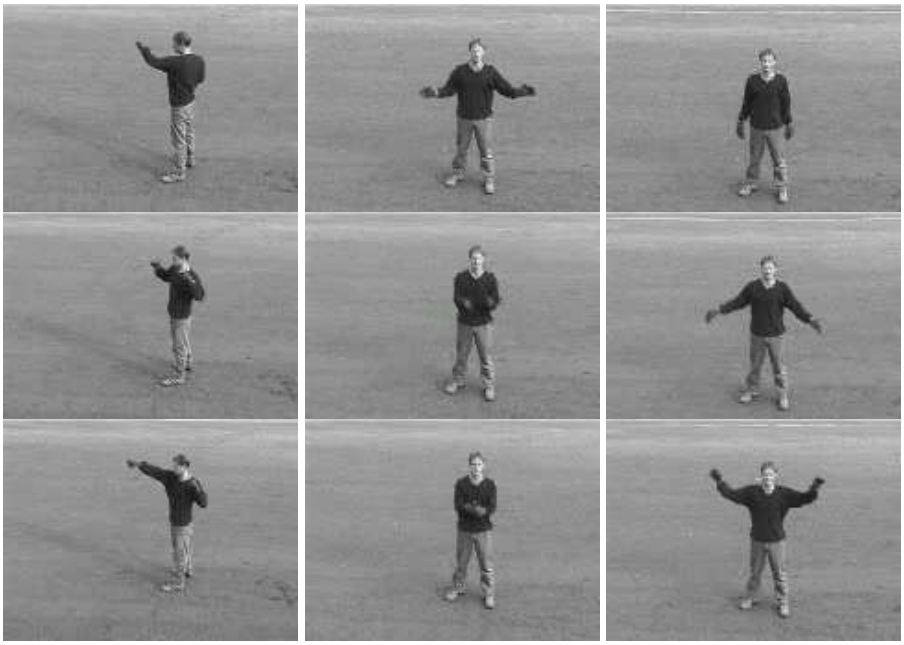
- Real-time application: algorithms requiring less computing resources will be prefered. For this reason, and as explained in the next chapter, we have chosen to develop a method based only on image plan information rather than on 3D world.
- Embeddable approach: The targeted applications request embedding the vision system in a hardware implying less resources (in terms of memory and processing) to be allocated to the gesture recognition process. Nowadays, engineers require systems with lower energy consuming (*i.e*. energy safe) which contrasts with the requirement of sophisticated image processing algorithms.

For the good usage of the approach, we also assume the subsequent hypotheses:

- Availability of a People Detector: whatever the acquiring device is, we assume that an adequate people detector is available to determine and identify where people are in the video stream. A people tracker may also be used in order to track individuals throughout image sequences. Once people are isolated, the gesture recognition system can be applied to each person in the video in order to detect relevant gestures.
- Video Sequence Quality: a reasonable image resolution and frame rate are required. Since we have chosen to implement a 2D plan information based method, a minimal frame-rate is required to capture most of motion. For instance, tested videos have a frame-rate around twelve frames per second. The image resolution must be taken in adequation with the distance of the targeted people from the camera: we expect that the shape of body parts can be discernible.
- Interaction and Crowd level: we are mainly interested in detecting single person gestures and not interaction gestures (*e.g*. hand shaking, hug). However, the developed method should be extensible to this type of interaction. A high density level of crowd is to be avoided since the increasing probabilities of occlusions and mismatches in people tracking. Nonetheless, if a robust people tracker is available for handling crowd scenes, the proposed system should perform as well.
- The developed method has to recognize gestures like hand waving, boxing and jogging (*c.f*. KTH database (Schuldt et al. 2004) or IXMAS database (Weinland et al. 2007)) and may be extended to more complex gestures for more challenging videos (*c.f*. Gerhome (Ger 2009) or TrecVid (Tre 2008)). Figure 1.2 illustrates the type of gestures that we want to recognize.

In this thesis, we try to answer two central questions in gesture recognition:

- Which representation of gesture can account faithfully of the associated motion ?
- Which strategy of recognition is the more adequate to give better results given the chosen representation ?



(a) boxing (b) clapping (c) waving

Figure 1.2: *Frames illustrating some gestures from our experiment dataset (KTH)*

Beneath this work, our contributions can be summarized as follows:

1. Proposing a new feature tracker to determine the regions of the human body where relevant motion can be extracted.
2. Proposing a new gesture representation: "Gesture Descriptor" based on "Local Motion Descriptors" which are computed from the tracked features. We model a gesture as a set of local motion signatures taking advantage of local motion descriptors. These descriptors are computed over the whole body and tracked over sufficiently large period of time. Therefore, we also benefit from the strength of global motion descriptors.
3. Proposing an effective and efficient learning-classification framework for gesture recognition based on the proposed gesture representation.

1.5 Overview of the Manuscript

In this section, we overview the remaining contents of this manuscript which is structured into five main chapters. The next chapter presents the state-of-the-art of human gesture recognition. The proposed method is over-viewed and detailed in chapters 3, 4 and 5. The seven and last chapter consists of a conclusion where a review of the contributions and an overview of perspectives are presented.

Chapter 2 recalls the previous work on gesture recognition by presenting an up-to-date state-of-the-art. After a brief presentation of the types of gesture, the enabling technologies

(*i.e.* sensors for gesture capture) and the main tools (*i.e.* techniques) for gesture recognition are over-viewed. Then, we discuss the different representations and models of gestures. Approaches for human gesture recognition are discussed and categorized into six groups: (1) feature extraction and statistical classification-based methods, (2) model-based methods, (3) template matching-based methods and (4) hybrid methods which combine several of the former techniques. Finally, we review the application areas of gesture recognition by presenting the specificity of each application domain and adequate methods for each of them.

Chapter 3 presents a global and broader view of the proposed approach. Subsequently to objectives and hypothesis enumeration, the two main parts of our work are presented: (1) gesture descriptor generation and (2) learning-classification algorithms. The design and implementation issues are then discussed.

Chapter 4 details the process of building a gesture descriptor. First, we present the pre-processing steps consisting of people detection and feature selection. Second, we describe how to compute 2D descriptor (*e.g.* HoG) associated to a feature point (*e.g.* corner point). Finally, we explain how to track this descriptor and how to build local motion descriptor which constitutes the gesture descriptor.

Chapter 5 concerns the learning-classification framework. A broader view of the framework is given. Then, we detail the learning algorithm (*i.e.* codebook generation) and the classification process. The implementation of the latter is discussed in two parts: (1) current work for the offline version and (2) future work for the on-line version.

Chapter 6 reports the experiments and their results. The experimental protocols is first presented and parameter settings are discussed. Second, the validation of feature tracking on synthetic data and real world video sequences (*e.g.* KTH (Schuldt et al. 2004)) is presented. Then, we compare the results to those obtained with our implementation of the KLT tracker. Finally, we evaluate the learning-classification framework and compare the obtained results to recent state-of-the-art methods of gesture recognition.

Chapter 2

State of the Art

I never came upon any of my discoveries through the process of rational thinking.

Albert Einstein

2.1 Introduction

Human gesture recognition consists of identifying and interpreting automatically human gestures using a set of sensors (*e.g.* cameras, gloves). In this chapter, we present an up-to-date review of the state-of-the-art in human gesture recognition which includes **gesture representations**, **recognition techniques** and **applications**. To study gesture recognition, it is essential to understand the definition and the nature of gesture as seen by the literature. Several questions rise when we try to define the word "gesture": What are the different categories of gesture ? Why do we use gestures ? What kind of information can be conveyed by gestures ? We try to answer these questions in the subsection 2.1.1. In subsection 2.1.2, we introduce the two main categories of existing approaches in human gesture recognition according to the type of the used sensors:

- Non-vision-based approaches: using instrumented gloves and tracking devices (*e.g.* joystick).
- Vision-based approaches: using one or many cameras.

Subsection 2.1.3 overviews the main tools used by human gesture recognition approaches. In our review, we focus mainly on vision-based approaches. So, the main following sections are related to this category. In section 2.2, we describe the different models used to represent gestures. The different techniques are discussed in section 2.3 and the different applications in the section 2.4. We conclude the chapter by the section 2.5 which discusses the challenges in human gesture recognition and introduces our approach.

2.1.1 Definition and Nature of Gesture

Generally speaking, we can define a gesture as a body movement. A gesture is a non-vocal communication, used instead of or in combination with a verbal communication, intended to express meaning. It may be articulated with the hands, arms or body, and also can be a movement of the head, face and eyes, such as winking, nodding, or rolling eyes. Gestures

constitute a major and an important mean of human communication. Indeed, (Pei 1984) enumerates seven hundred thousand of non-verbal communication signals which include fifty thousand two hundred facial expressions (Birdwhistell 1963) and five thousand of hand gestures (Krout 1935). However, the significance of a gesture strongly differs from a culture to another: there is no invariable or universal meaning for a gesture. For example, pointing with an extended finger is common in United-States and Europe but it is considered as a rude and offensive gesture in Asia. This implies that the semantic interpretation of a gesture depends strictly on the given culture. In addition, a gesture can be dependent on an individual state: for example, hand gestures are synchronous and coexpressive with speech, glance and facial expressions which reflect the individual mood. According to (Hall 1973), when two people engage a discussion, thirthy five per cent of their communiction is verbal and sixty five per cent is non-verbal. We can categorize the non-verbal communication into seven classes (Hall 1973):

1. Body language: facial expressions, postures, eye gaze (*e.g.* amount of gaze, frequency of glances, visual contact, patterns of fixation, blink rate), gestures, attitude.
2. Appearance: cloth, personal effects (*e.g.* jewelry, sunglasses).
3. Voice: pitch, tone, intonation, loudness, flow and pause, silence, laughter.
4. Space and distance: proxemics and proxemic behaviour categories.
5. Colors: cold or hot colors, color interpretation.
6. Chronemics (relation to time): punctuality, willingness to wait, speed of speech, willingness to listen, monochronic time schedule, polychronic time schedule.
7. Haptics: touching as non-verbal communication depends on the context of the situation, the relationship between communicators and the manner of touch. Touching is a particular type of gesture: handshakes, holding hands, kissing (cheek, lips, hand), high fives, licking, scratching.

Gestures can be categorized with respect to different criteria. For instance, (Ottenheimer 2005) distinguishes five types of gestures:

1. Emblems: an emblem (or quotable gesture or emblematic gesture) is a gesture which can be directly translated into short verbal communication such as goodbye wave in order to replace words. These gestures are very culture-specifics.
2. Illustrators: an illustrator is a gesture that depicts what the communicator is saying verbally (*e.g.* emphasis a key-point in the speech, illustrates a throwing action when pronouncing the words "he threw"). These gestures are inherent to the communicator's thoughts and speech. Also called gesticulations, they can be classified into five subcategories as proposed by (McNeill 1992):
    - beats: rythmics and often repetitive flicks (short and quick) of the hand or the fingers.

- deictic gestures: pointing gestures which can be concrete (pointing to a real location, object or person) or abstract (pointing abstract location, period of time).
- iconic gestures: it consists of hand movements that depict a figural representation or an action (hand moving upward with wiggling fingers to depict tree climbing).
- methaphoric gestures: gestures depicting abstractions.
- cohesive gestures: they are thematically related but temporally seperated gestures due generally to an interruption of the current communicator by another one.

3. Affect displays: an affect display is a gesture that conveys emotion or communicator's intentions (*e.g*. if the communicator is embarrassed). This type of gesture is less dependent on the culture.

4. Regulators: a regulator is a gesture that controls interaction (*e.g*. control turn-taking in conversation).

5. Adaptors: an adaptor is a gesture that enables the release of body tension (*e.g*. head shaking, quickly moving one's leg). These gestures are not used intentionally during a communication or interaction: they were at one point used for personal convenience and have turned into a habit though.

A gesture can be also conscious (intentional) or non-conscious (reflex, adaptors). In addition, a gesture can be dynamic or static. In the latter case, the gesture becomes a posture. Finally, we can classify gestures according to the body parts involved in the gesture: (1) hand gestures, (2) head/face gestures and (3) body gestures. Figure 2.1 illustrates a taxonomy of gesture categories which resumes all the criteria except the last one. This is not the unique way to classify gestures. Indeed, we can find in literature several other categorizations. However, we believe that the presented taxonomy accounts for almost all the aspects of gestures. In our work, we focus on dynamic gestures of the body and short actions independently from their semantic interpretation.

2.1.2 Enabling Technology

In this subsection, we overview the enabling technology for gesture recognition. As seen in the previous chapter and above, there are two main kinds of devices: (1) contact-based devices and (2) vision-based devices. Hereafter we discuss the two kinds of devices.

Contact-based technology

Contact-based devices are various: accelerometers, multi-touch screen, intrumented gloves are, for instance, the main used technologies. Some devices, like Apple©iPhone®, include several detectors: multi-touch screen and an accelerometer for instance. Other devices use only one detector: the acclerometers of the Nitendo©Wii-mote ®. Therefore, we can categorize these devices into five categories:

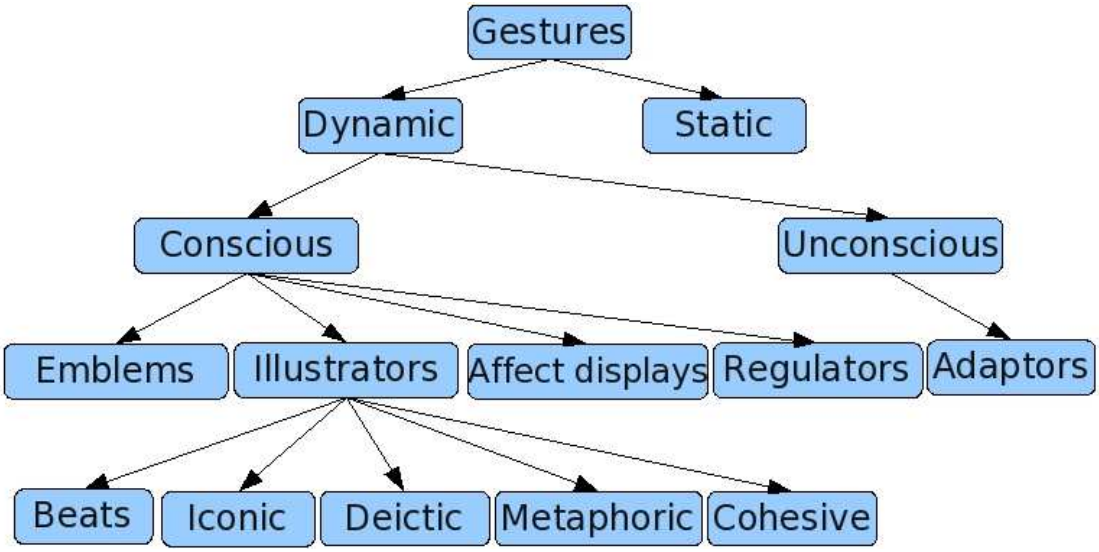


Figure 2.1: *A taxonomy of gesture categories*

- Mechanical: for instance, Immersion© proposes the "CyberGlove II®" which is a wireless intrumented gloved for hand gestures recognition. Animazoo©proposes a body suit called "IGS-190®" to capture body gestures. Figure 2.2 illustrates mechanical devices for gesture recognition. This kind of device is usually used in association with other types of device. For instance, (Kevin et al. 2004) introduce a method for trajectory modeling in gesture recognition with cybergloves and magnetic trackers. Similarly, the IGS-190 body suit is coupled with eighteen inertial devices (gyroscopes) which enable motion detection.

- Inertial: these devices measure the variation of the earth magnetic field in order to detect the motion. Two types of device are available: accelerometers (*e.g.* Wii-mote®) and gyroscopes (*e.g.* IGS-190®). (Schlömer et al. 2008) propose to recognize gestures with a wii-controller independently from the target system using Hidden Markov Models (HMM). The user can learn personalized gestures for multimodal intuitive media browsing. (Noury et al. 2003) and (Bourke et al. 2007) propose to detect fallings among normal gestures using accelerometers.

- Haptics: multi-touch screens become more and more common in our life (*e.g.* tablet PC, Apple©iPhone®). (Webel et al. 2008) propose to recognize multi-touch gestural interactions using HMM.

- Magnetics: these devices measure the variation of an artificial magnetic field for motion detection. Unlike inertial devices, magnetic devices have some health issues due to the artificial electro-magnetism.

- Ultra-sonic: motion trackers from this category are composed of three kinds of device: (1) sonic emmiters that send out the ultrasound, (2) sonic discs that reflect the ultrasound (wired by the person) and (3) multiple sensors that time the return pulse. The

(a) Instrumented glove: Immersion cyberglove 2 is equipped with (1) potentiometers that measure rotation of joints with known length and rigid links and with (2) optic fibers which measure the amount of light passing through cables in order to measure bend.

(b) Body suit: the Animazoo IGS-190 contains eighteen gyros for gesture recognition.

Figure 2.2: *Overview of mechanical gesture recognition devices*

position is computed according to the time of propagation/reflection and the speed of sound. The orientation is then triangulated. These devices are not precise and have low resolution but they are useful for environment that lacks light and has magnetic obstacles or noises.

Vision-based Technology

Vision-based gesture recognition systems rely on one or several cameras in order to analyze and interpret the motion from the captured video sequences. Similarly to contact-devices, vision-based devices are various. For instance, we can distinguish the following sensors:

- Infrared cameras: typically used for night vision, the infrared cameras give generally a brittle view of the human silhouette.

Criterion	Contact-devices	Vision-devices
User cooperation	Yes	No
User intrusive	Yes	No
Precise	Yes/No	No/Yes
Flexible to configure	Yes	No
Flexible to use	No	Yes
Occlusion problem	No (Yes)	Yes
Health issues	Yes (No)	No

Table 2.1: *Comparison between contact-devices and vision-devices*

- Traditional monocular cameras: the most common cameras due to their cheaper cost. Specific variant can be used such as fish-eyes cameras for wide-angle vision and time-of-flight cameras for depth (distance from the camera) informations.

- Stereocameras: stereovision delivers directly a 3D world information by embedding the triangulation process.

- PTZ cameras: Pan-Tilt-Zoom cameras enable the vision system to focus on particular details in the captured scene in order to identify more precisely its nature.

- Body markers: some vision systems require to place body markers in order to detect the human body motion. There are two types of marker: (1) passive such as reflective markers shining when strobes hit and (2) active such as markers flashing LED lights (in sequence). In such system, each camera, lighting with strobe lights or normal lights, delivers 2D frames with marker positions form its view. Eventually, a preprocessing step is charged of interpreting the views and positions into a 3D space.

Advantages and Disadvantages of both technologies

Both of the enabling technologies have its pros and cons. For instance, contact-devices require the user cooperation and can be unconfortable to wear for long time but they are precise. Vision-based devices do not require user cooperation but they are more difficult to configure and suffer from occlusion problems. Table 2.1 resumes the main advantages and disadvantages of both technologies. For instance, contact-devices are more precise except the utrasonic's. Also, they generally have not occlusion problems except the magnetic sensors (metal obstacles) and ultrasonic sensors (mechanical obstacles). Concerning health issues, we notice that some contact-devices can rise some problems: allergy for the mechanical sensor material, cancer risk for magnetic devices.

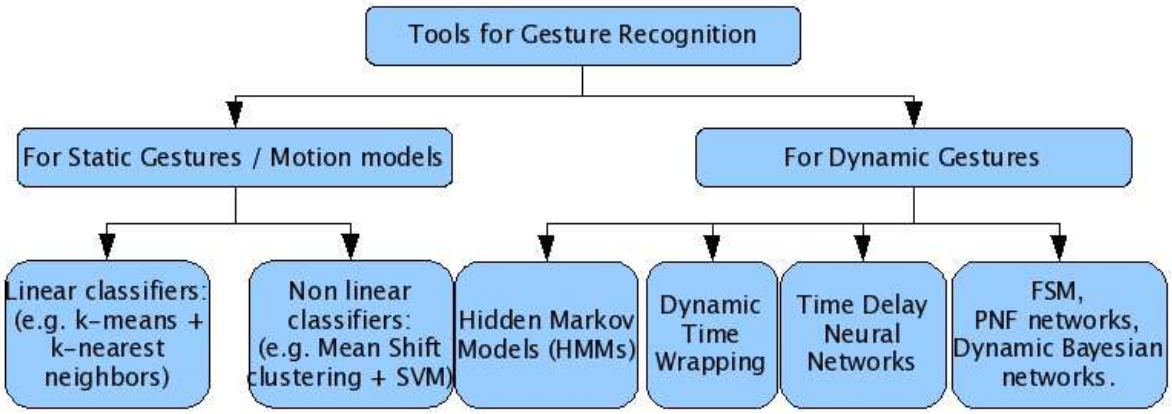


Figure 2.3: *Tools for gesture recognition: clustering and classifying algorithms (*e.g. *Mean Shift clustering + Support Vector Machine (SVM) classifier) are suitable for recognizing postures (*i.e. *static gestures) and gestures modelized by motion features (*i.e. *spatio-temporal features) since they do not handle the temporal aspect. However, when dealing with dynamic gestures with only spatial models (*e.g. *3D skeleton model), temporal aware tools are needed such as HMMs and Time Delay Neural Networks (TDNN).*

2.1.3 Tools for Gesture Recognition

The problem of gesture recognition can be generally divided in two sub-problems: (1) the gesture representation problem (*c.f*. next section) and (2) the decision/inference problem. Independently from the used device and the gesture representation, several tools for decision/inference can be applied to gesture recognition. To detect static gestures (*i.e*. postures), a general classifier or a template-matcher can be used. However, dynamic gestures have a temporal aspect and require tools that handle this dimension like Hidden Markov Models (HMM) unless the temporal dimension is modeled through the gesture representation (*e.g*. motion based model). Figure 2.3 presents the main tools for gesture recognition. Hereafter, we overview the three most common ones: (1) particle filtering and condensation algorithm, (2) learning algorithms for statistical modeling and (3) automata-based approaches (such as Finite State Machines (FSM)).

Particle Filtering and Condensation Algorithm for Gesture Recognition

The goal of particle filtering, also called Sequential Monte Carlo method (SMC), is a probabilistic inference of the object motion given a sequence of measurements. Introduced by (Isard & Blake 1996), condensation (*i.e*. Conditional Density Propagation) is an improvement of particle filtering for visual tracking which has been extended to gesture recognition (*c.f*. (Isard & Blake 1998) and (Black & Jepson 1998)). The main idea behind condensation is to estimate the future probability density by sampling from the current density and weighting the samples by some measures of their likelihood. Recently, (Lee 2008) extends the latter method to the two hand motion models. The author describes the state of a particle at a given time by four parameters: the integer index of the predictive model, the current posi-

tion in the model, a scaling factor of amplitude and a time-dimension scale factor. The three latter parameters are duplicated to take into account the motion of each hand. The recognition of gesture is done through three filtering stages: initialization, prediction and updating. A motion model, consisting of the average horizontal and vertical projections of the object velocities, is associated with the filtering process in order to recognize gestures.

Learning Algorithms for Gesture Statistical Modeling

Learning algorithms are essentially used for feature extraction based methods. There are two main variants of learning algorithms: (1) linear learner and (2) non-linear learner. The former is suited for linearly separable data and the latter for the other cases. Another way to categorize learning algorithms is to consider their outcome. Thus, we distinguish supervised learning (*i.e.* matching samples to labels), unsupervised learning (*i.e.* only sample clusters without labels), semi-supervised learning (*i.e.* mix of labelled and un-labelled data), reinforcement learning (*i.e.* learns policies given observations, *c.f.* (Darrell & Pentland 1996)), transduction (*i.e.* supervised learning with prediction, *c.f.* (Li & Wechsler 2005)) and learning to learn (*i.e.* learns his own inductive bias based on previous experience, *c.f.* (Thrun & Pratt 1998, Baxter 2000)). The choice of the learning algorithm depends mainly on the chosen gesture representation (*c.f.* next section). More details on the approaches using these tools are discussed in subsection 2.3.1). For example, (Ren & Zhang 2009) propose to recognize static hand gestures by learning the contour line's Fourier descriptor of a segmentation image obtained by mean shift algorithm (Cheng 1995). The classification is done by a support vector machine combined with the minimum enclosing ball (MEB) criterion.

Automata-based Approaches

With learning algorithms, automata-based methods are the most common approaches in the literature. For instance, FSMs, HMMs, PNF (*i.e.* Past-Now-Future network) are sort of automaton with a set of states and a set of transitions. The states represent static gestures (*i.e.* postures) and transitions represent allowed changes with temporal and/or probabilistic constraints. A dynamic gesture is then considered as a path between an initial state and a final state. (Lee & Kim 1999) propose an approach for gesture recognition using HMM-based threshold. (Lu & Little 2006*a*) present a method for recognizing human gestures using PCA-HOG global descriptor. The recognition is done by maximum likelihood estimation using HMM classifier proposed by (Yamato et al. 1992). (Pinhanez & Bobick 1997) detect human actions using PNF propagation of temporal constraints. The main limitation of the approaches based on automata is that the gesture model must be modified when a new gesture needs to be recognized. Moreover, the computational complexity of such approaches is generally huge since it is proportional to the number of gestures to be recognized which is not the case for methods based on other tools.

In the remaining of this chapter, we focus mainly on vision-based gesture recognition by detailing the different gesture representations and by discussing recognition methods and applications.

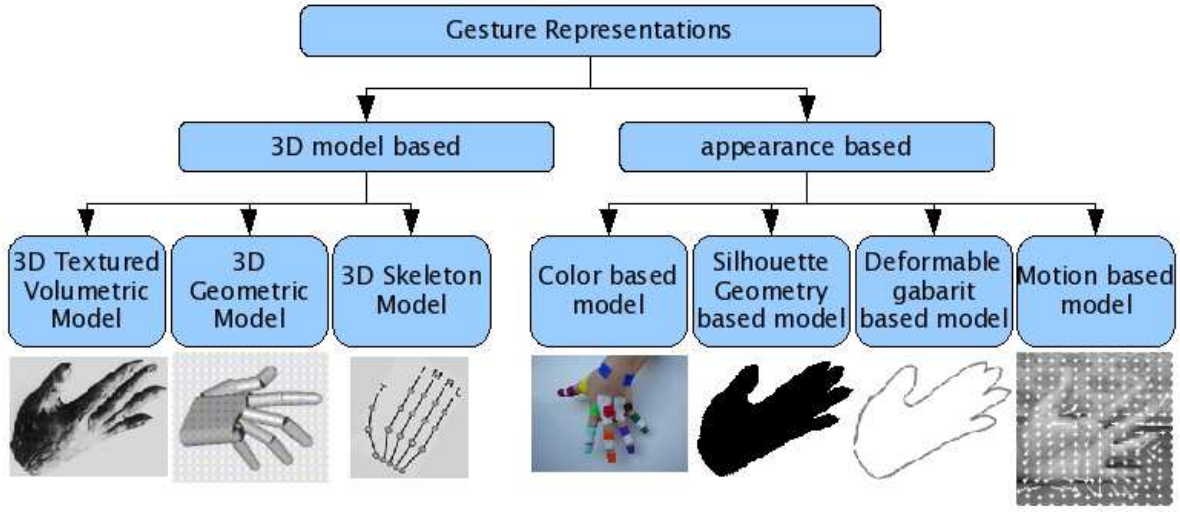


Figure 2.4: *The different representations of gesture.*

2.2 Representations of Gesture

Several gesture representations and models have been proposed to abstract and model human body parts motion. We distinguish two main categories of method: (1) 3D model based methods and (2) appearance based methods. Moreover, we can split the proposed models in two kinds according to the spatial and temporal aspects of gestures: (1) posture-automaton models in which the spatial and the temporal aspects are modeled separately and (2) motion models in which there is a unique spatio-temporal model. Figure 2.4 overviews the different representations of gestures. In this section, we overview the different gesture representations. We will focus mainly on motion representations since the proposed work belongs to this category. The methods for gesture recognition are discussed in the next section.

2.2.1 3D model based methods

A 3D model defines the 3D spatial description of the human body parts. The temporal aspect is generally handled by an automaton which generally divides the gesture time into 3 phases(*c.f*. (McNeill 1992)): (1) the preparation or prestroke phase, (2) the nucleus or stroke phase and (3) the retraction or poststroke phase. Each phase can be represented as one or several transition(s) between the spatial states of the 3D human model. The main advantage of 3D model based methods is to recognize gestures by synthesis: during the recognition process, one or more camera(s) are looking at the real target and then compute the parameters of the model that matches spatially the real target and then follows the latter motion (*i.e*. update the model parameters and check whether it matches a transition in the temporal model). Thus, the gesture recognition is generally precise (specially the start and the end time of the gesture). However, these methods tend to be computationally expensive unless implemented directly in dedicated hardware. Some methods (*e.g*. (Boulay 2007)) combine

silouhette extraction with 3D model projection fitting by finding the target self-orientation. Generally, three kinds of model are usually used:

- Textured kinematic/volumetric model: these models contain very high details of the human body: skeleton and skin surface information.
- 3D geometric model: these models are less precise than the formers in terms of skin information but still contain essentially skeleton information.
- 3D skeleton model: these are the most common 3D models due to their simplicity and higher adaptability: The skeleton contains only the information about the articulations and their 3D degree of freedom (DoF).

2.2.2 Appearance-based methods

Concerning appearance-based methods, two main sub-categories exist: (1) 2D static model-based methods and (2) motion-based methods. Each sub-category contains several variants. For instance, the most used 2D models are:

- Color-based models: methods with this kind of model use generally body markers to track the motion of the body or the body part. For example, (Bretzner et al. 2002) propose a method for hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering.
- Silhouette geometry based models: such models may include several geometric properties of the silhouette such as perimeter, convexity, surface, compacity, bounding box/ellipse, elongation, rectangularity, centroid and orientation. (Birdal & Hassanpour 2008) use the geometric properties of the bounding box of the hand skin to recognize hand gestures.
- Deformable gabarit based models: they are generally based on deformable active contours (*i.e.* snake parametrized with motion and their variants (Leitner & Cinquin 1993)). (Ju et al. 1997) used snakes for the analysis of gestures and actions in technical talks for video indexing.

For motion models, we can split them in two variants:

- Global motion descriptor: (Yilmaz & Shah 2008) have proposed to encode an action by an "action sketch" extracted from a silhouette motion volume obtained by stacking a sequence of tracked 2D silhouettes. The "action sketch" is composed of a collection of differential geometric properties (*e.g.*peak surface, pit surface, ridge surface) of the silhouette motion volume. For recognizing an action, the authors use a learning approach based on a distance and epipolar geometrical transformation for viewpoint changes. (Lu & Little 2006*b*) propose to recognize gestures via maximum likelihood estimation with hidden Markov models and a global HOG descriptor computed over the whole body. The authors extend their method in (Lu & Little 2006*a*) by reducing

the global descriptor size with principal component analysis. (Gorelick et al. 2007) extract space-time saliency, space-time orientations and weighted moments from the silhouette motion volume. Gesture classification is performed using nearest neighbors algorithm and Euclidean distance. Recently, (Calderara et al. 2008) introduce action signatures. An action signature is a 1D sequence of angles, forming a trajectory, which are extracted from a 2D map of adjusted orientation of the gradients of the motion-history image. A similarity measure is used for clustering and classification. As these methods are using global motion, they depend on the segmentation quality of the silhouette which influences the robustness of the classification. Furthermore, local motion, which can help to discriminate similar gestures, can easily get lost with a noisy video sequence or with repetitive self-occlusion.

- Local motion descriptor: local motion based methods overcome these limits by considering sparse and local spatio-temporal descriptors more robust to brief occlusions and to noise. For instance, (Scovanner et al. 2007) propose a 3-D (2D + time) SIFT descriptor and apply it to action recognition using the bag of word paradigm. (Schuldt et al. 2004) propose to use Support Vector Machine classifier with local space-time interest points for gesture categorization. (Luo et al. 2008) introduce local motion histograms and use an Adaboost framework for learning action models. More recently, (Liu & Shah 2008) apply Support Vector Machine learning on correlogram and spatial temporal pyramid extracted from a set of video-word clusters of 3D interest points.

  These methods are generally not robust enough since the temporal local windows (with short size and fixed spatial position) do not model the exact local motion but several slices of that motion instead. To go beyond the state of the art, we propose to track local motion descriptors over sufficiently long period of time thanks to a robust HOG (Histogram of Oriented Gradients) tracker. So, the spatial position is not fixed and the resulting trajectory of the local descriptor represents faithfully the local motion. The generated descriptors are used for gesture learning-clustering using the bag of word paradigm. Thus, we combine the advantages of global and local gesture descriptors to improve the quality of recognition.

2.3 Approaches for Human Gesture Recognition

Early approaches in gesture recognition from video sequences focused mainly on optical flow and motion history analysis. Throughout the past two decades, the number of techniques has been increased and different categories of approaches emerged. In this section, we outline the different methods of human gesture recognition from video sequences.

2.3.1 Feature Extraction and Statistical Classification

These methods are generally associated with a global or local motion model. The features (*i.e.* motion descriptors) are extracted from the input video sequence. Two stages are necessary for the recognition: (1) a learning stage where the extracted features from a training video dataset are categorized and (2) a classification stage where the extracted features are compared to the learned ones.

(Calderara et al. 2008) describe a method for action recognition using a classification algorithm based on mixtures of von Mises distributions of a proposed action signature. To obtain the action signature, the authors scan the motion-history image (Bobick & Davis 2001) along the direction given by the average gradient orientation, selecting only the points in which the motion energy is equal to one. The authors use Mixture of von Mises ditributions (Cucchiara et al. 2003) to describe an action signature. The parameters of these distributions are found by applying the Expectation-Maximization (EM) algorithm. A similarity measure based on a global alignment (inexact matching) and optimized by dynamic programming is used for training and classifying actions. To cope with the huge variability of actions, the authors adopt a learn-and-predict strategy in order to update and refine continously the action clusters of the learned database. The training-classification algorithm is an adaptation of the global alignment similarity measure proposed by (Gusfield 1997). The algorithm is validated with a set of 25 videos. Each video portrays a person performing a sequence of 8 actions.

(Liu & Shah 2008) describe a gesture recognition method by learning (using Support Vector Machine) a feature vector which is obtained by extracting correlogram and spatial temporal pyramid from a set of video-word clusters (VWCs). This set of VWCs is built by applying a Maximization of Mutual Information (MMI) algorithm on a codebook of video-word clusters. A video-word cluster is initialized as a video word codebook singleton. The video-word codebook is learned using k-mean, based on an appearance similarity, from cuboids extracted by separate linear filters in spatial and temporal directions. The number of video word codebooks (*i.e.* cluster number) is predefined (in the experiment is fixed to 1000). The cuboids are gradient-based descriptors characterising 3D spatial-temporal interest points (proposed by (Dollar et al. 2005)).

2.3.2 Model-based methods

As seen in previous section there are two types of gesture model: (1) 3D models and (2) 2D models. Unlike learning based methods, the recognition process is composed of a single stage where the parameters of the real target are extracted and then fitted to the adequate gesture model. The model fitting is driven by the attempt to minimize a residual measure between the projected model and the person contours (*e.g.* edges of the body) which requires a very good segmentation of body parts. Thus, such techniques require video sequences without very strong noise. (Chu & Cohen 2005) introduce a method for gesture recognition using

3D Visual Hull (3D geometric model). The approach reconstructs the 3D human body model from several point of views captured by several cameras (for instance four cameras). Then a 3D shape descriptor consisting of a set of geometric properties of the 3D model is computed. Finally, a "matching pursuit decomposition" is performed using a priori known 3D model of postures and a dual-state HMM. (Munoz-Salinas et al. 2008) propose to apply a depth silhouette (*i.e.* a 3D silhouette combining 2D silhouette and depth information) representation to three gesture recognition methods:

- Silhouette compression with PCA and learning with HMM: the sequence of gesture silhouettes is represented in terms of their principal components and the temporal behaviour of the gesture in the eigenspace is learnt by HMM.
- An examplar-based gesture recognition using HMM (*c.f.* subsection 2.3.4).
- Temporal template (*c.f.* next subsection) compressed with PCA and learned with SVM.

2.3.3 Template Matching

In template matching based methods there is neither a feature extraction nor a gesture model, the whole gesture is considered as a template. (Bobick & Davis 2001) use the motion history images (MHI) and motion energy images (which can be seen as gesture templates) to recognize gestures. During the learning process, a statistical model (mean and covariance matrix) of seven Hu moments of MHI and MEI is generated. For the matching process, a Mahalanobis distance is computed between the moment description of the new input and moments of learned gestures. Recently, (Roh et al. 2006) propose a volume motion template (VMT) for view-invariant gesture recognition. The training data-set consists of the projection images of the VMTs of each learned gesture. For matching a new input gesture, the k-nearest neighbor algorithm is used.

These methods are different from 2D methods since the temporal and spatial dimensions are included in the same model. So, there is no need for an automata-like recognizer. Also, this is different from motion-based models where local or global motion descriptors (*e.g.* HOG, SIFT) are extracted and then learned. Here, the whole gesture template (or a more compact representation) is used as a learning model. The disadvantage of such methods is the huge size of learned data which influences the computational cost of the matching process.

2.3.4 Hybrid and Miscellaneous Methods

(Munoz-Salinas et al. 2008) propose to combine a depth silhouette (3D model) to a temporal template matching method. The authors define a multi-layered silhouette history image (MLSHI). Each layer corresponds to motion history of one silhouette depth. Each template layer is reduced to its principal components using PCA. Then, the component are concatenated to fed a SVM classifier. (Boulay 2007) introduces an action recognition method using

Finite State Machines (FSM) and a Hybrid posture recognition process. The author defines a 3D posture avatar which models several postures of the human body. In the recognition process, the person orientation is estimated and then the 2D silhouette is compared to the projections of the a priori known postures from the 3D avatar according to the computed orientation. The silhouette comparison is done using different appearance based techniques (*e.g.* horizontal and vertical projections, Hu moments). The action recognition is performed by usind a FSM in which each state represents one or several postures.

Recently, exemplar based technique for gesture recognition have been introduced (Izo & Grimson 2007, Kale et al. 2004, Liu & Sarkar 2005). An exemplar is a compact representation for the set of images of a gesture. Assuming that $\{ s_i, i \in [1..n] \}$ denotes the set of n images reapresenting a particular gesture, the exemplar e corresponding to this gesture is the image which pixels satisfy formula 2.1.

$$e(x,y) = \frac{1}{n}\sum_{i=1}^{n}\kappa(s_i(x,y)) \tag{2.1}$$

where κ is a function that returns one if the pixel becomes to foreground and zero otherwise. A frame to exemplar distance (FED) is introduced to compute the distance between a new input gesture and an exemplar based learned data-set. (Munoz-Salinas et al. 2008) propose a method combining exemplar and depth silhouette and using HMM for gesture recognition. The authors introduce a depth exemplar: each pixel of a depth exemplar is an histogram (with m bins) representing the likelihood distribution of the depth values in the corresponding pixels in the image set of depth silhouettes of the gesture. An adequate new FED is also proposed. The exemplar representation can be seen as a trade-off between temporal template and global motion descriptor. However, the authors show that this method performs worse than the two other proposed methods (3D model-based and temporal template-based).

2.4 Application Areas

Gesture recognition has several domains of application. From sign language interpretation to virtual environments with smart human-machine interfaces, the number of domains increases continuously and the proposed solutions are more and more efficient. In this section, we overview the main domains of application of gesture recognition from video sequences.

2.4.1 Sign Language

While in speech recognition the goal is to transcribe speech to text, the aim of sign language recognition is to transcribe gestures of the sign language to text. Studies for this kind of application focus mainly on hand and head gesture recognition. A significant gesture in a particular sign language can be either static or dynamic. For example, (Swee et al. 2007) propose a system to recognize "Malay" sign language with a set of sensors consisting of

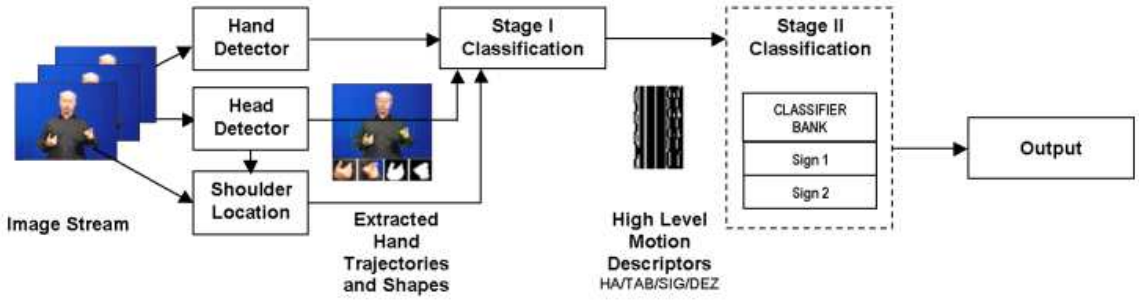


Figure 2.5: *The method proposed by (Bowden et al. 2004) for sign language recogntion*

accelerometers and flexure sensors to capture the movement of shoulders, elbows, wrists, palms and fingers. The system is able to recognize 25 common words signing with a HMM based method. (Bowden et al. 2004) try to recognise signed words from the "British" sign language by using video streams as unique input. The recognition process illustrated by figure 2.5 is composed of two main steps:

- Human upper body tracking with extraction of hand trajectories.
- Gesture classification using high level feature description based on sign linguistics. The classification process is divided in two stages. In the first stage, authors generate a description of hand shape and movement at the level of "the hand has shape 5 (an open hand) and is over the left shoulder moving right". This level of feature is based directly upon those used within sign linguistics to document signs. In the second stage, they apply Independent Component Analysis (ICA) to separate the channels of information from uncorrelated noise. Final classification uses a bank of Markov models to recognise the temporal transitions of individual words/signs.

2.4.2 Presentations / Gesture-to-Speech

In a presentation (*e.g.* weather narration, technical reports), significative gestures are recognized to identify more precisely what the speaker is talking about (*c.f.* (Ju et al. 1997)). Specially, we can be interested to detect the directional indication through pointing. This is useful for identifying the context of statements or instructions. Moreover, it can be very interesting to interpret gestures at the same time with the speaker speech since there are some correlation between them. (Kettebekov et al. 2005) propose to co-analyze an audiovisual stream based on prosody for co-verbal gesture recognition. The main idea is to co-analyze gestures and speech by relating a set of prosodic features in speech to hand kinematics. The proposed approach, which aims to determine when gesture and speech are synchronized, was validated on a weather narration application.

Gesture command	3D avatar response
I am hungry	Patient is hungry
I am thirsty	Patient is thirsty
I need going to bath	Patient needs to go to the bathroom
I have pain	Patient has pain
It is cold	Patient is cold - Room is cold
It is hot	Patient is hot - Room is hot
I need a doctor	Patient is calling the doctor
I need a nurse	Patient is calling the nurse
I need help	An emergency situation

Table 2.2: *(Keskin et al. 2007)'s healthcare gesture commands*

2.4.3 Virtual Reality

Virtual reality (*i.e.* virtual environments) allows a user to interact with a computer-simulated environment (whether that simulated world is an instance of the real world or an instance of imaginary world). It includes immersive gaming, flight simulators and remote control. Here gesture recognition is used as a mean of communication with the virtual world (eventually the real system).

2.4.4 Healthcare

(Keskin et al. 2007) propose a multimodal system integrating gesture recognition and 3D talking head technologies for a patient communication application at a hospital or healthcare setting for supporting patients treated in bed. Nine gesture commands are automatically recognized and then a feedback is provided using a 3D talking avatar. Table 2.2 overviews the gesture commands and Figure 2.6 illustrates the "I am thirsty" gesture command. The authors use a continuous hidden Markov model with online Kalman filtering, online training and gesture spotting for recognizing the gesture commands. Antoher kind of application is homecare for elderly people living at home as in Gerhome project (Ger 2009).

2.4.5 Video Surveillance

The main objectives of video surveillance are security and safety. The first thoughts that spring immediately to mind are about the security issue. As for gesture recognition, The main goal is to detect violent and/or furtive actions in some areas to secure. (Hiroshi et al. 2006) introduce an approach for detecting violent actions (*e.g.* bag-snatching) in an elevator car with optical flow analysis. The faces of violent actors are then recognized. Concerning the second objective (*i.e.* safety) of video surveillance, the goal is to check if the gestures and the actions of a certain operator are safe in a certain zone.

Figure 2.6: *The gesture for "I am thirsty"*

2.5 Discussion

As seen above, several approaches have been proposed to recognize gestures from vision-based devices for a wide range of applications. Hereafter, we review the pros and cons of each category of approaches and we introduce our method by underlying our contributions.

2.5.1 Effectiveness - Efficiency tradeoff

We have seen that the most effective methods for recognizing gestures from video sequences are those which use a 3D model of the human body. However the most efficient ones are those which rely on appearance. Indeed, we have seen in previous sections that model-based methods require a very good segmentation of body parts and that the design of such model can be complex in terms of handling the time dimension of gestures. Thus, we believe that methods based on motion models (global, local or template) provide the best tradeoff between effectiveness and efficiency. Concerning decision-inference techniques,learning algorithms based on statistical modeling of gestures appear to be more flexible since they do not limit the number of gestures to be recognized.

2.5.2 Simplicity - Faithfulness tradeoff

There is no doubt that the most faithful description of gestures is obtained by using 3D human body model with motion. Nevertheless to take into account all the aspects of motion, the conception and the parameterization of such model is usually too complex. Appearance based methods are simpler to model and to parameterize but some ambiguities can rise on the prestroke and poststroke phases (*i.e.*the begin and the end of gestures) especially when motion model is used. Among appearance methods, motion based model approaches

are easier to implement since they include in the same model the temporal and the spatial aspects of the gesture.

2.5.3 Proposed method and contributions

To go beyond the state of the art, we propose to track local motion descriptors over sufficiently long period of time thanks to a robust HOG tracker. The generated descriptors are used for gesture learning-clustering using the bag of word paradigm. Thus, we combine the advantages of global and local gesture descriptors to improve the quality of recognition. The local motion descriptors can be considered as a local version of the global action signature proposed by (Calderara et al. 2008) with the advantage of capturing also local motion. Compared to the global PCA-HOG descriptor proposed by (Lu & Little 2006*a*) (one global HOG descriptor for each gesture/action), the proposed gesture/action descriptor consists of a set of local motion descriptors which accounts more faithfully for local motion. Instead of computing a global HOG volume from a person already tracked, we use local HOGs tracked independently. Our method contrasts from common local motion methods by tracking salient 2D descriptors instead of computing arbitrary time-volume of 2D descriptors. Our contributions can be resumed as follow:

- Proposing a robust feature point tracker by using HOG and Kalman filtering.
- Proposing a novel gesture motion descriptor .
- Proposing a new gesture learning-classification framework based on the proposed gesture motion descriptor.

Chapter 3

Thesis Overview

The question is not what you look at, but what you see.

Henry David Thoreau

3.1 Introduction

Before presenting the proposed approach, we remember hereafter our objectives and the assumptions we made.

3.1.1 Objectives

Our objective is to recognize, on-line, body gestures (*e.g.* hand rising, hand waving, kicking, applauding) and more generally short actions (*e.g.* sitting, standing, falling, bending) accomplished by one or several individuals. The goal is to estimate the likelihood of gesture recognition according to the observation conditions.

3.1.2 Constraints

Many techniques have already been proposed for gesture recognition in specific environment (*e.g.* laboratory) using the cooperation of several sensors (*e.g.* camera network, individual equipped with markers). Despite these strong hypotheses, gesture recognition is still brittle and often depends on the position of the individual relatively to the cameras. We aim at reducing these hypotheses in order to conceive general algorithm enabling the recognition of the gesture of an individual involving in an un-constrained environment and observed through limited number of cameras. We restrict our study on scenes captured by one camera and containing gestures similar to those of these databases: KTH (Laptev 2005), IXMAS (Weinland et al. 2007), TrecVid 2008 (Tre 2008) and Gerhome (Ger 2009). The proposed approach could be validated through a home-care application aiming at helping elderly people living at home (Gerhome project (Ger 2009)).

3.2 Proposed Human Gesture Recognition Approach

In order to recognize human gestures from videos, we propose an approach composed of three stages. In the first stage, we compute for each detected human in the scene a set of 2D local descriptors, based on feature points (*i.e.* corners) and Histograms of Oriented

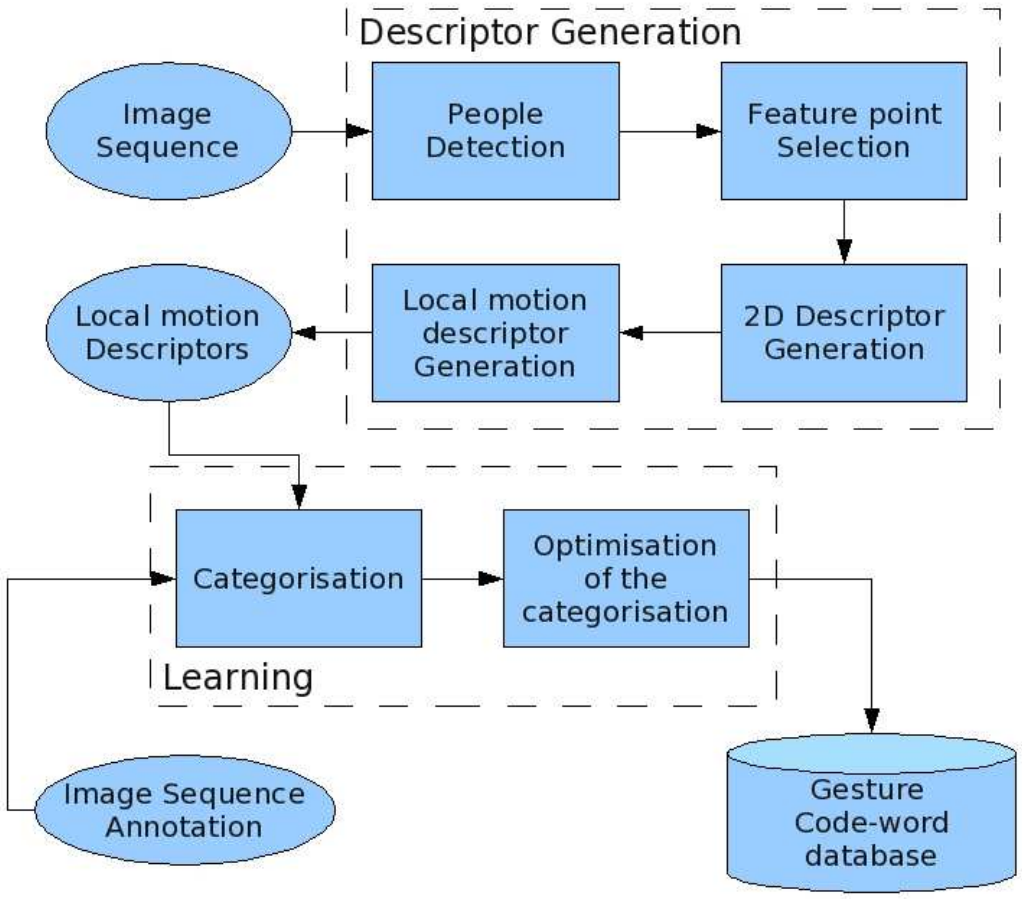


Figure 3.1: *Gesture descriptor generation and Code-book learning*

Gradients (HOG), which are tracked over time to build local motion descriptors. In the second stage, we learn the local motion descriptors for a given gesture creating code-words from a learning video data set: local motion descriptors are categorized into motion clusters. In the last stage, we classify the gesture of a person in a new video by extracting the person local motion descriptors as new code-words and comparing them with learned code-words. Figure 3.1 illustrates the two first stages and figure 3.2 illustrates the last one.

In the subsection 3.2.1, we present the first stage. The two remaining stages are presented in subsection 3.2.1.

3.2.1 Gesture Descriptor

For the first stage, we have developed a feature point tracking algorithm based on two feature point selectors: Shi-Thomasi corner selector (Shi & Tomasi 1994) and Features from Accelerated Segment Test (FAST) corner selector (Rosten & Drummond 2006). The tracking is done by associating to each corner a Histogram of Oriented Gradients (HOG) and matching the latter through time. The local motion descriptors are extracted in four steps: people

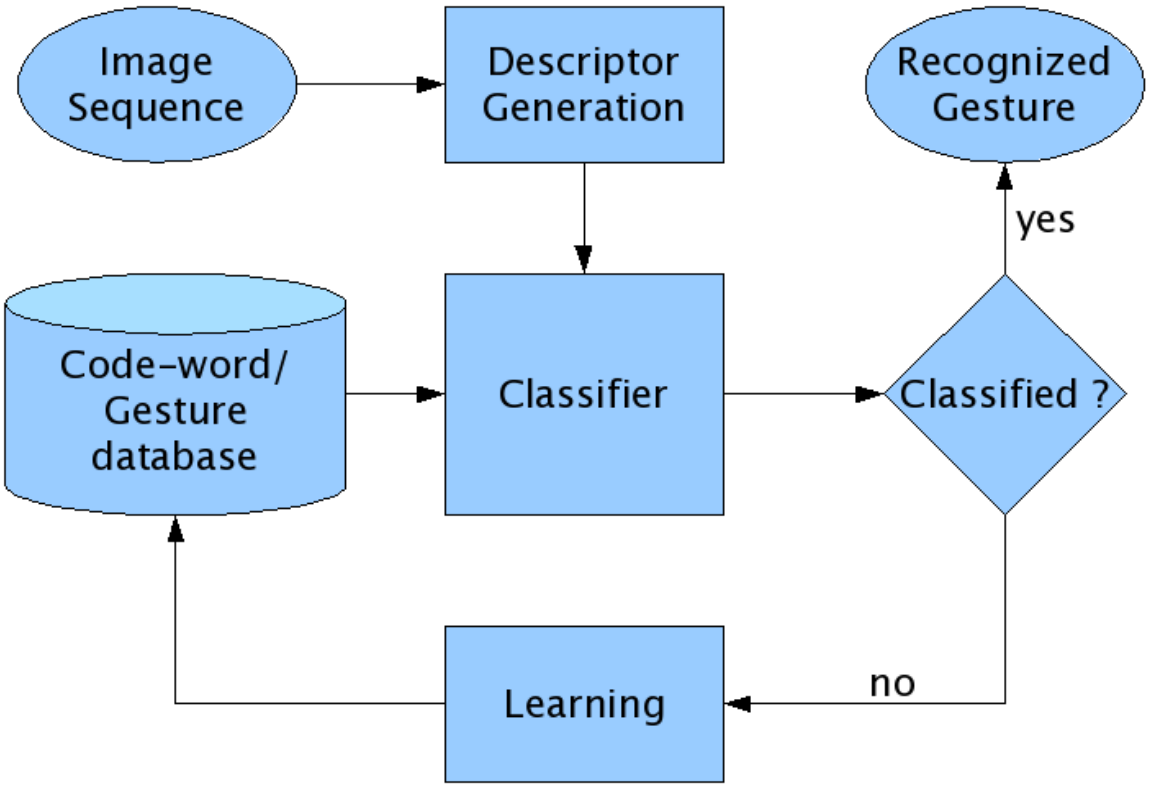


Figure 3.2: *Gesture classification*

detection, feature point selection, 2D descriptor generation, and local motion descriptor generation (*i.e.* temporal tracking and descriptor computation).

People Detection

Object detection is performed by background subtraction to determine moving regions followed by a morphological dilation. Then a people classifier is applied to determine bounding boxes of human mobile object. The people bounding boxes define a mask for feature point extraction. This step not only limits the search space of feature points but also segregates distinct moving regions: each moving region is processed independently from others except, eventually, when two regions are merging. This enables to apply the gesture recognition process to different persons in the scene until their movement is detected with too strong overlap (partial or total occlusion). In case of interaction between two persons (*e.g.* handshaking), we consider a new region resulting from the merging of the two initial regions corresponding to the interacting people.

Feature Points

Feature points are extracted for each detected person using Shi-Thomasi corner detector or Features from Accelerated Segment Test (FAST) corner detector. The masks generated by the previous step are used to define image regions where to extract features. Given an image I in gray-scale (corresponding to a moving region), we first compute gradients g_x (through the x axis) and g_y (through the y axis) by applying simple filters ($[-1\,0\,1]$ filter for g_x and $[-1\,0\,1]^T$ filter for g_y). Then, for each pixel p in the image and in a window (u, v) centered on the considered pixel, we compute the 2x2 Hessian matrix defined by the equation (3.1).

$$H_p = \sum_u \sum_v \begin{bmatrix} {g_x}^2 & g_x g_y \\ g_x g_y & {g_y}^2 \end{bmatrix} \tag{3.1}$$

After that, we compute eigen values λ_1 and λ_2 of the Hessian matrix H_p: $\min(\lambda_1, \lambda_2)$ measures the Shi-Thomasi corner strengthness. The FAST corner detector considers pixels in a Bresenham circle of radius r around the candidate point. The corner strength measure is computed as a function of the degree of brightness of the pixels in the circle. Whatever the used corner selector is, pixels are sorted in decreasing order according to the corner strength. After that, we select the most significant corners by ensuring a minimum distance between them. Thus, feature points enable us to localize points where descriptors have to be computed since they usually correspond to body parts (*i.e.* regions of the body) where the movement can be easily discernable. Indeed, it is simpler to detect the movement of a descriptor located at a body corner (gradient variability and intensity contrast) than detecting the movement of a descriptor located inside the body where gradients are almost inexistent.

2D Descriptors

The third step consists of computing 2D descriptors. For each feature point, we define a neighborhood (a small square centering on the considered feature point) called a "patch". A 2D descriptor is computed for each patch. Our 2D descriptor is based on Histogram of Oriented Gradient (HoG) (Dalal & Triggs 2005). Thus, we associate to each feature point a descriptor block composed of 3×3 cells; each of them has a pixel size of 5×5 or 7×7: The feature point is the center of the center cell of the descriptor block. In each cell, using the gradients g_x and g_y computed in the previous step, we associate to each pixel in the considered cell a weight-vote (gradient module or a function of the gradient module) for the orientation histogram of the cell. Therefore, the 2D descriptor is a vector concataining the descriptor block histograms. The dimension of this vector is noted dim; it corresponds to the cell number multiplied by the dimension of a cell histogram (*e.g.* 9×64 or 9×128). The decomposition of the descriptor block into cells improves trackability when we construct the temporal 2D descriptor. Indeed, each cell encapsulates a local and specific information about the 2D descriptor which will increase the trackability through the temporal dimension by taking into account the local appearance of each cell.

Local Motion Descriptors

Local motion descriptors are built by tracking 2D descriptors. Let us suppose that we have detected a 2D descriptor d_{t-1} in the frame f_{t-1}, we are now interested to determine the descriptor d_t in the frame f_t which can be identified to d_{t-1}. Two tracking solutions are considered: the former is based on feature point tracking (considering only the center of the descriptor) and the latter is based on tracking the whole descriptor. Concerning feature point tracking, the KLT algorithm can be used. **As for tracking the 2D descriptor as a whole, we have developed a new tracking algorithm.** The basic idea is to minimize a quadratic error function $\mathcal{E}(d_t, d_{t-1})$ in a neighborhood $\mathcal{V}_{f_t}$ in the frame f_t corresponding to the predicted position of d_{t-1} obtained by a Kalman filter. In the case when several descriptors $(d_t^1, d_t^2, ..., d_t^k)$ in this neighborhood satisfy the minimum of the error function, we compute the visual evidence (intensity difference in gray-scale) between each descriptor and the descriptor of the previous frame to track d_{t-1}. The tracker will choose the descriptor that has the nearest visual evidence to d_{t-1}. We define a temporal 2D descriptor as the vector obtained by the concatenation of two time instants (detection start time t_d and detection end time t_f) and all the components of the tracked 2D descriptors between t_d and t_f. The dimension of this vector is $2 + (t_f - t_d + 1) \times dim$. The local motion descriptor is obtained from the temporal 2D descripor by computing trajectory angles and then applying Principal Component Analysis (PCA) (Jolliffe 2002) to select principal axies. So, a gesture representation consists of a set of local motion descriptors.

3.2.2 Learn-and-predict Algorithm

The learn-and-predict algorithm combines gesture descriptor learning and gesture classification.

Local Motion Descriptor Learning

Each training video is annotated with one gesture label (*e.g.* boxing). We associate the gesture information of the image sequence to each computed descriptor of the training sequence. Local Motion Descriptors are grouped into categories called "video-word" using k-means algorithm and a similarity measure. Given as input all the training video sequences, the k-means algorithm consists of clustering n objects (local motion descriptors) into k classes (clusters) $(S_i, i = 1, 2, ..., k)$ corresponding to video-words. The choice of k value is important! We choose k large enough to correctly describe the set of the m gestures to learn and recognize ($k > 3 \times m$) and strictly less than the total number of descriptors n ($k < n$). The goal is to minimize the total variance between the k parts or the quadratic error function (*c.f.* equation 3.2).

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{3.2}$$

Where μ_i is the centroid of all the elements of the partition S_i (*i.e.* cluster). Each video-word is annotated by the corresponding gestures.

Then, we apply the Maximization of Mutual Information (MMI) algorithm in order to compact the video-words into an optimal set of code-words (Liu & Shah 2008). The MMI algorithm consists of reducing the number of video-words obtained by the k-means algorithm. Considering two joint discrete random variables X and Y, where $X \in \mathcal{X} = \{x_1, x_2, ..., x_k\}$, $Y \in \mathcal{Y} = \{y_1, y_2, ..., y_m\}$, $\mathcal{X}$ is the set of video-words and $\mathcal{Y}$ is the set of gesture labels. The mutual information between X and Y measures how much information from X is contained in Y. The formula (3.3) defines the mutual information between X and Y.

$$MI(X,Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} P(X = x, Y = y) \log \frac{P(X = x, Y = y)}{P(X = x)P(Y = y)} \tag{3.3}$$

The goal of the MMI algorithm is to incrementally reduce X (by fusion of video-word pairs) by optimizing (ensuring minimum loss) the mutual information between video-words and gesture labels. Before the optimization process $\mathcal{X}$ corresponds to video-words and after to code-words. The algorithm is described hereafter (Algorithm 1). Note that $\otimes$ is the fusion operator, ϵ is a maximal threshold of information loss and $Card$ is the cardinal operator.

Algorithm 1 *Maximization of Mutual Information (MMI) Algorithm*

Require: X, Y
1: $X_{min} \leftarrow X$
2: $minimalDistance \leftarrow \infty$
3: **for all** $x_1, x_2 \in X / x_1 \neq x_2$ **do**
4: $X_{previous} \leftarrow X$
5: $X_{next} \leftarrow (X - \{x_1, x_2\}) \cup \{(x_1 \otimes x_2)\}$
6: $distance \leftarrow MI(X_{previous}, Y) - MI(X_{next}, Y)$
7: **if** $distance < minimalDistance$ **then**
8: $minimalDistance \leftarrow distance$
9: $X_{min} \leftarrow X_{next}$
10: **end if**
11: **end for**
12: **if** $minimalDistance < \epsilon \& Card_{X_{min}} > Card_Y$ **then**
13: $X \leftarrow X_{min}$
14: MMI(X,Y)
15: **end if**

Gesture Classification

We use a "learn-and-predict" classification strategy: considering a new generated gesture descriptor, this gesture descriptor is transformed into a code-word and then compared to other code-words in the codebook (see figure 3.2). Based on this comparison, the classifier decides to classify it as belonging to an existent category or to create a new category (a new class of gesture or action not learned yet). We introduce also a new voting mechanism for handling the many-to-many mapping between video-words and gesture labels. This

strategy is applied to two classifiers. The first classifier is the k-nearest neighbor algorithm: considering a training database constituted of N input/output pairs (descriptor-gesture), to estimate the output associated to a new input x, the algorithm considers, with the use of an appropriate distance, the k-nearest inputs in the database to x and assigns it with the most common output among the associated outputs of the considered inputs.

The second classifier is based on Support Vector Machine (SVM). SVMs, thanks to two key ideas, transforms a non-linear classification problem to a quadratic optimization problem.

The first key idea is the concept of Maximum-Margin. A Margin is the distance between the separator hyperplane and the nearest samples called support-vectors. In a SVM, the separator hyperplane is chosen as the separator satisfying the maximum margin. This choice is the optimal one (Hearst 1998) according to Vapnik-Chervonenkis Statistic Learning Theory. The problem of determining this optimal separator can be resolved with quadratic optimization algorithms.

To deal with non-linear classification, the second key idea of SVMs is to transform the input space to another space with higher dimensionality (possibly infinity as dimension) in which the classification problem becomes linearly separable. This is done thanks to a kernel function which satisfies several constraints and permits the space switch without knowing the explicit transformation. In addition, the kernel function replaces the computing of the scalar product in the higher-dimension space by the evaluation of a function at a point. This technique is called "kernel trick".

For handling multi-classes case (when more than two classes have to be separated), we distinguish two general methods applicable to any binary classifier: "one-versus-all" method and "one-versus-one" method. Given training data samples classifiable into N classes $(C_1, ..., C_N)$, the "one-versus-all" method consists of constructing N binary classifiers for each class: attributing "Classified" label or "Unclassified" label according to the sample membership in the correspondent class or not. For classification of new instances, winner-takes-all strategy is applied: the classifier generating the higher margin wins the vote. In this method, there is no normalization between the margin values given by the classifiers so we can have some scale problems resulting in misclassification (Bishop 2006). In addition, the problem is not balanced, for example with N=10, we use only 10% of positive samples for 90% of negative samples.

Concerning the "one-versus-one" method, it consists of building $(N \times (N-1))/2$ binary classifiers by confronting one by one, each class pair formed from the N classes. For classification of new instances, max-wins voting strategy is applied: the class with the most votes determines the instance classification. Considering x the sample to be classified and $\Phi_{ij}(.)$ the classifier segregating class C_i from class C_j and returning the class label of the considered

sample, then the label associated to x is $\underset{k\in\{1,...,N\}}{\arg\max} Card(\{\Phi_{ij}(x)\}\cap\{k\}; i,j \in \{1,...,N\}, i<j)$. In this method, some ambiguities can rise when there is no majority vote (Bishop 2006). A generalization of these methods has been proposed by (Dietterich & Bakiri 1995) named ECOC, considering the output of the binary classifiers as codes in order to apply code error correction techniques.

3.3 Design and Implementation of the proposed approach

As seen in the first chapter, the proposed method is integrated to the SUP platform (Avanzi et al. 2005) which provides two algorithms (a people detector and a people tracker) to our approach. Thus the proposed gesture recognition framework must conform to the SUP design rules. The SUP platform was developed in order to provide a generic and extensible framework to compose vision algorithms in order to perform behavior understanding. Behavior understanding is broader than gesture recognition since it takes into account more complex and abstract events. For this purpose, the SUP platform provides a scenario recognition module and a scenario description language to describe events to be recognized. The proposed method can be seen as a complement of this module to enable the recognition of gestures and integrate them as possible classes of elementary (*i.e.* primitive) events.

The architecture of the SUP platform is overviewed in figure 3.3. Algorithms in the SUP platform are splitted in two categories: (1) Vision algorithms for visual cues extraction and object/feature detection/tracking and (2) Behavior Undestanding algorithms for learning, classifying, monitoring and more generally recognizing events. Similarly, we can split our gesture recognition framework in two parts: (1) The gesture descriptor generation part which belongs to the vision category of the SUP plateform and (2) The gesture learning-classification part which belongs to the behavior understanding category.

3.4 Discussion

The proposed Human Gesture Recognition approach brings several contributions:

- The approach consists of an online learn-and-predict algorithm introducing a new classification process with a novel voting mechanism for handling many-to-many mapping.
- The method introduces a new feature tracker based on local 2D HOG descriptor tracking.
- The proposed framework includes a new gesture representation based on the trajectories of relevant local motion descriptors.

However, it presents some issues that must be considered:

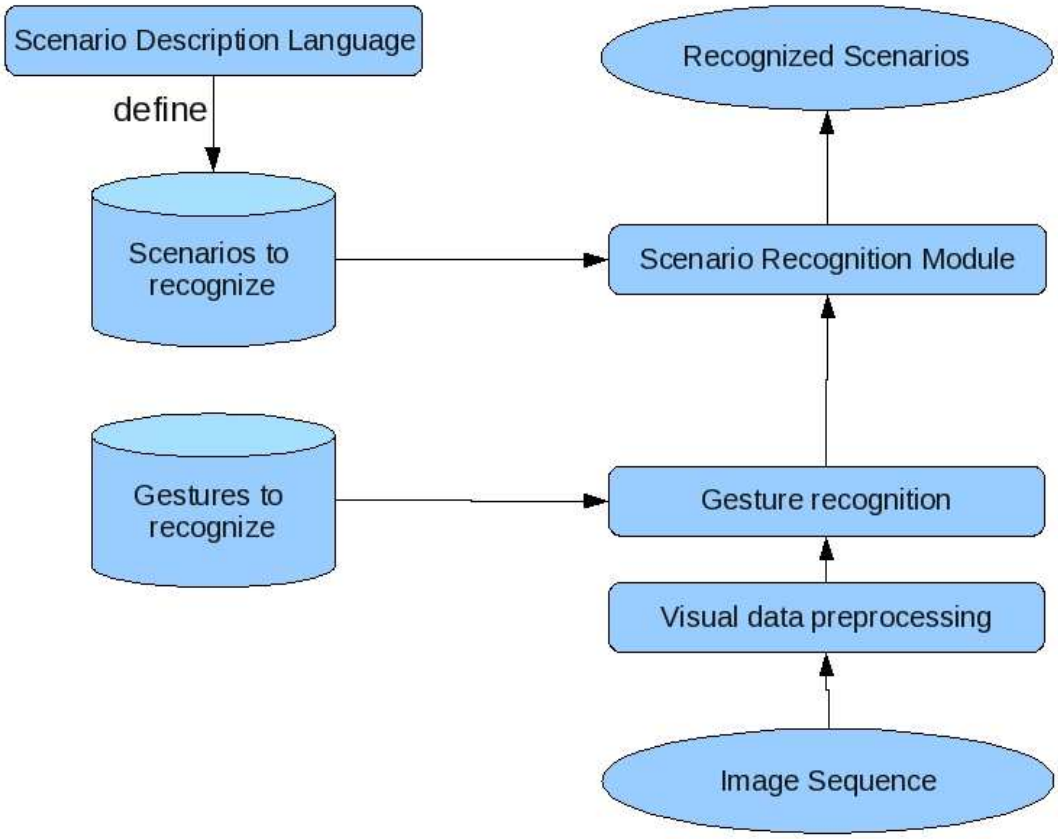


Figure 3.3: *The architecture of the SUP platform*

- The proposed gesture representation is dependent on the point of view. In order to enable multi-view gesture recognition, we must include the different views in the training dataset for the learning stage.
- The method does not handle occlusions and needs scenes with good resolution.

3.5 Conclusion

We have proposed a new method for online gesture recognition based on "learn-and-predict" classification strategy. To achieve this goal, a new feature tracker based on HOG tracking has been proposed in order to build local motion descriptors. The gesture descriptor, which consists of a set of local motion descriptors, describes a gesture observed from a certain point of view. In the next two chapters, we detail the three stages of the proposed approach.

Chapter 4

Gesture Descriptor

What we see depends mainly on what we look for.

John Lubbock

4.1 Introduction

In order to recognize human gestures, **we propose a novel Gesture Descriptor which is defined as a set of Local Motion Descriptor**. Our Local Motion Descriptor is a local and elementary representation of motion of a body part region. The main idea for building Local Motion Descriptor is to track a particular local 2D descriptor (*e.g*. HOG descriptor) over the time to build a temporal local descriptor (*e.g*. temporal HOG descriptor) which is factorized in a more compact information called Local Motion Descriptor. Each local 2D descriptor is computed for a particular local feature (*e.g*. corner point) which indicates the part of the human body where tracking motion can be relevant.

The choice of the appropriate gesture descriptor is essential for gesture recognition due to the complex aspect of the gesture (even restrained to one single person) which contains several spatial granularities (*e.g*. whole body or specific body parts) and several temporal granularities (*e.g*. instantaneous, repetitive, synchronized body motions). Hence, it is usually difficult to find the best tradeoff between the complexity of the gesture model (generally more complex models provide more accurate results since they represent more faithfully the gesture), the robusteness of this model to noise and environment changes (*e.g*. illumination changes, cloth changes), and the efficiency of the derived approach (*i.e*. real-time constraints). In many cases, one can privilege accuracy and robustness to computitional efficiency and vice-versa. However, the task becomes more difficult when one tries to conciliate these three aspects. Moreover, occlusions can introduce some ambiguities in case of one point view scene (*e.g*. side-view waving), whereas a multiple views scene introduces integration and fusion issues (*i.e*. matching and correspondance between gesture descriptors).

As seen in chapter 2 (section 2.2), several gesture representations have been proposed through the last decades. The most common representations are 2D motion based models and 3D skeleton models. Compared to 3D models, 2D motion based models require less computational resources since they handle in one and unique model the spatial and the temporal aspect of a gesture. More precisely, motion-based models can be splitted in two categories: (1) global motion models and (2) local motion models. Global motion models (Yilmaz & Shah 2008, Gorelick et al. 2007, Calderara et al. 2008) represent each gesture

as a unique and global motion signature. Local motion models (Scovanner et al. 2007, Luo et al. 2008, Liu & Shah 2008) capture several local motion features for each gesture. So, we choose to implement an appearance-based gesture representation since it is simpler and requires less processing time. Our main contribution is to model a gesture as a set of local motion signatures taking advantage of local motion descriptors. These descriptors are computed over the whole body and tracked over a sufficiently large period of time. Therefore this approach also benefits from the strength of global motion descriptors.

The proposed gesture descriptor is based on Histogram of Oriented Gradient (HOG) proposed by (Dalal & Triggs 2005) and is built through three steps:

- A preprocessing step in which we perform people detection and compute features (*e.g.* corners) to determine salient motion regions as described in section 4.2.
- The second step (section 4.3) consists of computing a 2D HOG descriptor for each feature selected in the previous step.
- In the last step (section 4.4), we track these 2D HOG Descriptors to build a Temporal HOG Descriptor identifying local motion of the 2D one. A Local Motion Descriptor (LMD) is then extracted from the Temporal HOG Descriptor.

4.2 Object Detection and Feature Selection

In order to compute the gesture descriptors, we are interested to compute feature points where significant and easy-to-track 2D descriptors can be extracted. In addition, we have to process independently the descriptors extracted from different mobile objects to characterize the motion of each of them. Thus, two preprocessing stages are compulsory before the gesture descriptors extraction: (1) People detection and (2) Feature extraction. In the following subsections we describe in details the two preprocessing stages.

4.2.1 People Detection

With good quality videos, human gesture recognition could be done directly using only gesture descriptors as proposed in this work. However, real world videos are often noisy and people are badly contrasted and often occluded. Also, such videos require preprocessing stage consisting of people detection and tracking. Thus, to detect efficiently human gestures, detecting and isolating individuals from each other in the input video sequences is often needed. So, a people classifier is used in this work to identify people and to assign features to each of them independently. More precisely, in order to improve people detection, we have chosen to use three levels of algorithm to achieve this task:

- A moving region detector.
- A people classifier.

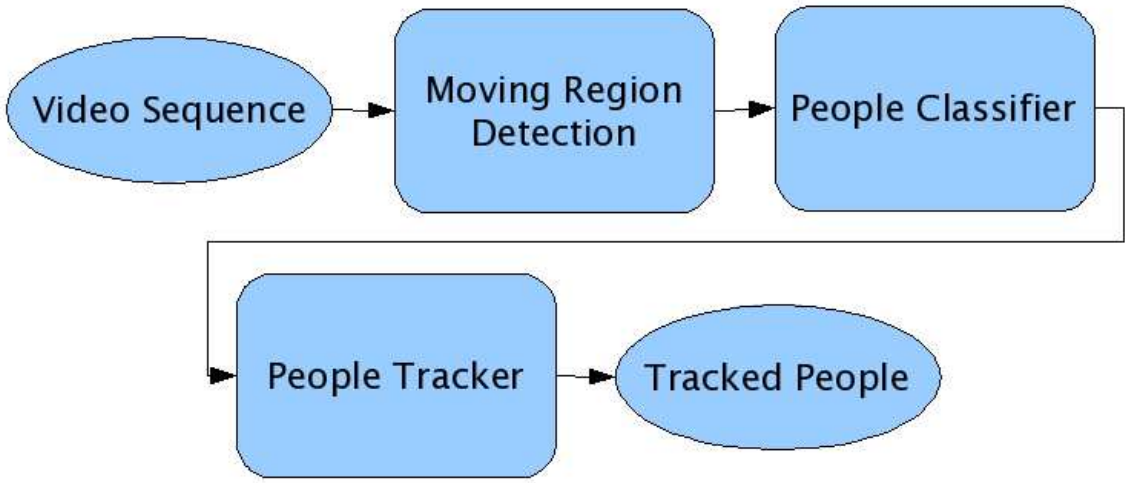


Figure 4.1: *People Detection Stage*

- A people tracker.

Figure 4.1 illustrates the people detection stage. Hereafter, we details the three levels of algorithm while motivating our choices.

Moving Region Detector

There are many methods to extract moving regions from a video sequence. Three categories are listed below:

1. Optical flow (Moeslund 2008): consists of computing optical flow (*i.e.* pattern of apparent motion of object pixels) between frames and selecting connected and coherent regions of motion as moving regions (*i.e.* motion different than average motion). Approaches based on optical flow are mainly used for mobile cameras, however they are computationally complex and expensive unless implemented on a graphical processing unit (GPU) or equivalent specialized hardware.

2. Temporal differencing (Moeslund 2008): consists of adaptative thresholding of the temporal difference between current frame and one or several previous frames and selecting the result as the boundary of moving regions. Approaches using this technique are suitable for dynamic environment, nonetheless the obtained shape is brittle and does not include all relevant features (usually only boundary of regions are detected).

3. Background subtraction (Moeslund 2008): This is the most used category in the literature and the most efficient for sequences captured with a fixed camera. Approaches from this category has to implement a background model and then subtract the reference image (*i.e.* background image) from the current image.

Since background subtraction is the most common category for moving region detection and the processed videos are captured from a fixed camera, we have chosen to use this method

as proposed by (Avanzi et al. 2005). In case of a mobile camera, a GPU-implemented optical flow method could be applied.

People Classifier

In order to determine the bounding boxes of a mobile person, we apply a people classifier (*c.f*. (Zuniga 2008)). Depending on people classifier, the approach can handle different levels of interaction and occlusion. The bounding boxes surrounding the people define a mask for feature point extraction. This step not only limits the search space of feature points but also segregates distinct moving regions: each mobile object is processed independently from others except when two objects are overlapping each other (*e.g*. people crossing). This enables to apply the gesture recognition process to different people in the scene until their movement is detected with overlap (partial or total occlusion). In case of interaction between two people (*e.g*. handshaking), we consider a new region resulting from the merging of the two initial regions corresponding to the interacting people. Eventually, a people tracker is used to separate the two persons while tracking them throughout the video.

People Tracker

A people tracker is required to improve the classification results and control the tracking of the 2D HOG Descriptor. Indeed, we are interested to delimit the tracking of the descriptors by looking mainly in the direction of the instantaneous speed of individuals. In addition, the individual speed is a parameter for the 2D HOG descriptor tracking algorithm. Various approaches of people tracking has been proposed (Avanzi et al. 2001, Bar-Shalom et al. 2007, Zuniga 2008). We expect from the people tracker to output the velocity (speed and movement direction) of the centroid for each tracked individual.

4.2.2 Feature Selection

Once people have been detected, we compute a set of features for each individual. Feature selection is a very difficult task since it deals with the detection of representative, repeatable and agregate information from images for dimentionality reduction purposes. There are four kinds of low-level features that can be selected from an image: Edges (first and/or second order gradients or phase congruency (Kovesi n.d.)), Corners (image curvatures), Blobs (region/patch analysis such as SIFT, Saliency) and Ridges (curves representing symmetry axis). We have chosen to implement gradient-based feature selectors (*e.g*. corners) since it is easier to detect motion in regions with high gradient. Corners are the finest features and they are generally associated to a local image patch around the feature in order to extract a descriptor (*e.g*. HOG, Fourier descriptors, moments) to improve their representiveness.

The masks generated by the previous step are used to define the image regions where to select features. Feature points are selected for each detected person using Shi-Thomasi corner detector (Shi & Tomasi 1994) or Features from Accelerated Segment Test (FAST) corner

detector (Rosten & Drummond 2006). Then, we sort the corners in descendant order according to the corner strength measurement provided by the corner detector. Finally, we select the most significant corners by ensuring a minimum distance between them. Thus, feature points enable us to localize points where descriptors have to be computed since they usually correspond to body parts where the movement can be discernible. Indeed, it is simpler to detect the movement of a descriptor located at a body corner (thanks to gradient variability and intensity contrast) than detecting the movement of a descriptor located inside the body where gradients are almost inexistent. The minimum distance between corners ensures that the computed descriptors will not overlaps and so will be independent (*i.e.* prevents tracking ambiguities). Thus, the processing time is reduced and the descriptors are better distributed over the body parts. Hereafter, we details the two chosen corner detectors.

Shi-Thomasi corner detector

Given an image I in gray-scale (corresponding to a moving region), we first compute gradients g_x (along the x axis) and g_y (along the y axis) by applying simple filters ($[-1\,0\,1]$ filter for g_x and $[-1\,0\,1]^t$ filter for g_y). This choice is justified since (Dalal & Triggs 2005) demonstrates that these filters are less processing time consuming than other gradient generators and are as efficient as any another one (*e.g.* Sobel operator, Derivative of Gaussian). Then, for each pixel p in the image and in a window (u, v) centered on the considered pixel, we compute the 2x2 Hessian matrix defined by the equation (4.1).

$$H_p = \sum_u \sum_v \begin{bmatrix} {g_x}^2 & g_x g_y \\ g_x g_y & {g_y}^2 \end{bmatrix} \tag{4.1}$$

After that, we compute the eigen values λ_1 and λ_2 of the Hessian matrix H_p (*i.e.* the roots of $det(H_p - \lambda I) = 0$): (Shi & Tomasi 1994) prove that $\min(\lambda_1, \lambda_2)$ is a better measure of the corner strengthness than the one given by the Harris corner detector.

FAST corner detector

The corner detector considers pixels in a Bresenham circle of radius r around the candidate point. If n contiguous pixels are all brighter (or all darker) than the considered point by at least t, then the candidate is considered to be a corner. Although r can in principle take any value, only a value of 3 is used (corresponding to a circle of 16 pixels circumference), and tests show that the best value of n is 9. This n value is the lowest one at which corners but not edges are detected. The resulting detector is reported to produce very stable features. Additionally, the ID3 algorithm (Quinlan 1990) (*i.e.* learning a decision tree) is used to optimize the order in which pixels are tested, resulting in the most computationally efficient corner detector available as claimed by the authors. The corner strength measure μ of a corner c is given by equation 4.2 ($\mathcal{B}$ is the Bresenham circle of radius $r = 3$ and $I(.)$ returns the gray scale intensity of the pixel in its argument).

$$\mu(c) = \sup\Big(\sum_{p \in \mathcal{B}, I(p) > I(c)} I(p) - I(c), \sum_{p \in \mathcal{B}, I(p) < I(c)} I(c) - I(p)\Big) \tag{4.2}$$

4.3 2D Gesture Descriptor

During the last three decades, several local descriptors have been proposed to represent local image structures and usually combined with one or several feature detectors.

To obtain a local descriptor, most of the state-of-the-art methods detect interest points and then consider for each interest point an image patch centered on it and from which a descriptor can be extracted. To build good descriptors from the patches, we must ensure four important properties:

- Distinctiveness: we expect that visually similar patches should have similar descriptors and that visually different patches should have different descriptors.
- Invariance: in addition, we can expect that despite a transformation (*i.e*. rotation, brightness change) two visually similar patches should still have similar descriptors. There are two ways to obtain such a descriptor: directly compute an invariant descriptor such as a normalized histogram or by applying some geometric and photometric normalizations on the patch and then extract the descriptor (*e.g*. correlogram).
- Robustness: also, visually similar patches should have similar descriptors despite the noise.
- Dimensionality: for efficiency and for generalization purpose (for tracking), we expect low dimensional descriptors (*e.g*. small number of histogram bins).

The most trivial local descriptor is a vector of patch pixels. Similarity score between two descriptors can be obtained by applying cross-correlation. Also, we can classify state-of-the-art descriptors into four categories:

- Distribution-based descriptors: using histograms of pixel intensities, of colors.
- Spatial-Frequency based descriptors: describe the frequency content of the image (such as Gabor transform).
- Differential descriptors: based on image derivatives.
- Miscellaneous descriptors: Van Gool's moments (Mindru et al. 2004, Van Gool et al. 1996).

We have chosen to use differential descriptors since they seem to outperform other types of descriptor and give the best accuracy-performance trade-off. The choice of local descriptors instead of global descriptors is justified by the fact that the formers can capture more detailed and specific information than the latters which agregate information. Indeed, local and specific motion can be detected by tracking local descriptors which is not necessary the case for global descriptors.

For each feature point, we define a neighborhood (a small square centering on the considered feature point) called a "patch". A 2D descriptor is computed for each patch. Our 2D descriptor is based on Histogram of Oriented Gradient (HOG) (Dalal & Triggs 2005). Thus,

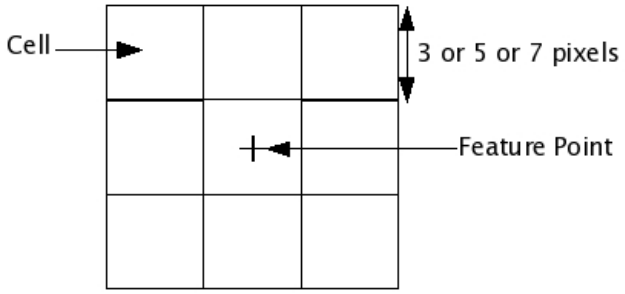


Figure 4.2: *A descriptor block*

we associate to each feature point a descriptor block composed of 3×3 cells; each of the cells has a pixel size of 3×3, 5×5 or 7×7. The feature point is the center of the center cell of the descriptor block as illustrated in figure 4.2. Let g_x and g_y the gradients computed in the previous step, we compute for all the pixels in the block the gradient magnitude g and the gradient orientation θ using respectively equation 4.3 and equation 4.4.

$$g(u,v) = \sqrt{g_x(u,v)^2 + g_y(u,v)^2} \tag{4.3}$$

$$\theta(u,v) = \arctan \frac{g_y(u,v)}{g_x(u,v)} \tag{4.4}$$

In order to be less dependent on background, we use the unsigned gradient orientation defined by equation 4.5. In addition, we choose to threshold the gradient magnitude, as defined by equation 4.6, to eliminate eventual noise.

$$\tilde{\theta}(u,v) = \begin{cases} \theta(u,v) + \pi & \text{if } \theta(u,v) < 0 \\ \theta(u,v) & \text{otherwise} \end{cases} \tag{4.5}$$

$$\tilde{g}(u,v) = \begin{cases} g(u,v) & \text{if } g(u,v) \geq T \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

For each cell $c_{ij}$where $(i,j) \in \{1,2,3\}^2$ in the block, we compute a feature vector f_{ij} by quantizing the unsigned orientation into K orientation bins weighted by the gradient magnitude as defined by equation 4.7.

$$f_{ij} = [f_{ij}(\beta)]^t_{\beta \in [1..K]} \tag{4.7}$$

where $f_{ij}(\beta)$ is defined by the equation 4.8.

$$f_{ij}(\beta) = \sum_{(u,v) \in c_{ij}} \tilde{g}(u,v)\delta[bin(u,v) - \beta] \tag{4.8}$$

The function $bin(u,v)$ returns the index of the orientation bin associated to the pixel (u,v) and the function $\delta[]$ is the Kronecker delta. Therefore, the 2D descriptor of the block is a

vector concatenating the feature vectors of all its cells normalized by the coefficient ρ which is defined in equation 4.9.

$$\rho = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{\beta=1}^{K} f_{ij}(\beta) \tag{4.9}$$

The dimension of this vector is noted dim; it corresponds to the cell number multiplied by the dimension of a cell feature vector (*i.e.* $9 \times K$). Note that each component of this descriptor takes its values in the interval $[0, 1]$. The decomposition of the descriptor block into cells improves the descriptor trackability. Indeed, each cell encapsulates a more local and specific information about the pixel neighborhood compared to the whole descriptor block.

4.4 Local Motion Descriptor

As explained previously, a gesture descriptor is a set of Local Motion Descriptors. Each Local Motion Descriptor is a compact representation of a Temporal 2D descriptor. Temporal 2D descriptors are built by tracking 2D descriptors. Let us suppose that we have detected a 2D descriptor d_{t-1} in the frame f_{t-1}, we are interested to determine the descriptor d_t in the frame f_t which can be identified to d_{t-1}. Two tracking solutions are considered: the former is based on feature point tracking (considering only the center of the descriptor) and the latter is based on tracking the whole descriptor.

Concerning feature point tracking, the KLT algorithm (Shi & Tomasi 1994) can be used. The main challenge for feature tracking is to define **where** and **how** to search for features in next frames once they have been detected. The way an algorithm answers to these questions influences the algorithm's robustness w.r.t. noise and environment changes. For example, the KLT algorithm tries to solve a displacement equation based on gradient temporal difference and grayscale temporal difference for each considered feature point which results in a noise dependent algorithm.

As for tracking 2D descriptor as a whole, we have developed a new tracking algorithm based on the least square method using Kalman filter.

This section details the proposed tracking algorithm and the resulting local motion descriptors. In subsection 4.4.1, we introduce the tracking assumptions and some descriptor metrics to be used by the tracker. Subsection 4.4.2 presents the tracking algorithm which effectively tracks (*i.e.* measures) the descriptor. Subsection 4.4.3 details the Kalman filtering process. The last two subsections explain respectively how to build/compute Temporal 2D Descriptors and Local Motion Descriptors.

4.4.1 Temporal filtering of a Descriptor

Before describing the descriptor tracking algorithm, we need to define metrics and the procedure to filter the descriptor throughout the sequence. First of all, we define a quadratic

error function $\mathcal{E}$ that measures the dissimilarity between two given descriptors $d^{(n)}$ and $d^{(m)}$ according to equation 4.10:

$$\mathcal{E}(d^{(n)}, d^{(m)}) = \sum_{i=1}^{9\times K} (d_i^{(n)} - d_i^{(m)})^2 \tag{4.10}$$

where $d_i^{(n)}$ and $d_i^{(m)}$ are respectively the i^{th} component of the descriptors $d^{(n)}$ and $d^{(m)}$. Note that the maximal value of the quadratic error function is $9 \times K$ since all the components take their values in $[0, 1]$. Let f_1 the frame (which can be any frame in the video sequence) in which we detect for the first time a descriptor $d^{(1)}$ at position x_1 relatively to the centroid of the correspondent individual. Assume, for instance, that we want to estimate the position x_t of this descriptor in the frame f_t given the previous estimated positions and constrained by the quadratic error function value between $d^{(t)}$ and $d^{(1)}$. This error function is defined to be less than 1% of the maximal value of the function *i.e.* $\frac{9\times K}{100}$. Let d the state of the descriptor (*i.e.* the unknown true value of the descriptor), we want to estimate $\hat{d}$ such that the least square error between measurements (*i.e.* $d^{(1)}, ..., d^{(t)}$) and the state is minimum. The equation 4.11 defines the square error $\mathcal{C}$ between the sate d of the descriptor and the measurements $d^{(1)}, ..., d^{(t)}$.

$$\mathcal{C} = \frac{1}{2}\sum_{i=1}^{t}(d^{(i)} - d)^2 \tag{4.11}$$

The least square error is obtained when the derivative of the square error is equal to zero:

$$\frac{\partial \mathcal{C}}{\partial d} = 0 = \sum_{i=1}^{t}(d^{(i)} - \hat{d}) = \sum_{i=1}^{t} d^{(i)} - t \times \hat{d} \tag{4.12}$$

So, the estimated state is:

$$\hat{d} = \frac{1}{t}\sum_{i=1}^{t} d^{(i)} \tag{4.13}$$

Since we aim at estimating the state of the descriptor at each step of the tracking, we use the recursive least square method by applying the results of equation 4.17. The estimation at a step t of the tracking is:

$$\hat{d}^{(t)} = \frac{1}{t}\sum_{i=1}^{t} d^{(i)} = \frac{1}{t}\sum_{i=1}^{t-1} d^{(i)} + \frac{1}{t} d^{(t)} \tag{4.14}$$

Hence the estimation at the step $t - 1$ is:

$$\hat{d}^{(t-1)} = \frac{1}{t-1}\sum_{i=1}^{t-1} d^{(i)} \tag{4.15}$$

We deduce that:

$$\hat{d}^{(t)} = \frac{t-1}{t}\hat{d}^{(t-1)} + \frac{1}{t} d^{(t)} \tag{4.16}$$

Which can be reformulated as:

$$\underbrace{\hat{d}^{(t)}}_{\text{estimate at step t}} = \underbrace{\hat{d}^{(t-1)}}_{\text{estimate at step t-1}} + \overbrace{\frac{1}{t}}^{\text{Gain}} \underbrace{(\overbrace{d^{(t)}}^{\text{Actual measure}} - \overbrace{\hat{d}^{(t-1)}}^{\text{Predicted measure}})}_{\text{Innovation}} \tag{4.17}$$

Here the gain specifies how much do we pay attention to the difference between what we expected and what we actually get. Note that the gain decreases while the tracking advance which means that we become more and more confident in the estimation while the tracking progress. This decision implies that if there is some changes in appearance (*i.e.* illumination changes) the descriptor may be lost. However, we believe that this is not a major inconvenient because new descriptors can be generated. The main advantages of this choice is to prevent mixing of two descriptors and to ensure the tracking of the same descriptors. To stop the tracking of the descriptor, the constraint using the estimate state of the descriptor becomes:

$$\mathcal{E}(\hat{d}^{(t)}, \hat{d}^{(1)}) \leq \frac{9 \times K}{100} \tag{4.18}$$

The actual measure of the descriptor d_t is obtained by computing the descriptor on the new tracked position.

Hereafter, we detail the tracking of the descriptor. In order to track the descriptor position, we use a Kalman filter with a motion model for prediction (Kalman 1960). The state vector $\mathcal{X}$ used for tracking is defined by equation 4.19.

$$\mathcal{X} = [\ p_x\ p_y\ v_x\ v_y\ d\ ]^T \tag{4.19}$$

where (p_x, p_y) is the position of the descriptor, (v_x, v_y) is its velocity and $\hat{d}$ is the estimation of its value given by equation 4.17. A candidate descriptor (which has not been tracked yet) is described by a measurement vector $\mathcal{Z}$ as defined in equation 4.20.

$$\mathcal{Z} = [\ p_x\ p_y\ d\ ]^T \tag{4.20}$$

Equation 4.21 define a distance $\mathcal{D}$ between a candidate descriptor $\mathcal{Z}$ in the current frame and a tracked descriptor $\mathcal{X}$ through previous frame.

$$\mathcal{D}(\mathcal{Z}, \mathcal{X}) = \alpha \frac{\sqrt{\mathcal{E}(\mathcal{Z}_d, \mathcal{X}_{\hat{d}})}}{\sqrt{||\mathcal{Z}_d|| + ||\mathcal{X}_{\hat{d}}|| + 1}\sigma_{\hat{d}}} + \beta \frac{||\mathcal{Z}_p - \mathcal{X}_p||}{\sigma_p} \tag{4.21}$$

where α and β are empirically derived weight parameters, $\mathcal{Z}_{\hat{d}}$, $\mathcal{X}_{\hat{d}}$, $\mathcal{Z}_p$, $\mathcal{X}_p$ are respectively the estimated value and the position of descriptors $\mathcal{Z}$ and $\mathcal{X}$, and $\sigma_{\hat{d}}$ and σ_p are the covariance parameters extracted from the Kalman filter's covariance matrix. The first term of the distance represents the scaled difference between the two descriptor values and the second term represents the difference between the candidate descriptor location and predicted descriptor location given by the Kalman filter scaled by the standard deviation of the position. The confidence $\mathcal{C}$ in the candidate descriptor $\mathcal{Z}$ to correspond to the tracked descriptor $\mathcal{X}$ is defined by equation 4.22.

$$\mathcal{C}(\mathcal{Z}, \mathcal{X}) = \frac{1}{1 + \mathcal{D}(\mathcal{Z}, \mathcal{X})} \tag{4.22}$$

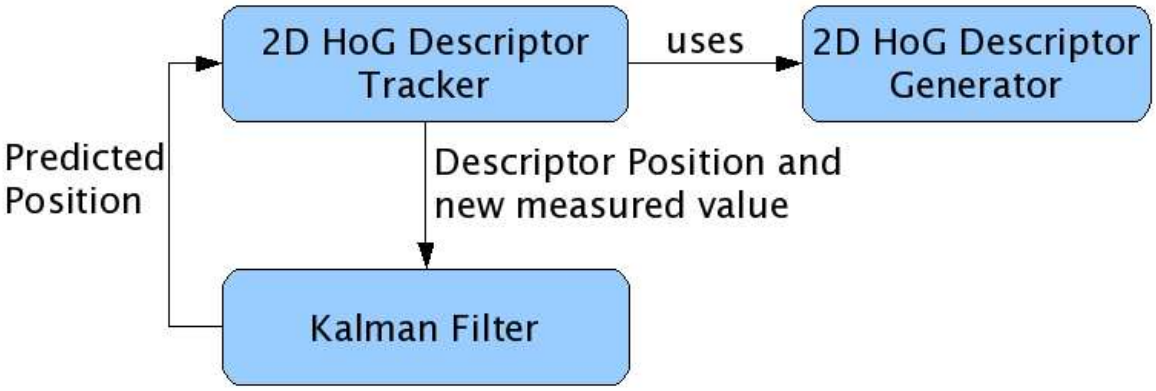


Figure 4.3: *2D HOG descriptor tracking using extended Kalman filter*

The basic idea of a Kalman filtering based tracker is to recursively estimate the state vector given the last estimate and a new measurement which is made by the traditional tracker. Figure 4.3 illustrates the tracking algorithm of 2D HOG Descriptors. For a newly detected descriptor, the tracking evolves in three steps:

1. Initialization (at the first frame where the descriptor is detected): We take the initial position given by the feature detector (*e.g.* corner point) at the position of the descriptor and the mobile velocity as its initial velocity and we compute the descriptor to build the initial state vector $\mathcal{X}^{(1)}$. We consider initially a big error tolerance $P^{(0)}$ which is defined in the subsection 4.4.3.

2. Prediction (for the next frames): We use the Kalman filter to predict the relative descriptor position and consider it as search center for the tracking algorithm.

3. Correction (for the next frames): We recover the measured position from the tracking algorithm and use it to carry out the state correction using the Kalman filter.

Steps 2 and 3 are carried out while the tracking runs. In the following subsections, we describe respectively the tracking algorithm, the Kalman filtering and the construction of the temporal 2D descriptor.

4.4.2 The HOG descriptor tracking algorithm

The tracking algorithm consists of searching the next position of the HOG descriptor knowing its last position in the previous frame and the predicted position from the Kalman filter. The search procedure is a downhill search (*i.e.* starts from the center and goes to the boundaries by exploring all possibilities) around the predicted position by minimizing the quadratic error function. Note that for computational enhancements, we should investigate heuristic search instead of exhaustive exploration. Thus, we determine the search regions for next measurements using last states, predicted states and state uncertainties and then we make new measurements (*i.e.* compute new HOG descriptors) for all pixels in the search

regions. The confidence measure (as defined in equation 4.22) is used to select the best solution, if the downhill search gives several solutions.

For each descriptor, the tracking algorithm of 2D HOG descriptors is run as described hereafter. Where the $ellipse$ procedure returns the ellipse with foci $(\hat{\mathcal{X}}^{(t-1)}, \hat{\mathcal{X}}_{-}^{(t)})$ and with

Algorithm 2 *2D HOG descriptor tracking algorithm*

Require: $\hat{\mathcal{X}}^{(t-1)}, \hat{\mathcal{X}}_{-}^{(t)}, P_t^-$ {Last estimated state, predicted state and error covariance matrix}
Ensure: $\mathcal{Z}^{(t)}$ {The actual measure}
1: $R \leftarrow ellipse(\hat{\mathcal{X}}^{(t-1)}, \hat{\mathcal{X}}_{-}^{(t)}, P_t^-)$ {Compute the search region}
2: $\mathcal{S} \leftarrow \emptyset$ {Initialize the set of candidate measures}
3: **for all** $\mathcal{Z} \in R$ **do**
4: **if** $\mathcal{E}(\mathcal{Z}, \hat{\mathcal{X}}_{-}^{(t)}) < \frac{9\,K}{100}$ **then**
5: $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathcal{Z}\}$
6: **end if**
7: **end for**
8: $maxConfidence \leftarrow 0$
9: **for all** $\mathcal{Z} \in \mathcal{S}$ **do**
10: $confidence \leftarrow \frac{\mathcal{C}(\mathcal{Z}, \hat{\mathcal{X}}_{-}^{(t)}) + \mathcal{C}(\mathcal{Z}, \hat{\mathcal{X}}^{(t-1)})}{2}$
11: **if** $confidence > maxConfidence$ **then**
12: $maxConfidence \leftarrow confidence$
13: $\mathcal{Z}^{(t)} \leftarrow \mathcal{Z}$
14: **end if**
15: **end for**

eccentricity e defined by equation 4.23.

$$e = \frac{c}{a} \tag{4.23}$$

where c is the focal distance (*i.e.* half of the distance between the two foci) and a is the semi-major axis of the ellipse which is computed as defined by formula 4.24.

$$a = \sqrt{c^2 + b^2} \tag{4.24}$$

where b is the semi-minor axis computed as defined by formula 4.25.

$$b = c + \sqrt{\sigma_x^2 + \sigma_y^2} \tag{4.25}$$

where σ_x and σ_y are the variance extracted from the covariance matrix P_t^-. Note that $b \geq c$ which implies that $b^2 + c^2 > 2\,c^2$ and thus $a \geq \sqrt{2}\,c$ (using equation 4.24). This ensures that the search region has a minimum size according to the focal distance which is the half of the predicted motion of the descriptor.

The presented tracking algorithm is controlled by the Kalman filter: It is used as non-linear measurement module of the current state of the descriptor. The algorithm reduces

the search region using the predicted state and a priori covariances from the Kalman filter. Compared to the KLT tracker, the algorithm assumes finest match criterion since the HOG descriptor difference in a search region relies much more on distinctiveness than the displacement equation of the KLT tracker. However, processing time can be improved by investigating an heuristic search.

4.4.3 The Kalman filter

We use the extended version of the Kalman filter since the dynamic model of the measurements is not linear. The extended Kalman filter is derived from two equations:

1. The state equation: also called the process equation, it relates the current estimation of the sate vector $\mathcal{X}^{(t+1)}$ with the last estimated state vector $\mathcal{X}^{(t)}$:

$$\mathcal{X}^{(t+1)} = f(t, \mathcal{X}^{(t)}) + w^{(t)} \tag{4.26}$$

where $w^{(t)}$ is a tweak factor also called process noise which is assumed to be additive, white and centered Gaussian (*i.e.* with zero mean) with covariance matrix defined by equation 4.27.

$$E[w^{(s)}\, w^{(t)^T}] = \begin{cases} Q_t & \text{for } s = t \\ 0 & \text{otherwise} \end{cases} \tag{4.27}$$

The function $f(t, \mathcal{X}^{(t)})$ is the transition function taking the state vector form time t to time $t + 1$. This function gives a model of motion of the 2D HOG descriptor for the prediction step of the filtering. As for our experiments, we consider three motion models for the 2D HOG descriptor:

- A linear motion (random-walk) model as given by equation 4.28. Linear motion models do not perform very well if the acceleration is variable particulary when the motion orientation changes (*i.e.* strong angular acceleration changes).

$$f = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.28}$$

- A Brownian motion model which performs better for motion with variable acceleration.

$$f = \begin{pmatrix} \exp(-\frac{1}{4}(p_x + 1.5v_x)) \\ \exp(-\frac{1}{4}(p_y + 1.5v_y)) \\ \exp(-\frac{1}{4}v_x) \\ \exp(-\frac{1}{4}v_y) \\ d \end{pmatrix} \tag{4.29}$$

2. The measurement equation: it relates the measurement $\mathcal{Z}^{(t)}$ (*i.e.* HOG descriptor and its position) to the state vector $\mathcal{X}^{(t)}$ at a given instant:

$$\mathcal{Z}^{(t)} = h(t, \mathcal{X}^{(t)}) + v^{(t)} \tag{4.30}$$

where $v^{(t)}$ is the measurement noise which is assumed to be additive, white and centered Gaussian with covariance matrix defined by equation 4.31. Moreover, it is supposed to be uncorrelated with the tweak factor $w^{(t)}$.

$$E[v^{(s)}\, v^{(t)^T}] = \begin{cases} R_t & \text{for } s = t \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

The function $h(t, \mathcal{X}^{(t)})$ is the measurement function which gives the actual measure of the state vector. This function represents the HOG descriptor tracking algorithm (*i.e.* h value is the measurement given by the tracking algorithm) and is obviously non-linear.

The first step for deriving the extended Kalam filter is to linearize the transition and the measurement functions by approximating (first-order Taylor development) them with their respective Jacobian matrix as defined by equations 4.32 and 4.33.

$$F_{t+1,t} = \left.\frac{\partial f(t, \mathcal{X}^{(t)})}{\partial \mathcal{X}}\right|_{\mathcal{X}=\hat{\mathcal{X}}_-^{(t)}} \tag{4.32}$$

$$H_t = \left.\frac{\partial h(t, \mathcal{X}^{(t)})}{\partial \mathcal{X}}\right|_{\mathcal{X}=\hat{\mathcal{X}}_-^{(t)}} \tag{4.33}$$

where $\hat{\mathcal{X}}_-^{(t)}$ is the prediction of the state vector given by applying the state equation on the last estimate of the state vector ($\hat{\mathcal{X}}_-^{(t)} = f(t, \hat{\mathcal{X}}^{(t-1)})$). The elements of the two Jacobian matrices are computed at each iteration given the last estimate $\mathcal{X}^{(t-1)}$ and the predicted new estimate $\mathcal{X}_-^{(t)}$. Note that, in the linear case $F_{t+1,t}$ is equal to f.

The Jacobian matrix of the measurement function defined in equation 4.33 is developped as given by equation 4.34.

$$H_t = \begin{pmatrix} \frac{\partial h_x}{\partial p_x} & \frac{\partial h_x}{\partial p_y} & \frac{\partial h_x}{\partial v_x} & \frac{\partial h_x}{\partial v_y} & \frac{\partial h_x}{\partial d} \\ \frac{\partial h_y}{\partial p_x} & \frac{\partial h_y}{\partial p_y} & \frac{\partial h_y}{\partial v_x} & \frac{\partial h_y}{\partial v_y} & \frac{\partial h_y}{\partial d} \\ \frac{\partial h_d}{\partial p_x} & \frac{\partial h_d}{\partial p_y} & \frac{\partial h_d}{\partial v_x} & \frac{\partial h_d}{\partial v_y} & \frac{\partial h_d}{\partial d} \end{pmatrix} \tag{4.34}$$

Where h_x,h_y and h_d are the three components of $h(t, \mathcal{X}^{(t)})$. Since $\frac{\partial h_x}{\partial p_y} = \frac{\partial h_y}{\partial p_x} = \frac{\partial h_x}{\partial v_y} = \frac{\partial h_y}{\partial v_x} = \frac{\partial h_x}{\partial d} = \frac{\partial h_y}{\partial d} = \frac{\partial h_d}{\partial p_x} = \frac{\partial h_d}{\partial p_y} = \frac{\partial h_d}{\partial v_x} = \frac{\partial h_d}{\partial v_y} = 0$, $\frac{\partial h_d}{\partial d} = 1$ and assuming that $v_x^h = \frac{\partial h_x}{\partial p_x}$, $v_y^h = \frac{\partial h_y}{\partial p_y}$, $a_x^h = \frac{\partial^2 h_x}{\partial^2 p_x}$ and $a_y^h = \frac{\partial^2 h_y}{\partial^2 p_y}$, H_t is then given by this formula:

$$H_t = \begin{pmatrix} v_x^h & 0 & \frac{v_x^h}{a_x^h} & 0 & 0 \\ 0 & v_y^h & 0 & \frac{v_y^h}{a_y^h} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.35}$$

The second step is to apply the dynamic filtering equation 4.36 which incorporates the static filtering equation 4.17, is derived from the state and measurement equations by minimizing the expectation of the square error between measurements and state.

$$\underbrace{\hat{\mathcal{X}}^{(t)}}_{\text{estimate at step t}} = \underbrace{\hat{\mathcal{X}}_{-}^{(t)}}_{\text{Predicted state at step t}} + \overbrace{G_t}^{\text{Gain}} \underbrace{(\overbrace{\mathcal{Z}^{(t)}}^{\text{Actual measure}} - \overbrace{h(t, \hat{\mathcal{X}}_{-}^{(t)})}^{\text{Predicted measure}})}_{\text{Innovation}} \tag{4.36}$$

where G_t is the gain defined by equation 4.37, $h(t, \hat{\mathcal{X}}_{-}^{(t)}) = H_t \hat{\mathcal{X}}_{-}^{(t)}$ is the predicted measure and $Z^{(t)}$ is the actual measure provided by the tracking algorithm.

$$G^t = \frac{P_t^- \, H_t^T}{H^t \, P_t^- \, H_t^T + R_t} \tag{4.37}$$

where P_t^- is the a priori covariance matrix for prediction error defined by equation 4.38 which is computed from the last posteriori covariance matrix for the estimation of error P_{t-1} as defined by equation 4.39 at instant t.

$$P_t^- = F_{t,t-1} \, P_{t-1} \, F_{t,t-1}^T + Q_t \tag{4.38}$$

$$P_t = (I - G_t \, H_t) P_t^- \tag{4.39}$$

where I is the identity matrix. The value of the initial posteriori covariance matrix P_0 is given by formula 4.40.

$$P_0 = \begin{pmatrix} 0 & 0 & e_x & 0 & 0 \\ 0 & 0 & 0 & e_y & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.40}$$

Thus, we assume that we are sure about descriptor position but unsure about its velocity. The errors e_x and e_y on velocity components are empirical standard deviations of the velocity components. Figure 4.4 illustrates the Kalman filtering process.

The process noise and the measurement noise are the parameters of the filter and their values will be discussed in chapter 6. The proposed filter suffers from the drawbacks of the extended Kalman filter. First, unlike the linear variant, the filter is not an optimal estimator (*i.e.* not defined as the maximization of the likelihood). Moreover, if the initial state is not precise or the parameters are not adequate the filter can quickly diverge due to the linearization approximation. Furthermore, the used linear motion model allows that great speed becomes wrongly likely since the process noise increases the speed variance while the tracking progress (*c.f.* the IOU Model and MTST in (Washburn 2007)). Another variant of Kalman filter (*i.e.* unscented Kalman filter) is to be considered since it provides a better approximation for the non-linearity than the first-order Taylor expansion used in the extended Kalman filter (Julier & Uhlmann 1997).

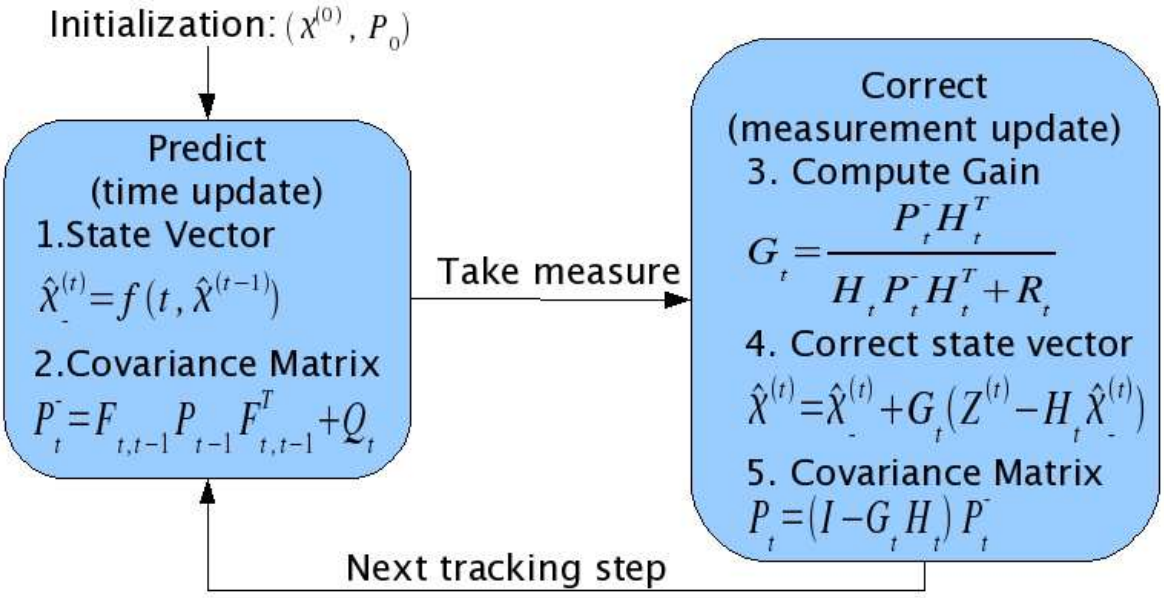


Figure 4.4: *The extended Kalman filtering process*

4.4.4 The temporal 2D descriptor

The temporal 2D descriptor is the vector obtained by the concatenation of the final descriptor estimate $\hat{d}$ and the positions of the descriptor during the tracking process. The dimension of this vector is $9 \times K + 2 \times \ell$ where ℓ is the number of the 2D tracked positions. Assuming that $\mathcal{T}^d = [(x_1, y_1), ..., (x_\ell, y_\ell)]^T$ is the array of the descriptor d locations, then the temporal 2D descriptor is the heterogeneous vector $\mathcal{V} = [\hat{d}\ \mathcal{T}^d]^T$. Note that it encapsulates at the same time the texture information (*i.e.* normalized histogram of oriented gradient) and the motion information (*i.e.* associated movement of the textured region) which can be significantly large. Hereafter, we detail how to build a homogeneous and dense vector to model this local motion named Local Motion Descriptor.

4.4.5 Local Motion Descriptor

Generally, standard learning-classification frameworks need a feature space which can be reprensented as a vectorial space with a unique, finite and fixed dimension. The set of temporal 2D descriptors does not match these requirements. Indeed, the descriptors are heterogenous with variable size and variable component domains. Hence, we need to transform and condense this set into a homogeneous and dense set which can be seen as a vectorial space with finite dimension. To satisfy this, a local motion descriptor is extracted from each temporal 2D descriptor. Given the trajectory array $\mathcal{T}^d$, we define the line trajectory vector $\mathcal{L}$ as:

$$\mathcal{L}^d = [(w_1, h_1), ..., (w_{\ell-1}, h_{\ell-1})]^T \tag{4.41}$$

where $w_i = x_{i+1} - x_i$ and $h_i = y_{i+1} - y_i$. The trajectory orientation vector $\Theta^d = [\theta_1, ..., \theta_{\ell-2}]^T$ is computed thanks to the formula defined by equation 4.42.

$$\forall i \in [1, \ell - 2];\ \theta_i = \arctan(h_{i+1}, w_{i+1}) - \arctan(h_i, w_i) \tag{4.42}$$

where the $\arctan$ function returns the orientation of the given line with respect to the x axis. Since $-2\pi \leq \theta_i \leq 2\pi$, we normalize the vector by dividing all its components by 2π. The resulting vector is noted $\tilde{\Theta}^d$. The local motion descriptor is defined as the concatenation of the descriptor estimation $\hat{d}$ which indicates the texture involved in the motion and the normalized trajectory orientation vector $\tilde{\Theta}^d$ which represents the motion. Its dimension is $9 \times K + \ell - 2$. At this level, the descriptor components have the same domain which is $[-1, 1]$ even though the texture components are positive values. Moreover, the motion representation is invariant under direct image plan similarities (*e.g.* direct isometries and positive homothecies) and covariant under indirect ones (*e.g.* indirect isometries and negative homothecies). However, the size of the descriptors still variable. To reduce the dimension of local motion descriptors and uniformize their size, we apply Principal Component Analysis (PCA) and project the θ_i on the three first principal axis $\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3$. Also known as the Karhunen Loeve (KL), PCA uses factorization to transform a set of data to a dense representation according to their statistical properties (*c.f.* (Jolliffe 2002)). Thus we get a final local motion descriptor $\hat{\Theta}^d = [d\ \hat{\theta}_1\ \hat{\theta}_2\ \hat{\theta}_3]^T$ which is invariant to 2D scale and rotation changes.

4.5 Conclusion

We have presented in this chapter how to build the local Motion descriptors. A Gesture Descriptor is a set of local motion descriptors which characterizes the correspondant gesture. The proposed descriptor is invariant in the image plane and agregates at the same time the local texture and its respective motion. It combines the strength of local and global motion descriptors thanks to tracking HOG descriptors. Indeed, the spatial position of the descriptor is not fixed and the resulting trajectory of the local descriptor represents faithfully the local motion which contrasts with traditional local descriptors consisting of arbitrary small time-volume of 2D descriptors with fixed spatial position. We have introduced a new algorithm for tracking 2D HOG descriptors which is an interesting contribution for feature tracking since it proposes to track more robustly the feature points by associating them to HOG descriptors and track the latters instead of the formers.

Chapter 5

Learning and Classification algorithms

Learning from experience is a faculty almost never practiced.

Barbara Tuchman

5.1 Introduction

Recognizing human gestures from video sequences is generally composed of two tasks: (1) gesture descriptor generation which includes motion tracking and (2) the decision/inference process where gestures are effectively recognized. These tasks are mutually related since the choice of a particular gesture representation influences those of the inference process and vice-versa. We have proposed to recognize gestures using an appearance based model: Local Motion Descriptors (LMDs) as detailed in the previous chapter. The proposed gesture representation is based on local descriptors of local motion. Since the complexity and the limits of automaton-based inference processes, we believe that the adequate inference process for local descriptors representation comes to be a learning-classification process. In fact, this kind of process is simpler to extend and more flexible according to the gesture variability as seen in chapter 2.

In this chapter, we explain the proposed decision/inference process for gesture recognition which consists of a learning-classification framework. Indeed, we use the bag of words paradigm in order to categorize the LMDs according to the gestures that we want to recognize. First, for each video in the training data-set, we generate all LMDs and annotate them with the corresponding gesture. Second, we cluster into k clusters the LMDs of each training video by applying the k-means algorithm. The k parameter is set up empirically. Then, each cluster is associated to its corresponding gestures, so similar clusters can be labeled with different gestures. Finally, by taking all generated clusters as a database, the k-nearest neighbors algorithm is run to classify gestures occurring in the test data-set. A video is classified according to the amount of neighbors which have voted for a given gesture providing the likelihood of the recognition. We also discuss the extension of this learning-classification framework with the Maximization of Mutual Information (MMI) algorithm for the learning step and with Support Vector Machine (SVM) algorithm for the classification step.

Section 5.2 introduces the general structure of the learning-classification framework for the on-line recognition process. Section 5.3 presents the learning step of gestures and its extension with MMI algorithm. Section 5.4 discusses the classification step and its extension with SVM algorithm.

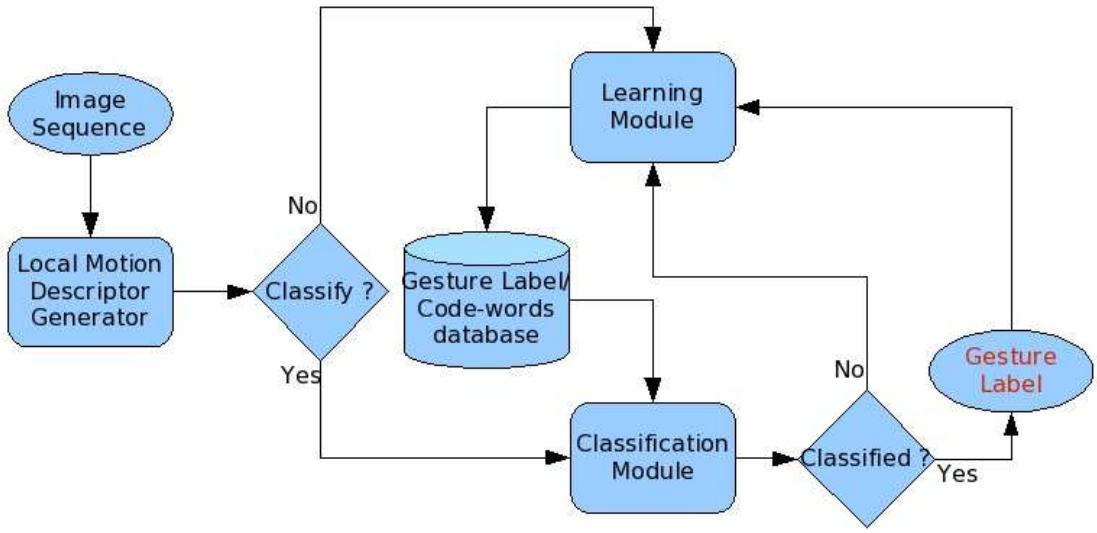


Figure 5.1: *The learning-classification framework: in the first diamond "Classify ?", we check whether we want to classify or to learn the given gesture; in the second diamond "Classified ?" we check whether we have succeed or not to classify the corresponding gesture.*

5.2 General Framework

In order to recognize human gestures from video sequences, we propose to use a learning-classification framework associated with the algorithm of local motion descriptor generation (*c.f.* previous chapter) as described by figure 5.1. Indeed, we choose to use a learn-and-predict strategy which consists of trying to classify a test sample from the training data-set or integrate it to the learning process if it fails to do so. Then, the new test sample can be considered as a classified one or can serve to update the training data-set.

Recently, (Signer et al. 2007) proposed a java based general framework for pen-based gesture recognition. We propose to adapt this framework to human body gesture recognition from video sequences. The main problem of designing such framework is the extensibility and cross-application re-usability. To deal with these constraints, it is important to provide a generic way for gesture representation. In our case, we choose to model gestures as proposed by the figure 5.2. The general architecture ensures that a gesture can be represented by several descriptors and eventually several descriptor clusters (a cluster is represented by its centroid which is a descriptor). Each recognition algorithm contains a set of predefined gestures that it can recognize. A gesture is characterized by its name and the set of local motion clusters associated to it. A local motion cluster is a set of local motion descriptors represented by their mean and standard deviation. We also define a recognition algorithm model that uses the gesture model in order to recognize gesture (*i.e.* learn and classify). A general abastraction of each type of algorithms (*i.e.* gesture representation generation, learners and classifiers) is provided to define the general interaction between the conerned type of algorithms and the gesture recognition module. This model was designed to fit into the SUP platform (*i.e.* former VSIP (Avanzi et al. 2005)) and is overviewed in figure 5.3.

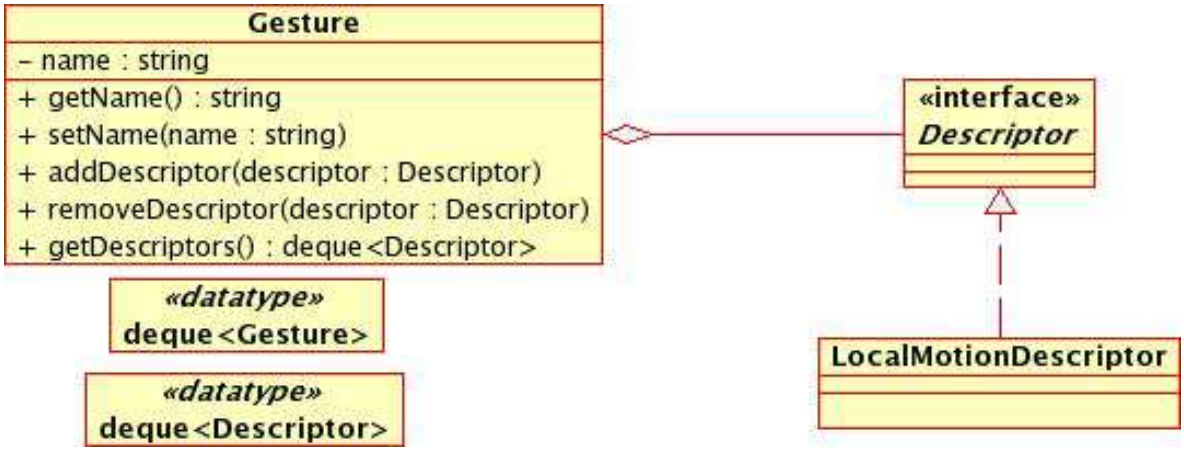


Figure 5.2: *Gesture Model for learning-classification framework: A deque is a double ended queue that modelize a set of instances of the Class passed in its generic parameter.*

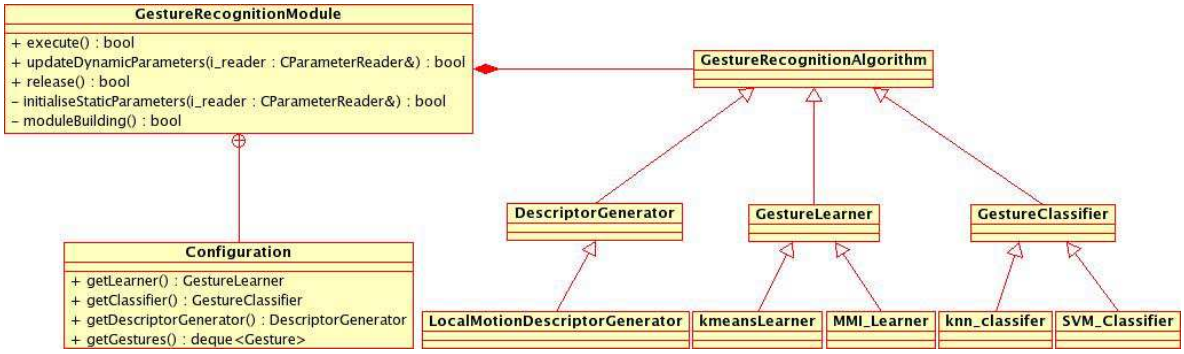


Figure 5.3: *Recognition algorithm model for gesture learning and classifying*

A gesture configuration algorithm (*i.e.* configuration class) handle the intitialization of the gesture recognition module and all subsequent chosen algorithms. So , we ensure that the gesture recognition module can be configured to run different versions of learners, classifiers and eventually use different gesture representation generation.

5.3 Gesture Learning and Codebook generation

For learning gestures, we assume that the training data-set is built of video sequences, each of them contains one and only one gesture instance. In order to build local motion descriptor cluster, each training video is annotated with the corresponding gesture label. For each training video sequence, we compute local motion descriptors and associate each of them with the gesture label of the video sequence. Then, we apply the k-means algorithm in order to group these descriptors into clusters which we call "video-words". The k-means algorithm needs a similarity measure (*i.e.* a distance) for comparing two different local motion descriptors. We choose to use the Euclidean distance as a similarity measure since it seems to be more adequate for texture distance and the normalized motion representation.

Indeed, we have tested several distances and the one that performs best is the Euclidean distance (*c.f*. next chapter).

Thus, given as input all the training video sequences, the k-means algorithm, described in algorithm 3, consists of classifying the set S of local motion descriptors (*i.e*. feature vectors) into k classes (*i.e*. clusters) $(S_i, i = 1, 2, ..., k)$ corresponding to video-words. The value of k is empirically chosen so that is large enough to describe correctly the set of the m gestures to learn ($k > 3 \times m$) and strictly less than the total number of local motion descriptors. The lower bound for the choice of k is justified by analyzing the videos illustrating gestures and according to the state of the art: gestures are usually composed of three units of coherent motion (*i.e*. pre-stroke, stroke and post-stroke) and in our representation, these units of motion correspond to local motion descriptor clusters (*c.f*. next chapter for k choice). The goal of k-means algorithm is to minimize the total variance between the k clusters or the quadratic error function (5.1).

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{5.1}$$

Where μ_i is the centroid of the cluster S_i and x_j are the local motion descriptors belonging to S_i.

A video-word (a cluster) is characterized by its centroid (the mean) and its standard deviation which can be computed using the cluster membership map. Then, each video-word is annotated by the corresponding gestures (the union of labels of the cluster members).

Once we have obtained the video-words, an optional step is to reduce their dimensionality by compacting them into code-words. To achieve this goal, we propose to apply Maximization of Mutual Information (MMI) algorithm (Liu & Shah 2008) on the clusters generated by the k-means algorithm. Let C be the discrete random variable of video-words: the centroids of the generated clusters $C \in \mathcal{C} = [\mu_1..\mu_k]$. Let G be the discrete random variable of gesture labels $G \in \mathcal{G} = [g_1..g_m]$. With the cluster membership map $A : S \rightarrow \mathcal{C}$, we define the conditional probability distributions $P(C|G)$ and $P(G|C)$ by equations 5.2 and 5.3.

$$P(C = \mu_i | G = g_i) = \frac{Card(Label^{-1}(g_i) \cap A^{-1}(\mu_i))}{Card(Label^{-1}(g_i))} \tag{5.2}$$

$$P(G = g_i | C = \mu_i) = \frac{Card(Label^{-1}(g_i) \cap A^{-1}(\mu_i))}{Card(A^{-1}(\mu_i))} \tag{5.3}$$

Where the function $Label$ is the map between feature vectors (*i.e*. local motion descriptors) and gesture labels; $Card(.)$ is the cardinal operator. Note that the cluster membership map A (repectively the label function $Label$) is a surjection but not an injection form S to $\mathcal{C}$ (respectively from S to $\mathcal{G}$) and therefore $A^{-1}(\mu_i)$ (respectively $Label^{-1}(g_i)$) returns the preimage of the cluster μ_i (respectively of the gesture label g_i) which is a subset of S defined by $\{x \in S / A[x] = \mu_j\}$ (respectively $\{x \in S / Label(x) = g_i\}$).
By taking as definition of the marginal distributions of C and G the formulas 5.5 and 5.6, we

Algorithm 3 *k-means clustering algorithm*

Require: k {the number of clusters}
S {the set of feature vectors}
$sim(x, y)$ {the similarity function}
Ensure: $\mu_i, i \in [1..k]$ {the centroids of the k clusters}
A {the cluster membership map}

$S' \leftarrow S$
for $i \leftarrow 1$ to k **do** {choose k random vectors to intialize the clusters}
  $j \leftarrow random(Card(S'))$
  $\mu_i \leftarrow S'[j]$
  $S' \leftarrow S' - \{\mu_i\}$ {remove that vector from S' so we cannot choose it again}
end for
for $i \leftarrow 1$ to $Card(S)$ **do** {assign initial cluster μ_j with feature vector $S[i]$}
  $A[i] \leftarrow \arg\min_{j \in [1..k]} \{sim(S[i], \mu_j)\}$
end for
$change \leftarrow true$
while $change$ **do** {perform clustering}
  for $i \leftarrow 1$ to k **do** {recompute cluster centroids if a change has occurred}
    $sum, count \leftarrow 0$
    for $j \leftarrow 1$ to $Card(S)$ **do**
      if $A[j] = i$ **then**
        $sum \leftarrow sum + S[j]$
        $count \leftarrow count + 1$
      end if
    end for
    $\mu_i \leftarrow sum/count$
  end for
  $change \leftarrow false$ {assume there is no change}
  for $i \leftarrow 1$ to $Card(S)$ **do** {reassign feature vectors to clusters}
    $a \leftarrow \arg\min_{j \in [1..k]} \{sim(S[i], \mu_j)\}$
    if $not(a = A[i])$ **then**
      $A[i] \leftarrow a$
      $change \leftarrow true$ {change of affiliation – loop again to recompute cluster centroids}
    end if
  end for
end while

can verify that these definitions (*i.e.* equations 5.2 and 5.3) match the conditional probability definition (*c.f.* equation 5.4).

$$P(G = g_i | C = \mu_i) = \frac{P(G = g_i, C = \mu_i)}{P(C = \mu_i)} = \frac{P(C = \mu_i | G = g_i)\ P(G = g_i)}{P(C = \mu_i)} \tag{5.4}$$

$$P(C = \mu_i) = \frac{Card(A^{-1}(\mu_i))}{Card(S)} \tag{5.5}$$

$$P(G = g_i) = \frac{Card(Label^{-1}(g_i))}{Card(S)} \tag{5.6}$$

Thus, we can deduce the joint distribution of C and G from equation 5.4 which gives:

$$P(G = g_i, C = \mu_i) = \frac{Card(Label^{-1}(g_i) \cap A^{-1}(\mu_i))}{Card(S)} \tag{5.7}$$

Hence, the mutual information between C and G which measures how much information from C is contained in G is:

$$MI(C, G) = \sum_{\mu_i \in \mathcal{C}, g_i \in \mathcal{G}} P(C = \mu_i, G = g_i) \log \frac{P(C = \mu_i, G = g_i)}{P(C = \mu_i)P(G = g_i)} \tag{5.8}$$

The goal of MMI algorithm is to reduce incrementally the size of the video-words $\mathcal{C}$ in order to obtain a compact set of code-words $\hat{\mathcal{C}}$ by keeping the value of $MI(\hat{C}, G)$ as high as possible and the value of $MI(\hat{C}, C)$ (which measures the compactness of $\hat{\mathcal{C}}$ with respect to $\mathcal{C}$) as low as possible. Note that $\hat{C}$ is an associated discrete random variable with the final optimal set $\hat{\mathcal{C}}$. At each step of the algorithm, the pair of video-words that gives the minimum loss of mutual information when merged, is chosen as candidate for merge. The merge is actually done if and only if the loss of mutual information (*c.f.* formula 5.10) generated by the merge of this optimal pair is not larger than a predefined threshold ϵ or if the minimal number of clusters is reached. Before the optimization process, the set $\mathcal{C}$ corresponds to video-words and after that process, the optimal set $\hat{\mathcal{C}}$ corresponds to code-words.

So, the trade-off between the compactness of the optimal set and the discrimination criterion (maximum of mutual information) must be resolved by the algorithm. This comes to solve the minimization problem defined by the equation 5.9 given the conditional probability distribution $P(\hat{C}|C)$.

$$\min(MI(\hat{C}, Y) - \lambda^{-1} MI(\hat{C}, C)) \tag{5.9}$$

Where λ^{-1} is the Lagrange multiplier. The details of the solution of this problem are given in (Tishby et al. 1999). (Liu & Shah 2008) demonstrate that the loss of information caused by the merge of a pair of video-words μ_i and μ_j can be computed thanks to the formula 5.10.

$$\Delta MI(\mu_i, \mu_j) = \sum_{k \in \{i,j\}} P(C = \mu_k) D_{KL}(P(G = .|C = \mu_k)||[P(G = .|C = \mu)]) \tag{5.10}$$

Where $D_{KL}(.||.)$ is the Kullback-Leibler divergence (*c.f.* formula 5.11), $[P(G = g|C = \mu)]$ is defined by equation 5.12 and μ is the resulting merged video-word.

$$D_{KL}(P(.|y)||Q(.|z)) = \sum_x P(x|y) \log(\frac{P(x|y)}{Q(x|z)}) \tag{5.11}$$

$$[P(G=g|C=\mu)] = \frac{P(C=\mu_i)}{P(C=\mu_i)+P(C=\mu_j)}\,P(G=g|C=\mu_i) + \frac{P(C=\mu_j)}{P(C=\mu_i)+P(C=\mu_j)}\,P(G=g|C=\mu_j) \tag{5.12}$$

The non-recursive version of the MMI algorithm is described hereafter (Algorithm 4). Note that $\otimes$ is the merging operator which applies to two video-words.

Compared to the code-words of (Liu & Shah 2008), our code-words already integrate the

Algorithm 4 *Maximization of Mutual Information (MMI) Algorithm*

Require: $\mathcal{C}$, $\mathcal{G}$, C, G
Ensure: $\hat{\mathcal{C}}$, $\hat{C}$
1: $\hat{\mathcal{C}} \leftarrow \mathcal{C}$
2: $minimalLoss \leftarrow 0$
3: **while** $minimalLoss < \epsilon$ & $Card(\hat{\mathcal{C}}) > Card(\mathcal{G})$ **do**
4: $minimalLoss \leftarrow \infty$
5: **for all** $\mu_i, \mu_j \in \hat{\mathcal{C}}/\mu_i \neq \mu_j$ **do**
6: $\Delta MI(\mu_i,\mu_j) \leftarrow \sum_{g\in\mathcal{G}}\sum_{k\in\{i,j\}} P(C=\mu_k)D_{KL}(P(G=g|C=\mu_k)||[P(G=g|C=\mu)])$
7: **if** $\Delta MI(\mu_i,\mu_j) < minimalLoss$ **then**
8: $minimalLoss \leftarrow \Delta MI(\mu_i,\mu_j)$
9: $merge_i \leftarrow \mu_i$
10: $merge_j \leftarrow \mu_j$
11: **end if**
12: **end for**
13: **if** $minimalLoss < \epsilon \& [Card(\hat{\mathcal{C}})-1] > Card(\mathcal{G})$ **then**
14: $\hat{\mathcal{C}} \leftarrow \hat{\mathcal{C}} - \{merge_i, merge_j\}$
15: $\hat{\mathcal{C}} \leftarrow \hat{\mathcal{C}} \cup \{merge_i \otimes merge_j\}$
16: Compute the new conditional density $\hat{C}$
17: $C \leftarrow \hat{C}$
18: **end if**
19: **end while**

spatiotemporal structural information which is not the case of the formers. Indeed, a code-word is a compact information of local motion descriptor clusters. Hence, we do not need an extra-step for the extraction of this information from the code-words.

5.4 Gesture Classification

The problem of classification with a learned database is usually a regression problem: Classifying is "guessing" the associated output (*i.e.* gesture label) of a new input (*i.e.* new generated local motion descriptors generated from a new input video) by considering known pairs of inputs/outputs. While the role of learning is to approximate and estimate the general map between training pairs, the role of classifying is to predict the associated

output of a new input based on the map estimation. In this section, we discuss two classifying strategies of the proposed learning-classification framework: (1) k-nearest neighbors algorithm and (2) Support Vector Machine classifier. We also discuss how to generate the likelihood of recognition.

5.4.1 k-nearest neighbors classifier

The k-nearest neighbor algorithm is one of the most common classifier in the literature. The main idea behind this algorithm is to select the k-nearest neighbors of a certain input from a training database and then assign it to the output that cast a majority vote among the ones associated to the selected inputs. In order to obtain always a majority vote, the "k" parameter is usually an odd number since even ones can cause ties in case of two-class classification problem. The main advantage of this algorithm is that it is an universal approximator and can model any many-to-one mapping very well. The drawbacks consist of the lack of robustness for high dimension spaces and low computational complexity with huge training data-set.

In order to adapt this algorithm to our training data-set, we must answer three questions:

- How to cope with the many-to-many mapping (A cluster can correspond to many gestures and vice-versa) ?
- How to deal with the dimensionality of the feature space ?
- How can we decrease the size of the learned database (defined below) obtained from the training data-set ?

We have already answered the two last questions since we have used the Principal Component Analysis (PCA) for dimensionality reduction and the k-means (eventually the MMI algorithm) to reduce the size of the training data-set. However, the first question remains unsolved! A suitable answer is to make a voting mechanism which transforms the many-to-many mapping into a many-to-one mapping.

Let $\mathcal{T} = \{(c, g)/c \in \mathcal{C} \& g \in \mathcal{G} \& g \in Label^{-1}(c)\}$ our final learned database with cardinal N. We recall that $Card(\mathcal{G}) = m$ and we assume that $Card(\mathcal{C}) = n$. The likelihood $L(c|g)$ of a particular cluster c given a gesture g is defined by equation 5.13.

$$L(c|g) = P(G = g|C = c) \tag{5.13}$$

We define the likelihood **measure** of a gesture g according to k observed clusters $c'_i,\ i \in [1..k]$ by:

$$L(g|c'_1,\ \dots, c'_k) = \frac{\sum_{i=1}^{k} L(c'_i|g)}{\sum_{h \in \mathcal{G}} \sum_{i=1}^{k} L(c'_i|h)} \tag{5.14}$$

Note that this likelihood measure satisfies the equation 5.15.

$$\sum_{g \in \mathcal{G}} L(g|c'_1, \ \dots, c'_k) = 1 \tag{5.15}$$

During the classification process, test sample (*i.e.* training video) generates several local motion descriptors $lmd_i, \ i \in [1..M]$. Each descriptor casts votes for k nearest clusters. If we note $L(g|lmd_i)$ the likelihood measure of a gesture g according to the k nearest clusters from lmd_i, then the gesture associated to the test sample is defined by equation 5.16 and its recognition likelihood is defined by equation 5.17.

$$g_{recognized} = \arg\max_{g \in \mathcal{G}} \sum_{i=1}^{M} L(g|lmd_i) \tag{5.16}$$

$$recognitionLikelihood(g_{recognized}) = \frac{\sum_{i=1}^{M} L(g_{recognized}|lmd_i)}{M} \tag{5.17}$$

When ties (*i.e.* several gestures with the same likelihood) occur, the classifier is unable to classify the new input. Then, the new input is fed to the learner which prompts the user for the gesture label. Two cases can be distinguished:

- The new gesture is already learned: in that case the likelihood tie vote for several gestures with correct likelihood values. The user decides which gesture wins the vote and the learned clusters are updated according to this choice.
- The new gesture has not been learned: in that case the likelihood tie has a very small value to choose any of the learned gestures. The user gives the appropriate gesture label and existing clusters are updated and eventually new clusters are created for the new gesture label.

In order to reduce the frequency of ties and specially the ones from the first case, we should introduce a weighted version of the likelihood measure. For each observed cluster $c'_i, \ i \in [1..k]$ casted by a detected local motion descriptor, the weighted likelhood measures for a given gesture is defined by equation 5.18.

$$L(g|c'_1, \ \dots, c'_k) = \frac{\sum_{i=1}^{k} \omega_i \ L(c'_i|g)}{\sum_{h \in \mathcal{G}} \sum_{i=1}^{k} \omega_i \ L(c'_i|h)} \tag{5.18}$$

Where the $\omega_i, \ i \in [1..k]$ are defined by formula 5.19.

$$\omega_i = \frac{d_i}{\sum_{j=1}^{k} d_j} \tag{5.19}$$

Where d_i is the Euclidean distance between the local motion descriptor and the cluster c'_i. Note that the sum of the weights $\omega_i,\ i \in [1..k]$ is equal to one which ensures the same property for the sum of weighted likelihood over gesture labels.

For on-line recognition, we cannot wait for all local motion descriptors to be computed in order to estimate the likelihood of gesture recognition. So, similarly to the static filtering of the HoG descriptor seen in the previous chapter, we can derive a recursive equation from equation 5.16 by considering that local motion descriptors $lmd_i,\ i \in [1..M]$ are indexed by their chronological order of computation which gives the equation 5.20.

$$g^M_{recognized} = \arg\max_{g \in \mathcal{G}} Likelihood^M(g) \tag{5.20}$$

Where $Likelihood^1(g) = L(g|lmd_1)$ and for $M > 1$, $Likelihood^M(g)$ verifies the recursion defined by equation 5.21.

$$Likelihood^M(g) = Likelihood^{M-1}(g) + \frac{1}{M}(L(g|lmd_M) - Likelihood^{M-1}(g)) \tag{5.21}$$

Algorithm 5 describes the modified version of the k-nearest neighbors for our learning-classification framework. This version of the algorithm supposes that each test sequence

Algorithm 5 *k-nearest neighbors - offline version*

Require: $\mathcal{T}$ {The training data-set}
$lmd_i,\ i >= 1$ {The generated local motion descriptors from the test sequence}
Ensure: $g_{recognized}, recognitionLikelihood(g_{recognized})$
1: $M \leftarrow 1$
2: **while** an lmd_i is generated **do**
3: execute the usual k-nearest neighbors for lmd_i
4: $g^M_{recognized} \leftarrow \arg\max_{g \in \mathcal{G}} Likelihood^M(g)$
5: $recognitionLikelihood(g_{recognized}) \leftarrow \frac{\sum_{i=1}^{M} L(g_{recognized}|lmd_i)}{M}$
6: $M \leftarrow M + 1$
7: **end while**

contains one and only one gesture. Now, we are interested to adapt this algorithm for on-line detection where several gestures can occur in a video sequence. Thus, we must integrate the time duration of a gesture in the learning-classification process to decide when to stop the recognition process and starts a new one. We assume that the duration of any gesture is ruled by a duration of life law (*i.e.* poisson law). So, the samples (*i.e.* videos) of the training data-set for a given gesture are instances of a random variable with exponential distribution. We know that if s is the number of videos in the training data set and $d_i,\ i \in [1..s]$ are the duration of these samples, then an $100(1-\alpha)\%$ exact confidence interval for the mean $\frac{1}{\lambda}$ is

given by equation 5.22.

$$\frac{1}{\hat{\lambda}}\frac{2s}{\chi^2_{2s;\alpha/2}} < \frac{1}{\lambda} < \frac{1}{\hat{\lambda}}\frac{2s}{\chi^2_{2s;1-\alpha/2}} \tag{5.22}$$

Where $\hat{\lambda}$ is defined by equation 5.23 and $\chi^2_{k;x}$ is the value of the chi squared distribution with k degrees of freedom that gives x cumulative probability.

$$\hat{\lambda} = \frac{s}{\sum_{i=1}^{s} d_i} \tag{5.23}$$

Then, we can consider that a gesture is recognized if and only if its duration is in the confidence interval and we have reached a local maximum of likelihood. So the on-line version of the k-nearest neighbors can be described by algorithm 6. To use this online-version, we can

Algorithm 6 *k-nearest neighbors - on-line version*

Require: $\mathcal{T}$ {The training data-set}
$lmd_i,\ i >= 1$ {The generated local motion descriptors from the test sequence}
Ensure: $g_{recognized}, recognitionLikelihood(g_{recognized})$
1: $M \leftarrow 1$
2: $duration \leftarrow 0$
3: $previousLikelihood \leftarrow 0$
4: **repeat**
5: $duration \leftarrow duration + 1$
6: save the previous likelihood if any in $previousLikelihood$
7: **while** a lmd_i is generated **do**
8: execute the usual k-nearest neighbors for lmd_i
9: $g^M_{recognized} \leftarrow \arg\max_{g \in \mathcal{G}} Likelihood^M(g)$
10: $recognitionLikelihood(g_{recognized}) \leftarrow \frac{\sum_{i=1}^{M} L(g_{recognized}|lmd_i)}{M}$
11: $M \leftarrow M + 1$
12: **end while**
13: **until** $duration \in confidenceInterval(g_{recognized})$ & $previousLikelihood > recognitionLikelihood(g_{recognized})$

imagine a sliding window algorithm (Kim, Song & Kim 2007) which detects the prestroke phase of gestures and calls the on-line classifier. The issue of overlapping prestrokes must be resolved by this algorithm. Also, the time precedence constraints among local motion descriptors should be studied. However, since different gestures are composed of different local motion patterns, this feature would be superfluous. Nonetheless, if we want to differentiate between two very similar gestures then it will be convinient to use it.

5.4.2 Support Vector Machine classifier

The main drawback of the k-nearest neighbors algorithm is its computational complexity: even if the usual implementation uses the kd-trees to find the k-nearest neighbors, the repetition of the execution of the algorithm for each new local motion descriptors increases considerably the processing time. An alternative to the k-nearest neighbors classifier is to use Support Vector Machine (SVM) classifier. Commonly, a SVM is a binary classifier which transforms a non-linear two-class classification problem to a two-class linear problem. Considering the training data-set T as defined in the previous subsection, we divide the single multiclass classification problem into multiple binary problems. To do so, there are two alternatives: using the "one-versus-all" splitting method or the "one-versus-one" splitting method. Note that the input of each binary SVM is a local motion descriptor and the output is one or zero depending on the descriptor membership to the associated cluster. Hereafter, we detail the appropriate voting mechanism for each strategy:

- "one-versus-all" strategy: we construct n (*i.e.* the number of clusters in our training data-set) binary classifiers for each cluster. Then, we attribute "Classified" label or "Unclassified" label according to the sample (*i.e.* local motion descriptor) membership in the corresponding cluster or not. For classification of new instances, winner-takes-all strategy is applied: the classifier generating the higher margin (*i.e.* the score given by the SVM) wins the vote. In this method, there is no normalization between the margin values given by the classifiers so we can have some scale problems resulting in misclassification (Bishop 2006). In addition, the problem is not balanced, for example with N=10, we use only 10% of positive samples for 90% of negative samples.

- "one-versus-one" strategy: we build $(n\times(n-1))/2$ binary classifiers by confronting one by one, each cluster pair formed from the n clusters. For classification of new instances, max-wins voting strategy is applied: the class with most votes determines the instance classification. Considering x the sample (*i.e.* local motion descriptor) to be classified and $\Phi_{ij}(.)$ the classifier segregating cluster c_i from cluster c_j and returning the cluster of the considered sample, then the cluster associated to x is $\underset{k\in\{1,...,N\}}{\arg\max}\, Card(\{\Phi_{ij}(x)\} \cap \{k\}; i,j \in \{1,...,N\}, i<j)$. In this method, some ambiguities can rise when there is no majority vote (Bishop 2006). A generalization of these methods has been proposed by (Dietterich & Bakiri 1995) named ECOC, considering the output of the binary classifiers as codes in order to apply code error correction techniques.

Once the local motion descriptors have been classified into learned clusters, the most likely gesture still need to be determined. In both strategy, each local motion descriptor will cast a vote for one cluster, so we can use the equation 5.16 for voting to recognize gestures. The difference here is that $L(g|lmd_i)$ is not computed by equation 5.14 anymore but it is equal to $P(G=g|C=c)$ instead; where c is the cluster casted by the local motion descriptor.

Now, that the problem is reduced to solve a set of binary SVM classifier, we explain hereafter how to build a binary SVM classifier for our training data-set. While the k-nearest

neighbors can deal directly with multi-class clustering, a SVM is a binary linear classifier which tries to find the hyperplane that separates the two considered clusters and maximize the margin (*i.e.* the distance) between each cluster members and the separating hyperplane. (Hearst 1998) demonstrates that these criteria give the optimal separator according to Vapnik-Chervonenkis Statistic Learning Theory. The nearest cluster members to this optimal hyperplane are called support-vectors.

In order to deal with a non-linear classification problem, an improved version of SVM is to use the "kernel trick" (*i.e.* kernel methods). The goal is to find a non-linear mapping Φ from the original feature space S to a new feature space $\Phi(S)$ with a higher dimensionality (it can have an infinite dimension) in which the classification problem becomes linearly separable. Then, a real valued function K from the feature space pairs, called the kernel, is defined such as its value is equal to the dot product of the images of its arguments by the non-linear mapping Φ (*c.f.* equation 5.24).

$$K(f, g) =< \Phi(f), \Phi(g) > \tag{5.24}$$

Such function must be positive-defined and symmetric. As long as the existence of Φ is guaranteed, there is no need to explicit Φ since the dot product in the target space is completely defined by the kernel function K which applies on the initial feature space pairs. Thus, the complex computation of the dot product in the higher dimensional space is replaced by the simple evaluation of a kernel function between the pairs of the initial feature space. Then we can apply the SVM classifier on our training data-set by using an appropriate kernel function. We choose to use the Gaussian radial basis function (*c.f.* formula 5.25) as a kernel since the corresponding mapping transforms the feature space into an Hilbert space of infinite dimension which guarantees a high probability for the linear separability.

$$k(x, x') = \exp(-\frac{||x - x'||^2}{2\ \sigma^2}) \tag{5.25}$$

SVM classifier can run well on a small training data-set with a good separability criterion. This contrasts with the k-nearest neighbor algorithm which guarantees a low error rate only with great training data-set (*c.f.* (Ming et al. 2003)). Indeed, as the amount of data approaches infinity, the k-nearest neighbor algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate which is the minimum achievable error rate given the distribution of the data. (Ming et al. 2003) propose to combine SVM with k-nearest neighbor algorithm in order to maximize the accuracy-effeciency trade-off of the classification process.

5.5 Conclusion

In this chapter, the proposed learning-classification framework for gesture recognition has been described. The learning step transforms the local motion descriptors generated from a training data-set into a codebook of video-words by using the k-means clustering algorithm. Optionally, the video-words can be compacted into code-words enabling dimension-

ality reduction and consequently speeding up the classification process. For the classification step, we use the k-nearest neighbors algorithm in order to recognize new gestures with a certain likelihood. We have adapted this algorithm to work with our training data-set by proposing an adequate voting mechanism. Alternatively, we can use a Support Vector Machine for the classification which provides a better separability criterion (Ming et al. 2003). In the next chapter, we overview the experiments and the main results of the proposed method.

Chapter 6

Evaluation

No amount of experimentation can ever prove me right; a single experiment can prove me wrong.

Albert Einstein

6.1 Introduction

We have proposed a learning-classification framework for gesture recognition using a HOG tracking algorithm. The proposed framework is based on a statistical model of local motion signatures (*i.e.* descriptors) and provides a likelihood measure for recognized gestures. In this chapter, we present the experiments carried out to evaluate and validate this framework. The goal is to demonstrate the effectiveness of the method and to evaluate its potential for real world video sequences. Generally, we can split the proposed framework in two phases: (1) HOG tracking phase and (2) Gesture learning-classification phase. In the tracking phase, the system tries to find local motion of local features by matching corresponding HOG descriptors over the time. In the learning-classification phase, the system learns several gestures through the clustering of the generated local motion descriptors. Then, a new and unknown gesture is classified by a statistical voting process. We have performed an experiment for each phase in order to validate them independently. However, concerning the validation of the second phase, we have only evaluated a part of the framework consisting of k-means algorithm for learning and the offline version of the k-nearest neighbors for classification. So, the computational efficiency of the recognition phase has been evaluated only for the version based on k-nearest neighbors algorithm. The validation of the recognition phase based on SVM is also discussed.

For the HOG tracking phase, we evaluate the tracking performance through two different data-set: (1) a synthetic data-set and (2) a real world data-set. The results of the algorithm are compared with the ones obtained with the KLT algorithm. The main advantage for testing the tracking algorithm on synthetic data is to demonstrate the effectiveness of the tracker. Concerning the gesture learning-classification phase, we validated the proposed framework with the KTH and IXMAS databases and compared the results to several state-of-the-art methods.

The ground truth associated to the evaluation gesture databases is presented in section 6.2. In section 6.3, we define the evaluation metrics for each experiment. Section 6.4

introduces the experimental protocol and details the results of the different experiments. Section 6.5 concludes this chapter by discussing the main results and the robustness of the approach to noise and occlusions. The extensibility of the approach is also discussed.

6.2 Gesture Databases

The gesture databases used for the evaluation and the validation of the proposed method consist of the KTH action/gesture database (Schuldt et al. 2004) and the IXMAS action/gesture database (Weinland et al. 2007).

The KTH database contains 600 videos illustrating six actions/gestures: (1) walking, (2) jogging, (3) running, (4) hand waving, (5) hand clapping and (6) boxing. Each action/gesture is performed many times by 25 actors for four different scenarios:

1. Scenario "s1": The action/gesture is performed outdoors without any scale change (*i.e.* constant distance from the camera).
2. Scenario "s2": The action/gesture is performed outdoors with scale changes (*i.e.* the distance form the camera variates while action is performed).
3. Scenario "s3": The action/gesture is performed outdoors with different clothes (*i.e.* color/contrast variation).
4. Scenario "s4": The action/gesture is performed indoors (*i.e.* illumination variation).

Figure 6.1 illustrates the different actions/gestures with the different scenarios. Thus, there are $4 \times 6 = 24$ videos per actor. The database is split into three independent data-sets: (1) a training data-set (8 actors), (2) a validation data-set for tuning parameters (8 actors) and (3) a testing data-set for evaluation (9 actors). All videos from this database were taken over homogeneous background thanks to a static camera with 25fps frame rate. The spatial resolution of each video is 160x120 pixels.

The IXMAS database contains 468 action clips for 13 gestures and each of them is performed three times by 12 actors. Each video clip has a spatial resolution of 390x291 pixels, a frame-rate of 23fps and it is captured by five cameras from different points of view (*i.e.* five video sequences for each clip). The gestures of the database are : (1) check watch, (2) cross arms, (3) scratch head, (4) sit down, (5) get up, (6) turn around, (7) walk, (8) wave, (9) punch, (10) kick, (11) point, (12) pick up and (13) throw. Figure 6.2 and Figure 6.3 illustrate the different actions of the IXMAS database.

6.3 Evaluation Criteria/Method

Hereafter, the different metrics used for each experiment are defined. The validation of a learning-classification framework is generally carried out with respect to standard metrics. In such framework, two kinds of classification error can occur: (1) Statistical errors and

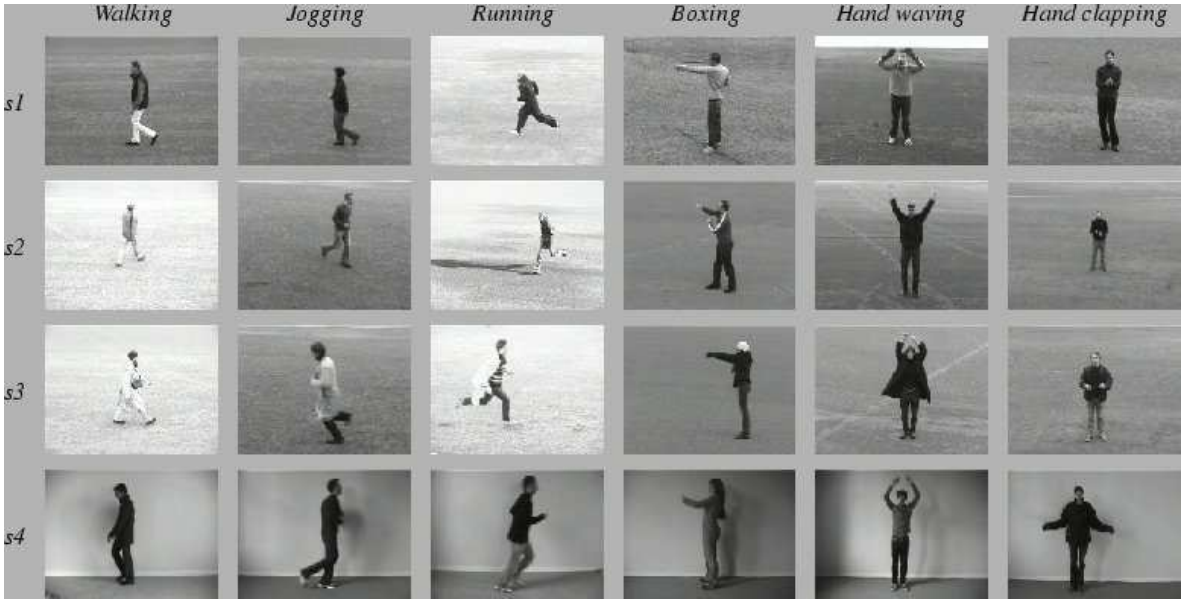


Figure 6.1: *The different actions/gestures of the KTH database with the different scenarios*

(2) Systematic errors. A statistical error is generally caused by random, and inherently unpredictable fluctuations in the measurement apparatus (camera outputs and eventually the preprocessing process outputs) or the system being studied. However, a systematic error is generally caused by non-random fluctuations from an unknown source (*i.e.* a constant drift), and which, once identified, can usually be eliminated. Since the proposed method is based on a statistical model of local motion signatures, we assume that all the errors generated by the framework are statistical ones which is actually not true. Given a ground truth, the outcome of a classifier belongs necessarily to one of these four categories:

- True positive: also known as positive classification, it occurs if the classifier detect something where in ground truth is the same thing.

- False positive: also known as an α error, it is the error of rejecting a classification hypothesis when it is actually true. Plainly speaking, it occurs when we are observing an event where in ground truth there is none. An example of this would be if a classifier detects that a gesture is hand waving when in reality it is not. Thus, it is the error of commission (*i.e.* excessive credulity) when certain events that are one thing are classified as another thing.

- False negative: also known as a β error, it is the error of failing to reject a classification hypothesis when it is in fact not true (*i.e.* it should have been rejected). In other words, this is the error of failing to observe an event where in ground truth there is one. An example of this would be if a classifier assumes that there is no gesture when in reality it is. Thus, it is the error of omission (*i.e.* excessive skepticism) when a certain event is not classified as what it is.

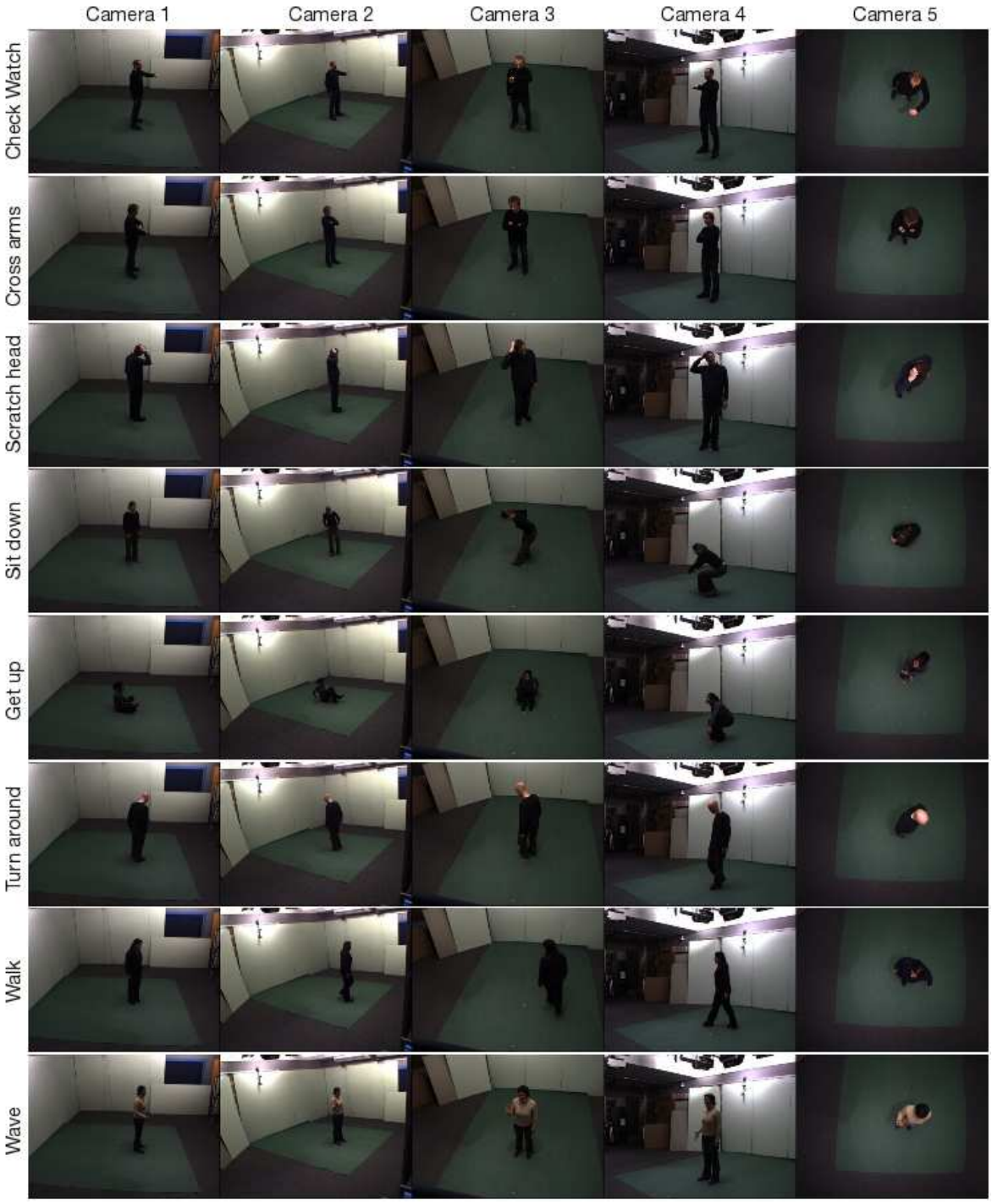


Figure 6.2: *The 8 first actions/gestures of the IXMAS database with the different views*

- True negative: also known as negative classification, it occurs if the classifier does not detect anything where in ground truth there is nothing.

Based on these four categories of outcome, several quality measures can be defined to eval-

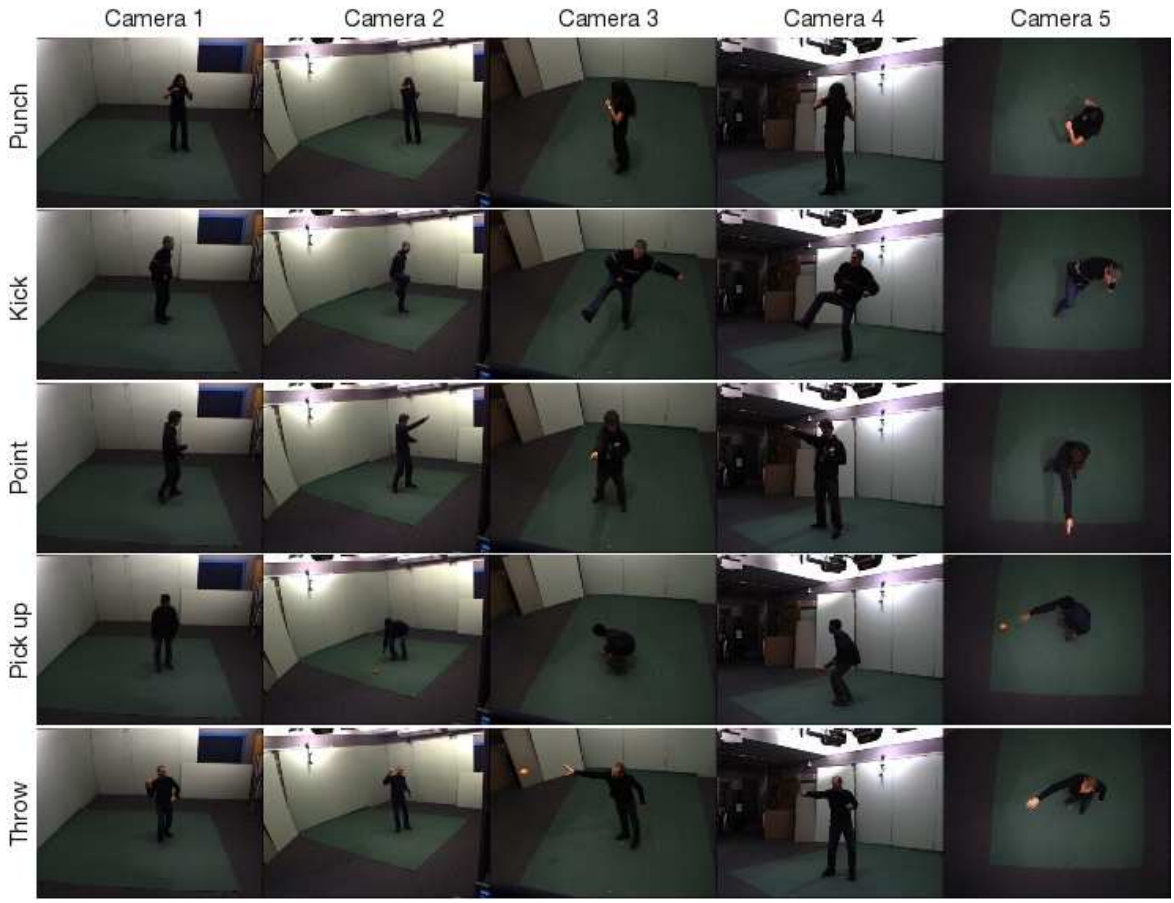


Figure 6.3: *The 5 remaining actions/gestures of the IXMAS database with the different views*

uate the efficiency and the effectiveness of a classifier:

- accuracy: accuracy measures the degree of veracity of the classifier. In other words, it is the degree of closeness of a classifier outcome to its actual value (ground truth). Equation 6.1 explicits the formula to compute it.

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}} \tag{6.1}$$

  it is too difficult to compute accuracy since the number of true negatives cannot be objectively determined. Moreover, with the assumption of that only statistical errors exist, the measure of accuracy does not match the exact definition of a system accuracy. A more adequate metrics, in this case, are precision and recall. A " F-score" (*c.f.* formula 6.2), which consists of the harmonic mean of recall and precision, can be computed to approximate the system efficiency.

$$F - score = \frac{2 \times precision \times recall}{precision + recall} \tag{6.2}$$

Table 6.1: *Classification metrics*

		Ground Truth		Metrics
		Positive	Negative	
Classifier outcome	Positive	True Positive	False Positive	Precision
	Negative	False Negative	True Negative	NPV
Metrics		sensitivity	specificity	accuracy

- sensitivity (*i.e.* recall): sensitivity (also called recall or statistical power) measures the proportion of successful classification of positive ground truth. It can be seen as the probability that the classifier succeeds given that there is an event in the ground truth. The higher the sensitivity, the fewer ground truth events go undetected.

$$\text{sensitivity} = \frac{\text{number of True Positives}}{\text{number of True Positives} + \text{number of False Negatives}} \tag{6.3}$$

- specificity measures the proportion of successful classification of negative ground truth. As with sensitivity, it can be viewed as the probability that the classifier does not recognize anything given that there is nothing in the ground truth. The higher the specificity, the fewer false events (eventually false alarms) are risen by the classifier. Like accuracy, it is too difficult to compute the specificity.

$$\text{specificity} = \frac{\text{number of True Negatives}}{\text{number of True Negatives} + \text{number of False Positives}} \tag{6.4}$$

- positive predictive value (*i.e.* precision) : precision measures the amount of correct classifications among all positive classifications.

$$\text{precision} = \frac{\text{number of True Positives}}{\text{number of True Positives} + \text{number of False Positives}} \tag{6.5}$$

- negative predictive value measures the amount of correct negative classifications among all negative classifications.

$$\text{negative predictive value} = \frac{\text{number of True Negatives}}{\text{number of True Negatives} + \text{number of False Negatives}} \tag{6.6}$$

Table 6.1 summarizes the different metrics for classification. Since the difficulty of determining true negatives, we choose to measure only the precision and the sensitivity. Then, the F-score of our classifier can be computed. Note that there is a trade-off between recall and precision: when we want to increase the precision generally this comes with a decrease of the sensitivity and vice-versa. Moreover, when the classifier outcome is positive but its class is different from a corresponding in the ground truth, we consider that the classifier has made at the same time the α and β errors: it casts one false positive and one false negative.

6.4 Experimental Protocol and Results

The following experiments have been performed on a Fedora Core 10 operating system (64-bit) and on an Intel Xeon workstation (4-core@2.83 GHz). Three experiments have been conducted to evaluate the proposed method:

- Testing the HOG tracker on synthetic data.
- Testing the HOG tracker on real data (the testing data-set of the KTH database, Gerhome database and IXMAS database).
- Testing the learning-classification framework on the KTH database and the IXMAS database.

In order to tune the parameters of the proposed algorithms, we run them several times with different values of the parameters using the validation data-set of the KTH database. For the proposed feature tracker, we found out that the best values are $K = 9$ for histogram orientation bins so the size of a 2D HOG Descriptor is 81. We have found that the optimal values for the empirical weights α and γ of the distance $\mathcal{D}$ are respectively 5 and 2. The minimum distance between corner points (i.e. HOG descriptors) is set to nine pixels. Concerning the parameters of the Kalman filter, we have found that the value of seven for the standard deviation of the process noise and the measurement noise gives the best tracking results.

Since the KTH database contains six gestures, we have tested all the values of k between 18 and 57 for the k-means clustering algorithm. We realized that the best classification results is when $k = 27$. Finally, the best value of the k parameter of the k-nearest neighbor algorithm is $k = 5$. For the IXMAS database, we have selected the value $k = 197$ for the learning phase and $k = 5$ for the classification phase. Hereafter, we detail the results for each experiment.

6.4.1 Tracking Results on Synthetic Sequence

In order to demonstrate the effectiveness of the HOG tracker, we have tested it on a synthetic data-set as illustrated by figure 6.4. The generated sequence is composed of 247 frames. During the whole sequence our tracker has lost only 39 descriptors while the mean number of descriptors per frame is 35. The lost of descriptors occurs when there is a sudden and strong change in the motion direction. This is due to the linear motion model used by the Kalman filter. An improvement to cope with this high motion is to use more sophisticated motion model (e.g. Brownian motion). As for the KLT tracker, the number of lost descriptors is about 547 and the mean number of descriptors per frame is 29. Hence, our algorithm outperforms the KLT tracker for synthetic data.

6.4.2 Tracking Results on Real Sequences

To go forward in the validation of the developed tracker, we have tested it on the validation data-set of the KTH database. Figure 6.5 illustrates the results on KTH database and

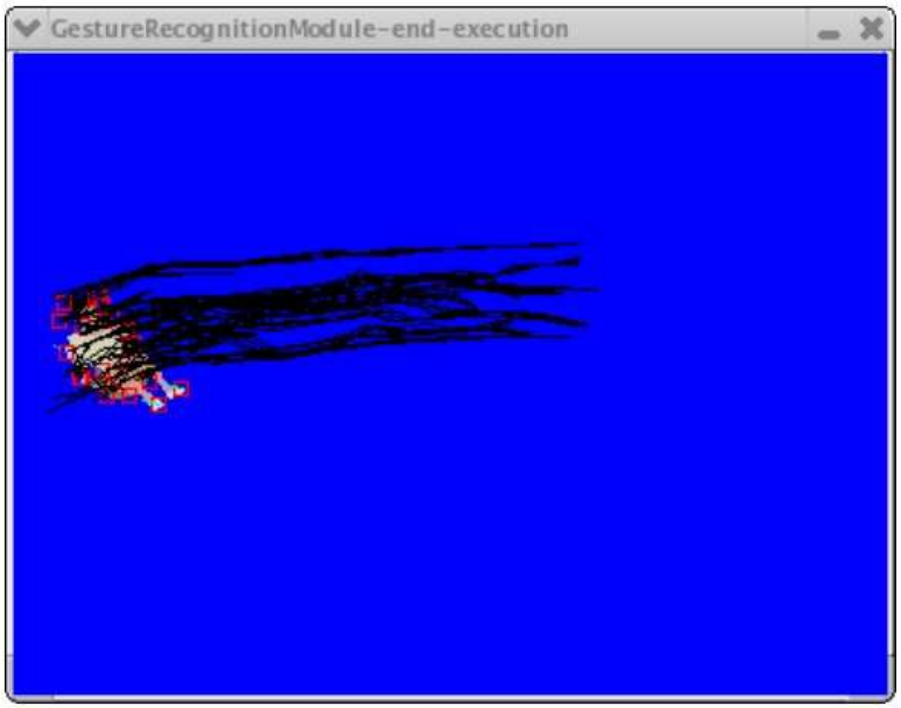
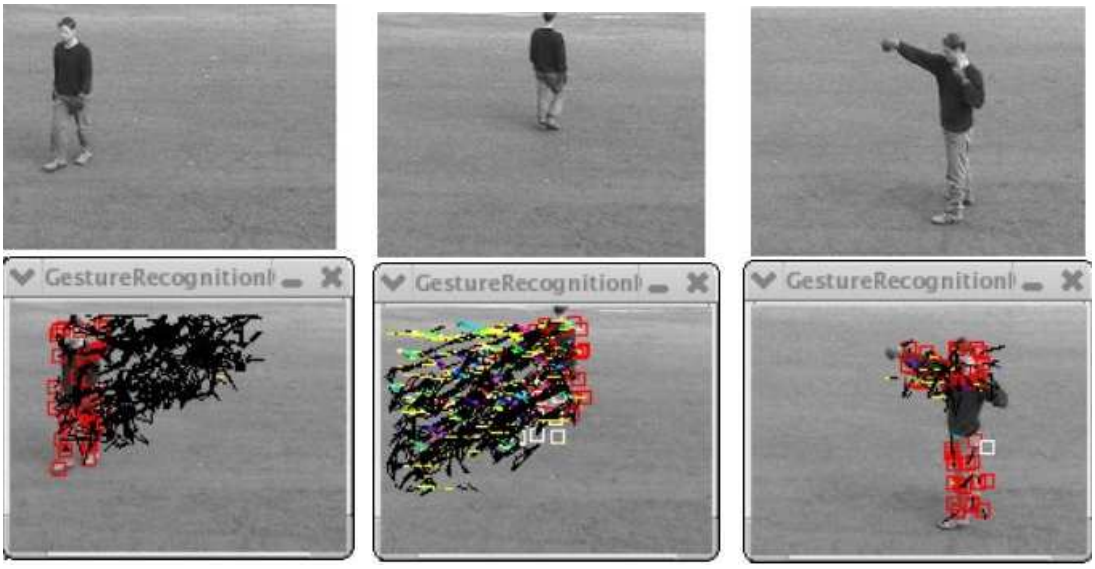


Figure 6.4: *Synthetic data-set: red rectangles represent the tracked descriptors and the lines represent their trajectories.*

Table 6.2: *Results of HOG tracking module with the validation data-set of the KTH database.*

	Mean	Var	Min	Max
number of descriptors per frame	22.32	03.37	15.15	34.38
number of tracked descriptors per frame	20.70	03.57	15.00	27.88
number of lost descriptors per frame	01.62	01.50	00.15	06.50

table 6.2 resumes the obtained results. This table describes the mean, variance, minimum and maximum values of the number of descriptors (detected, tracked and lost) per frame. The proposed tracker outperforms the KLT feature point tracker (Shi & Tomasi 1994) (there are only nine tracked feature points per frame in average) which looses many more feature points. We notice that the gap between our tracker and the KLT tracker is more significant for real data. This is due to the fact that the KLT tracker is very sensible to noise. Note that there is a slight drift between the descriptor position and the actual position of the corner points. This is a common drawback of texture correlation-based techniques for feature tracking to which our method belongs. However, this issue is not so important for our gesture recognition framework since the "global" local motion of the drifted descriptor is equivalent to the actual local motion of the corner. The average processing time of the proposed tracker is about 15fps which can be considered as near real-time performance. Moreover a GPU implementation can significantly increase the computational performance of the algorithm.



(a) Walking (1) (b) Walking (2) (c) Boxing

Figure 6.5: *KTH data-set: red rectangles represent the tracked descriptors, white ones represent the newly detected descriptors and the lines represent the trajectories of tracked descriptors.*

Table 6.3: *Results of HOG tracking module with the video sequences of the IXMAS database.*

	Mean	Var	Min	Max
number of descriptors per frame	57.00	02.25	48.00	60.00
number of new descriptors per frame	00.18	00.02	00.15	00.21
number of tracked descriptors per frame	56.75	01.97	51.57	59.75
number of lost descriptors per frame	00.12	00.03	00.09	00.17

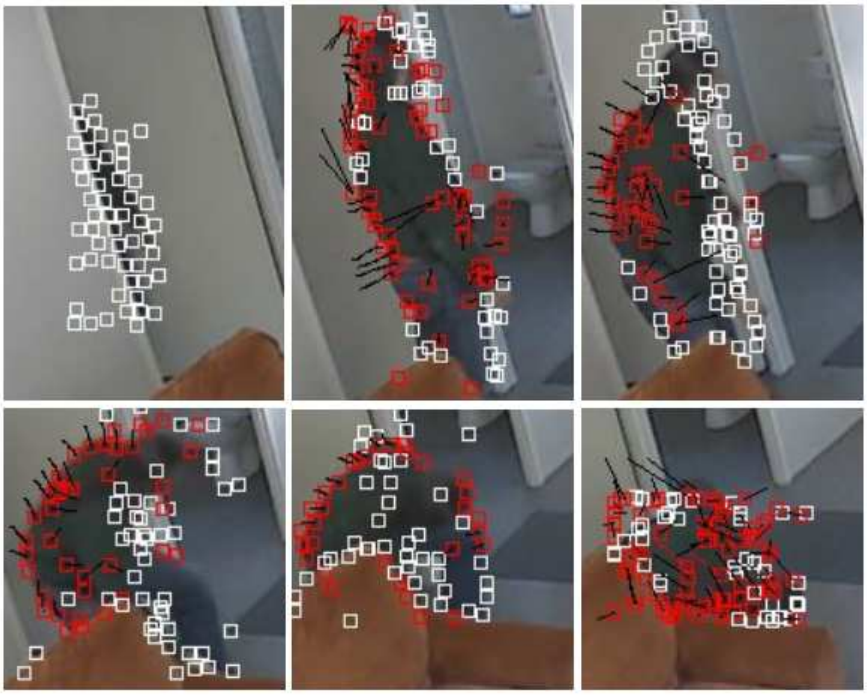
Furthermore, we have carried out tests of the proposed tracker on the Gerhome and IXMAS video databases. Figure 6.6 illustrates a falling action from the Gerhome database while figure 6.7 shows the output of our tracker and the KLT tracker when only short term motion (five frames) is displayed. Similarly, figure 6.8 presents several frames from the IXMAS database illustrating a turn-around action and figure 6.9 represents the correspondent output of our tracker with short term and long term motion. Table 6.3 resumes the obtained measurements, which are the same as for table 6.2, for the IXMAS database. The better performance of our algorithm on IXMAS database can be explained by the fact that the video resolution (390x291) is better than in KTH database (160x120). Additionally, the actors in IXMAS are always near the camera and noise is almost nonexistent. Unfortunately, we have not obtained statistically significant results for Gerhome database yet. This is due to the number of tested videos which is still low and the non-optimization of the segmentation algorithm and the people detector.

Figure 6.6: *Frames from Gerhome database illustrating a falling action*

6.4.3 Classification results for Gesture Recognition on KTH database

We train our algorithm on the KTH training data-set and test it on the corresponding test data-set. We recall that all the parameters of the framework are tuned with respect to the validation data-set. Results are illustrated by the confusion matrix 6.4 and are compared to the state of the art method in table 6.5. We obtain better or slightly better results than recent methods. We also find out that FAST corners outperform Shi-Tomasi corners which is consistent with results in (Rosten & Drummond 2006). Note that even though (Kim, Wong & Cipolla 2007) obtain slightly better results, their results are not comparable to ours since they use a different experimental protocol (Leave-one-out cross-validation). Table 6.6 shows the precision, recall and F-score of the proposed framework. Figure 6.10 illustrates the variation of precision and recall according to the "k" parameter of the k-means clustering algorithm while figure 6.11 overviews the correspondent precision-recall graph. The proposed framework has a high sensitivity which means that there is few false negatives. However, the precision can be improved. Thus, the next step will consist of testing the whole learning-classification framework (*i.e.* adding MMI to compact the code-book and using SVM and the on-line version of k-nearest neighbors algorithm). We can see that very few descriptors are lost and most of them are correctly tracked throughout the video. Only some descriptors on legs and arms are regularly lost due to self occlusions. Finally, figure 6.12 shows the Receiver Operator Characteristic (ROC) curve which overviews the true positive rate w.r.t. false positive rate. From this graph, we can deduce that our algorithm has a high detection rate w.r.t. false alarm rate.

(a) Output of the KLT tracker: Each white line represents the history of motion of the correspondent corner point.



(b) Output of the proposed tracker: Tracked HOG descriptors are in red and new ones are in white. Each black line corresponds to the short term trajectory of the associated descriptor.

Figure 6.7: *Outputs of the proposed tracker and the KLT tracker for the videos from Gerhome database 6.6: Only short term motion is shown.*

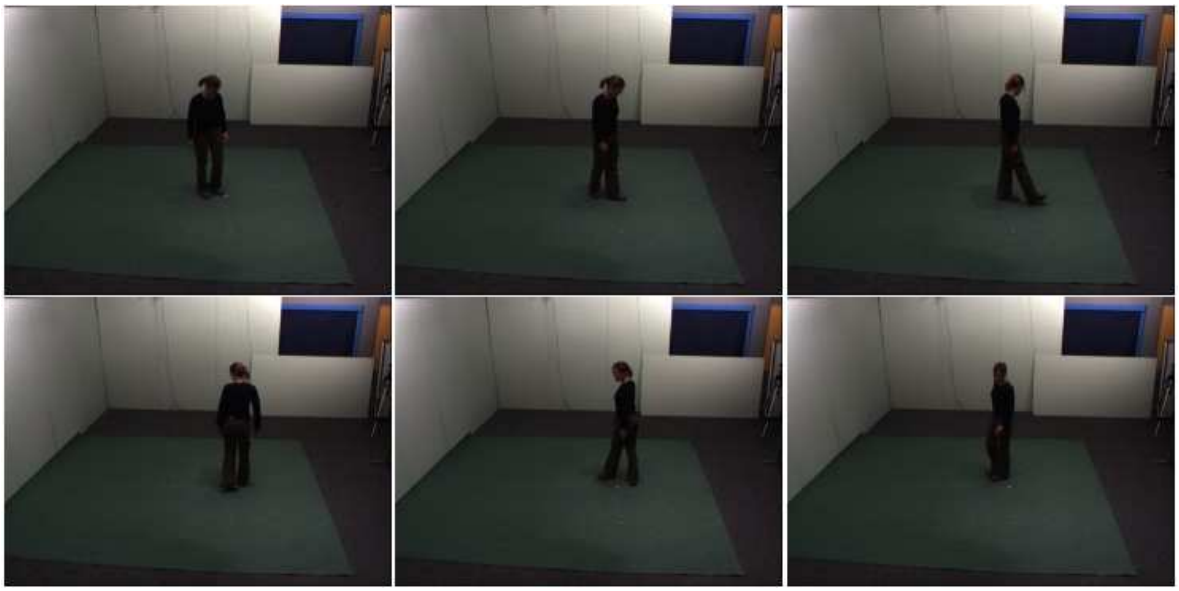


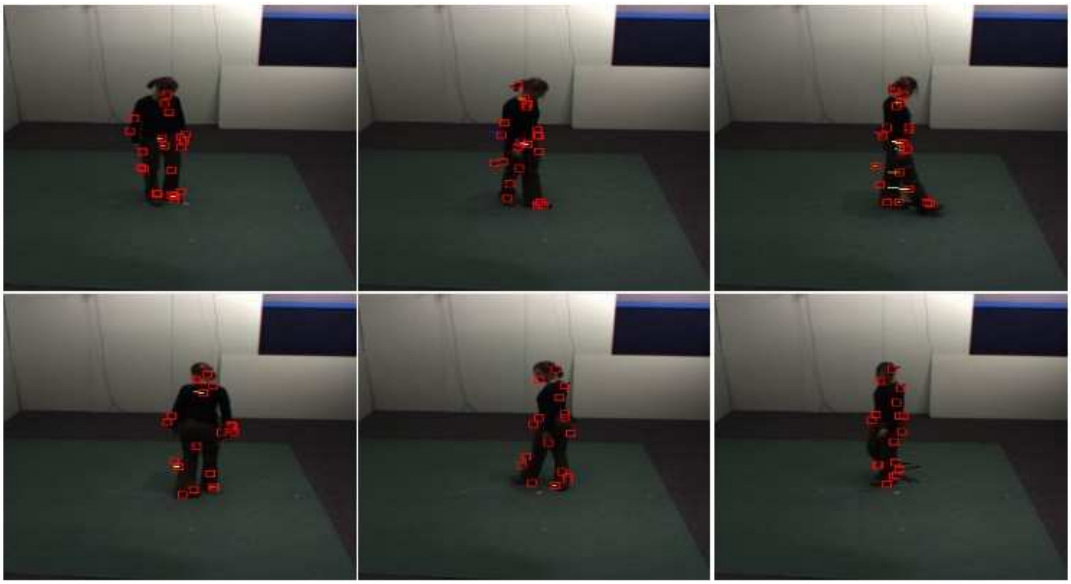
Figure 6.8: *Frames from the IXMAS database illustrating a turn-around action.*

Table 6.4: *Confusion matrix for the classification on KTH database using Shi-Tomasi (upper values) and FAST corner points (lower values).*

	W.	J.	R.	B.	H.C.	H.W.
W.	$\frac{0.95}{0.97}$	$\frac{0.03}{0.03}$	$\frac{0.02}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$
J.	$\frac{0.03}{0.02}$	$\frac{0.85}{0.91}$	$\frac{0.10}{0.07}$	$\frac{0.02}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$
R.	$\frac{0.05}{0.03}$	$\frac{0.07}{0.05}$	$\frac{0.88}{0.92}$	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$
B.	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.95}{0.97}$	$\frac{0.03}{0.02}$	$\frac{0.02}{0.01}$
H.C.	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.05}{0.03}$	$\frac{0.88}{0.92}$	$\frac{0.07}{0.05}$
H.W.	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.00}{0.00}$	$\frac{0.02}{0.01}$	$\frac{0.01}{0.00}$	$\frac{0.97}{0.99}$

6.4.4 Classification results for Gesture Recognition on IXMAS database

For IXMAS database, we adopt a leave-one-out cross-validation scheme. Since each action is captured from five points of view, we have selected $k = 197$ for the k-means clustering algorithm. Actually, the three phases of a gesture (*i.e.* prestroke, stroke and poststroke) can generate different 2D motion patterns from different points of view. So for each action, we can expect a maximum of $3 \times 5 = 15$ motion patterns. Due to the fact that the IXMAS database contains 13 gestures, the expectation grows to $13 \times 15 = 195$ motion patterns. For the classification phase, we used the same value of the k parameter (*i.e.* 5) as for KTH database. The classification is carried out independently for each gesture clip corresponding to the remaining actor (*i.e.* discarded by the leave-one-out rule). So, for each actor (1 out



(a) Output of the proposed tracker with short term motion: only few descriptors and their correspondent motions are displayed.

(b) Output of the proposed tracker with long term motion: different colors indicate different motion directions

Figure 6.9: *Outputs of the proposed tracker with short term and long term motion.*

of 12), each gesture (1 out of 13) and at a particular step of the cross-validation, there are $5 \times 3 = 15$ video sequences to be classified versus $11 \times 5 \times 3 = 165$ video sequences used for learning. In addition, we choose to carry out this experiment with the FAST corner detector only since it is the best one (*c.f.* results on KTH database). The confusion matrix for this experiment is given by table 6.7 and table 6.9 presents the correspondent efficiency metrics.

Table 6.5: *Comparison of different results of the KTH database.*

Method	Variant	Precision
Our method	Shi-Tomasi	91.33%
	FAST	94.67%
Liu and Shah (Liu & Shah 2008)	SVM VWCs	91.31%
	VWC Correl.	94.16%
Luo et al. (Luo et al. 2008)		85.10%
Kim et al. (Kim, Wong & Cipolla 2007)		95.33%

Table 6.6: *Precision, recall and F-score of the proposed method on KTH database.*

	Precision	Recall	F-score
with Shi-Tomasi	91.33	99.07	95.04
with FAST	94.67	99.78	97.15

We compare the results of our method to those of (Weinland et al. 2007) in table 6.8. Note that, even though, other methods (*e.g.* (Lv & Nevatia 2007, Liu & Shah 2008)) have been validated on the IXMAS database, we cannot compare their results to ours since they do not follow the same experimental protocol. For instance, (Liu & Shah 2008) learn and classify only from four camera views (excluding the top view which has the worst recognition rate) and (Lv & Nevatia 2007) do not use a leave-one-out cross-validation. Nevertheless, we have obtained a better precision than these two approaches (82.80% for (Liu & Shah 2008) and 80.60% for (Lv & Nevatia 2007)). Unsurprisingly, gestures with large motion (*e.g.* sit down, get up, turn around, walk) are much better recognized than gestures with small motion (*e.g.* scratch head, wave, point) which, besides, share some motion patterns. The mean processing time of the offline k-NN classifier for the IXMAS database is 35 seconds per gesture which is quite correct knowing that a gesture is in mean illustrated by 80 frames.

6.4.5 Discussion and Further evaluation

The value of the k parameter for both k-means and k-nearest neighbor algorithms is mainly dependent on the number of gestures to be recognized, the gesture database size (*i.e.* number of gestures, number of view points per gesture). Indeed, when we process a multiview database of n gestures all captured under m views, the constraint on the parameter k of the k-means algorithm becomes $k > 3 \times n \times m$ since local motion patterns for each gesture phase (*i.e.* prestroke, stroke and poststroke) can be different from several points of views. To avoid parameter tuning, the Mean Shift clustering algorithm (Georgescu et al. 2003) and the SVM classifier can be used. The evaluation of SVM, with k-means as clustering algorithm, should

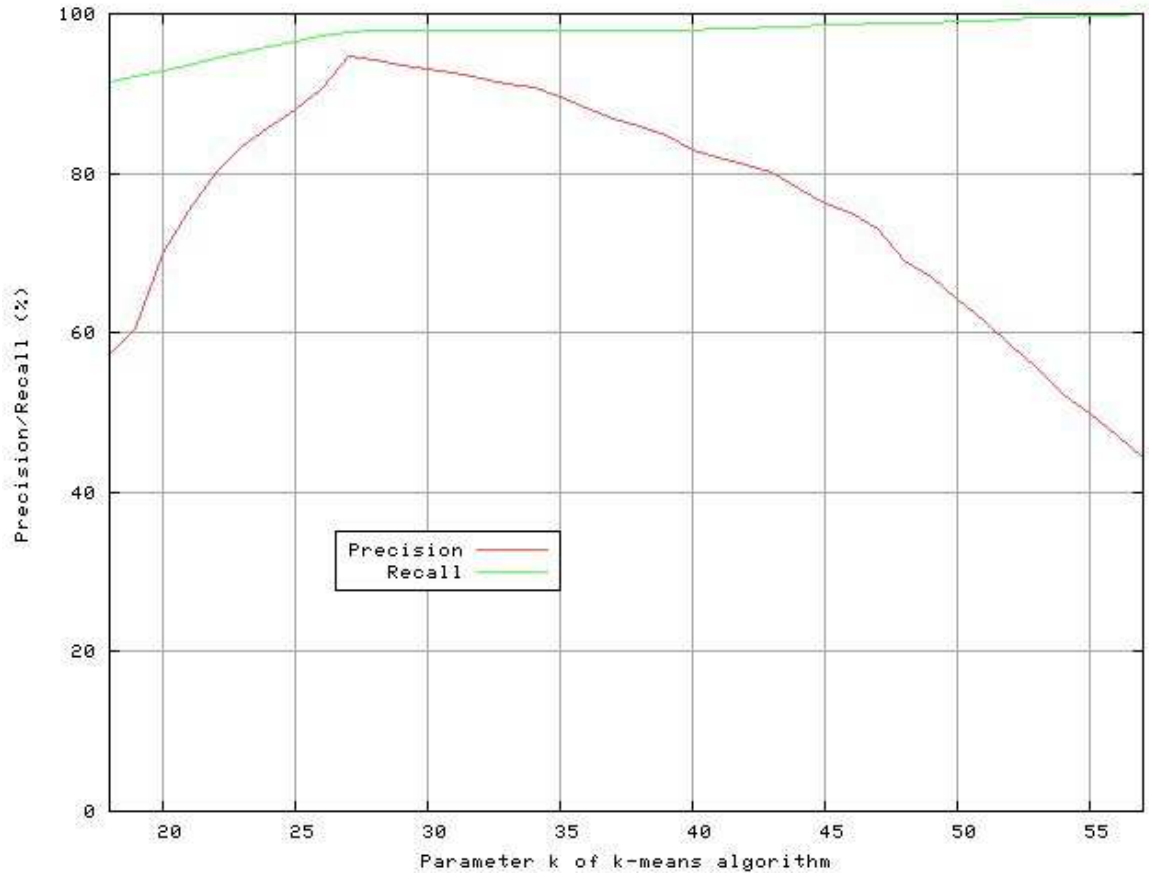


Figure 6.10: *The variations of precision and recall w.r.t. k-means parameter for KTH database.*

be carried out to compare the results to those of the k-nearest neighbor algorithm. Moreover, we need to carry out more experiments on the IXMAS database by testing its robustness to the variations of the k parameters of the learner (*i.e.* k-means) and the classifier (*i.e.* kNN) and by drawing ROC and precision-recall graphs.

6.5 Conclusion

In this chapter, we have demonstrated the efficiency and the effectiveness of the proposed learning-classification framework. The HOG tracker used for local motion descriptor generation has been validated on synthetic and real data. The proposed tracker outperforms the KLT tracker specially for real data where it shows its robustness to noise. As for the gesture recognition process including learning and classification algorithms, it has been validated with the KTH action/gesture database. The results show that the proposed framework outperforms (slightly) the state-of-the-art methods. Moreover, we have carried out preliminary validation on IXMAS multi-view data-set and satisfactory results have been obtained. However, more validation is needed to evaluate the whole framework with IXMAS data-set and more challenging databases such as the multi-view KUG database (Hwang et al. 2006) or a

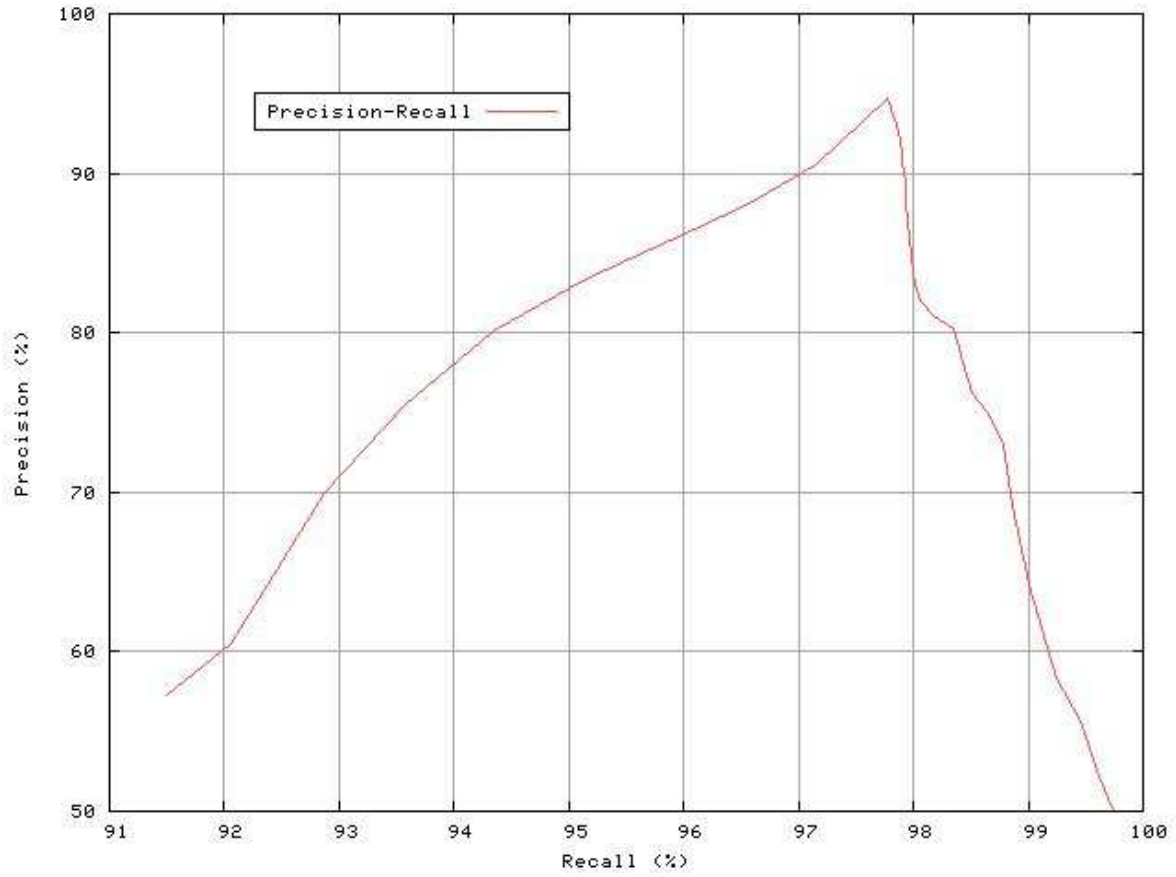


Figure 6.11: *The precision-recall graph for KTH database.*

real-world situation database (*e.g.* TrecVid and/or Gerhome). In the next and last chapter, we review the contributions of this thesis, we discuss the pros and cons of the proposed framework and we present the short-term and long-term future work.

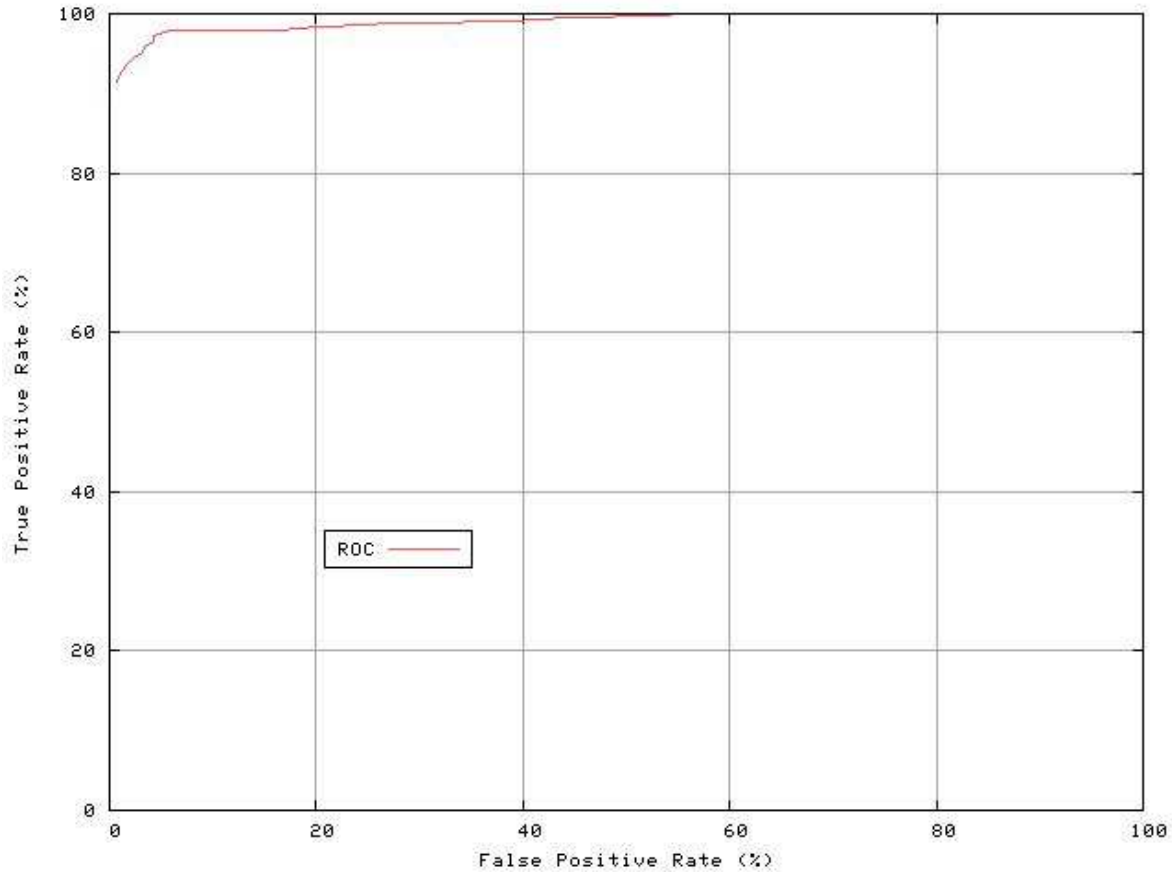


Figure 6.12: *ROC curve for the KTH database.*

Table 6.7: *Confusion matrix for the classification on IXMAS database using FAST corner points.*

	C.W.	C.A.	S.H.	S.D.	G.U.	T.A.	Wl.	Wv.	Pu.	K.	Po.	P.U.	T.
C.W.	0.87	0.05	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.03	0.00	0.00
C.A.	0.10	0.75	0.05	0.00	0.00	0.00	0.00	0.03	0.03	0.00	0.04	0.00	0.00
S.H.	0.07	0.08	0.69	0.00	0.00	0.00	0.03	0.03	0.07	0.00	0.03	0.00	0.00
S.D.	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
G.U.	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
T.A.	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Wl.	0.00	0.00	0.00	0.00	0.00	0.05	0.95	0.00	0.00	0.00	0.00	0.00	0.00
Wv.	0.05	0.03	0.12	0.00	0.00	0.00	0.00	0.62	0.03	0.03	0.09	0.00	0.03
Pu.	0.04	0.04	0.00	0.00	0.00	0.02	0.00	0.03	0.75	0.07	0.00	0.01	0.04
K.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.97	0.00	0.00	0.00
Po.	0.03	0.00	0.06	0.00	0.00	0.03	0.00	0.08	0.20	0.03	0.57	0.00	0.00
P.U.	0.00	0.00	0.00	0.05	0.00	0.02	0.03	0.00	0.01	0.00	0.00	0.89	0.00
T.	0.00	0.00	0.05	0.00	0.04	0.00	0.00	0.06	0.03	0.00	0.06	0.00	0.76

Table 6.8: *Comparison of different results of the IXMAS database.*

Method	Variant	Precision
Our method	FAST	83.23%
Weinland et al. (Weinland et al. 2007)		81.27%

Table 6.9: *Precision, recall and F-score of the proposed method on IXMAS database.*

	Precision	Recall	F-score
with FAST	83.23	87.46	85.29

Chapter 7

Conclusion

Once we accept our limits, we go beyond them.

Albert Einstein

During this thesis, we have proposed a promising framework for human gesture recognition from video sequences. We have validated this approach on the KTH and IXMAS gesture databases and we have obtained good results. Thus, we have mainly reached our objectives even though we still need to perform on-line experiments with everyday life videos. We believe that the proposed approach is sufficiently robust and flexible to deal with this kind of videos. Indeed, we have theoretically proved that our learning-classification framework can be adapted to on-line recognition. Moreover, we have proposed several solutions to improve the efficiency and the effectiveness of our method. For instance, we can learn normal gestures and consider as abnormal the non-recognized ones. The performance of the algorithm can be then improved incrementally by learning from new life experience.

The proposed learning-classification framework for gesture recognition involves three processing steps consisting of **gesture descriptor generation**, **gesture learning** and **gesture classification**. This approach has introduced a novel gesture representation which combines local and global motion descriptor advantages. It contains an adequate learning-classification framework for gesture recognition using these new descriptors. The spatio-temporal aspect of gestures is taken into account since the low level algorithm tracks corner points through HoG and Kalman filtering.

In order to take into account noise and ambiguities that can rise in real world applications, we have integrated to our framework a likelihood measure of the recognized gestures. A gesture is recognized when its likelihood is sufficiently larger than the other ones. The likelihood measure complements advantageously the similarity measure generally used by common classifiers. Indeed, the likelihood accounts the coherence and the quality of the recognition according to the observations.

This chapter is structured into five sections which conclude the present thesis manuscript. In section 7.1 we overview our contributions with regards to feature tracking. Section 7.2 presents the benefits of the proposed gesture descriptor. Section 7.3 outlines the contributions for the learning-classification framework. Then we discuss the pros and cons of the global approach in section 7.4. Last, Section 7.5 encloses this manuscript by underlining the horizons for future work.

7.1 About Feature Tracking

Feature point tracking algorithms are, with optical flows, the most common approaches for motion extraction from moving cameras in computer vision. We have proposed a novel texture correlation- based technique by using HOG and Kalman filters. Instead of tracking corner points directly like with the KLT tracker, we associate to each corner point a local HOG descriptor and we track the latter with a Kalman filtering process. During the measure step of the Kalman filter, the algorithm searches for the best match of the previous descriptor in an ellipse build based on the descriptor velocity. Experiments show that our tracking algorithm outperforms the KLT tracker:

- The tracker has a correct computational efficiency which is comparable to the performance of KLT tracker.
- In term of effectiveness, the proposed tracker is able to match more features than KLT tracker.
- In term of reproducibility, we have found that the FAST based version performs better than the Shi-Tomasi based version.

7.2 About Gesture Descriptor

Our gesture descriptor is a novel representation of gesture which consists of a trade-off between local motion descriptor and global motion descriptor. Indeed, to go beyond the state of the art, we have proposed to track HOG descriptors over sufficiently long period of time thanks to a robust HoG tracker. The generated descriptors are used for gesture learning-clustering using the bag of word paradigm. From local descriptors, the proposed local motion descriptors take the advantage of coping with local motion. From global descriptors, we have inherited the track of motion through time of each local body corner, whereas traditional local motion descriptors consider a temporal window which is shorter and does not correspond exactly to the same local body region. Thus, we combine the advantages of global and local gesture descriptors to improve the quality of recognition. This gesture descriptor enables also two levels of spatial representation. Local description (*i.e.* texture) is obtained thanks to the HOG descriptors and global description at the person level can be provided by the position of the HOG descriptors relatively to the person gravity center.

We are not the first to attempt to use "tracked" HOG descriptors for action recognition, (Lu & Little 2006*a*) have proposed a method for action recognition using PCA-HOG. The main difference between our approach and this method is that, for one person, we track several local HOG descriptors whereas the others compute a single global HOG over the whole body of an already tracked person (it can be seen as a temporal global volume). Moreover, this approach was designed for recognizing coarse people gestures in low-resolution videos.

7.3 About Learning and Classification algorithms

We have proposed a learning-classification framework for gesture recognition using local motion descriptors as gesture representation. While state-of-the-art frameworks deal with many-to-one mapping between descriptors and gestures, the proposed method for learning and classifying gestures is based on a many-to-many mapping. Thus, we have introduced a novel voting process which transforms this complex mapping into a more conventional one. The key idea is to use a likelihood measure for each gesture given the observed local motion descriptors. The gesture with the higher likelihood wins the vote and its correspondent likelihood is considered as the recognition likelihood. The KTH database has been used to evaluate the proposed framework and our results show that it outperforms recent state-of-the-art methods validated on the same video database. Additionally, we have carried out a validation process on IXMAS database and we have obtained good preliminary results.

7.4 Discussion

Hereafter, we discuss how we have managed to reach our objectives and what at the end the limitations of the proposed method are. In the first chapter of this manuscript, we have postponed several objectives with several constraints and hypotheses. Below, we check if we have answered all of them:

- Mono-camera application: the proposed learning-classification framework is suitable for this kind of application since it can recognize gestures from various point of views. For that purpose, it is necessary and sufficient to learn each gestures from the different point of views.
- Fixed camera: as currently implemented the learning-classification framework deals only with fixed cameras. However, with appropriate preprocessing algorithms (*i.e.* people detector and tracker) for mobile camera, the framework is still valid. Nevertheless, the quality of the feature tracker should be checked for that kind of preprocessing algorithms.
- Real-time application: the local motion descriptor generator performs on-line and processes about 15 fps. We believe that with the use of an on-line version of the SVM classifier and with GPU implementation, we can reach nearly real-time performance for gesture recognition.
- Embeddable approach: the proposed algorithms are highly adaptable to parallel computing on GPU. Moreover, in all stages of the framework, we try to reduce data dimensionality: PCA for feature space dimension and MMI for learning database size. Hence, we should be able to embed this framework into a smart camera.
- Validity of the hypotheses: the proposed framework currently needs a people detector and a people tracker (*c.f.* SUP/VSIP platform), however it can be implemented to

require only a people detector. Since the people tracker helps the feature tracker to reduce the search zone for features and eliminates at a certain level of ambiguity for crowded scenes, the computational cost difference should be first evaluated. Moreover, the gesture classifier can be applied to crowded scene with the help of a good people tracker in order to recognize individual gestures learned on single actor video database. Finally, the feature tracker is more robust to noise than KLT tracker, so we can apply the proposed method to real world videos.

The main limitations of the proposed method are:

- Feature tracking: the proposed tracker is based on an extended Kalman filtering process. The main drawback of this tracker is the drift between the tracked HoG descriptor position and the actual feature (*i.e.* corner) position. Moreover, the used motion model does not account for rapid change in motion direction.

- Gesture representation: in the proposed framework, a gesture is modeled as a set of local motion descriptors. However, there is no time precedence constraints among the different local motion descriptors which constitutes a loss of information. Nevertheless, we believe that this loss does not influence the recognition process since different gestures contain different motion patterns (*e.g.* pre-stroke, stroke, post-stroke) which is sufficient to differentiate between them.

- Learning-classification framework: the learning and classification algorithms (k-means and k-nearest neighbors) require a priori knowledge of their parameter k. Hence, these algorithms require a tuning stage of the parameters which can be tedious and sensitive.

7.5 Future Work

The proposed learning-classification framework can be improved in several ways. In this section, we overview the future work split into short term perspectives and long term perspectives.

7.5.1 Short term perspectives

For short term perspectives, we are going forward in the evaluation process in order to test and validate the whole framework:

- Feature tracker evaluation: we are planning to go forward in the tracker evaluation by testing it on more challenging videos and validating it against a suitable ground truth. The results of the tracker shall be confronted to those of (Kim, Wong & Cipolla 2007) and (Lu & Little 2006*a*). Also, we will evaluate the impact of the choice of the preprocessing algorithms (*i.e.* people detector and tracker) on the tracker performance.

- Testing the on-line version of the k-nearest neighbors (k-NN) algorithm and estimating its computational efficiency.
- Testing the support vector machine (SVM) classifier and comparing its performances to those of the k-NN algorithm.
- Parameter tuning: we should improve the parameter tuning of the proposed algorithms by performing an Expectation-Maximization (EM) algorithm. Specially we should determine the k parameter for k-means clustering algorithm and the one for kNN.
- We should evaluate the incidence of the MMI algorithm on the classification effectiveness and efficiency for both classifiers (*i.e*. k-NN and SVM). Eventually, we should determine automatically the maximum tolerated threshold of loss of information given a particular training data-set.
- Re-evaluation of the whole framework on IXMAS multi-view database should be carried out in order to obtain more accurate results and to test the robustness of the framework with respect to the point of view change (*i.e*. learn from four views and classify from the remaining view). The Korea University Gesture (KUG) database(*c.f*. figure 7.1 and (Hwang et al. 2006)) can also be used to validate the multi-view independence of the proposed framework.
- Testing the classifiers on real-world videos such as TrecVid and Gerhome (*c.f*. Figure 7.2) with a predefined database of learned gestures (*c.f*. normal and abnormal gestures in KUG database) is to be carried out. (Smeaton et al. 2006) introduce general goals, guidelines, metrics and general results for TrecVid. Recently, (Smeaton et al. 2009) introduce the TrecVid's work in high-level feature extraction. A similar evaluation-validation process can be carried out for Gerhome database.

7.5.2 Long term perspectives

In long term, the learning-classification framework can be extended in different ways:

- Improving the HOG descriptor tracker: we could try the use of the unscented Kalman filter for HOG tracking to improve the quality of the tracking. Moreover, in order to eliminate the drift between HOG descriptors and the true position of corner points, we can explore some energy relaxation techniques such as proposed by (Galvin et al. 1999).
- Improving the gesture representation: A more compact gesture representation can be used by reducing the size of the texture information (*i.e*. HOG) of the local motion descriptor with PCA. Also, we can test the impact of adding the initial position w.r.t. the person gravity center of the local motion descriptor to the gesture representation.
- Improving the voting strategy of the gesture classifier by introducing weights. We believe that the introduction of the weights to the voting mechanism will reinforce the

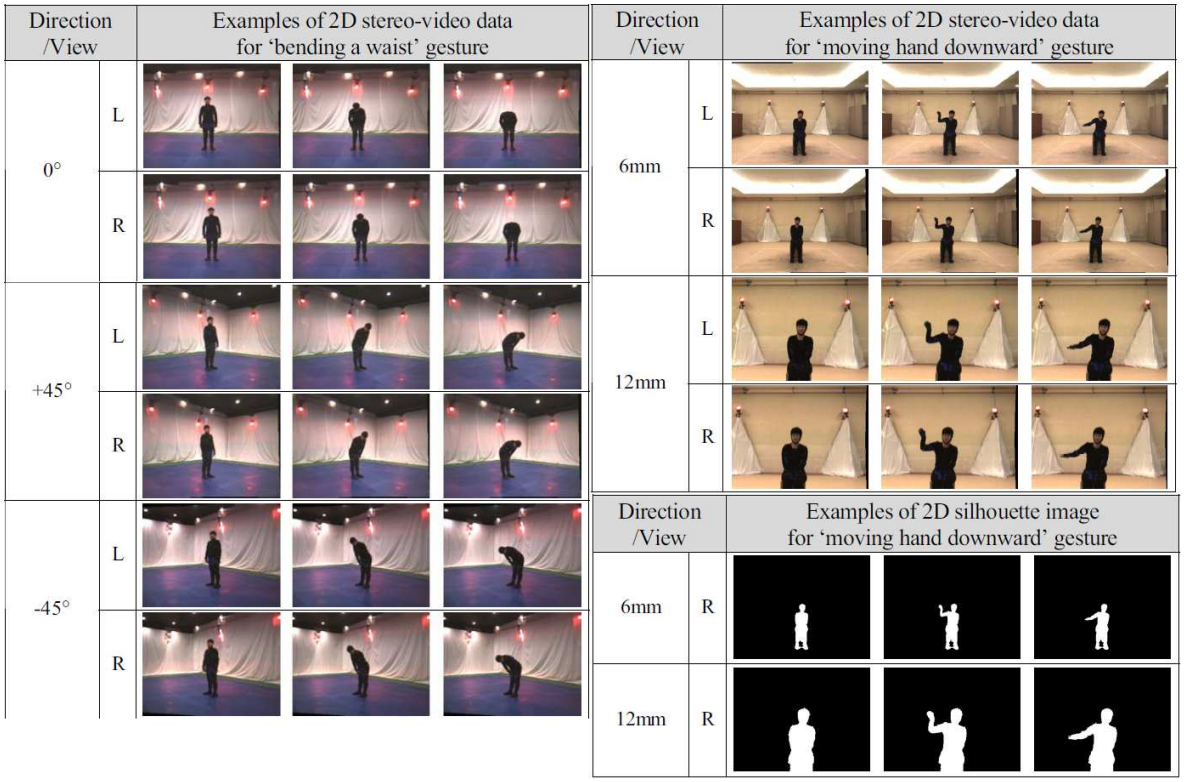


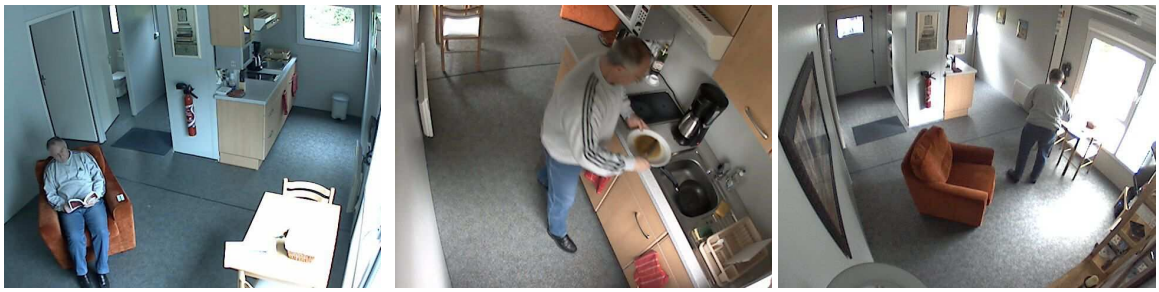
Figure 7.1: *Frames illustrating some gestures from the KUG video database*

pertinent descriptors and improve the quality of the recognition. Moreover, we can try the Mean Shift Clustering algorithm (Georgescu et al. 2003) instead of k-means which has the advantage of not requiring any prior knowledge of the number of clusters (*i.e.* non-parametric algorithm) and not constraining the shape of clusters which can be non-linearly separable. Mean Shift Clustering contrasts with k-means where the clusters are constrained to be spherically symmetric and their number has to be known a priori. Thus the Mean Shift Clustering algorithm becomes more adequate when the number of gestures to recognize (and specially the number of similar gestures) increases.

- Combine SVM and k-NN classification: some previous work of (Rong et al. 2001) and (Ming et al. 2003) proved that a combination of SVM with k-NN classifier provides a better trade-off between accuracy (*i.e.* precision and sensitivity) and computational efficiency compared to both of the isolated classifiers. So, we shall explore this combination in order to build a more accurate and efficient classifier (*c.f.* (Blanzieri & Bryl 2007)).

- Processing time improvement and embedding: the computational efficiency of the proposed framework can be improved by (1) optimizing the different algorithms and/or (2) reimplementing them on the Nvidia®CUDA platform (Standard C language for parallel computing with dedicated libraries) for Graphical Processing Unit (GPU) ex-

(a) TrecVid dataset



(b) Gerhome data-set

Figure 7.2: *Illustration of TrecVid and Gerhome video data-sets*

ecution. The former option has been discussed previously in this chapter and during this manuscript. The latter option is a step towards embedding the recognition algorithm into a smart camera: in fact, GPU miniaturization is progressing exponentially and their power consumption (TDP) decreases quickly with respect to their parallel computing performance: by the end of this year the first chips that integrate at the same time a CPU and a GPU will be in the market (AMD®Fusion chip and Intel®Havendale and Auburndale chips). Thus, we believe that the way to embed the proposed framework on a camera is to associate a GPU, a ROM memory for all algorithms and learning data and a RAM memory for execution. However, we can consider a more scalable implementation by embedding only the local motion descriptor generator into the camera and performing gesture classification on a remote server. In that way, adding a new gesture will impact only the server (*i.e.* updating the gesture codebook) and the improvement of the local motion descriptor generator will concern only the camera firmware (*i.e.* the embedded software into the camera).

Bibliography

Avanzi, A., Bérmond, F. c. & Thonnat, M. (2001), Tracking multiple individuals for video communication, *in* 'ICIP '01: Proceedings of the International Conference on Image Processing (ICIP01', Vol. 2, pp. 379–382.

Avanzi, A., Brémond, F., Tornieri, C. & Thonnat, M. (2005), 'Design and assessment of an intelligent activity monitoring platform.', *EURASIP Journal on Applied Signal Processing (JASP)* **14**, 2359–2374.

Baklouti, M., Monacelli, E., Guitteny, V. & Couvet, S. (2008), Intelligent assistive exoskeleton with vision based interface, *in* 'Proceedings of the 6th international conference on Smart Homes and Health Telematics', Vol. 5120 of *Lecture Notes In Computer Science*, Springer-Verlag, Berlin, Heidelberg, pp. 123–135.

Bar-Shalom, Y., Blackman, S. S. & Fitzgerald, R. J. (2007), 'Dimensionless score function for multiple hypothesis tracking', *Aerospace and Electronic Systems, IEEE Transactions on* **43**(1), 392–400.

Baxter, J. (2000), 'A model of inductive bias learning', *Journal of Artificial Intelligence Research* **12**, 149–198.

Birdal, A. & Hassanpour, R. (2008), Region based hand gesture recognition, *in* '16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision', pp. 1–7.

Birdwhistell, R. L. (1963), 'The kinesic level in investigations of the emotions', *Knapp P H (ed.) Expression of the Emotions in Man* **II**, 123–139.

Bishop, C. M. (2006), *Information Theory, Inference, and Learning Algorithms*, Springer. ISBN 0387310738.

Black, M. J. & Jepson, A. D. (1998), A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions., *in* 'European Conference on Computer Vision', Vol. 1406 of *Lecture Notes in Computer Science*, Springer, pp. 909–924.

Blanzieri, E. & Bryl, A. (2007), Evaluation of the highest probability svm nearest neighbor classifier with variable relative error cost, *in* 'Fourth Conference on Email and Anti-Spam (CEAS)'.

Bobick, A. F. & Davis, J. (2001), 'The recognition of human movement using temporal templates', *IEEE Trans. Pattern Analysis and Machine Intelligence* **23**(3), 257–267.

Boulay, B. (2007), Human posture recognition for behaviour understanding, PhD thesis, Université de Nice-Sophia Antipolis.

Bourke, A., O'Brien, J. & Lyons, G. (2007), 'Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm', *Gait & Posture* **26**(2), 194–199.
URL: *http://www.sciencedirect.com/science/article/B6T6Y-4MBCJHV-1/2/f87e4f1c82f3f93a3a5692357e3fe00c*

Bowden, R., Windridge, D., T., K., Zisserman, A. & Brady, M. (2004), A linguistic feature vector for the visual interpretation of sign language, *in* 'European Conference on Computer Vision', Vol. 1, Springer-Verlag, pp. 391–401.

Bretzner, L., Laptev, I. & Lindeberg, T. (2002), Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering, *in* 'Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on', pp. 405–410.
URL: *http://dx.doi.org/10.1109/AFGR.2002.1004190*

Calderara, S., Cucchiara, R. & Prati, A. (2008), Action signature: A novel holistic representation for action recognition, *in* 'International Conference on Advanced Video and Signal Based Surveillance', IEEE Computer Society Press, Washington, DC, USA, pp. 121–128.

Cheng, Y. (1995), 'Mean shift, mode seeking, and clustering', *IEEE Trans. Pattern Analysis and Machine Intelligence* **17**(8), 790–799.

Chu, C.-W. & Cohen, I. (2005), Posture and gesture recognition using 3d body shapes decomposition, *in* 'CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops', IEEE Computer Society, Washington, DC, USA, pp. 69–76.

Cucchiara, R., Grana, C., Piccardi, M. & Prati, A. (2003), 'Detecting moving objects, ghosts, and shadows in video streams', *IEEE Trans. Pattern Analysis and Machine Intelligence* **25**(10), 1337–1342.

Dalal, N. & Triggs, B. (2005), Histograms of oriented gradients for human detection, *in* 'International Conference on Computer Vision and Pattern Recognition', Vol. 1, IEEE Computer Society Press, San Diego, CA, USA, pp. 886–893.
URL: *http://www.acemedia.org/aceMedia/files/document/wp7/2005/cvpr05-inria.pdf*

Darrell, T. & Pentland, A. (1996), Active gesture recognition using partially observable markov decision processes, *in* 'International Conference on Pattern Recognition', Vol. 3, IEEE Computer Society Press, Washington, DC, USA, pp. 984–989.

Dietterich, T. G. & Bakiri, G. (1995), 'Solving multiclass learning problems via error-correcting output codes', *Journal of Artificial Intelligence Research* **2**, 263–286.

Dollar, P., Rabaud, V., Cottrell, G. & Belongie, S. (2005), Behavior recognition via sparse spatio-temporal features, *in* 'Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on', pp. 65–72.
URL: *http://dx.doi.org/10.1109/VSPETS.2005.1570899*

Fang, G., Gao, W. & Zhao, D. (2007), 'Large-vocabulary continuous sign language recognition based on transition-movement models', *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* **37**(1), 1–9.

Galvin, B., McCane, B. & Novins, K. (1999), Robust feature tracking, *in* 'Fifth International Conference on Digital Image Computing, Techniques, and Applications (DICTA)'.

Georgescu, B., Shimshoni, I. & Meer, P. (2003), Mean shift based clustering in high dimensions: A texture classification example, *in* 'International Conference on Computer Vision', Vol. 1, IEEE Computer Society Press, Los Alamitos, CA, USA, p. 456.

Ger (2009), 'Gerhome website'.
URL: *http:://gerhome.cstb.fr/*

Gorelick, L., Blank, M., Shechtman, E., Irani, M. & Basri, R. (2007), 'Actions as space-time shapes', *IEEE Trans. Pattern Analysis and Machine Intelligence* **29**(12), 2247–2253.

Gusfield, D. (1997), *Algorithms on strings, trees, and sequences: computer science and computational biology*, Cambridge University Press, New York, NY, USA.

Hall, E. T. (1973), *The silent language*, Anchor Books. ISBN-13: 978-0385055499.

Hearst, M. A. (1998), 'Support vector machines', *IEEE Intelligent Systems* **13**(4), 18–28.

Hiroshi, K., Makito, S., Kazuhiko, S., Ken-ichi, T. & Kazuo, K. (2006), Pattern recognition for video surveillance and physical security, Technical Report 375, The Institute of Electronics, Information and Communication Engineers.
URL: `http://ci.nii.ac.jp/naid/110005717906/en/`

Hwang, B.-W., Kim, S. & Lee, S.-W. (2006), A full-body gesture database for automatic gesture recognition, *in* 'Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on', IEEE Computer Society Press, pp. 243–248.

Isard, M. & Blake, A. (1996), Contour tracking by stochastic propagation of conditional density, *in* 'European Conference on Computer Vision', Vol. 1064 of *Lecture Notes in Computer Science*, Springer, pp. 343–356.

Isard, M. & Blake, A. (1998), A mixed-state condensation tracker with automatic model-switching, *in* 'International Conference on Computer Vision', Narosa Publishing House, pp. 107–112.

Izo, T. & Grimson, W. E. L. (2007), 'Simultaneous pose recovery and camera registration from multiple views of a walking person', *Image and Vision Computing* **25**(3), 342–351.

Jaimes, A., Tseng, B. L. & Smith, J. R. (2003), Modal keywords, ontologies, and reasoning for video understanding, *in* 'Proceedings of the International Conference On Image and Video Retrieval', Vol. 2728 of *Lecture Notes in Computer Science*, Springer, pp. 248–259.

Jolliffe, I. T. (2002), *Principal Component Analysis: Second Edition*, Springer Verlag. ISBN-13: 978-0387954424.

Ju, S. X., Black, M. J., Minneman, S. & Kimber, D. (1997), Analysis of gesture and action in technical talks for video indexing, Technical report, American Association for Artificial Intelligence. AAAI Technical Report SS-97-03.

Julier, S. J. & Uhlmann, J. K. (1997), A new extension of the kalman filter to nonlinear systems, *in* I. Kadar, ed., 'Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series', Vol. 3068 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 182–193. Provided by the SAO/NASA Astrophysics Data System.
URL: `http://adsabs.harvard.edu/abs/1997SPIE.3068..182J`

Kale, A., Sundaresan, A., Rajagopalan, A., Cuntoor, N., Roy-Chowdhury, A., Kruger, V. & Chellappa, R. (2004), 'Identification of humans using gait', *Image Processing, IEEE Transactions on* **13**(9), 1163–1173.

Kalman, R. E. (1960), 'A new approach to linear filtering and prediction problems', *Transactions of the ASME–Journal of Basic Engineering* **82**(Series D), 35–45.

Keskin, C., Balci, K., Aran, O., Sankur, B. & Akarun, L. (2007), A multimodal 3d healthcare communication system, *in* 'IEEE 3DTV Conference', IEEE Computer Society Press, pp. 1–4.

Kettebekov, S., Yeasin, M. & Sharma, R. (2005), 'Prosody based audiovisual coanalysis for coverbal gesture recognition', *Multimedia, IEEE Transactions on* **7**(2), 234–242.

Kevin, N. Y. Y., Ranganath, S. & Ghosh, D. (2004), Trajectory modeling in gesture recognition using cybergloves®and magnetic trackers, *in* 'TENCON 2004. 2004 IEEE Region 10 Conference', Vol. A, pp. 571–574.

Kim, D., Song, J. & Kim, D. (2007), 'Simultaneous gesture segmentation and recognition based on forward spotting accumulative hmms', *Pattern Recognition* **40**(11), 3012–3026.

Kim, T.-K., Wong, S.-F. & Cipolla, R. (2007), Tensor canonical correlation analysis for action classification, *in* 'International Conference on Computer Vision and Pattern Recognition', pp. 1–8.

Kovesi, P. (n.d.), 'Image features from phase congruency', *Videre: A Journal of Computer Vision Research.* **1**(3).

Krout, M. H. (1935), 'Autistic gestures: An experimental study in symbolic movement', *Psychological Monographs* **46**, 119–120.

Laptev, I. (2005), 'On space-time interest points', *International Journal of Computer Vision* **64**(2/3), 107–123.

Lee, H.-K. & Kim, J. H. (1999), 'An hmm-based threshold model approach for gesture recognition', *IEEE Trans. Pattern Analysis and Machine Intelligence* **21**(10), 961–973.

Lee, Y. W. (2008), Application of the particle filter for simple gesture recognition, *in* 'International Conference on Intelligent Computing', Vol. 5227 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 534–540.

Leibe, B., Seemann, E. & Schiele, B. (2005), Pedestrian detection in crowded scenes, *in* 'International Conference on Computer Vision and Pattern Recognition', Vol. 1, pp. 878–885.

Leitner, F. & Cinquin, P. (1993), From splines and snakes to snake splines, *in* 'Selected Papers from the Workshop on Geometric Reasoning for Perception and Action', Springer-Verlag, London, UK, pp. 264–281.

Li, F. & Wechsler, H. (2005), 'Open set face recognition using transduction', *IEEE Trans. Pattern Analysis and Machine Intelligence* **27**(11), 1686–1697.

Liu, J. & Shah, M. (2008), Learning human actions via information maximization, *in* 'International Conference on Computer Vision and Pattern Recognition', IEEE Computer Society Press, Anchorage, Alaska, USA, pp. 1–8.
URL: `http:://longwood.cs.ucf.edu/~vision/papers/cvpr2008/5.pdf`

Liu, Z. & Sarkar, S. (2005), 'Effect of silhouette quality on hard problems in gait recognition', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **35**(2), 170–183.

Lu, W.-L. & Little, J. J. (2006*a*), Simultaneous tracking and action recognition using the pca-hog descriptor, *in* 'Computer and Robot Vision, 2006. The 3rd Canadian Conference on', Quebec, Canada, pp. 6–13.

Lu, W. L. & Little, J. J. (2006*b*), Tracking and recognizing actions at a distance, *in* 'Proceedings of the ECCV Workshop on Computer Vision Based Analysis in Sport Environments (CVBASE '06)', Graz, Austria.

Luo, Q., Kong, X., Zeng, G. & Fan, J. (2008), 'Human action detection via boosted local motion histograms', *Machine Vision and Applications* .
URL: `http://dx.doi.org/10.1007/s00138-008-0168-5`

Lv, F. & Nevatia, R. (2007), Single view human action recognition using key pose matching and viterbi path searching, *in* 'International Conference on Computer Vision and Pattern Recognition', IEEE Computer Society Press.

McNeill, D. (1992), *Hand and Mind: What Gestures Reveal about Thought*, University Of Chicago Press. ISBN: 9780226561325.

Mikolajczyk, K., Schmid, C. & Zisserman, A. (2004), Human detection based on a probabilistic assembly of robust part detectors, *in* 'Computer Vision - ECCV 2004', Vol. 3021, Springer Berlin / Heidelberg, pp. 69–82.

Mindru, F., Tuytelaars, T., Van Gool, L. & Moons, T. (2004), 'Moment invariants for recognition under changing viewpoint and illumination', *Computer Vision and Image Understanding* **94**(1-3), 3–27.

Ming, T., Yi, Z. & Songcan, C. (2003), Improving support vector machine classifier by combining it with k nearest neighbor principle based on the best distance measurement, *in* 'Proceedings of the IEEE Conference on Intelligent Transportation Systems', Vol. 1, pp. 373–378.

Moeslund, T. B. (2008), *Image and Video Processing*, Aalborg Universitet, Denmark.

Munoz-Salinas, R., Medina-Carnicer, R., Madrid-Cuevas, F. J. & Carmona-Poyato, A. (2008), 'Depth silhouettes for gesture recognition', *Pattern Recognition Letters* **29**(3), 319–329.

Nickel, K. & Stiefelhagen, R. (2007), 'Visual recognition of pointing gestures for human-robot interaction', *Image Vision Computing* **25**(12), 1875–1884.

Noury, N., Barralon, P., Virone, G., Boissy, P., Hamel, M. & Rumeau, P. (2003), A smart sensor based on rules and its evaluation in daily routines, *in* 'Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE', Vol. 4, pp. 3286–3289.

Ottenheimer, H. J. (2005), *The Anthropology of Language: An Introduction to Linguistic Anthropology*, Wadsworth Publishing. ISBN-13: 978-0534594367.

Papageorgiou, C. & Poggio, T. (2000), 'A trainable system for object detection', *International Journal of Computer Vision* **38**(1), 15–33.

Pei, M. (1984), *The Story of Language*, Plume; Rep Rev edition. ISBN-13: 978-0452008700.

Pinhanez, C. & Bobick, A. (1997), Human action detection using pnf propagation of temporal constraints, *in* 'International Conference on Computer Vision and Pattern Recognition', pp. 898–904.

Quinlan, J. R. (1990), Induction of decision trees, *in* J. W. Shavlik & T. G. Dietterich, eds, 'Readings in Machine Learning', Morgan Kaufmann. Originally published in *Machine Learning* 1:81–106, 1986.

Qureshi, F. Z., Terzopoulos, D. & Jasiobedzki, P. (2004), A cognitive vision system for space robotics, *in* 'Proceedings of the ECCV 2004 Workshop on Applications of Computer Vision', pp. 120–128.

Ren, Y. & Zhang, F. (2009), Hand gesture recognition based on meb-svm, *in* 'Embedded Software and Systems, Second International Conference on', Vol. 0, IEEE Computer Society, Los Alamitos, CA, USA, pp. 344–349.

Roh, M.-C., Shin, H.-K., Lee, S.-W. & Lee, S.-W. (2006), Volume motion template for view-invariant gesture recognition, *in* 'ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition', IEEE Computer Society, Washington, DC, USA, pp. 1229–1232.

Rong, L., Shiwei, L. & Zhongzhi, S. (2001), A effective classified algorithm of support vector machine with multi-representative points based on nearest neighbor principle, *in* 'International Conferences on Info-tech and Info-net (ICII)', Vol. 3, pp. 113–119.

Rosten, E. & Drummond, T. (2006), Machine learning for high-speed corner detection, *in* 'European Conference on Computer Vision', Vol. 1, Springer, Graz, Austria, pp. 430–443.

Schlömer, T., Poppinga, B., Henze, N. & Boll, S. (2008), Gesture recognition with a wii controller, *in* 'TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction', ACM, New York, NY, USA, pp. 11–14.

Schuldt, C., Laptev, I. & Caputo, B. (2004), Recognizing human actions: a local svm approach, *in* 'International Conference on Pattern Recognition', Vol. 3, IEEE Computer Society Press, Cambridge, UK, pp. 32–36.

Scovanner, P., Ali, S. & Shah, M. (2007), A 3-dimensional sift descriptor and its application to action recognition, *in* 'ACM International Conference on Multimedia', ACM, New York, NY, USA, pp. 357–360.

Shi, J. & Tomasi, C. (1994), Good features to track, *in* 'International Conference on Computer Vision and Pattern Recognition', Springer, Seattle, WA, USA, pp. 593–600.

Signer, B., Kurmann, U. & Norrie, M. C. (2007), igesture: A general gesture recognition framework, *in* 'Proceedings of ICDAR 2007, 9th International Conference on Document Analysis and Recognition'.

Smeaton, A. F., Over, P. & Kraaij, W. (2006), Evaluation campaigns and trecvid, *in* 'MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval', ACM Press, New York, NY, USA, pp. 321–330.

Smeaton, A. F., Over, P. & Kraaij, W. (2009), High-Level Feature Detection from Video in TRECVid: a 5-Year Retrospective of Achievements, *in* A. Divakaran, ed., 'Multimedia Content Analysis, Theory and Applications', Springer Verlag, Berlin, Germany, pp. 151–174.

Swee, T. T., Salleh, S.-H., Ariff, A., Ting, C.-M., Seng, S. K. & Huat, L. S. (2007), Malay sign language gesture recognition system, *in* 'Intelligent and Advanced Systems, 2007. ICIAS 2007. International Conference on', pp. 982–985.

Thrun, S. & Pratt, L. (1998), *Learning to Learn*, Kluwer Academic Publishers, Norwell, Massachusetts, USA.

Tishby, N., Pereira, F. C. & Bialek, W. (1999), The information bottleneck method, *in* 'The 37th annual Allerton Conference on Communication, Control, and Computing', pp. 368–377.

Tre (2008), 'Trecvid 2008 website'.
URL: *http://www-nlpir.nist.gov/projects/tv2008/*

Van Gool, L., Moons, T. & Ungureanu, D. (1996), Affine / photometric invariants for planar intensity patterns, *in* 'European Conference on Computer Vision', Lecture Notes in Computer Science, pp. 642–651.
URL: *http://www.springerlink.com/content/f31215u5843p4058*

Wang, L., Hu, W. & Tan, T. (2003), 'Recent developments in human motion analysis', *Pattern Recognition* **36**(3), 585–601.
URL: *http://www.sciencedirect.com/science/article/B6V14-4771RWD-2/2/41d0f27ca154e3140e27fdf68a9df519*

Washburn, A. R. (2007), 'A short introduction to kalman filters'. Naval Postgraduate School.
URL: *http://diana.cs.nps.navy.mil/~arwashbu/Files/KalmanIntro.pdf*

Webel, S., Keil, J. & Zoellner, M. (2008), Multi-touch gestural interaction in x3d using hidden markov models, *in* 'VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology', ACM, New York, NY, USA, pp. 263–264.

Weinland, D., Boyer, E. & Ronfard, R. (2007), Action recognition from arbitrary views using 3d exemplars, *in* 'International Conference on Computer Vision', IEEE Computer Society Press, Rio de Janeiro, Brazil, pp. 1–7.
URL: *http::/perception.inrialpes.fr/Publications/2007/WBR07*

Wu, B. & Nevatia, R. (2005), Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors, *in* 'ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)', Vol. 1, IEEE Computer Society, Washington, DC, USA, pp. 90–97.

Yamato, J., Ohya, J. & Ishii, K. (1992), Recognizing human action in time-sequential images using hidden markov model, *in* 'International Conference on Computer Vision and Pattern Recognition', pp. 379–385.

Yilmaz, A. & Shah, M. (2008), 'A differential geometric approach to representing the human actions', *Computer Vision and Image Understanding* **109**(3), 335–351.

Zuniga, M. (2008), Incremental Learning of Events in Video using Reliable Information, PhD thesis, Université de Nice-Sophia Antipolis.

Index

RÉSUMÉ

Dans cette thèse, nous voulons reconnaître les gestes (par ex. lever la main) et plus généralement les actions brèves (par ex. tomber, se baisser) effectués par un individu. De nombreux travaux ont été proposés afin de reconnaître des gestes dans un contexte précis (par ex. en laboratoire) à l'aide d'une multiplicité de capteurs (par ex. réseaux de cameras ou individu observé muni de marqueurs). Malgré ces hypothèses simplificatrices, la reconnaissance de gestes reste souvent ambigüe en fonction de la position de l'individu par rapport aux caméras. Nous proposons de réduire ces hypothèses afin de concevoir un algorithme général permettant de reconnaître des gestes d'un individu évoluant dans un environnement quelconque et observé à l'aide d'un nombre réduit de caméras. Il s'agit d'estimer la vraisemblance de la reconnaissance des gestes en fonction des conditions d'observation. Notre méthode consiste à classifier un ensemble de gestes à partir de l'apprentissage de descripteurs de mouvement. Les descripteurs de mouvement sont des signatures locales du mouvement de points d'intérêt associés aux descriptions locales de la texture du voisinage des points considérés. L'approche a été validée sur une base de données de gestes publique KTH et des résultats encourageants ont été obtenus.

Mots-clés: reconnaissance de gestes, vision par ordinateur, reconnaissance de comportements, suivie de points d'intérêts, descripteurs de mouvements, filtres de Kalman, apprentissage et classification statistiques.

ABSTRACT

In this thesis, we aim to recognize gestures (*e.g.* hand raising) and more generally short actions (*e.g.* fall, bending) accomplished by an individual. Many techniques have already been proposed for gesture recognition in specific environment (*e.g.* laboratory) using the cooperation of several sensors (*e.g.* camera network, individual equipped with markers). Despite these strong hypotheses, gesture recognition is still brittle and often depends on the position of the individual relatively to the cameras. We propose to reduce these hypotheses in order to conceive general algorithm enabling the recognition of the gesture of an individual involving in an unconstrained environment and observed through limited number of cameras. The goal is to estimate the likelihood of gesture recognition in function of the observation conditions. Our method consists of classifying a set of gestures by learning motion descriptors. These motion descriptors are local signatures of the motion of corner points which are associated with their local textural description. We demonstrate the effectiveness of our motion descriptors by recognizing the actions of the public KTH database.

Keywords: gesture recognition, computer vision, behavior recognition, feature point tracking, motion descriptors, Kalman filters, statistical learning and classification.