Dynamic and Robust Object Tracking for Activity Recognition

Duc Phu Chau

UNIVERSITE DE NICE-SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour l'obtention du grade de

Docteur en Sciences

de l'Université de Nice - Sophia Antipolis

Mention : INFORMATIQUE

presentée et soutenue par

Duc Phu CHAU

DYMANIC AND ROBUST OBJECT TRACKING FOR ACTIVITY RECOGINITION

Thèse dirigée par Monique THONNAT
et co-dirigée par François BREMOND

soutenue le 30/3/2012

Jury :

Jean-Paul	RIGAULT	Professeur, Université de Nice - Sophia Antipolis	Président
Frédéric	LERASLE	Professeur, Université Paul Sabatier	Rapporteur
Cina	MOTAMED	MdC HDR, Université du Littoral Côte d'Opale	Rapporteur
James L.	CROWLEY	Professeur, INP de Grenoble	Examinateur
Monique	THONNAT	DR1, INRIA Sophia Antipolis - Méditerranée	Directrice de thèse
François	BREMOND	DR2, INRIA Sophia Antipolis - Méditerranée	Co-directeur de thèse

T H È S E

DYNAMIC AND ROBUST OBJECT TRACKING FOR ACTIVITY RECOGNITION

DUC PHU CHAU

3 - 2012

ACKNOWLEDGMENTS

I would like to thank Frédéric LERASLE, professor of Paul Sabatier University, and Cina MOTAMED, Maître de Conférences HDR of University of Littoral Côte d'Opale for accepting to review my PhD manuscript and for their pertinent feedbacks.

Thank Prof. James CROWLEY for participating in the jury committee of my PhD defence. I also would like to give my thanks to Prof. Jean-Paul RIGAULT for accepting to be the president of the committee. Thank Eric DALMASSO, manager of Accenture, for his remarks on my work.

I sincerely thank my thesis supervisors Monique THONNAT and François BREMOND for what they have done for me. It is my great chance to work with them. Thanks for teaching me how to communicate with the scientific community, for being very patient to repeat the scientific explanations several times due to my limitations on knowledge and foreign language.

I appreciate Monique THONNAT, my main supervisor, for being always strict with me during my PhD study. Her high requirements have helped me to obtain significant progress in my research capacity. She guided me the necessary skills to express and formalize the scientific ideas. Thanks for giving me a lot of new ideas to improve my thesis. I am sorry not to be a good enough student to understand quickly and explore all these ideas in this manuscript. Without her supervision, my PhD study cannot be completed on time. A great thank goes for all of them.

I thank François BREMOND, my co-supervisor, for his availability and kindness. He has taught me the necessary scientific and technical knowledge as well as redaction aspects for my PhD study. He also gave me all necessary supports so that I could complete this thesis. I have also learned from him how to face to the difficult situations and how important the human relationship is. I really appreciate them.

I then would like to acknowledge Christine CLAUX and Vanessa WALLET for helping me to

solve a lot of complex administrative problems that I never imagine.

Many special thanks are also to all of my colleagues in the STARS team for their kindness as well as their scientific and technical supports during my thesis period, especially Etienne, Daniel, Erwan, Julien, Bernard, Slawomir, Anh Tuan, Guillaume and Annie. All of them have given me a very warm and friendly working environment. More appreciation is for Etienne, Luis, Slawomir, Veronique, Anh Tuan and Nadia for our friendship.

Big thanks are to my Vietnamese friends for helping me to overcome my homesickness, specially to Dan, Thang, Tram, Van, Trung, Yen and the Vietnamese group living at the Jean-Médecin student residence (Canh, Duong, Huong, Hue and Hong). I will always keep in mind all good moments we have spent together.

I also appreciate my colleagues from the Technology Department of Phu Xuan Private University (Hue city, Vietnam) who have given me the best conditions so that I could completely focus on my study in France. I sincerely thank Dr. TRAN Quang Tuyet, vice-director of the University, for his kindness and supports to my study plan.

Thank my teachers at The Information Technology Department of Hue University of Sciences (Hue city, Vietnam) and at The French Institute for Computer Science (IFI) (Hanoi city, Vietnam) for teaching me computer science knowledge from basic to advanced levels. I appreciate IFI giving me the opportunity for studying in France. I also sincerely thank Dr. TRUONG Cong Tuan for his supervision of my license thesis as well as his encouragement during my PhD study.

A big thank to my all family members for their full encouragements and perfect supports during my studies. It has been more than five years since I lived far from family. It is short and quick but still long enough for helping me to recognize how important my family is in my life.

The most special and greatest thanks are for my wife, HOANG Thi Vui. Thanks for supporting me entirely and perfectly all along my PhD study. Thanks for being always beside me and sharing with me all happy as well as hard moments. This thesis is thanks to her and is for her.

Finally, I would like to thank and to present my excuses to all the persons I have forgotten to mention in this section.

CHAU Duc Phu

Sophia Antipolis (France), spring 2012

RÉSUMÉ

Cette thèse présente une nouvelle approche pour contrôler des algorithmes de suivi d'objets mobiles. Plus précisément, afin de s'adapter aux variations contextuelles de suivi, cette approche apprend à régler les paramètres des algorithmes de suivi d'objets basés sur l'apparence ou sur les points d'intérêt. Le contexte de suivi d'une vidéo est défini par un ensemble de caractéristiques : la densité des objets mobiles, leur niveau d'occultation, leur contraste et leur surface 2D. Dans une phase d'apprentissage supervisée, des valeurs de paramètres satisfaisantes sont déterminées pour chaque séquence d'apprentissage. Puis ces séquences sont classifiées en groupant leurs caractéristiques contextuelles. A chaque contexte sont associées les valeurs de paramètres apprises. Dans la phase de contrôle en ligne, deux approches sont proposées. Pour la première approche, quand le contexte change, les paramètres sont réglés en utilisant les valeurs apprises. Pour la deuxième, le réglage des paramètres est réalisé quand le contexte change et quand la qualité de suivi (calculée par un algorithme d'évaluation en ligne) n'est pas assez bonne. Un processus d'apprentissage en-ligne met à jour les relations contextes/paramètres. L'approche a été expérimentée avec des vidéos longues, complexes et plusieurs ensembles de vidéos publiques. Cette thèse propose cinq contributions : (1) une méthode de classification des vidéos pour apprendre hors-ligne les paramètres de suivi, (2) un algorithme d'évaluation en-ligne du suivi, (3) une méthode pour contrôler en ligne le suivi, (4) un algorithme de suivi pouvant s'adapter aux conditions de la scène, (5) une méthode de suivi robuste basée sur le filtre de Kalman et un suivi global.

Mot clés : contrôleur, réglage des paramètres en ligne, suivi d'objets, apprentissage, évaluation en ligne.

ABSTRACT

This thesis presents a new control approach for mobile object tracking. More precisely in order to cope with the tracking context variations, this approach learns how to tune the parameters of tracking algorithms based on object appearance or points of interest. The tracking context of a video sequence is defined as a set of features : density of mobile objects, their occlusion level, their contrasts with regard to the background and their 2D areas. Each contextual feature is represented by a code-book model. In an offline supervised learning phase, satisfactory tracking parameters are searched for each training video sequence. Then these video sequences are classified by clustering their contextual features. Each context cluster is associated with the learned tracking parameters. In the online control phase, two approaches are proposed. In the first one, once a context change is detected, the tracking parameters are tuned using the learned values. In the second approach, the parameter tuning is performed when the context changes and the tracking quality (computed by an online evaluation algorithm) is not good enough. An online learning process enables to update the context/parameter relations. The approach has been experimented on long, complex videos and some public video datasets. This thesis proposes five contributions : (1) a classification method of video sequences to learn offline the tracking parameters, (2) an online tracking evaluation algorithm, (3) a method to tune and learn online the tracking parameters, (4) a tunable object descriptor-based tracking algorithm enabling adaptation to scene conditions, (5) a robust mobile object tracker based on Kalman filter and global tracking.

Keywords : controller, online parameter tuning, self-adaptation, object tracking, machine learning, online tracking evaluation.

CONTENTS

FIGURES

TABLES

1

INTRODUCTION

1.1 Motivations

Nowadays video surveillance systems are installed worldwide in many different sites such as airports, hospitals, banks, railway stations and even at home (see figure 1.1). The surveillance cameras help a supervisor to oversee many different areas from the same room and to quickly focus on abnormal events taking place in the controlled space. However one question arises : how can a security officer analyse and simultaneously dozens of monitors with a minimum rate of missing abnormal events (see figure 1.2) in real time ? Moreover, the observation of many screens for a long period of time becomes tedious and draws the supervisor's attention away from the events of interest. The solution to this issue lies in three words : intelligent video monitoring.

The term "intelligent video monitoring" expresses a fairly large research direction that is applied in different fields : for example in robotics and home-care. In particular, a lot of researches and works are already achieved in video surveillance applications. Figure 1.3 presents a processing chain of a video interpretation system for action recognition. Such a chain includes generally different tasks : image acquisition, object detection, object classification, object tracking and activity recognition. This thesis studies the mobile object tracking task.

Mobile object tracking has an important role in the computer vision applications such as home care, sport scene analysis and video surveillance-based security systems (e.g. in bank, parking, airport). In term of vision tasks, the object tracking task provides object trajectories for several tasks such as activity recognition, learning of interest zones or paths in a scene and detection of events of interest.

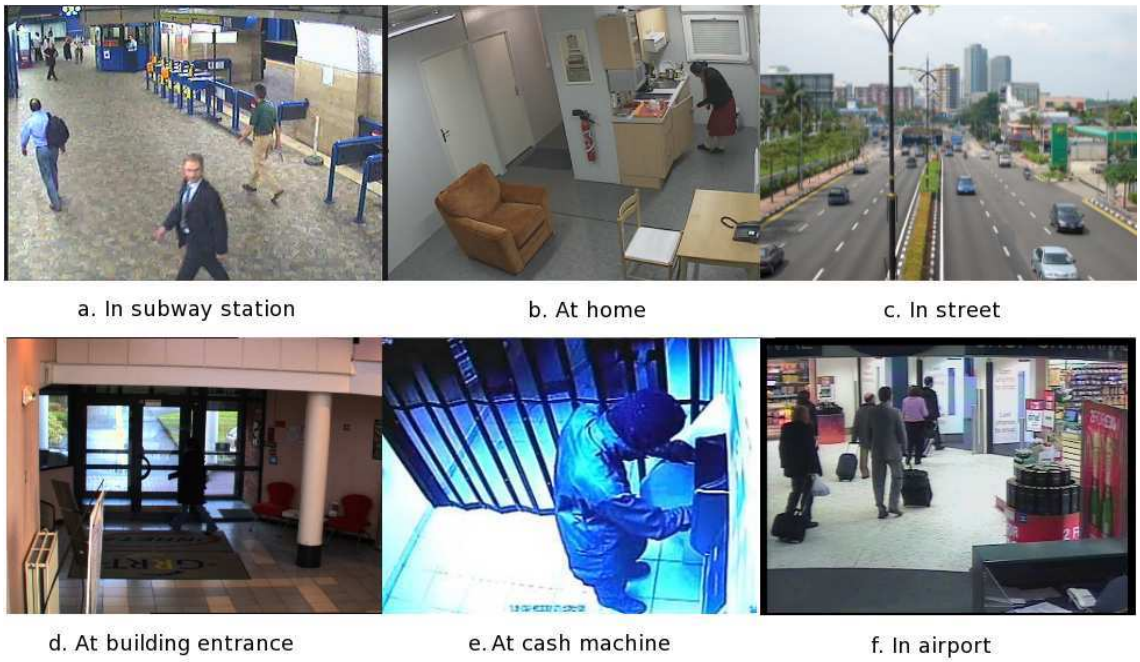


Figure 1.1 – Illustration of some areas monitored by video cameras

Figure 1.2 – A control room for video surveillance (source [securite-surveillance.com, 2009])

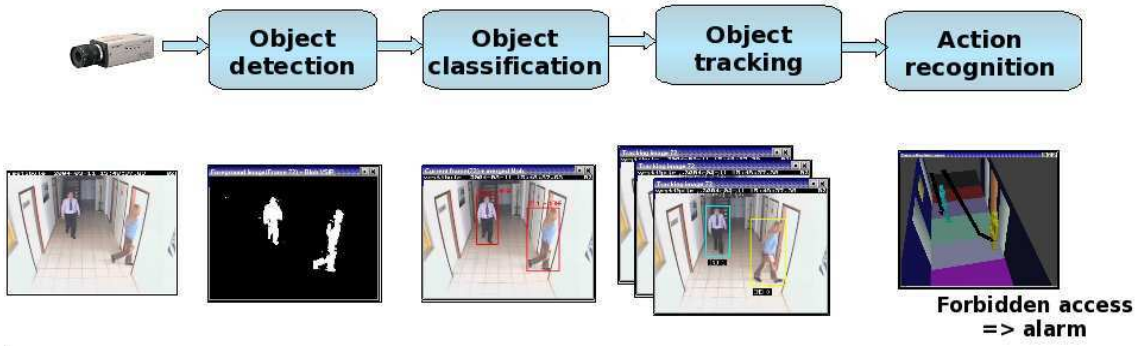


Figure 1.3 – Illustration of a video interpretation system. The first row presents the task names. The second row presents result illustrations of corresponding tasks.

The tracking quality can be influenced by the quality of vision tasks performed at lower levels such as object detection, object classification, or by some video features such as complexity of object movements, scene illumination intensity, low contrast, high density and occlusion frequence of mobile objects. Therefore, the performance of a tracking algorithm often depends on the considered video sequence. In particular, for a long video sequence (i.e several hours or days) in which the variation of these properties happens frequently, the tracking quality is still an issue. The problems are the following : How can an automatic system robustly track mobile objects in different conditions and situations such as the ones cited above. And in those complex cases, how can the user regulate the tracking parameters to get an optimal tracking quality ?

1.2 Objectives and Hypotheses

1.2.1 Objectives

In this thesis, we propose a new method for controlling tracking algorithms. The first ideas about an automatic control system which helps to adapt system performance to the problem of contextual variations, have been studied by [Shekhar et al., 1999, Thonnat et al., 1999, Khoudour et al., 2001]. Depending on the existence of a human interaction (e.g with an expert, an end-user) during the testing phase, we can classify the control methods in two categories : manual control (i.e. with human interaction) and automatic control (i.e. without human interaction).

The objective of the proposed method is to define an automatic control algorithm which is able to adapt online the tracking task to the scene variations in a long video sequence (i.e. several hours or days) by tuning the tracking parameters over time.

We aim to build a control algorithm which is : **generic**, **flexible** and **intelligent**. The term

"**generic**" means that our method can handle different tracking algorithms. In this work, our objective is to control object tracking algorithms which rely on object appearance or on points of interest. These algorithms are selected because their approaches are largely studied in the state of the art. The term "**flexible**" implies that the structure of the proposed control algorithm can be applied for handling other tracking algorithm classes (e.g. object contour-based tracking) just by changing the definition of the video context. The term "**intelligent**" means that this approach requires less human interaction than the control methods in the state of the art.

In order to reach these three objectives, the four following issues have to be tackled :

- **Define the video context :** The quality of object appearance-based tracker usually depends on some features such as the density, contrast, size of mobile objects appearing in the scene. In order to optimize the tracking quality in all situations, we need to model the videos according to these features. The first issue is thus to define a video context based on pertinent features characterizing the video sequences.
- **Optimize the tracking parameter values for a video context :** Once a video context is computed, we need to find the best tracking parameter values for this context. In this work, we rely on the ground-truth data of detection and tracking as well as an optimization algorithm (e.g. exhaustive search, Genetic algorithm, Adaboost algorithm) to compute these values. The second issue is to define an optimization task to find the best tracking parameters for a given video context.
- **Context classification :** Similar contexts can obtain different tracking parameter values because the optimization algorithm does not find the best solution or the context is not accurate enough to play the role for determining the best parameter values. Therefore we need to classify the contexts and compute the best parameters for context clusters. For each video sequence, we also need to determine the learned context cluster to which the current video context belongs. The best parameter values associated to this cluster are used for tuning the tracking parameters. Therefore the third issue is to define a classification task for video contexts.
- **Evaluate online the tracking quality :** In the tracking process, when the tracking quality is not sufficient, the tracking parameters need to be tuned to improve the tracking quality. To this end, the fourth issue is to define an online tracking evaluation algorithm.

1.2.2 Hypotheses

The control method presented in this manuscript is done with the five following assumptions :

1. The considered tracking algorithms have at least one tunable parameter which influences significantly the tracking quality. The idea of our control algorithm is to tune tracking

parameters to adapt the tracker performance to the scene variations. Therefore this assumption is mandatory. The performance of our method is directly proportional to the influence levels (i.e. significances) of the controlled parameters on the tracking quality. In this work, our study is limited to numeric parameters.

2. There exists a training set of videos representative of the video contexts. In our method, we propose an offline phase to learn the tracking parameters for different video contexts. Greater the number of learned contexts is, more precise the training phase is.
3. There exists a number of contexts which have an impact on the tracking quality. In other words, for a tracking algorithm, there exists a function f mapping a video context c to satisfactory parameter values p (i.e. parameter values for which the tracking quality is greater than a predefined threshold ϵ) :

$$\begin{aligned} \exists f: \quad & C \rightarrow P \\ & c \longmapsto p \end{aligned} \tag{1.1}$$

where C is the set of video contexts and P is the set of the satisfactory tracking parameters. Let g be a function mapping a video v to its context c :

$$\begin{aligned} g: \quad & V \rightarrow C \\ & v \longmapsto c \end{aligned} \tag{1.2}$$

$c_i = g(v_i)$ is the context of v_i. The function f is assumed to satisfy the following property :

$$\forall v_1, v_2 : \text{if } |c_1 - c_2| < \epsilon_1 \tag{1.3}$$

$$=> |Q(v_1, f(c_1)) - Q(v_2, f(c_2))| < \epsilon_2 \tag{1.4}$$

where ϵ_1, ϵ_2 are predefined thresholds ; $Q(v_i, f(c_i))$ represents the tracking quality for video v_i when parameters $f(c_i)$ are used.

4. The video context in the training phase keeps unchanged within an enough large number of frames (e.g greater than 100 frames) so that the proposed controller can detect it and adapt the parameters to it.
5. The input video is produced by a monocular camera. This assumption is given to limit the tracking algorithm approaches which are considered in the control process.

1.3 Context of Study

This work is done in the PULSAR (is STARS now) project-team at INRIA Sophia Antipolis (The French National Institute for Research in Computer Science and Control). PULSAR means

"Perception, Understanding and Learning Systems for Activity Recognition". PULSAR research direction focuses on the real-time semantic interpretation of dynamic scenes observed by sensors. Thus, PULSAR studies long-term spatio-temporal activities performed by human beings, animals or vehicles in the physical world. PULSAR proposes new techniques in the field of cognitive vision and cognitive systems for physical object recognition, activity understanding, activity learning, system design, evaluation and focuses on two main application domains : safety/security and healthcare. PULSAR has accumulated a strong expertise in this area throughout the last decade. The team has participated in many French and European projects in this domain such as CARETAKER (Content Analysis and REtrieval Technologies to Apply Extraction to massive Recording), GERHOME(GERontology at HOME), CoFriend (Cognitive and Flexible learning system operating Robust Interpretation of Extended real scenes by multi-sensors datafusion), VideoID, SIC... Working in PULSAR team, we have opportunity to test the proposed method on several video datasets recorded in real world, and to apply the result of the proposed method for the activity recognition.

1.4 Contributions

Compared to the state of the art, this thesis brings five significant contributions in which the first three ones relate to the control task and the two last ones concern the mobile object tracking task.

1.4.1 Contributions on the Control Task

– **A New Video Classification Method for Learning Offline Tracking Parameters :** In the state of the art, many approaches have been proposed to improve the tracking quality, but most of them either use prior scene information [Chau et al., 2009a] or are only specific for their tracking algorithms [Kuo et al., 2010, Santner et al., 2010]. Therefore these approaches cannot be generic enough for different tracking algorithms or video scenes. In this work, we define the notion of "context" of a video chunk that includes a set of six feature values influencing the tracking quality. These six contextual features are : the density, the occlusion level of mobile objects appearing in this video sequence, their contrast with regard to the surrounding background, their contrast variance, their 2D area and their 2D area variance. Thanks to this context notion, we can divide automatically a video sequence into chunks of uniform feature values. Video chunks are classified by clustering their contextual features to create context clusters. The best satisfactory parameter values are learned for each context cluster. Therefore, the proposed control method can be applied for any scene type and for several tracking algorithms whose quality is influenced

by the defined contextual features.

- **An Online Tracking Evaluation Algorithm (published in ICDP 2009 [Chau** et al.**, 2009**b**]) :** In order to tune and learn online the tracking parameters, an online tracking quality evaluation is necessary. Therefore we propose an algorithm to evaluate online the performance of tracking algorithms. In this work, the trajectory confidence is computed by considering the coherence of object appearances (i.e. 2D size, color), speed, movement direction. We also take into account the temporal length and exit zone of a trajectory for determining its confidence. This algorithm gives as output a score estimating online the tracking performance over time. The advantages of the approach over the existing state of the art approaches are : (1) little prior knowledge is required (only exit zones in the camera view are required), (2) the method can be applied in complex scenes containing several mobile objects.
- **A New Approach to Tune and Learn Online the Tracking Parameters (published in ICIP 2012 [Bak** et al.**, 2012]) :** While parameter tuning for static image applications or for object detection in videos has been largely studied in the state of the art [Martin and Thonnat, 2008, Bhanu and Das, 1995, Nghiem, 2009], online parameter tuning for the tracking task is not really addressed. In this thesis, we present a new method to tune and learn online the tracking parameters thanks to a learned database resulting from a supervised or an unsupervised learning phase. This parameter tuning helps to adapt the tracking algorithms to the video context variation.

1.4.2 Contributions on Tracking Algorithms

In this work, we present three tracking algorithms to experiment the proposed control method in which two trackers are original.

- **A Tunable Object Descriptor-based Tracking Algorithm Enabling Adaptation to Scene Conditions (published in ICDP 2011 [Chau** et al.**, 2011**a**] and in ICVS 2011 workshop [Zaidenberg** et al.**, 2011])** . Object appearance is widely used in the state of the art for object tracking [Zhou et al., 2006, Kwolek, 2009, Bak et al., 2009, Monari et al., 2009] but most of them cannot perform robustly in different contexts. We propose in this manuscript a tracking algorithm which is effective in different scene contexts. First an object descriptor pool is used to compute the matching score between two detected objects. This pool includes 2D, 3D positions, 2D sizes, color histogram, histogram of oriented gradient (HOG), color covariance and dominant color descriptors. In the tracking process, a temporal window is defined to establish the matching links between the detected objects. The temporal window enables to find the object trajectories even if the objects are misdetected in some frames. A trajectory filter is defined to remove trajectories considered as noise. The object descriptor weights have a strong influence on the tracker quality. We propose

to use a Adaboost algorithm in which each descriptor plays the role as a weak classifier to learn offline their values for each tracking context. The descriptor weights can be tuned online to cope with the contextual variations. This tracking algorithm brings some contributions over the state of the art trackers : (1) a robust tracking algorithm based on an object descriptor pool, (2) a new method to quantify the reliability of HOG descriptor, (4) a combination of color covariance and dominant color descriptors with a spatial pyramid kernel to manage the case of object occlusion. .

– **A Tracking Algorithm based on Kalman Filter and Global Tracking (published in VISAPP 2011 [Chau** et al.**, 2011**b**])**. In this work, we propose an algorithm to track mobile objects based on their trajectory properties. The proposed tracker includes two stages : tracking and global tracking. The tracking stage follows the steps of a Kalman filter including estimation, measurement and correction. First for each tracked object, its state including position and 2D bounding box is estimated by a Kalman filter. Second, in the measurement step, this tracked object searches for the best matching object based on four descriptors : 2D position, 2D area, 2D shape ratio and color histogram. Third, the best matching object and its estimated state are combined to update the position and 2D bounding box sizes of the tracked object. However, the mobile object trajectories are usually fragmented because of occlusions and misdetections. Therefore, the global tracking stage aims at fusing the fragmented trajectories belonging to the same mobile object and removing the noisy trajectories. The advantages of our approach over the existing state of the art ones are : (1) no prior knowledge information is required (e.g. no calibration and no contextual models are needed), (2) the tracker can be effective in different scene conditions : single/several mobile objects, weak/strong illumination, indoor/outdoor scenes, (3) a global tracking stage is defined to improve the object tracking performance.

We have tested the proposed control and tracking algorithms on three public datasets : Caviar[1] , ETISEO[2], TRECVid [Smeaton et al., 2006], and on some long and complex videos belonging to the following projects : Gerhome[3], the Caretaker[4] and Vanaheim[5] European projects.

1.5 Manuscript Organization

This manuscript is organized as follows.

Chapter 2 presents first a state of the art on the mobile object tracking. Tracking algorithms are classified according to their approaches. Second, a state of the art on the control methods is

[1]http ://homepages.inf.ed.ac.uk/rbf/CAVIAR/
[2]http ://www-sop.inria.fr/orion/ETISEO/
[3]http ://gerhome.cstb.fr/en/home/introduction.html
[4]http ://sceptre.king.ac.uk/caretaker
[5]http ://www.vanaheim-project.eu/

described. These methods are divided into three different approach categories : offline-based, online-based and hybrid approaches combining the two previous approaches.

Chapter 3 presents a general description of the proposed control method including two phases : offline learning and online control. We describe the execution process step by step in both phases.

Chapter 4 details a scheme to learn the best satisfactory tracking parameters for each video context. First, we present the definition of video context (or called tracking context) in this work. The video contexts are classified by clustering their features. An optimization process is performed to determine the best satisfactory parameter values for each context cluster.

Chapter 5 describes the proposed online control process. The objective of this process is to to tune and learn online the tracking parameters to adapt the tracking task to context variations using a learned database. It includes two stages : initial parameter configuration and parameter adaptation. For the parameter adaptation stage, we will present two approaches called context-based parameter adaptation and evaluation-based parameter adaptation.

Chapter 6 presents three tracking algorithms used for testing the proposed controller. The first tracker relies on a pool of eight object descriptors. In this tracker, a trajectory filter is defined to remove trajectories considered as noise. The second tracking algorithm uses a Kalman filter associated with a set of four object descriptors. A global tracker is proposed for this tracker to track mobile objects in the case of misdetections and also to remove the noisy trajectories. The last tracker relies on the tracking of Kanade-Lucas-Tomasi (KLT) features located on mobile objects.

Chapter 7 is dedicated to the experimentation and validation of the proposed methods. We present the results of the tracking task in both cases without and with the control process. A comparison with several trackers from the state of the art is also described to highlight the proposed approach performance.

Chapter 8 presents the concluding remarks and limitations of the thesis contributions. We also discuss about the short-term and long-term perspectives of this study.

2

STATE OF THE ART

This chapter presents first a state of the art in object tracking. Object tracking algorithms are classified into three categories : point tracking, appearance tracking and silhouette tracking. Second, we describe a state of the art in control methods whose goal is to adapt vision tasks to contextual variations of images or video sequences. These methods are divided into three different categories : offline-based, online-based and hybrid (i.e. combining the two previous ones) approaches.

2.1 Tracking Algorithms

The aim of an object tracking algorithm is to generate the trajectories of objects over time by locating their positions in every frame of video. An object tracker may also provide the complete region in the image that is occupied by the object at every time instant.

The tracking algorithms can be classified by different criteria. In [Motamed, 2006], based on the techniques used for tracking, the author divides the trackers into two categories : the model-based and feature-based approaches. While a model-based approach needs the model for each tracked object (e.g. color model or contour model), the second approach uses visual features such as Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005], Haar [Viola and Jones, 2003] features to track the detected objects. In [Almanza-Ojeda, 2011], the tracking algorithms are classified into three approaches : appearance model-based, geometry model-based and probability-based approaches. The authors in [Aggarwal and Cai, 1997] divide the people tracking algorithms into two approaches : using human body parts and without using human body parts.

In this manuscript, we re-use the object tracking classification proposed by [Yilmaz et al., 2006]

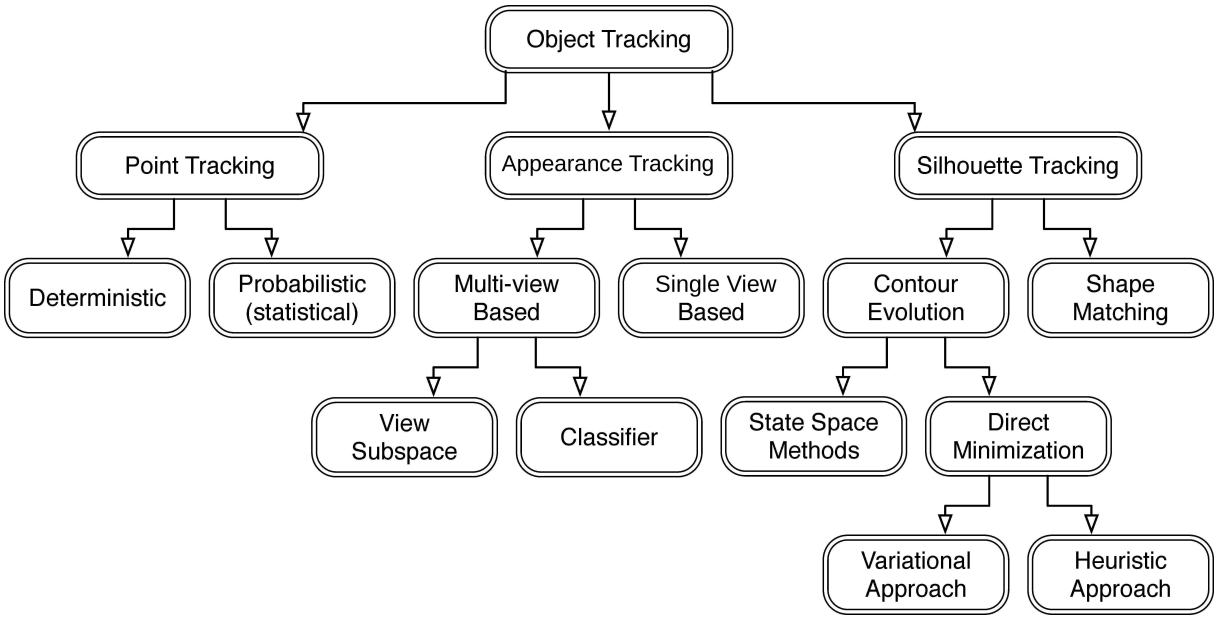


Figure 2.1 – Taxonomy of tracking methods (adapted from [Yilmaz et al., 2006]).

because this classification represents clearly and quite completely the tracking methods existing in the state of the art. This taxonomy method relies on the "tracked targets". The tracked targets can be points of interest, appearance or silhouette of mobile object. Corresponding to these target types, three approach categories for object tracking are determined : point tracking, appearance tracking and silhouette tracking. Figure 2.1 presents the taxonomy of tracking methods proposed by this paper.

- **Point tracking** : The detected objects are represented by points, and the tracking of these points is based on the previous object states which can include object positions and motion. An example of object correspondence is shown in figure 2.2(a).
- **Appearance tracking** (called "kernel tracking" in [Yilmaz et al., 2006]) : The object appearance can be for example a rectangular template or an elliptical shape with an associated RGB color histogram. Objects are tracked by considering the coherence of their appearances in consecutive frames (see example in figure 2.2(b)). This motion is usually in the form of a parametric transformation such as a translation, a rotation or an affinity.
- **Silhouette tracking** : The tracking is performed by estimating the object region in each frame. Silhouette tracking methods use the information encoded inside the object region. This information can be in the form of appearance density and shape models which are usually in the form of edge maps. Given the object models, silhouettes are tracked by either shape matching or contour evolution (see figures 2.2(c), (d)).

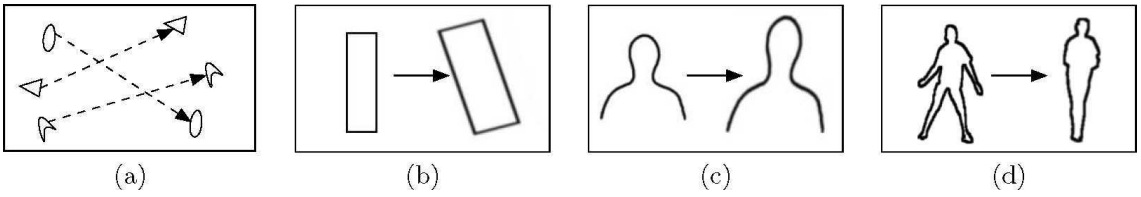


Figure 2.2 – Illustration of different tracking approaches. (a) Multipoint correspondence, (b) Parametric transformation of a rectangular patch, (c, d) Two examples of silhouette matching. (Source [Yilmaz et al., 2006]).

2.1.1 Point Tracking

2.1.1.1 Deterministic Approaches

According to [Wikipedia, 2011b], a deterministic system is a system in which no randomness is involved in the development of the future states of the system. A deterministic model thus always produces the same output from a given starting condition or initial state. In order to apply this idea for object tracking, the object movements are generally assumed to follow some trajectory prototypes. These prototypes can be learned offline, online, or constructed based on a scene model. We can find in the state of the art many tracking algorithms based on this idea [Scovanner and Tappen, 2009, Bilinski et al., 2009, Baiget et al., 2009].

In [Scovanner and Tappen, 2009], the authors present a method to learn offline some tracking parameters using ground-truth data. In the offline phase, the authors define an energy function to compute the correctness of the people trajectories. This function is denoted $E(x_t)$ where x_t is a 2D vector containing the pedestrian's location at time t.

The authors assume that a pedestrian path is constrained by the four following rules. Each rule is represented by an energy function.

1. The displacement distance of people between two consecutive frames is not too large. The energy function expressing this rule is denoted $E_1(x_t)$.
2. The speed and direction of people movement should be constant. The energy function corresponding to this rule is denoted $E_2(x_t)$.
3. People movements should reach their destinations. The energy function representing this rule is denoted $E_3(x_t)$.
4. People movements intend to avoid people in the scene. The energy function of this rule is denoted $E_4(x_t)$.

The complete energy $E(x)$ is a weighted combination of these components :

$$E(x_t) = \sum_{i=1}^{4} \theta_i E_i(x_t) \tag{2.1}$$

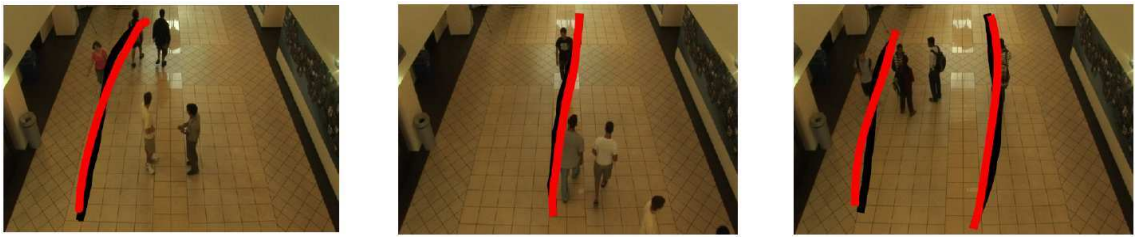


Figure 2.3 – Examples of pedestrian paths, shown in black, and predicted paths, shown in red. The model accurately predicts the deflection of pedestrians due to oncoming obstacles (source [Scovanner and Tappen, 2009]).

where θ_i represents the weight of the energy function i. This complete energy function is used to predict the pedestrian locations in the next frame. The pedestrians should move to the locations that minimize this energy. The objective of the training phase is to learn the values θ_i that make the predicted pedestrian tracks match corresponding tracks in the ground-truth data. To accomplish this, the authors define a loss function $L(x^*, g)$ that measures the difference between a predicted track x^* and the ground-truth track g as follows :

$$L(x^*, g) = \sum_{i=1}^{N_s} \sqrt{||x_t - g_t||^2 + \epsilon} \tag{2.2}$$

where x_t and g_t are locations of predicted track and ground-truth track at time t, and N_s is the number of positions of the considered track. The learned values θ_i are used later in the testing phase to predict pedestrian trajectories. Figure 2.3 shows some examples of the predicted paths (in red color) and their corresponding reference paths (in black color).

The advantage of this approach is that its performance does not depend on quality of the object detection process. However the used rules can be incorrect for complex people movements. The pedestrian destination can be changed. Obstacles are often not neither stable throughout the time. Pedestrian velocity is only correct if he/she is always detected correctly. Experimentation is only done with simple sequences.

In [Bilinski et al., 2009], the authors present a tracking algorithm based on a HOG descriptor [Dalal and Triggs, 2005]. First, the FAST algorithm [Rosten and Drummond, 2006] is used to detect the points of interest. Each point is associated with a HOG descriptor (including gradient magnitude and gradient orientation). The authors compute the similarity of the HOG points located in the consecutive frames to determine the couples of matched points. The object movements can be determined using the trajectories of their points of interest (see figure 2.4). In the case of occlusion, the authors compare the direction, speed and displacement distance of the point trajectories of occluded objects with those of objects in previous frames to split the bounding box of occluded objects (see figure 2.5). This approach can be well performed in

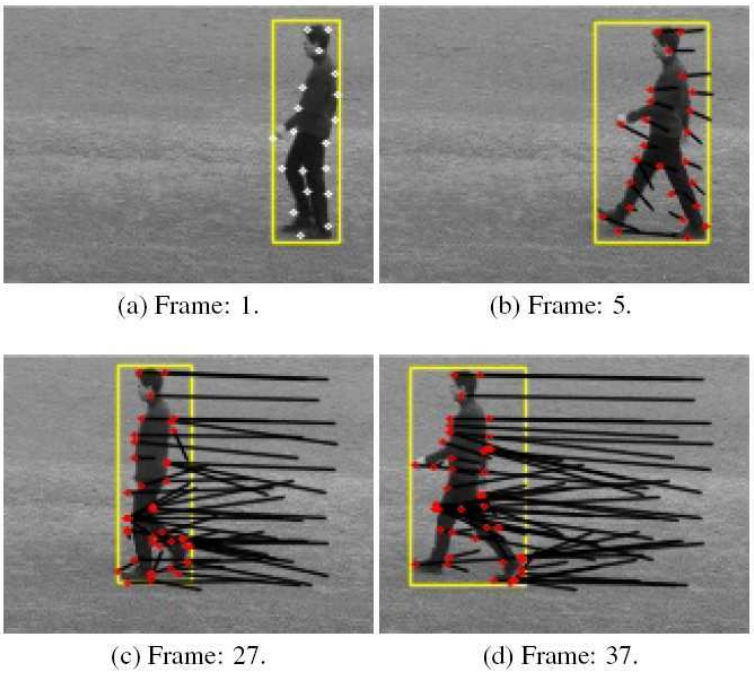


Figure 2.4 – Illustration of a point tracker for KTH dataset (source [Bilinski et al., 2009]).

the case of occlusions in which object appearance is not fully visible. However, the HOG descriptor reliability decreases significantly if the contrast between the considered object and its background is low.

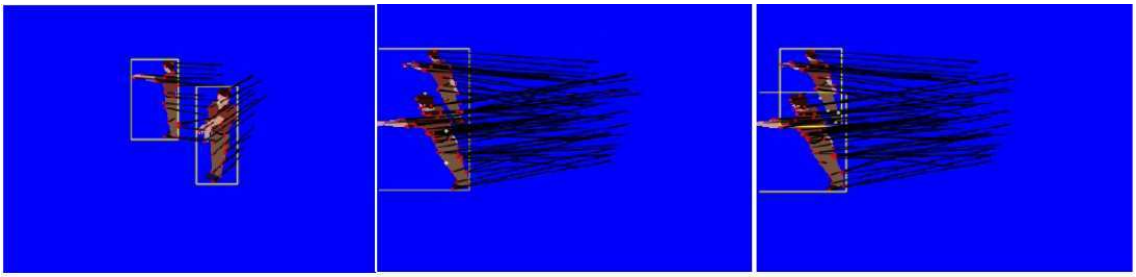
2.1.1.2 Probabilistic Approaches

Probabilistic approaches represent a set of object tracking methods which rely on the probability of object movements. In this approach, the tracked objects are represented as one or many points. One of the most popular methods of this approach is Kalman filter-based tracking. A Kalman filter is essentially a set of recursive equations that can help to model and estimate the movement of a linear dynamic system. Kalman filtering is composed of two steps : prediction and correction. The prediction step uses the state model to predict the new state of variables :

$$X_t^- = DX_{t-1}^+ + W \tag{2.3}$$

$$P_t^- = DP_{t-1}^+ D^T + Q^t \tag{2.4}$$

where X_t^- and X_{t-1}^+ are respectively the predicted and corrected states at time t and $t-1$; P_t^- and P_{t-1}^+ are respectively the predicted and corrected covariances at time t and $t-1$. D is the state transition matrix which defines the relation between the state variables at time t and $t-1$,



a. Frame 12 b. Frame 40 c. Frame 41

Figure 2.5 – Illustration of split a merged-object bounding box for a synthetic video sequence : a. Before merging b. Merging c. Split (source [Bilinski et al., 2009]).

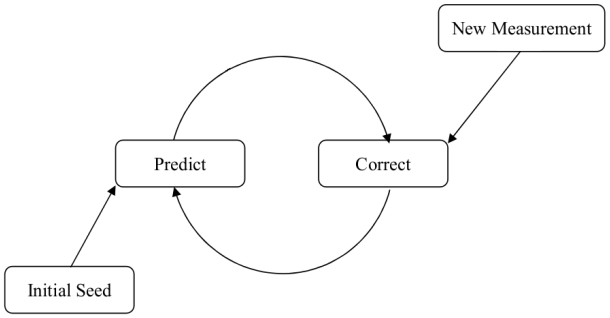


Figure 2.6 – Illustration of Kalman filter steps (source [Johnson, 1998]).

W is a noise matrix, Q is the covariance of the noise W. Similarly, the correction step uses the current observations Z_t to update the object's state :

$$K_t = P_t^- M^T [M P_t^- M^T + R_t]^{-1} \tag{2.5}$$

$$X_t^+ = X_t^- + K_t[Z_t - M X_t^-] \tag{2.6}$$

$$P_t = P_t^- - K_t M P_t^- \tag{2.7}$$

where M is the measurement prediction matrix, K is the Kalman gain and R is the covariance matrix of noise in measurement. An illustration of the Kalman filter steps can be found in figure 2.6. The Kalman filter is widely used in the vision community for tracking [Beymer and Konolige, 1999, Broida and Chellappa, 1986, Brookner, 1998].

In [Robert, 2009], the authors present a tracking algorithm for vehicles during the night time (see figure 2.7). In this work, vehicles are detected and tracked based on their headlight pairs. Assuming that the routes are linear, a Kalman filter is used to predict the movement of the headlights. When a vehicle turns, its Kalman filter is re-initialized.

Figure 2.7 – Illustration of a night time traffic surveillance system (source [Robert, 2009]).

Because the Kalman filter assumes that the variation of the considered variables draws from a Gaussian distribution, these approaches can be only applied for tracking objects with linear movements, or with movements of simple variations of direction, speed. In order to overcome these limitations, an Extended Kalman filter [Bar-Shalom and Fortmann, 1988] or particle filter [Almeida et al., 2005] can be used.

2.1.2 Appearance Tracking

Appearance tracking is performed by computing the motion of the object, which is represented by a primitive object region, from one frame to the next. The tracking methods belonging to this type of approaches are divided into two sub-categories : single view-based (called template-based in [Yilmaz et al., 2006]) and multi view-based.

2.1.2.1 Single View-based Approaches

This approach category is widely studied in the state of the art for tracking mobile objects in a single camera view. Many methods have been proposed to describe the object appearance. In [Snidaro et al., 2008], the authors present a people detection and a tracking algorithm using Haar [Viola and Jones, 2003] and Local Binary Pattern (LBP) [Ojala et al., 2002] features combined with an online boosting (see figure 2.8). The main idea is to use these features to describe the shape, the appearance and the texture of objects. While Haar features encode the generic shape of the object, LBP features capture local and small texture details, thus having more discriminative capability. First, the image is divided into cells and the Haar features are applied in each cell to detect people. Each detected person is divided into a grid of 2×3 blocks. Each block is divided in 9 sub-regions. For each region, the pixel grey values are used to apply the

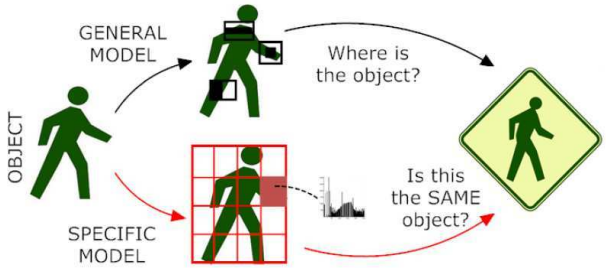


Figure 2.8 – A Haar-like features classifier is employed as a generic detector, while an online LBP features recognizer is instantiated for each detected object in order to learn its specific texture (source [Snidaro et al., 2008]).

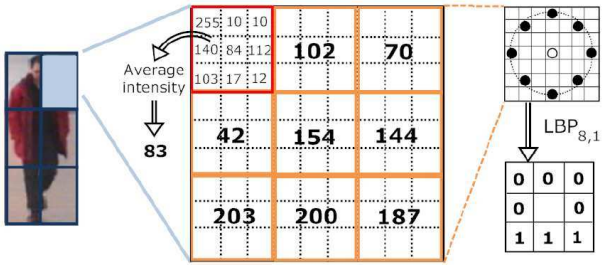


Figure 2.9 – Illustration of the LBP features computation (source [Snidaro et al., 2008]).

8-neighbours LBP calculus scheme (see figure 2.9). The LBP features are then used to track people.

Both classifiers (Haar and LBP) are combined with an online boosting [Grabner and Bischof, 2006]. The application of these two features in each cell (for the Haar features) or in each region (for the LBP features) is considered as the weak classifiers. These weak classifiers cluster samples by assuming a Gaussian distribution of the considered features. This online boosting scheme can help the system to adapt to specific problems which can take place during the online process (e.g. change of lighting conditions, occlusion). However, the online training is time consuming. Also, the authors do not explain clearly enough how to determine positive or negative samples in this training. It seems that the system has to learn in a sequence in which there is only one person before handling complex detection and tracking cases. The tested sequences are still simple (e.g few people in the scene, simple occlusion).

In [Zhou et al., 2006], the authors present a method to detect occlusion and track people movements using the Bayesian decision theory. Mobile object appearance is characterized by color intensity and color histogram. For each object pair detected in two consecutive frames, if the similarity score is higher than a threshold, these two objects are considered as matched and their templates are updated. If the matching score is lower than this threshold, the authors assume that an occlusion occurs. A mobile object is divided into sub-parts and the similarity

scores are computed for these parts. If the matching score of one object part is high enough while the other ones are low, an occlusion is detected. The mobile object can be still tracked but its template is not updated. This paper proposes a method to detect and track objects in occlusion cases. The authors define a mechanism to distinguish between an object appearance change due to occlusion or a real change (e.g due to the change of scene illumination or object distance to camera location). However the features used for characterizing the object appearance (i.e. intensity and color histogram) are not reliable enough in the case of poor lighting condition or weak contrast. The tested video sequences are not complex enough to prove the effectiveness of this approach.

2.1.2.2 Multi-view Approach

The methods belonging to this type of approaches are applied for tracking objects in a multi-video camera system. Because cameras can be different in rendering of colors or illumination, a color normalization step is usually necessary to make comparable object colors from different cameras (e.g. use grey intensity or compute mean and standard deviation values of color distributions).

In [Monari et al., 2009], the authors present an appearance model to describe people in a multi-camera system. For each detected object, its color space is reduced using a mean-shift-based approach proposed in [Comaniciu and Meer, 1997]. Therefore, the color texture of the observed object is reduced to a small number of homogeneous colored body segments. For each color segment, the area (in pixels) and the centroid are calculated and segments smaller than 5% of the body area are removed. Figure 2.10 illustrates the steps of this object appearance computation. Finally, for approximative spatial description of the detected person, the person is subdivided in three sections as follows : starting from the bottom, the first 55% as lower body, the next 30% as upper body, and the remaining 15% as head. The appearance descriptor is now composed, by assigning the color segments to the corresponding body part by its centroids. Identical colors, which belong to the same body part, are merged to one. In doing so, the spatial relationships within a body part are lost but at the same time this approach leads to an invariant representation of the object in different camera views.

Let $C = (c_1, c_2, ..., c_n)$ be the collection of all n color segments, with $c_d = [L_d, u_d, v_d, w_d, bp_d]^T$, where

- $d = 1..n$.
- L, u are the chromatic values and v is the luminance value of the homogeneous segment in CIE $L \times u \times v$ color space [Wikipedia, 2011a].
- $w \in \{0..1\}$ (weight) is the area fraction of the color segment relative to the object area.
- $bp = \{head, upperbody, lowerbody\}$ is the body part index which the centroid of the segment belongs to.

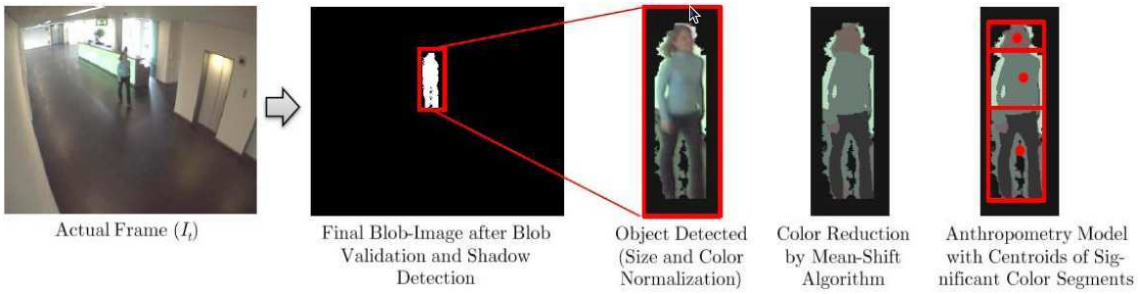


Figure 2.10 – Extraction of significant colors and mapping to dedicated body parts using a simple anthropometric model (source [Monari et al., 2009]).

The appearance feature set is defined by $F^{app} \subseteq C$, with F^{app} is the subset of the color segments, with a body part related fraction (w_d) higher than a minimum weight (e.g. 10%). For the similarity calculation of two appearance feature sets F^1_{app} and F^2_{app}, the Earth Mover's Distance (EMD) [Rubner et al., 1998] is used.

In [Bak et al., 2010a], the authors define a signature for identifying people over a multi-camera system. This method studies the Haar and dominant color features. For each single camera, the authors adapt the HOG-based technique used in [Corvee and Bremond, 2009] to detect and track people. The detection algorithm extracts the histograms of gradient orientation, using a Sobel convolution kernel, in a multi-resolution framework to detect human shapes at different scales. With Haar features, the authors use Adaboost [Freund and Schapire, 1997] to select the most discriminative feature set for each individual. This feature set forms a strong classifier. The main idea of dominant color feature is to select the most significant colors to characterize the person signature. The human body is separated into two parts : the upper body part and the lower body part. The separation is obtained by maximizing the distance between the sets of dominant colors of the upper and the lower body (see figure 2.11). The combination of the dominant color descriptors of upper and lower body is considered as a meaningful feature to discriminate people. An Adaboost scheme is applied to find out the most discriminative appearance model.

In [Colombo et al., 2008], the authors compare and evaluate three appearance descriptors which are used for estimating the appropriate transform between each camera's color spaces. These three appearance descriptors are : (1) mean color, (2) covariance matrix of the features : color, 2D position, oriented gradients for each channel and (3) MPEG-7 Dominant Color descriptor. In order to compare the color descriptors from two cameras, two techniques are presented to normalize color space and to improve color constancy. The first one (First-Order Normalization) consists in computing the mean value for each color component (YcbCr) over a training set of tracked objects in both cameras and to compute the linear transformation between both mean values. In the second one (Second Order Normalization), the authors consider

(a) original image (b) upper body part (c) lower body part

Figure 2.11 – The dominant color separation technique : a) original image ; b) upper body dominant color mask ; c) lower body dominant color mask (source [Bak et al., 2010a]).

the possibilities of rotation and translation of pixel color values. The term “rotation” means the difference of luminance and chromatic channels between two cameras. If there is no mixing between the luminance and the two chromatic channels, the rotation is not considered. The authors have tested these techniques for tracking the movement of a pedestrian over a camera network in subway stations. The result shows that the application of the color normalization techniques does not improve significantly the performance of covariance and dominant color descriptors. Also, the mean color descriptor brings the best result (compared to the two other techniques) when it is combined with the second normalization color technique. The paper gets some preliminary results on evaluation of different descriptors but the authors should extend their work on the case of multi-object tracking.

2.1.3 Silhouette Tracking

Objects may have complex shapes, for example, hands, head, and shoulders that cannot be well described by simple geometric shapes. Silhouette-based methods provide a more accurate shape description for these objects. The object model can be in the form of a color histogram or the object contour. According to [Yilmaz et al., 2006], silhouette trackers are divided into two categories, namely, shape matching and contour tracking. While the shape matching methods search for the object silhouette in the current frame, the contour tracking evolves from an initial contour to its new position in the current frame by either using the state space models or direct minimization of some energy functions.

2.1.3.1 Contour Tracking

In [Xu and Ahuja, 2002], the authors present an object contour tracking approach using graph cuts based active contours (GCBAC). Given an initial boundary near the object in the first frame, GCBAC can iteratively converge to an optimal object boundary. In each frame thereafter, the resulting contour in the previous frame is taken as initialization and the algorithm consists in two steps. In the first step, GCBAC is applied to the image area which is computed by the difference between a frame and its previous one to produce a candidate contour. This candidate contour is taken as initialization of the second step, which applies GCBAC to current frame directly. If the amount of difference within a neighbour area of the initial contour is less than a predefined threshold, the authors consider that the object is not moving and the initial contour is sent directly to the second step. So the initialization of the second step will be either the contour at the previous frame, or the resulting contour of the first step. Figure 2.12 presents this object contour tracking algorithm sketch. By using the information gathered from the image difference, this approach can remove the background pixels from object contour. However, this approach only works effectively if the object does not move too fast and/or the object does not change a lot in consecutive frames. It means that this approach cannot handle the cases of object occlusion. Figure 2.13 presents a head tracking result when the head is rotating and translating.

The authors in [Torkan and Behrad, 2010] present a contour tracking algorithm based on an extended greedy snake technique combined with a Kalman filter. The contour of a mobile object includes a set of control points (called snaxels). Firstly the system computes the centroid of an object contour by calculating the average value of the coordinates of its control points. A contour is represented by its centroid and the vectors corresponding to the coordinates of the control points relatively to the centroid. The tracking algorithm then uses a Kalman filter to estimate the new centroid position in the next frame. The new control points are calculated based on this new centroid, the vectors determined in the last frame, the shape scale and the scaling factor. A new initial contour is also constructed thanks to its new control points. After that, the greedy snake technique is applied to reconstruct the contour of the mobile object. For each point of the 8 neighbour points of a snaxel, the algorithm computes a snake energy value and the control point is updated with the neighbour point which has minimum energy. The contour is so updated according to the new control points. The snake energy includes an internal and an external energy. In the internal energy there are continuity energy and curvature energy. While the internal energy determines the shape of the contour, the external energy prevents contour from improper shrink or shape change and always holds it close to the target boundary. In this paper, the authors use the Kalman filter to estimate the position of contour centroid in the next frame. The field energy value and the application of the Kalman filter are useful for tracking targets with high speed and large displacement. However, only

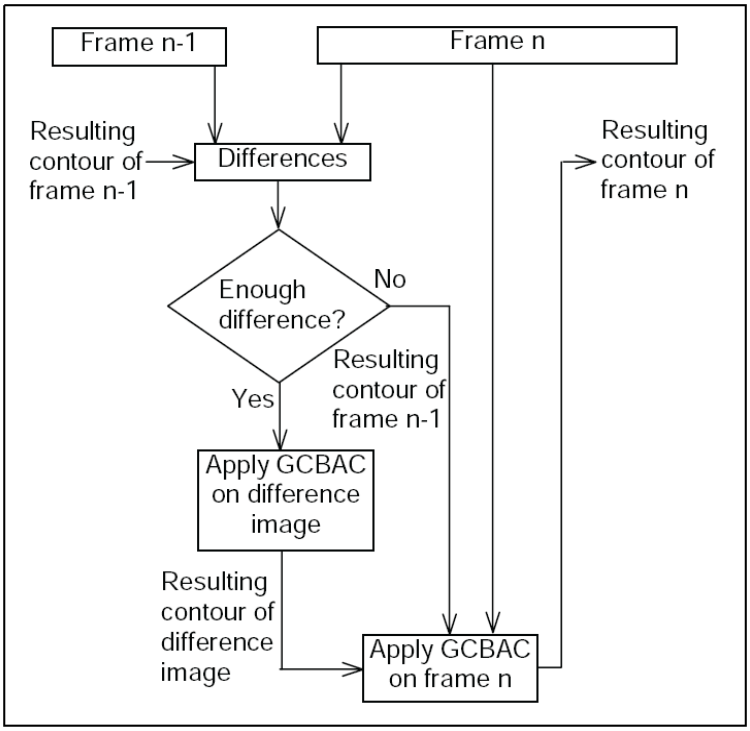


Figure 2.12 – Sketch of the object contour tracking algorithm using GCBAC (source [Xu and Ahuja, 2002])

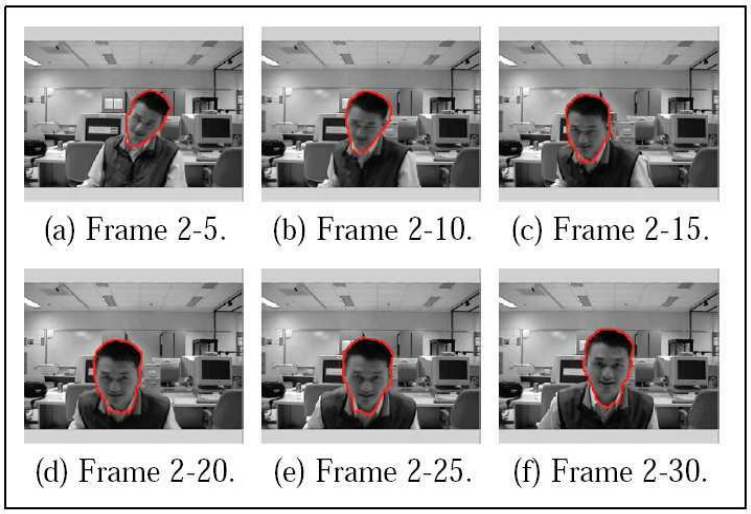


Figure 2.13 – Head tracking result when the head is rotating and translating (source [Xu and Ahuja, 2002])

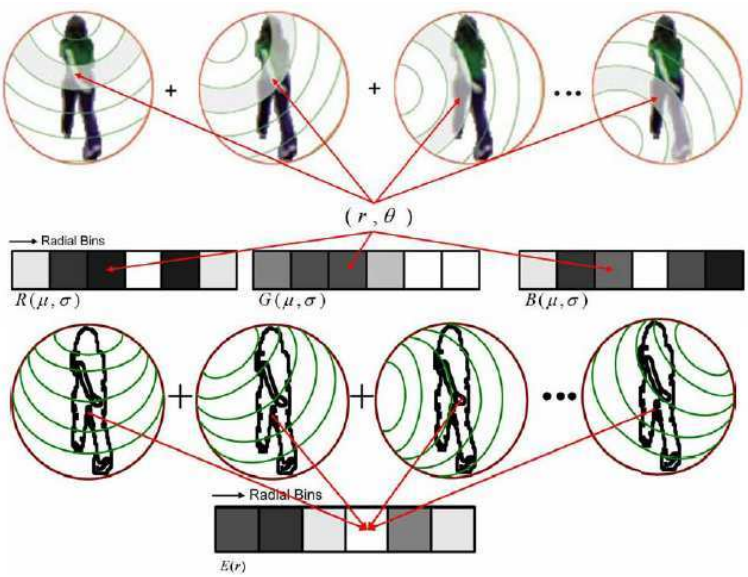


Figure 2.14 – Computation of the color and shape based appearance model of detected moving blobs (source [Kang et al., 2004])

three illustrations are provided. These illustrations are too simplistic. The object to track is black on a white background. A classical color segmentation should be able to detect correctly the unique object in the scene.

2.1.3.2 Shape Matching

In [Kang et al., 2004], the authors present an approach to compute the shape similarity between two detected objects. The object shape is described by a Gaussian distribution of RGB color of moving pixels and edge points. Given a detected moving blob, a reference circle C_R is defined as the smallest circle containing the blob. This circle is uniformly sampled into a set of control points P_i. For each control point P_i, a set of concentric circles of various radii are used to define the bins of the appearance model. Inside each bin, a Gaussian color model is computed for modeling the color properties of the overlapping pixels between a circle and the detected blob. Therefore, for a given control point P_i we have a one-dimensional distribution $\gamma_i(P_i)$. The normalized combination of the distributions obtained from each control point P_i defines the appearance model of the detected blob : $\Lambda = \sum \gamma_i(P_i)$.

An illustration of the definition of the appearance model is shown in figure 2.14 where the authors sample the reference circle with 8 control points. The 2D shape description is obtained by collecting and normalizing corresponding edge points for each bin as follows :

$$E(j) = \frac{\sum_i E_j(P_i)}{max_j(\sum_i E_j(P_i))} \tag{2.8}$$

where $E(j)$ is the edge distribution for the j^{th} radial bin, and $E_j(P_i)$ is the number of edge points for the j^{th} radial bin defined by the i^{th} control point P_i.

The defined model in this approach is invariant for translation. Rotation invariance is also guaranteed, since a rotation of the blob in the 2D image is equivalent to a permutation of the control points. This is achieved by taking a larger number of control points along the reference circle. Finally, the reference circle defined as the smallest circle containing the blob guarantees an invariance to scale. This approach is interesting but the authors only test with simple video sequences in which there are only two moving people.

The classification of the tracking algorithms presented in [Yilmaz et al., 2006] are only relative because there are still many tracking algorithms which combine different approaches such as between point-based and appearance-based [Kuo et al., 2010], between contour-based and shape-based [Yilmaz and Shah, 2004], or between contour-based and deterministic-based [Erdem et al., 2003].

2.2 Control Algorithms

Although many tracking approaches are proposed in the state of the art, object tracking quality always depends on the properties of the considered scenes such as illumination, camera position, scene path complexity or the properties of mobile objects such as number, density, occlusion frequence and size of objects. A tracker can produce good performance for a scene at an time instant, but fails in other cases (i.e other scenes, or the same scene but at another moment).

In order to overcome these limitations, automatic control systems have been studied to adapt the system to context variations. The main idea of the control approaches is to perceive the context information (the notion “context” is dependent on problems, approaches) and to handle the vision algorithms (e.g tuning parameters, activating pertinent algorithms) to adapt themselves to the variations of the considered images or video sequences.

Although there are some studies on control process for tracking task, these studies are quite specific for their tracking algorithms [Kuo et al., 2010, Santner et al., 2010]. Our objective is to address a generic control which can handle different tracking algorithms, therefore many studies presented in the following sections are not related to the object tracking task, but the principle of these control approaches can still be inherited for defining a control method for the object tracking algorithms. We divide the control approaches into three categories : offline-based, online-based and hybrid approaches.

2.2.1 Offline-based Approaches

The main idea of these approaches is to use an offline phase for training or building a knowledge base to adapt the system performance to the contextual variation of the processed image or video sequence. The knowledge mentioned here can be either the optimal parameter values for a given context [Martin and Thonnat, 2008, Thonnat et al., 1999, Yong et al., 2005] or a set of rules for tuning the parameters [Shekhar et al., 1999, Sherrah, 2010]. The offline learning process helps the online adaptive task to decrease significantly the processing time. Also, this approach can benefit the human interaction and knowledge when building a knowledge base.

In [Moisan et al., 1995], the authors present a real time knowledge-based system for program supervision in a vehicle driving assistance system. The objective of the program supervision is to select and adapt perception modules with respect to a goal and to the available resources. In order to reach these objectives, an knowledge base is built offline. This knowledge base contains the request-to-action mapping rules, the context description, the requests, the perception module descriptions and their tasks. The supervision task depends on the current context. In this paper, the following contexts are defined : none, motor way, road, intersection and stop at intersection. Depending on the detected context, the suitable task is performed. For example, in the "motorway" context, the front and rear cameras are switched on, while in the "intersection" context the left and right cameras are switched on. The experimental results show that the system can handle with three situations : driving on a straight road, at intersection, and detection of obstacles. This study gets some satisfactory results. The principles are quite general and applicable to various control approaches. However a limitation of this work is not to control the result.

The authors in [Thonnat et al., 1999] present a framework which is able to integrate expert knowledge and uses it to control the image processing programs. The framework is experimented on three different applications : road obstacle detection, medical imaging and astronomy. By considering both context and evaluation criteria, the system can find the best algorithm among a predefined algorithm set and tune its parameters to obtain the best possible performance. However, the construction of a knowledge base for this system requires a lot of time and data. The paper limits its study to static image processing and not to video processing.

In [Hall, 2006], the author presents an architecture for a self-adaptive perceptual tracking system including three stages : "auto-criticism", "auto-regulation" and "error recovery" (see figure 2.15). The "auto-criticism" stage plays the role of an online evaluation process. This stage returns the goodness score of trajectories. To do that, in the offline phase, the system takes as input typical trajectories and uses K-means clustering [MacQueen, 1967] to learn the scene reference model. In the online phase, the obtained trajectories are compared with the learned clusters using the Gaussian Mixed Model (GMM) to compute their goodness scores.

The goal of the "auto-regulation" stage is to find a parameter setting that increases the

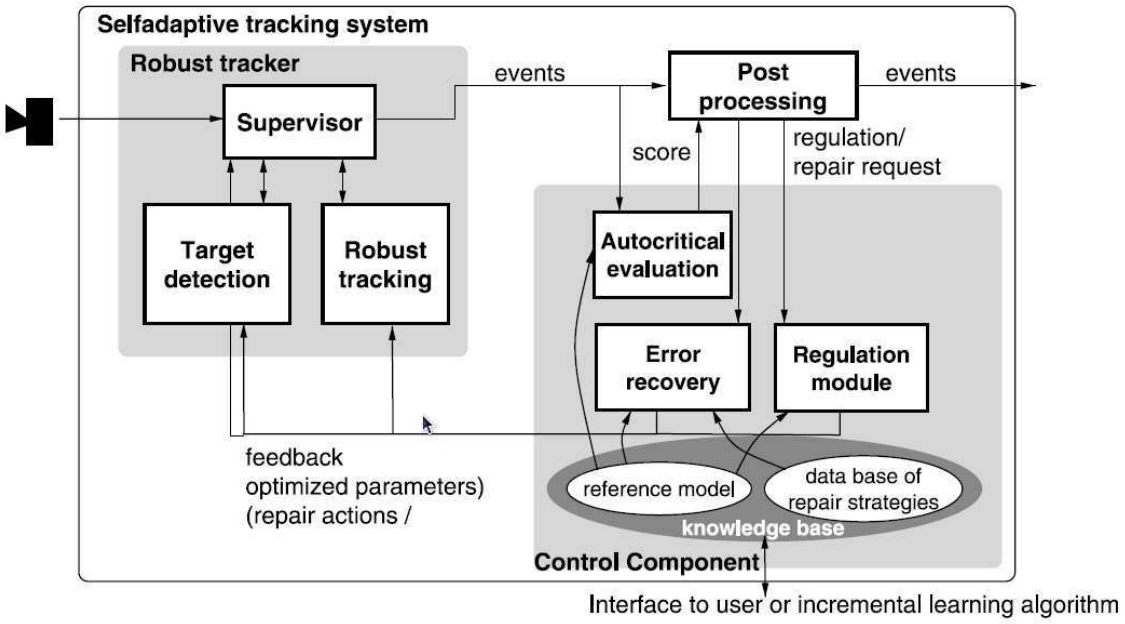


Figure 2.15 – Architecture of a self-adaptive tracking system (source [Hall, 2006]).

system performance. Two strategies for tuning parameters are performed : enumerative strategy and genetic algorithm. With the enumerative strategy, the system scans the parameter values in a predefined parameter space and searches for the best parameter set. In the second strategy, the system uses a genetic algorithm [Goldberg, 1989] to search for the parameter values.

The “error recovery” stage plays the role of repairing trajectory errors. It includes three components : “Error Detector”, “Error Classifier” and “Error Processor”. The “Error Detector” monitors the output of a perceptual component and is responsible for error detection. When the “Error Detector” detects an abnormal trajectory in the system output, it extracts the trajectory properties that serve then as input for the “Error Classifier”. The task of the error classifier is to classify the observed trajectory. If the classifier is unable to classify the trajectory, the example is labeled as “unknown”.

The “Error Processor” takes an error trajectory and its error-class label as input. It then applies the learned repair rule associated with the corresponding class to repair the error (e.g. remove noise, update the background image, merge the neighbour targets...). If the trajectory is classified as “unknown”, no repair rule is executed. Instead, the trajectory is stored in a database for later manual treatment, and is incorporated into the training data. In this way, new examples of error cases are collected automatically, to ensure that there is a sufficient number of available error examples for the design and test of the knowledge base.

Let s_t be the score at instant t representing online the tracking performance provided by the “auto-criticism” stage, S denotes a predefined threshold, the authors define a mechanism

for activating the "auto-regulation" and "error recovery" stages as follows :

- continue processing as normal if $s_t > S$ and $s_{t-\Delta t} > S$
- activate "error recovery" if $s_t > S$ and $s_{t-\Delta t} \leq S$
- activate "auto-regulation" if $s_t \leq S$ and $s_{t-\Delta t} \leq S$

In this paper, the control of the tracking process is able to interact with the detection algorithm to improve the tracking quality (e.g. change background image). However, this approach is too expensive in terms of processing time. The parameter tuning in the online phase is time consuming. Also, the searching methods for optimal parameters have some limits. The enumerative strategy is only appropriated for low dimensional parameter space and small range for each parameter. The genetic algorithm can output only local solutions. This approach requires an offline training process of the typical trajectories to evaluate online the tracker quality. It cannot be applied for the scenes where there are not clearly paths or roads (e.g. park, place).

In [Yong et al., 2005] the authors present a method to select a segmentation algorithm for a given image using a supervised training phase. Images are classified by clustering their grey intensity histograms. Each image cluster is segmented by all candidate algorithms. The system then searches for the best segmentation algorithm for each image cluster thanks to the user evaluation. In the testing phase, the system determines the cluster to which the processing image belongs, and the corresponding learned algorithm is used to perform the segmentation. In this paper, the authors assume that the same segmentation algorithm for images sharing the same features can produce the same performance. While the notion of "image features"is general, this assumption is too strong because it depends on the segmentation approach.

The paper [Sherrah, 2010] presents a method for automatic tuning of parameters for object tracking. Two assumptions are given. First, the algorithm quality expression is a function of the considered parameters and this function has no local extremums. Second, the optimal parameter values change slowly over time so that the presented method can successfully find them. In the training phase, the system studies the relation between parameter variations and the algorithm quality to define the rules for parameter tuning. In the online phase, the system can tune parameter values to improve the performance using these rules. While the first assumption is too strong, the authors do not mention how to compute online the algorithm quality. The experimental results should be compared with ground-truth data to validate the performance of this approach.

In [Shekhar et al., 1999], the authors present a knowledge-based framework which can automatically select algorithms and set parameters until the desired results are obtained. Figure 2.16 presents the functioning of the control engine. The knowledge base in this work is defined as a set of rules supporting the control process involving in different stages : planning, evaluation and repair. We can find below an example of a rule in the planning stage which helps to select an operator. This framework has been tested in two applications. The first one relates

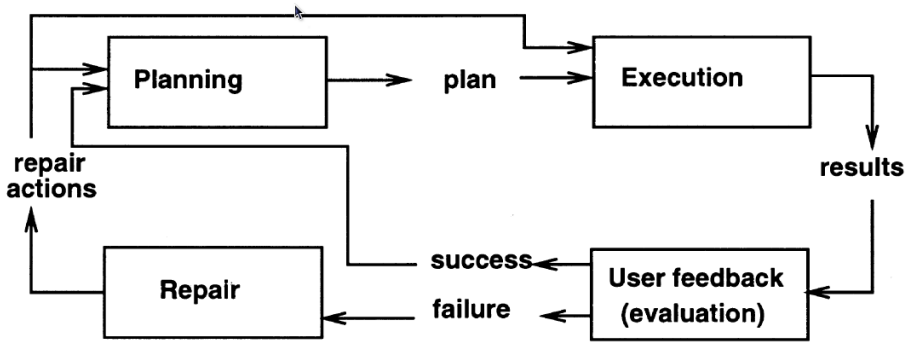


Figure 2.16 – Functioning of the control engine (source [Shekhar et al., 1999]).

to identify objects such as targets, buildings and roads. The second application is on vehicle detection in aerial images.

```
-------------------------------------------------
if
  the data have property x
  AND context field f has value y
  AND the request has a constraint z
then
  choose an operator with characteristic b
  AND do not choose operator w
-------------------------------------------------
```

Depending on the size, the properties of training data and the mechanism of the training scheme, a knowledge base might contain some incorrect or redundant knowledge. In these cases, a filtering process for knowledge base is necessary. In [R. Vincent and Thonnat, 1994], the authors propose a method to refine the knowledge base by combining several learned rules or solving the conflicts between them. A method for computing the similarity of two contexts is also described using premises belonging to each context. However, the priorities of premises are not considered. Also this approach does not remove the premises which might be useless.

In [Caporossi et al., 2004], the authors compare the tracker results with corresponding ground-truth data to exploit the importance of each parameter for each context and to exploit the influence of each parameter variation on tracker performance. The authors assume that parameter variations are independent. This is a strict hypothesis because the parameters are usually dependent on each other.

An open problem is how to find the optimal parameter values for a given context ? The parameter space is in general very large. An exhaustive search cannot be effective. In order to

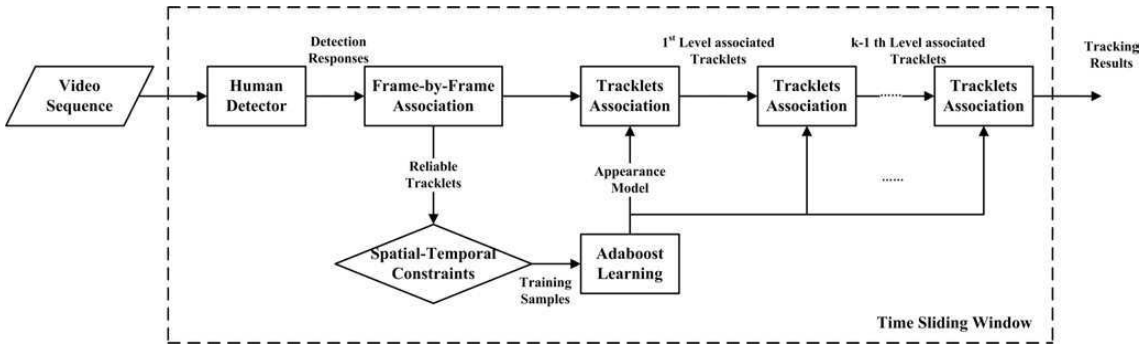


Figure 2.17 – Illustration of a training scheme for fusing object tracklets (source [Kuo et al., 2010]).

solve this problem, in [Bhanu and Das, 1995] the authors present a new method to search for optimal parameters. The main idea is to combine a Genetic algorithm (GA) [Goldberg, 1989] and a hill climbing algorithm (HC) [Russell and Norvig, 2003]. When the GA finds a new optimal point (i.e. parameters), the HC takes this point as a starting point. The HC then scans all "adjacent points". Adjacent points of a given point are defined in two ways. First, it can denote a set of points that are at an Euclidean distance apart from the given point. Thus the adjacent points are located in the neighbourhood of the given point. Second, adjacent points can denote the set of points that are at a unit Hamming distance apart from the given point. If the local maximum point is found, it will replace the initial maximum point and the GA proceed with the updated population.

2.2.2 Online-based Approaches

The offline-based approaches require generally a lot of processing time and human intervention. Also, the application of an offline learned knowledge for an online process phase (i.e. the decision step) can produce some errors leading to a decrease of performance of the processing task. Moreover, these approaches cannot handle the new cases for which the knowledge base has not learned yet. In order to overcome these limits, we present in this section some approaches based completely on an online process (i.e. without offline phase) to adapt their tasks to the context variations [Crowley and Reignier, 2003, Kuo et al., 2010, Santner et al., 2010].

The authors in [Kuo et al., 2010] present a method to associate tracklets using online learned discriminative appearance models (see figure 2.17).

First, the object tracklets are determined based on considering simple object features : position, size and color histogram in two consecutive frames. Second, the authors use an Adaboost algorithm for training automatically the appearance models of tracklet pairs belonging to a same trajectory. The training samples are collected assuming the two hypotheses : (i) one tracklet describes exactly one target, (ii) two tracklets which overlap in time represent different

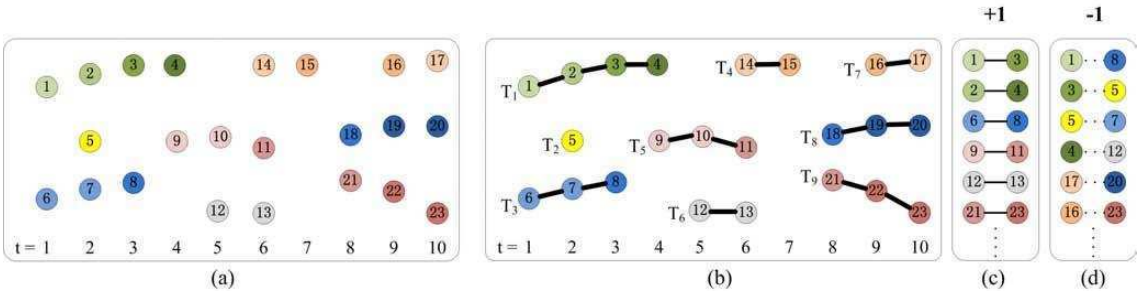


Figure 2.18 – The overview of the process of obtaining online training samples. (a) : The raw detection responses. (b) The result of reliable tracklets. (c) Positive training samples. (d) Negative training samples (source [Kuo et al., 2010]).

targets. Figure 2.18 illustrates the process of collecting the training samples.

The tracklets are described by several local features. These features are constructed based on RGB color histogram, covariance matrix and histogram of oriented gradients (HOG). The RGB color histogram is used to represent the color appearance of a local image patch. The goal of the covariance matrix [Tuzel et al., 2006] is to describe the image texture. The HOG feature [Dalal and Triggs, 2005] is used to capture object shape information. In this paper, a 32D HOG feature is extracted over the region R ; it is formed by concatenating 8 orientation bins in 2×2 cells over R. These features are computed at different object locations and scales to increase the descriptive ability. A feature similarity computation between corresponding sub-regions of two objects is considered as a weak classifier. An Adaboost algorithm is used to learn online the weights (i.e. importance) of the weak classifiers. At the end of the Adaboost algorithm, a strong classifier is defined as a weighted combination of the weak classifiers. This strong classifier is able to compute the similarity between two tracklets to decide to associate tracklets considered as belonging to the same mobile object.

In this paper, for the training task, the two given assumptions are too strong. In the case of occlusions or crowded scenes, it seems that these hypotheses cannot be true because the features used for tracklet construction (i.e. position, size and color histogram of mobile objects) are too simple. The authors should experiment with videos with poor illumination and/or low contrasted videos because these elements influence strongly the color appearance model.

The paper [Crowley and Reignier, 2003] presents a method to build a context aware system for computer vision applications. The system is organized as a multi-stage system. Each stage corresponds to a specific task and process. In order to control the processes, the authors associate each process with the three following capabilities : “auto-regulation”, “auto-description” and “auto-criticism”. An “auto-regulation” process means that this process is able to tune parameters for keeping the optimal quality. The second capability means that the process can provide its services or its command set to a more abstract meta-controller. Finally, an auto-critical

process means that this process can estimate online its quality.

In [Santner et al., 2010], the authors present a tracking framework which is able to control a set of different trackers to get the best performance. The system runs three tracking algorithms in parallel : normalized cross-correlation (NCC), mean-shift optical flow (FLOW) and online random forest (ORF). FLOW is used firstly as a main tracker. If the tracker quality of ORF is better, FLOW is replaced by ORF. When NCC quality is better than the one of ORF, it takes the main role of ORF. The approach is interesting but the authors do not mention how to evaluate online the tracker quality. Also, the execution of three trackers in parallel is very expensive in terms of processing time.

2.2.3 Hybrid Approaches

The notion of "hybrid approach" means a method using an offline process for building a knowledge base or for training data, and also including an online control strategy to adapt themselves to context variations of the video sequences. This type of approach benefits from the advantages of both previous approaches. Typical works of this type of approach can be found in [Georis et al., 2007, Nghiem, 2009].

The method proposed in [Georis et al., 2007] can be considered as a typical method of this approach. In this paper, the authors present a controlled video understanding system based on a knowledge-base. The system is composed of three main components : a library of programs, a knowledge base and a control process (see figure 2.19). Two additional components (the evaluation and the learning tool) enable the expert to evaluate the system and to learn system parameters.

The library of programs is organized following a model of the video understanding process which consists in three main tasks : 1) object detection and classification, 2) spatio-temporal analysis and 3) event recognition.

The second component is a knowledge base which includes three knowledge types. The first type is the end-user goal. This knowledge type is defined to enable a goal-directed control and to ensure that the system will match end-user expectations. The second type is the knowledge of the scene environment. This kind of knowledge helps the system to obtain the necessary information for adapting the system to the variations of the scene. The authors build this knowledge thanks to data provided about the camera, the scene (e.g. 3D geometric model), the physical objects (e.g. object type) and the video sequences (e.g. frame rate). The third knowledge type is the knowledge of the video processing programs. This knowledge defines the operators (i.e. primitive and composite operators) for different contexts. A learning tool helps the experts to complete a prior knowledge. This knowledge base is built using a supervised learning.

The last component is a control component which includes steps for managing all of the online processes of the system (see figure 2.20).

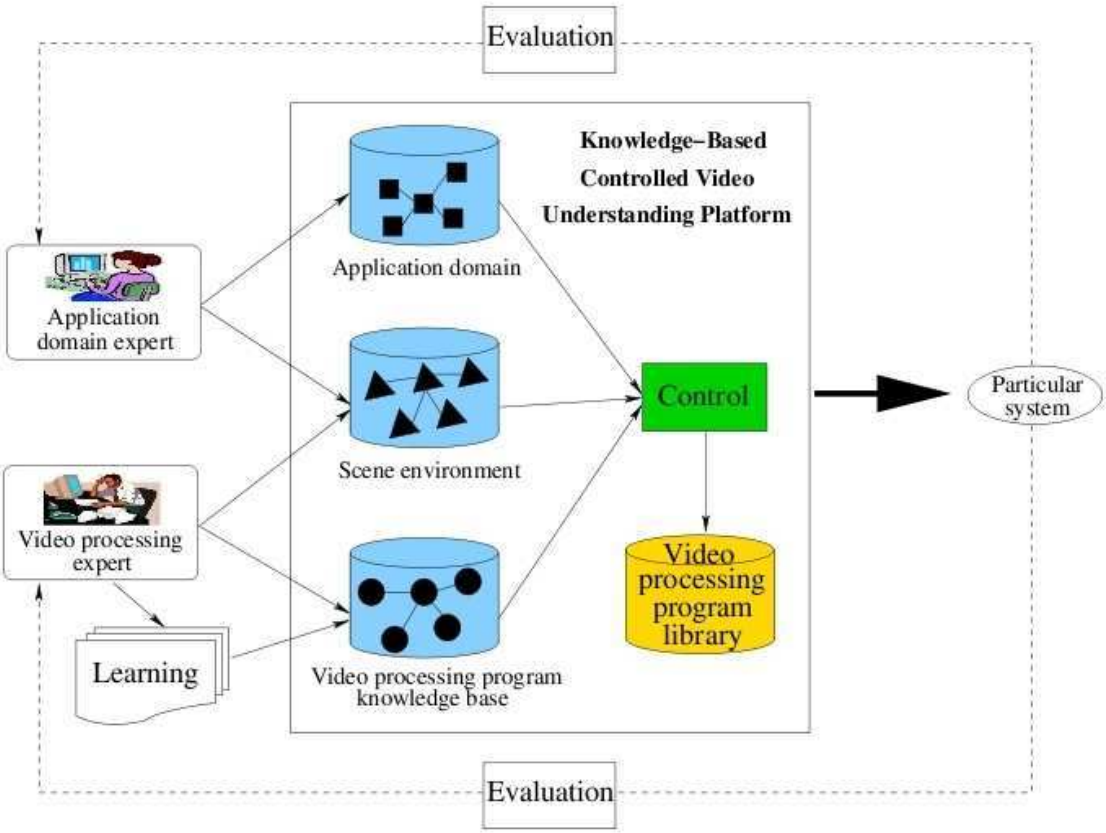


Figure 2.19 – A knowledge-based controlled video understanding platform (source [Georis et al., 2007]).

- Planning : This phase analyses the end-user request in order to select the best operator. Then, it creates a plan to solve the request.
- Program execution : This phase runs the video processing programs that have been ordered by the previous phase after having assigned values to parameters. Results are produced at this phase.
- Result evaluation : This phase evaluates the returned results and produces assessment values. If the evaluation results are good enough, the process can go on.
- Repair step : If the result assessment is negative, this phase decides the appropriate corrective actions to execute. This may lead to either reconsidering the arrangement of programs or changing some parameter values.

For repair step, the authors define five strategies :

- Goal-directed behavior : This strategy helps the system to satisfy the end-user request. For example, the system decides to use the people group tracking algorithm (instead of a single person tracking algorithm) for recognizing a fighting event.
- Data-driven behavior : The input data (e.g. video sequence, scene environment information) can help the system to take decision.
- Closed-loop behavior : this strategy corresponds to a feedback of information from high-level to low-level processes : the new processed results become the available information for the reasoning at the next time step.
- Local or global repair behavior : this strategy first needs the evaluation of results. Only a bad evaluation triggers the execution of repair criteria. In a local repair strategy, a repair criterion decides to re-execute the process and triggers parameter adjustment criteria. In a global repair strategy, a repair criterion can transmit the problem to a father process in the hierarchy.
- Real-time behavior : The system enforces an execution in a specified time so that the reasoning engine can keep the processing time to be faster than the video frame rate.

This approach is interesting, but it does not address specifically the tracking task. Also, the authors do not explain clearly enough the online evaluation algorithm. The notion “context” is not clearly formalized. The tested sequences are not complex enough.

In [Nghiem, 2009], the author presents an adaptive method for detecting mobile objects in videos. The objective of this method is to tune the parameter values of a background subtraction algorithm to adapt its functions to the current video context.

The processing chain includes three phases : execution, evaluation, and repair. While the execution phase is performed by a foreground detection task, the two other phases are done by a controller. In the evaluation phase, the authors define five error indicators (e.g. number of noisy pixels, area of unknown-classification objects) to evaluate the quality of the object detection task. When this quality is low, the repair phase is activated. This phase tries to find the optimal

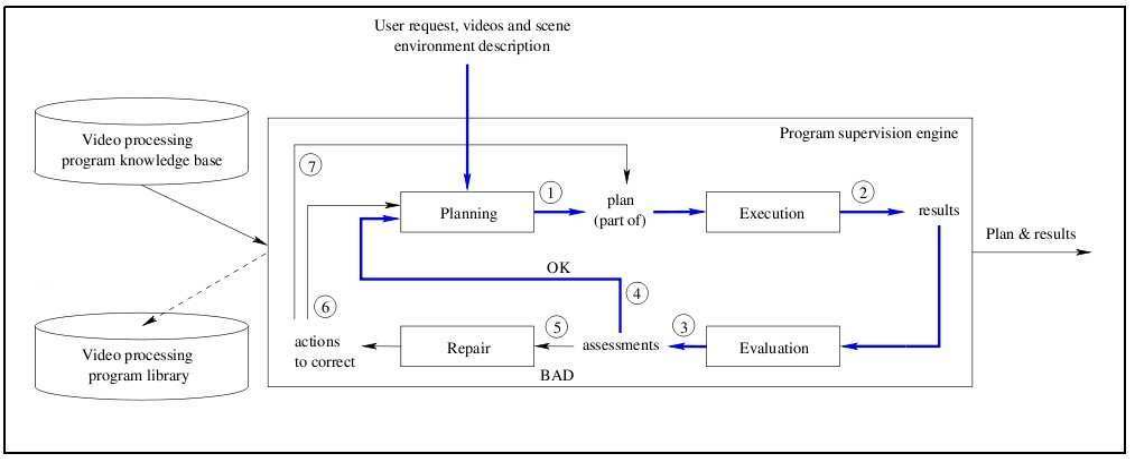


Figure 2.20 – Main phases of the reasoning engine (source [Georis et al., 2007]).

parameter values so that the foreground detection results are consistent with the feedback from the classification and tracking tasks. To do this, the controller has a set of parameter tuners. Each tuner is responsible for reducing the values of one or several error indicators. Depending on their approaches, these parameter tuners are divided into two categories : context-based and evaluation-based. The tuning process works as follows :

1. The controller takes as input the foreground detection results and computes the values of the five error indicators for each region in the image.
2. The controller uses the expert knowledge to verify if the values of the error indicators are good or not.
3. If the value of one error indicator in one region is not good, the controller activates the corresponding parameter tuner.
4. If the parameter tuner can improve the value of this error indicator, the controller requests the foreground detection task to change its parameter values at the affected region.
5. If the parameter tuner fails or if no parameter tuner can reduce the value of this error indicator, the controller informs human operators and stores the current video sequence for further offline analysis.

When a parameter tuner is activated (at step 3), the context-based tuner is called first. This tuner is specific to one particular background subtraction algorithm. It has two phases : online and offline. The set of learned contexts is empty at the beginning and it is constructed incrementally.

In the online phase, the tuner determines the context of the current video and then applies the corresponding optimal parameter values as follows :

1. The tuner verifies if the context of the current video belongs to one of the learned contexts.
2. If the context of the current video belongs to the learned context c_i then the tuner applies the learned optimal parameter values p_i for the detecting objects.
3. If the condition of the current video does not belong to any learned context, the tuner requests the controller to store the video for an offline phase.

The objective of the offline phase is to find the optimal parameter values for unknown contexts given by the online phase. This phase works as follows :

1. For each video stored by the controller in the online phase :
   a. Create ground-truth data for this video.
   b. Find optimal parameter values for this video using the ground-truth data.
2. Compare the optimal parameter values of different learned contexts. If two contexts c_1, c_2 have the same optimal parameter values, merge them to make c_3.

If the context-based tuner cannot solve the error, the "evaluation-based parameter tuner" is then activated. This parameter tuning process is based on the value of the error indicator for that this tuner is responsible. The tuning process works as follows :

1. The auto-critical function computes the value of the error indicator related to the parameter tuner.
2. If the error indicator value is good :
   a. The evaluation-based parameter tuner tries to improve the current parameter value.
   b. If the tuner cannot, the tuning process finishes successfully with the current parameter values as the tuned parameter values.
3. If the error indicator value is not good :
   a. The evaluation-based parameter tuner tries to produce how new parameter values which may help the foreground detection task to improve the values of the error indicator.
   b. If the evaluation-based parameter tuner cannot produce such new parameter values, the tuning process is considered unsuccessful.
   c. If the evaluation-based parameter tuner can produce such new parameter values, the controller requests the foreground detection task to set the new parameter values.
   d. The foreground detection task uses these new parameter values to detect the foreground in the next frame, then the tuning process returns to step 1.

While the context-based parameter tuner needs a knowledge base which is learned in an offline phase, the evaluation-based parameter tuner relies completely on information retrieved during the testing phase.

This work has several limitations. First, in the context-based parameter tuner, the author assumes that a same context cannot own different optimal parameter values. This is a strong assumption because in many cases, due to the optimization process, we find only the local optimal parameter values. Also, in general, the context definition can be not good enough to determine the unique optimal parameter values. Second, the authors assume no relationships between parameters when finding the optimal parameter values. Another important assumption is that this method requires the knowledge of the tuned parameters (e.g. which parameter to change for a given problem, the effect of changing parameter values). This prevents this method from being generic enough to control other object detection algorithms without this knowledge.

An important task in the control process of tracking algorithms is the online tracking quality evaluation. The objective of this task is to estimate online the performance of a tracking algorithm. Thanks to this, a controller can detect in real time (or with a low temporal delay) when the tracker fails and proposes suitable strategies for improving the tracking performance. In the following section, we will present some methods to evaluate online the object tracking quality.

- Online tracking quality evaluation

Online evaluation of tracking algorithms receives some attention in the existing literature [Erdem et al., 2004, Wu and Zheng, 2004, Wu et al., 2007]. In [Erdem et al., 2004], the authors define a metric called inter-frame color histogram metric for evaluating the tracking quality. The objective of this metric is to compare the color histogram of an object at every time instant t (denoted H_t) with the average color histogram of object within interval $[t-i, ..., t+i]$ (denoted $\overline{H_t}$). As the total number of pixels in the two histograms can be different, a normalization step is done. In this paper, the authors propose to use one of four different distances named L_1, L_2, χ^2 and histogram intersection distances to compute the difference between two histograms H_t and $\overline{H}_t$.

In [Wu and Zheng, 2004], the principle is to consider the coherence of five object descriptors over time : velocity, distance displacement, area, shape ratio and color. The quality of an output trajectory is said good if the values of the above descriptors corresponding to objects belonging to this trajectory do not undergo a large variation. At each frame, for each tracked object, the authors compute the variation of these descriptors by comparing the values of object descriptors between two consecutive frames. For each descriptor, by using predefined thresholds, this variation is mapped into a binary score : if the variation is not significant (i.e. lower than the corresponding threshold) then score is set to 1, otherwise score equals 0. These five binary scores (corresponding to the five object descriptors) are combined to quantify the tracking quality for the considered tracked object.

These two methods can well perform in videos in which the detection quality is good. However many drawbacks can be found. For the first one, the color histogram is not reliable enough to evaluate the tracking quality in some cases, for example in the case of occlusion. For the second method, with the five presented descriptors, there are still some tracking errors that the algorithm cannot take into account. For example the tracker can lose an object track even if this object remains in the scene. It is a popular error of most tracking algorithms and no descriptor presented in this method has been proposed to characterize this error. The experimentation of both methods are only done with some simple sequences in which there are only single object movement.

Beside the two mentioned methods, there exist several tracking approaches using an online evaluation. The tracking algorithms [Yang et al., 2005, Souded et al., 2011] which rely on particle filter belong to this category. The notion of "particle" (or sample) represents elements used to describe an object state. Although the objective of these algorithms are not to evaluate directly the tracking quality, they only evaluate online particles to filter out the low quality particles. This idea can be used to evaluate the object tracking quality. The particle filter is a Bayesian sequential importance sampling technique, which recursively approximates the posterior distribution using a finite set of weighted samples. First, a set of samples is created. Each sample is associated with a weight value representing its reliability. The particle filter algorithm consists in three steps : prediction, correction and re-sampling. In the prediction step, for each sample, a new sample is generated by a transition model. In the correction step, the new samples are corrected using the predicted and measurement results. Each new particle is weighted by its likelihood. In the re-sampling step, the particles whose weights are small enough are removed. The best particles are used to produce a new set of particles to ensure a constant number of particles over time. Depending on the notion of particle (e.g. object local feature such as Shift or HOG descriptor, or object global feature such as object color, size), we can use the particle filter to evaluate online the tracking quality in different cases. Moreover, a control process on the prediction (e.g. change the transition model) and re-sampling steps (e.g. change the re-sampling method or the number of particles) can be proposed to adapt the tracker performance to the context variation. However, the reliability of this method depends on the reliability of particles. The selection of suitable features for describing a particle is a hard task.

In this thesis, we propose a control algorithm for trackers based on object appearance or on points of interest in a single camera because these approaches are the most common ones. The algorithms belonging to this category are classified in "single view" or "point tracking" approaches in the taxonomy tree (see figure 2.1). In order control this tracker category, we

need to study the scene/object conditions which have significant influences on the tracking quality. We propose some following features to study :

- Contrast of objects, denoted C.
- 2D area of objects, denoted A.
- Scene illumination intensity, denoted I.
- Complexity of object trajectories, denoted Comp.

We divide the feature values into three levels from low to high, denoted 1, 2, 3.

Table 2.1 presents some object tracking approaches and the studied feature values of the tested videos. From this table, for each tracking approach, we can find that the tested video sequences have the same levels on contrast and 2D area of objects. Only one change is found in [Kwolek, 2009] (at the contrast condition). Otherwise, the values of the two left conditions (scene illumination and object complexity) vary. It means that the tracking approaches presented in this table can only cope with constant conditions on the contrast and 2D area of objects. So, these two conditions might influence significantly the quality of the object appearance-based tracking algorithms.

2.3 Discussion

Since the appearance of the first papers on object tracking in the years 1980s [O'Rourke and Badler, 1980], this topic has always attracted the attention of researchers. Many object tracking methods have been proposed in the state of the art and obtain some good results. However, as other vision tasks, the object tracking is dependent on processed video sequences. The variations of occlusion level, contrast and size of mobile objects within the processed video sequence can influence significantly the tracking performance. In order to solve this problem, some tracking control approaches have been proposed in the recent years. These studies can be classified into three categories : offline-based [Hall, 2006, Sherrah, 2010, Caporossi et al., 2004], online-based [Kuo et al., 2010, Santner et al., 2010] or hybrid approaches [Georis et al., 2007]. There are also many other approaches whose objective is to control other vision tasks [Thonnat et al., 1999, Yong et al., 2005, Martin and Thonnat, 2008, Nghiem, 2009]. However their ideas can be inherited to define a tracker controller. These methods have the following issues.

The first issue relates to the context notion. Depending on the property and objective of the study, each method has its proper definition of context. Moreover most of them relate to the static image applications [Martin and Thonnat, 2008, Shekhar et al., 1999, Thonnat et al., 1999, Yong et al., 2005]. Only few of them address video processing [Caporossi et al., 2004, Georis et al., 2007]. We have not found any paper about context study for object tracking.

The second issue pertains to the assumptions of these studies. Many approaches have too strong assumptions. For example in [Sherrah, 2010], the algorithm quality expression is descri-

Approach	Object descriptors	Video description	C	A	I	Comp
[Snidaro et al., 2008]	Haar and LBP	Building hall (Caviar dataset)	1	1	3	1
		Shop corridor (Caviar dataset)	1	1	2	2
		Parking	1	1	3	1
[Bak et al., 2009]	color and texture	Airport (TRECVid dataset)	3	2	3	1
		Home-care	3	2	2	2
[Kwolek, 2009]	color covariance	Two persons occluded	3	1	3	1
		Shop corridor (Caviar dataset)	1	1	2	2
		Building hall (PETS 2006)	3	1	2	1
[Maggio et al., 2007]	color and gradient direction	Person head (6 sequences)	2	1	3	1
		Outside scenes (6 sequences : Pets, Caviar)	2	1	3	2

Table 2.1 – An enumeration of features influencing the tracking quality. The column **Object descriptors** presents the object descriptors used for these tracker approaches. **C** : Contrast of objects ; **A** : 2D area of objects ; **I** : Scene illumination intensity ; **Comp** : Complexity of object trajectories. The features have three values 1, 2, 3 representing the feature value level from low to high.

bed as a function of control parameters. This function is assumed not to have local extremums. This hypothesis might be right when there is only one control parameter. Otherwise, the quality function varies depending on the number and properties of the control parameters.

The third issue relates to the online evaluation of the tracking quality. This evaluation task is very useful because it can detect when the performance of tracking task decreases to activate the parameter tuning process. Although different control approaches have been presented, only few of them (for example in [Georis et al., 2007, Nghiem, 2009]) use an online evaluation as a complementary information to support the control process. Moreover, this part is not explained clearly enough in these papers.

The fourth issue is about the knowledge base used in some control methods. As presented above, the knowledge base is often multiform. It can be either a set of rules for tuning parameters, or the optimal parameter values for a given context. The knowledge base construction requires generally user or expert interaction. This can ensure a reliable knowledge base compared to the unsupervised training acquisition of the knowledge. However, human interaction is time consuming. So the open problem is how to reduce at most this interaction.

The fifth issue pertains to the generic level of the control methods. We can see that most of the control studies address static image applications or video object detection. In the object tracking process, some authors have proposed control algorithms to adapt the tracking performance to the scene variations. However these methods are designed specifically for their tracking algorithms and are not generic enough to handle the other tracking approaches.

In this manuscript, we propose a control method for object tracking algorithms addressing these issues. The control strategies can regulate the tracking parameters for the current context to improve the tracking quality. The objective of the proposed approach is generic (remove the strong assumptions, control different tracking algorithms), flexible (easy to control other tracker approach types) and intelligent (decrease as most as possible human interaction, evaluate online the tracking quality).

We propose a method to control the object trackers based on object appearance and on points of interest (see figure 2.1). These tracking approaches are selected because they are the most popular ones and are largely studied in the state of the art.

3

Overview of the Proposed Approach

In this manuscript, we propose a control method which is able to tune and learn online tracking parameters to adapt the tracking task to the video scene variations. The proposed approach includes two phases : an offline learning phase and an online testing phase. This chapter presents a general description of these two phases. The details of these phases can be found in the next two chapters.

3.1 Offline Learning

The objective of this learning phase is to create or update a learned database which supports the control process of a tracking algorithm. This base contains the best satisfactory parameter values of the tracking algorithm for various scene conditions on the density, occlusion level of mobile objects, the object contrast and the object 2D area. The learned parameter values are associated with two values representing their reliability.

The learning is done in an offline phase to avoid the constraint related to the processing time. Also, thanks to this, we can use a manual annotation of the detection and tracking tasks to increase the precision of the learned database.

This phase takes as input training video sequences, annotated detected objects (including 2D positions and 2D bounding box sizes), annotated trajectories, a tracking algorithm and its control parameters. The term “control parameters” refers to parameters which are considered in the control process (i.e. to look for best satisfactory values in the learning phase and to be tuned in the online phase). The performance of the control algorithm depends strictly on how much the control parameters influence the tracking quality. For each control parameter, this phase requires a name, range of values and step value. The step value of a parameter is defined

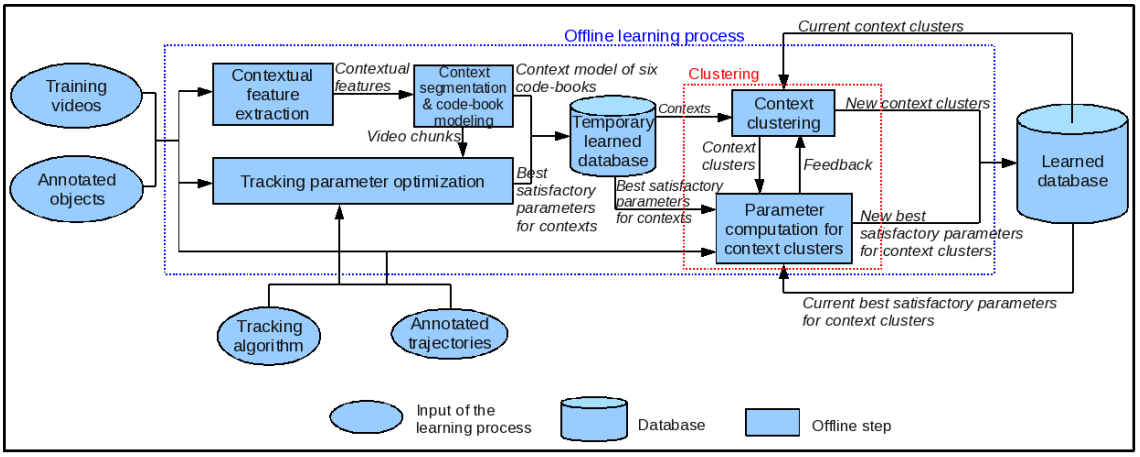


Figure 3.1 – Illustration of the offline learning scheme

as the minimal variation which can cause a significant change on the tracking quality. This value helps to avoid scanning the entire parameter space when searching its best satisfactory value for the tracking quality.

At the end of the learning phase, a learned database is created (if this is the first learning session) or updated (from the next learning sessions). A learning session can process many video sequences. Fig 3.1 presents the proposed scheme for building the learned database.

In the following sections, we describe an overview of the learning process as well as the execution steps including contextual feature extraction, context segmentation and code-book modeling, tracking parameter optimization, and clustering. The clustering step includes two sub-steps : context clustering and parameter computation for context clusters.

3.1.1 Learning Process Description

In this work, a tracking context of a video sequence is defined as a set of six features : the density, occlusion level of mobile objects appearing in this video sequence, the object contrast, object contrast variance, object 2D area and object 2D area variance. For each training video, we extract the contextual features from the annotated object detection and then use them to build a model of six code-books. In parallel, we perform the optimization process to determine the best satisfactory tracking parameter values. These parameters and the model of code-books are inserted into a temporary learned database.

After computing the best satisfactory parameters for all training videos, we perform a clustering of the contexts. At the end of a learning session, the learned database is created or updated by new context clusters and new best satisfactory tracking parameters.

3.1.2 Execution Steps

3.1.2.1 Contextual Feature Extraction

The notion of "tracking context" (or "context") in this work represents elements in the videos which influence the tracking quality. For different tracking approach types, the "context" definition can be different. For example, for tracking objects in a multi-camera network, the distance between camera views is very important to decide the parameter value on the duration for the object displacement from a camera view to another one. Therefore, the context of tracking algorithms for a multi-camera network should contain the information on the distance between camera views belonging to this network.

In this manuscript, we focus on the tracking algorithms which rely on object appearances (e.g. object color, texture). Therefore the context of a video sequence is defined as a set of six features which influence strongly object appearances : the density, occlusion level of mobile objects, the object contrast, object contrast variance, object 2D area and object 2D area variance. This step takes as input a training video and its annotated mobile objects. The annotation includes positions and 2D bounding box sizes of objects. This step gives as output the values of the contextual features over time.

3.1.2.2 Context Segmentation and Code-book Modeling

The context of a training video sequence can undergo a large variation. Therefore it is not correct enough to keep the same parameter values all along video. In order to solve this problem, we propose an algorithm to segment the training videos in consecutive chunks whose context has to be stable enough.

When a tracker fails in some frames, the tracking quality in the following frames are consequently influenced. So some low frequent contextual feature values can take an important role for keeping a good tracking quality. For example, when the density of mobile objects is high in few frames, the tracking quality can decrease significantly from that moment, even if in the following frames the density of objects decreases. Therefore, we decide to use a code-book model [Kim et al., 2004] to represent the contextual feature values of video chunks because this model can estimate complex and sparse distributions.

For each training video chunk, this step takes as input the values of the six contextual features. A contextual feature distribution is represented by a code-book model whose codewords describe the values of the corresponding feature. At the end of this step, each training video is divided into a set of chunks. Each video chunk context is represented by a model of six code-books corresponding to the six contextual features : the density, occlusion level of mobile objects, the object contrast, object contrast variance, object 2D area and object 2D area variance.

3.1.2.3 Tracking Parameter Optimization

The objective of the tracking parameter optimization step is to find the values of the control parameters which ensure the best possible tracking quality for each video context chunk. This quality is greater or equal to the threshold ϵ presented in hypothesis 3, section 1.2.2. These parameters are called "best satisfactory parameters". The optimization task is done using the manually annotated trajectories.

This step receives as input the annotated detected objects, a tracking algorithm, a video chunk and a set of control parameters for this tracker. The annotated objects are used as the object detection result. This step gives as output the best satisfactory parameter values. The information of a control parameter includes its name, value range and step value.

If the search space of the control parameters is small, an exhaustive search can be used to scan the values of these parameters. Otherwise, we use a genetic algorithm [Goldberg, 1989] or enumerative search. In some cases, we can convert an optimization problem to a classification problem, an Adaboost algorithm [Freund and Schapire, 1997] can be used for finding the best satisfactory parameter values.

Thanks to the annotated trajectories, we can evaluate the tracking results using some pre-defined tracking evaluation metrics to find the best satisfactory parameter values.

In order to represent the reliability, the best satisfactory parameter values are associated with two values. The first one is the number of frames of the training video in which the mobile objects appear (called "number of training frames"). The second one is the quality of the tracking output compared to the manual annotated trajectories.

The best satisfactory parameter values, their reliability values and the contextual code-book model corresponding to the training video chunk are stored into a temporary learned database.

3.1.2.4 Clustering

In some cases, two similar contexts can have different best satisfactory parameter values because the optimization algorithm only finds the local optimal solution. Another problem is that the context of a video sequence can be not sufficient for playing the role of a key value for determining the best satisfactory tracking parameter values. A context clustering process is thus necessary to group similar contexts and to compute the best satisfactory parameter values for the context clusters. This process includes two sub-steps : context clustering and parameter computation for context clusters.

The clustering step is done at the end of each learning session. This step takes as input the data stored in the temporary learned database and in the learned database resulting from the previous learning sessions. These data include a set of contexts or context clusters associated with their best satisfactory tracking parameter values. The data from the learned database is

only required as input from the second learning session and the followings. These contexts and context clusters are then grouped by classifying the contextual features.

The reliability of the best satisfactory parameter values of a component context plays the role of a weight value for computing context cluster parameter values. This weight value is directly proportional to the number of training frames and to the tracking output quality. The best satisfactory tracking parameter values of a context cluster are defined as a weighted combination of the best satisfactory parameter values of contexts belonging to this cluster. The best satisfactory parameters of a context cluster are also associated with the number of training frames and tracking quality of the best satisfactory parameters in component contexts to represent their reliability.

At the end of this step, a learned database is created (for the first learning session) or updated (for from the second learning session) with the new context clusters, new best satisfactory tracking parameter values and their weights.

3.2 Online Testing

The objective of the online testing phase is to tune and learn online the parameter values for obtaining the best tracking performance. This phase is divided into two stages : initial parameter configuration and parameter adaptation stages. While the objective of the first stage is to set values for control parameters at the first frame and for the parameters of the proposed controller, the second stage is responsible for tuning and for learning these parameters over time to adapt them to the video context variation.

3.2.1 Initial Parameter Configuration

At the first frames, as the control process does not have enough information to compute the context (a context should be computed for a sequence of at least 50 frames), it cannot decide how to set the tracking algorithm parameters. Therefore we define a mechanism to initialize the parameter values, for example the mean values or preference values (defined by an expert or user). Also, this step assigns the necessary values for the proposed controller parameters.

3.2.2 Parameter Adaptation

In the parameter adaptation, we propose two approaches called “context-based parameter adaptation” and “evaluation-based parameter adaptation”. While the first approach only relies on the information of the current video context to tune the tracking parameters (see figure 3.2), the second approach uses an online tracking evaluation and a clustering task to support the online control process (see figure 3.3).

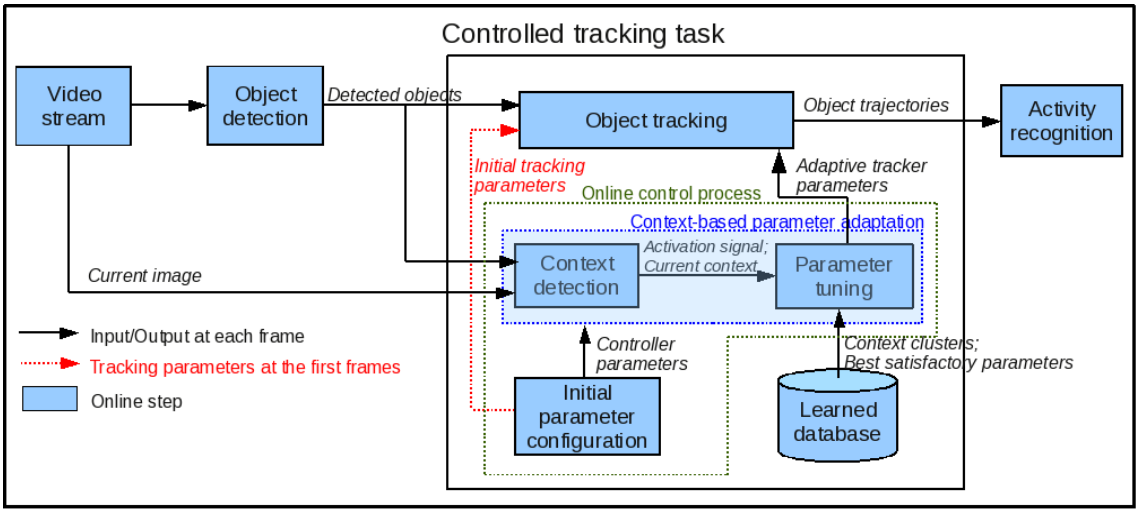


Figure 3.2 – Illustration of the tracking task with the context-based parameter adaptation approach

3.2.2.1 Context-Based Parameter Adaptation Approach

The objective of the context-based parameter adaptation approach is to detect contextual changes and to adapt the tracking parameter values to these changes. This approach takes as input the test video stream, the list of detected objects at every frames, the offline learned database and gives as output the adaptive tracking parameter values for every new contexts detected in the test (see figure 3.2).

In the following section, we describe the approach as well as the different steps of this approach including the context detection and parameter tuning steps.

- Approach Description

At the first frames, the controller sets the initial parameter values for the object tracking algorithm. After each predefined number of frames (e.g. 50 frames), if the context detection step detects a variation of context, the parameter tuning step looks for the best satisfactory parameters corresponding to the new detected context in the offline learned database. The tracking parameters are set to the found parameter values. If the context is not in the database, the tracking parameters are kept unchanged. In this case, the detected context is marked to be learned in the next learning sessions.

- Execution steps

+ Context Detection

When the tested video stream context changes (i.e change of the density, occlusion level, contrast, contrast variance, 2D area or 2D area variance of mobile objects), the current tracking

parameters may not be effective anymore. The objective of this step is to extract the contextual features of the current video chunk and to detect whether a contextual change has happened.

To do that, for each video chunk of similar temporal length, denoted N (e.g. N = 50 frames), its context is computed based on the object detection results. Then we search for the closest context cluster in the offline learned database. A contextual change is detected when the context of the current video chunk does not belong to the context cluster of the previous video chunk. If so, this step sends an activation signal to the parameter tuning step, otherwise nothing is done.

+ Parameter Tuning

The objective of the parameter tuning step is to find the best satisfactory parameter values which are adapted to the current video context. This step receives as input the activation signal and the current context from the "context detection" step and gives as output new adaptive tracking parameter values. The principle is that once this task receives an activation signal, it searches for the learned context cluster which best matches the input current context. The best satisfactory parameters corresponding to this found cluster are then used to parameterize the tracking algorithm. If no context cluster in the learned database matches with the current context, this context is marked to be learned in an offline phase.

3.2.2.2 Evaluation-Based Parameter Adaptation Approach

Similar to the context-based parameter adaptation, the evaluation-based parameter adaptation approach also takes as input the video stream, the list of detected objects at every frame and the learned database. However this approach requires the tracking result, control tracking parameter values at every frame and the exit zones of the camera view. The learned database is the result of the offline learning process but in this approach, this base can be updated online with an unsupervised learning process. The output of this approach is the set of control parameter values corresponding to the video contexts detected over time and the updated learned database.

Compared to the context-based parameter adaptation approach, we add two sub-tasks which are "online tracking evaluation" and "clustering" (see figure 3.3) to better optimize the control processing time, update online the learned database and to detect the limitations of this database.

In the following sections, we present the approach description as well as the execution steps including context detection, online tracking evaluation, parameter tuning and learning, and clustering.

- Approach Description

After each predefined number of frames (e.g. 50 frames), we compute the outputs of the

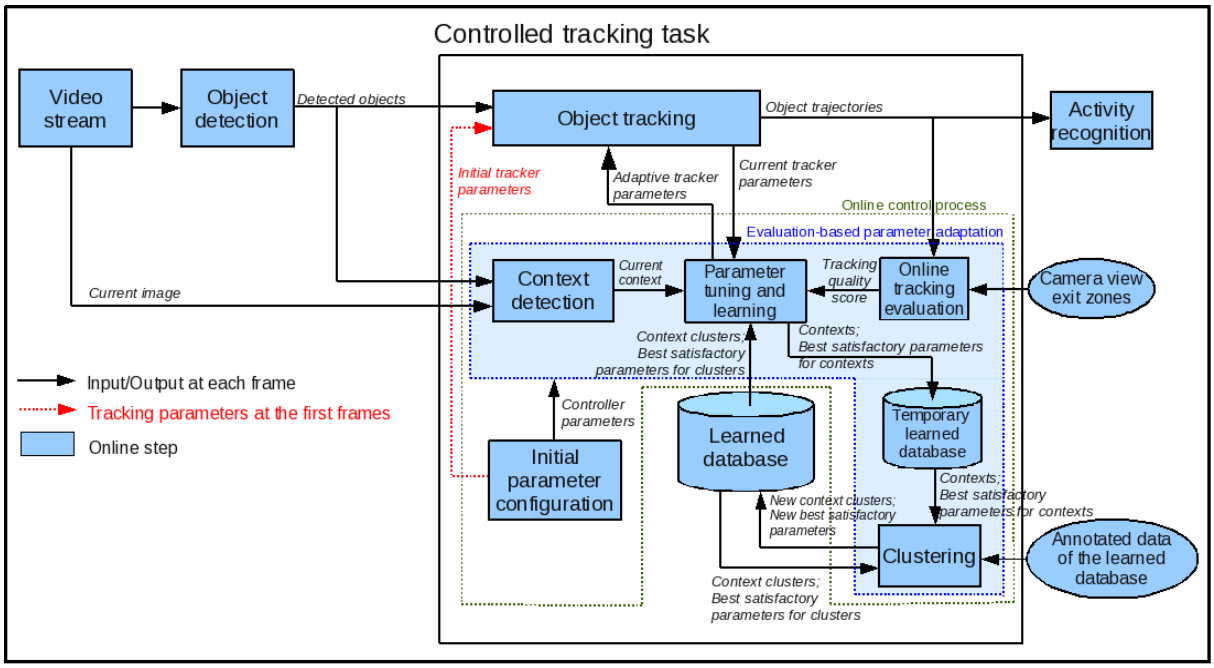


Figure 3.3 – Illustration of the tracking task with the evaluation-based parameter adaptation approach

context detection and online tracking evaluation tasks for the current video chunk. Using these values, the parameter tuning and learning step decides whether to tune the tracking parameters and to update the learned database (see the description of "parameter tuning and learning step" for more details).

After every predefined temporal interval (e.g. 5 minutes) or at the end of the processed video stream, the clustering step is activated to re-group the learned database to add the new contexts.

- Execution steps

+ Context Detection

The definition of the context detection step is similar to the one defined in the context-based parameter adaptation approach (see the "context detection" description in section 3.2.2.1 for more details).

+ Online Tracking Evaluation

The objective of the online tracking evaluation step is to evaluate online the tracking algorithm performance without ground-truth data. This step receives as input the object trajectories, exit zones of the camera view and gives as output every predefined number of N-frames (N is defined in section 3.2.2.1) a score corresponding to the tracking quality. This score is high (e.g.

greater than 0.5) when the considered tracker quality is good, and is low (e.g. lower than 0.5) when the tracker fails.

+ Parameter Tuning and Learning

The objective of the parameter tuning and learning step is to find the best satisfactory parameter values which are adapted to the current video context and to update the learned database using an unsupervised learning process. For each predefined number of N-frames (this number is equal to the one defined in context detection and online tracking evaluation steps), this step receives as input the current video context from the "context detection" step, the current tracking parameters and the tracking quality score from the "online tracking evaluation" step. The output of this step is new adaptive tracking parameter values and the updated learned database.

The execution mechanism is done as follows :

– If the tracking quality is good enough, the context of the considered chunk associated with the current tracking parameters are updated into the learned database.
– If the tracking quality is low and a contextual change is detected, it searches for the learned context cluster which best matches to the input current context. The best satisfactory parameters corresponding to this found cluster are then used to parameterize the tracking algorithm. If no context cluster in the learned database matches with the current context, the tracking parameters are kept unchanged. This context is marked to be learned in an offline learning phase.
– If the tracking quality is low and a contextual change is not detected, it means that the learned best satisfactory parameters for this video context are not sufficient. This testing video is marked to be relearned within an offline learning later.

We can find that the online tracking evaluation task helps to better optimize the control processing time because the parameter tuning is only activated when the tracker fails. Also, this approach can update online the learned database using an unsupervised learning algorithm. Finally it can detect the limitations of the learned database.

+ Clustering

In this approach, the learned database can be updated online with new detected contexts and their best satisfactory tracking parameters. Therefore a clustering process for the learned database is necessary to group the new added contexts. This step takes as input the learned database and gives as output the updated learned database. This clustering process is similar to the one defined in the offline learning phase (see section 3.1.2.4).

3.3 Summary

In this chapter, we have presented a general description on the proposed control algorithm. This algorithm includes two phases. An offline learning phase is first performed to determine the best satisfactory tracking parameters for each learning video context. The learning contexts are then classified by grouping their contextual features. Each context cluster is associated with the learned best satisfactory tracking parameters, number of training frames and tracking error. The online phase includes two stages : initial parameter configuration and parameter adaptation. For the parameter adaptation stage, two approaches are proposed : context-based parameter adaptation and evaluation-based parameter parameter adaptation. Compared to the first approach, the second one is better in term of processing time and the learned database can be updated online with an unsupervised learning process. However, the performance of this approach depends mostly on the online tracking evaluation quality. The next two chapters present in detail these two phases.

4

OFFLINE TRACKING PARAMETER LEARNING

This chapter describes in detail the execution steps of the offline learning phase of the proposed method. The objective of this learning phase is to create or update a learned database which supports the control process for a tracking algorithm. This base contains the best satisfactory parameter values of the tracking algorithm for various scene conditions on the density, occlusion level of mobile objects, their contrast with regard to the surrounding background, their contrast variance, their 2D area and their 2D area variance. The learned parameter values are associated with two values representing their reliability (see figure 4.24).

This phase takes as input training video sequences, annotated objects (including 2D positions and 2D bounding box sizes), annotated trajectories, a tracking algorithm and its control parameters. The term "control parameters" refers to parameters which are considered in the control process (i.e. to look for best satisfactory values in the learning phase and to be tuned in the online phase). For each control parameter, this phase requires a name, range value and step value. The step value of a parameter is defined as the minimal variation which can cause a significant change on the tracking quality. This value helps to avoid scanning the entire parameter space when searching its best satisfactory value for the tracking quality. In this work we consider only numeric parameters, however the proposed method can be applied also to symbolic parameters.

At the end of the learning phase, a learned database is created (if this is the first learning session) or updated (if not). A learning session can process many video sequences. Fig 4.1 presents the proposed scheme for building and updating the learned database.

In the following sections, first we study the influence of some contextual features for the

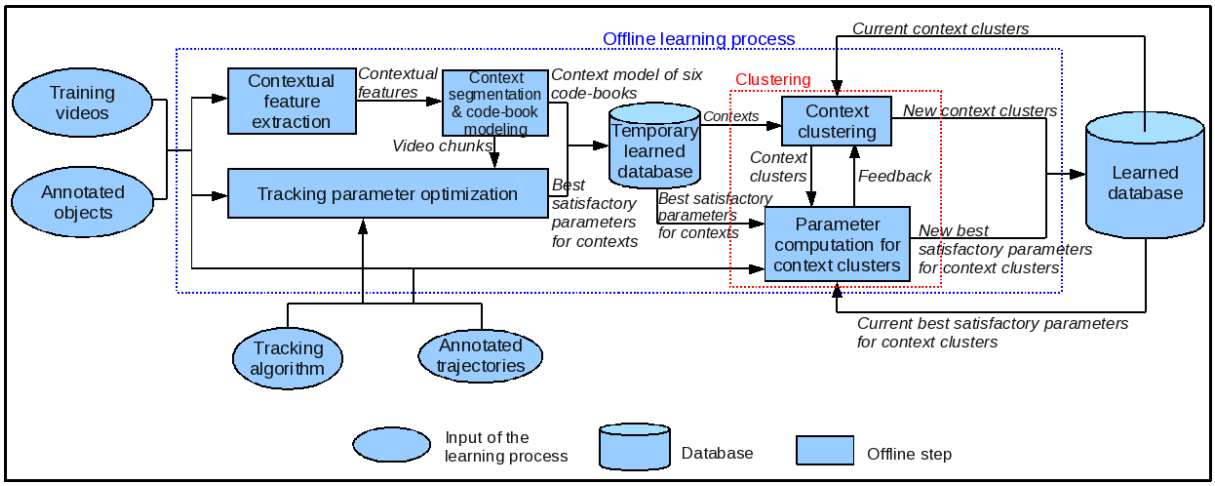


Figure 4.1 – Illustration of the offline learning scheme

object tracking performance. Second, we describe the execution steps of the offline learning process including contextual feature extraction, context segmentation and code-book modeling, tracking parameter optimization, clustering (including context clustering and parameter computation for context clusters).

4.1 Contextual Feature Study

The notion of "context" (or "tracking context") in this work represents elements in the videos which influence the tracking quality. The objective of this section is to search for contextual features for object trackers. Table 2.1 analyses some features extracted from the tested videos of some object appearance-based trackers. The analysed features are : the contrast of mobile objects with regard to their surrounding background, their 2D area, their movement complexity as well as scene illumination. The result shows that the three features : the object contrast and the 2D area can have a strong influence on these trackers. In order to verify this conclusion, we propose to study these features in this section. Moreover, we add two features to study which is the density and occlusion level of mobile objects. The density of mobile objects is defined as the ratio between the total of object 2D areas and the image 2D area.

In order to verify the influence of these features on the tracking quality, we test object appearance-based trackers with some video sequences whose these features undergo largely. If the tracking performances between these trackers are much different, we conclude that the performance of this tracker category is influenced by these features.

The tested tracking algorithms rely on the similarities of object descriptors over time : position (including 2D and 3D positions), 2D size (including 2D area and 2D shape ratio), color

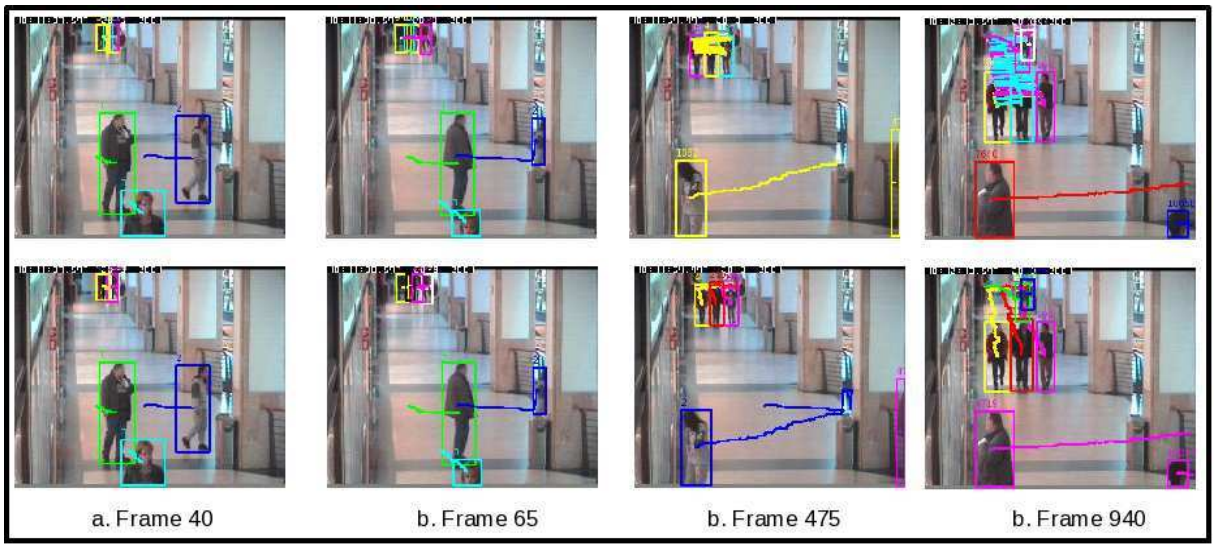


Figure 4.2 – Illustration of the tracking result when object density changes : First row : 2D size descriptor-based tracking ; Second row : color histogram descriptor-based tracking.

histogram and histogram of oriented gradients (HOG) to build object trajectories. The HOG-based tracker includes two stages. First the points of interest are detected using Fast algorithm [Rosten and Drummond, 2006]. Second, these points are tracked by computing the similarity of their HOG descriptors [Bilinski et al., 2009].

The following sections study respectively the tracking performance variation for the change of these four features.

4.1.1 Density of Mobile Objects

The density of mobile objects is defined as the occupancy of their 2D areas over the 2D camera view area. Figure 4.2 shows the object tracking in the sequence ThreePastShop1cor (from Caviar video dataset[1] at some frames in two cases : the first row illustrates the tracking result using 2D size descriptor ; the second row illustrates the tracking result using the color histogram descriptor. When there are only three persons appearing in the scene (at frame 40 and 65), both cases give good tracking quality. At frames 475 and 940, when three more persons appear, while the color histogram descriptor-based tracker still keeps a high tracking performance (in the second row), the tracking quality corresponding to the 2D size descriptor is very bad (in the first row). This can be explained by the fact that the 2D sizes of the new three persons are quite similar, so the size descriptors cannot discriminate them correctly. We conclude that the object density influences the tracking performance.

[1] http ://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

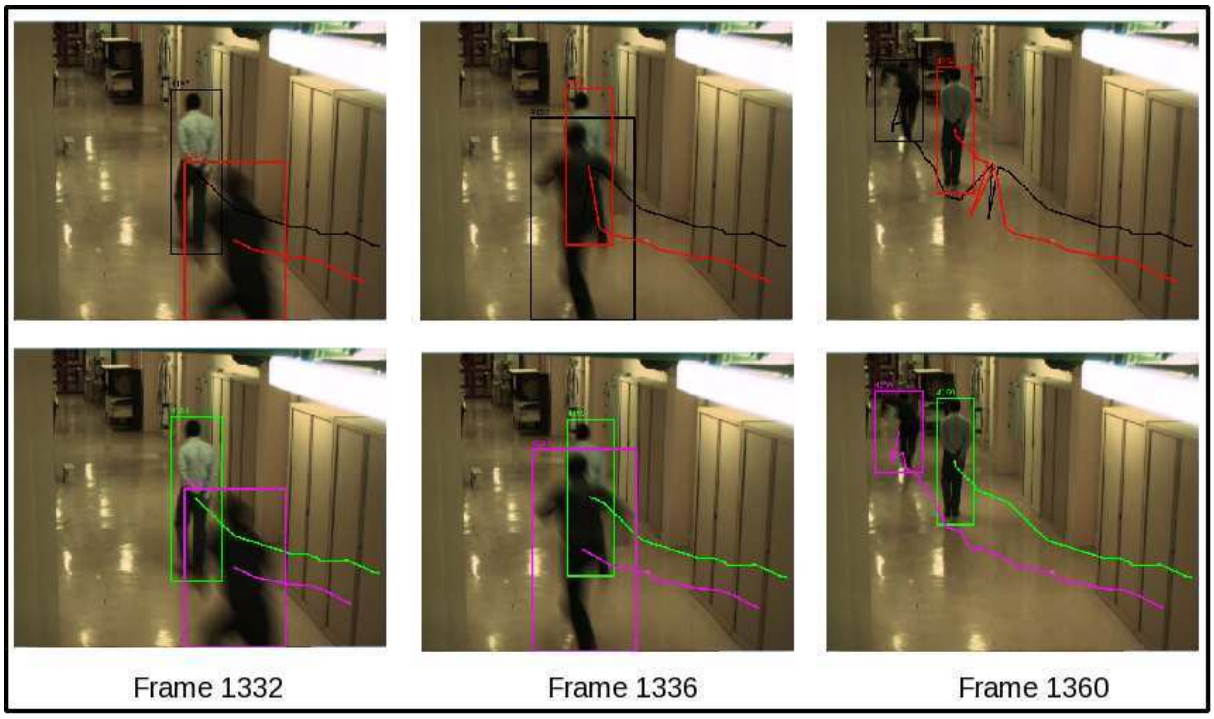


Figure 4.3 – Illustration of the tracking performance change when object occlusion occurs. First row : color histogram descriptor-based tracking ; Second row : 2D size descriptor-based tracking.

4.1.2 Occlusion Level of Mobile Objects

We study in this section the tracking performance variation when the object occlusions occur. Figure 4.3 shows the tracking result in the sequence ETI-VS1-BC-12-C1 (from ETISEO dataset[2] at three frames in two cases : the first row illustrates the tracking result using the color histogram descriptor ; the second row illustrates the tracking result when the object size descriptor is used. When there is no object occlusion (at frame 1332), both trackers give good results. At frame 1336, the object occlusion occurs, while the 2D size descriptor-based tracker still ensures a good quality, the color histogram-based tracker cannot. We conclude that the object occlusion level influences the tracking performance.

4.1.3 Object Contrast

The contrast of an object is defined as the color intensity difference between the object and its surrounding background. Figure 4.4 shows the tracking result in the sequence ETI-VS1-BC-13-C4 (from the ETISEO dataset) at some frames in two cases : the first row illustrates the tracking result using the position descriptor ; the second row illustrates the tracking result using

[2]http ://www-sop.inria.fr/orion/ETISEO/

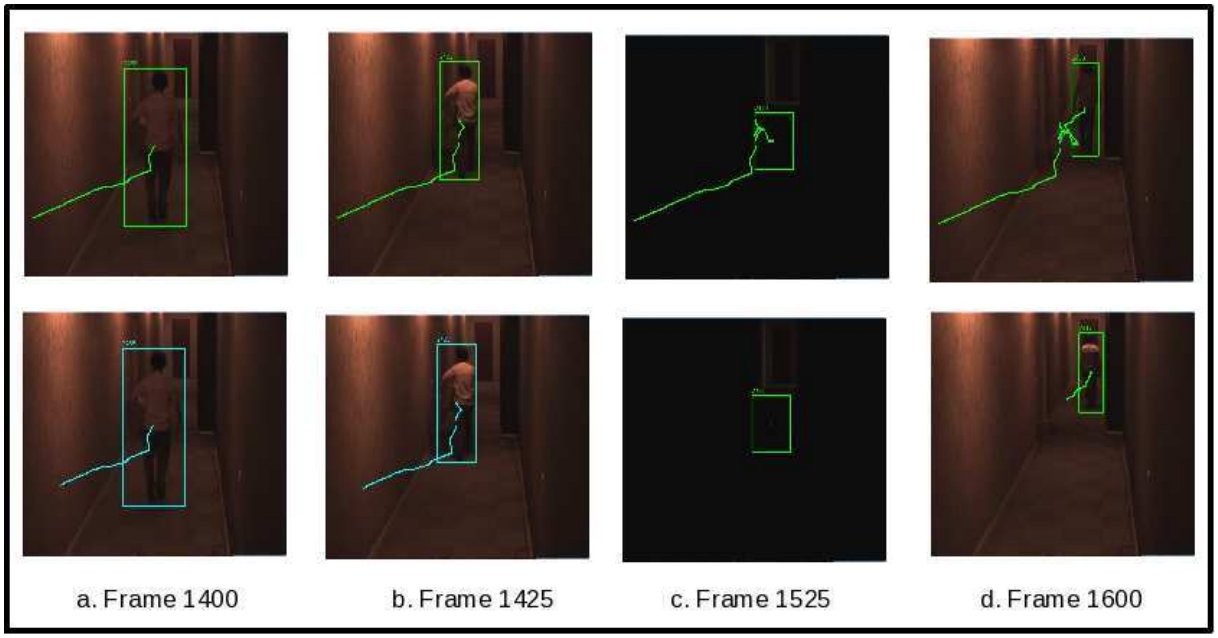


Figure 4.4 – Illustration of the tracking performance variation when object contrasts change. First row : position descriptor-based tracking ; Second row : HOG descriptor-based tracking.

the HOG descriptor. From frame 1400 to frame 1425, the object contrast does not change and the tracking results are good in both cases. From frame 1425 to frame 1525, the object contrast decreases, and from frame 1525 to frame 1600 it increases. While the 2D position-based tracker still ensures a good performance during these two periods, the HOG-based tracker cannot keep good tracking result. When the object contrast changes, the HOG descriptor reliability decreases. Therefore the HOG tracking fails. So the variation of object contrasts influences the tracking performance.

4.1.4 Object Area

In this section we study the tracking performance variation when object areas change. Figure 4.5 shows the tracking result in sequence ETI-VS1-BC-12-C1 (from the ETISEO dataset) at some frames in two cases : the first row presents the HOG descriptor-based tracking result ; the second row shows the color histogram-based tracking result. There are two persons in the camera view. The HOG-based tracker cannot track correctly the small person motion while the large person is still tracked correctly. This can be explained by the fact that for the small person, the number of points of interest is too small. Therefore the tracking of this person fails. For the color histogram descriptor-based tracker, the tracking quality is good in both cases. We conclude that the variation of object areas influences the tracking performance.

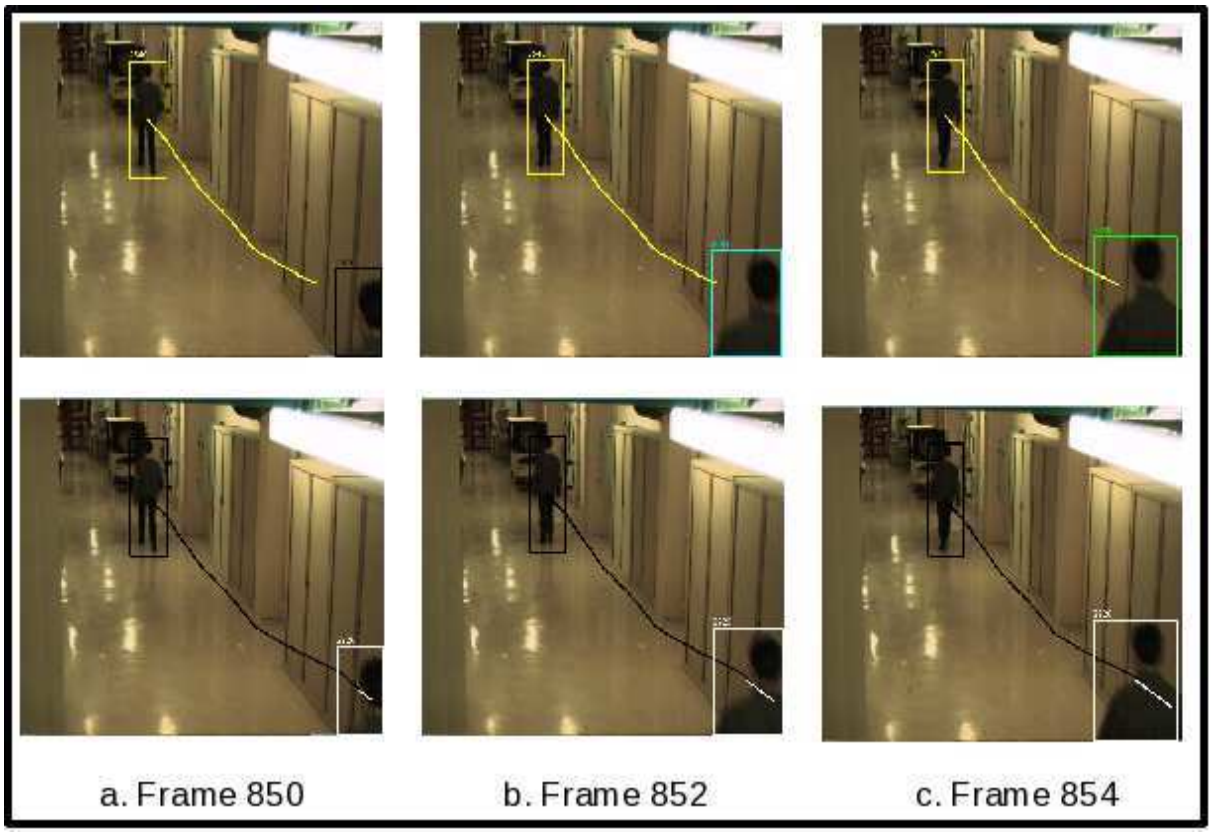


Figure 4.5 – Illustration of the tracking performance variation when object areas change. First row : HOG descriptor-based tracking ; Second row : color histogram descriptor-based tracking.

4.1.5 Conclusion

From the above analyses, we find that the features : the occlusion level, the density of mobile objects, their contrast with regard to the surrounding background, their 2D area influence significantly the tracking performance. In most cases, when one of these contextual features changes, the tracking quality of some object appearance-based trackers varies too. For example, when object density increases, the object 2D size descriptors are less reliable. Or when the object contrast decreases, the HOG descriptor cannot ensure a good tracking quality.

Therefore, we select the six following features to represent the "tracking context" (or "context") of a video sequence : occlusion level, density of mobile objects, their contrast intensity with regard to the surrounding background, the variance of their contrast, their 2D area and the variance of their 2D area.

4.2 Contextual Feature Extraction

For every frame, we extract six contextual features from annotated objects. This section presents in detail how this step can be done.

4.2.1 Density of Mobile Objects

The density of mobile objects influences significantly the tracking quality. A high density of objects causes a decrease of object detection and tracking performance. In particular, a variation of object density over time is challenging for tracking algorithms. Figure 4.6 illustrates the variation of object density in a corridor over time. Figure 4.6a. shows the view at frame 135, only one person appear in the scene. Figure 4.6b. presents the same camera view at frame 599 (around one minute later), many more people appear.

In this work, the density of mobile objects at instant t, denoted d_t is defined in function of their 2D area occupancy over the 2D camera view area at t. Greater the object 2D areas are, greater the occlusion probability and occlusion duration are. The mobile object density at t is defined as follows :

$$d_t = min(\frac{\sum_{i=1}^{n_t} a_t^i}{a_{img}}, 1) \tag{4.1}$$

where n_t is the number of mobile objects appearing at instant t, a_t^i is the 2D area of mobile object i at instant t and a_{img} is the 2D area of the camera view.

Figure 4.6 – Illustration of scene in two cases : a. At frame 135 : Low density of mobile objects b. At frame 599 : High density of mobile objects

4.2.2 Occlusion Level of Mobile Objects

The occlusion level of mobile objects is the first element which influences the tracking quality. An occlusion occurence makes the object appearance partially or completely not visible. It decreases the object detection and tracking performance. In particular, a variation of object occlusion level over time gives more challenges because the coherence of object appearance changes significantly. Figure 4.7a. presents a camera view at an airport with low density of mobile objects : only two persons appear in the scene. Figure 4.7b. presents the same camera view after two hours. We see that many more people appear and many occlusions occur.

Given two objects i and j, we compute their occlusion level based on their area overlap as follows :

$$ol_t^k = \frac{a_t^{ij}}{min(a_t^i, a_t^j)} \tag{4.2}$$

where k denotes the index value of this occlusion in the set of occlusions occuring at time t, a_t^{ij} is the overlap area of objects i and j at t. Two objects i and j are considered as in an occlusion state if ol_t^k is greater than Th_{ol}. Let m be the number of object occlusion occurrences at instant t, ol_t^k is the occlusion level of case k ($k = 1..m$). The occlusion level of mobile objects in a scene at instant t, denoted o_t, is defined as follows :

$$o_t = min(\frac{\sum_{k=1}^{m} ol_t^k \times 2}{n_t}, 1) \tag{4.3}$$

Figure 4.7 – Illustration of scene in two cases : a. At moment t : Low occlusion level of mobile objects b. At $t + 2h$: High occlusion level of mobile objects

Figure 4.8 – Illustration of object contrast variation over space : a. High contrast b. Low contrast

where n_t is the number of mobile objects at t. The multiplication by 2 in the formula is explained by the fact that an occlusion occurrence is related to two objects.

4.2.3 Contrast of Mobile Objects

The contrast of an object is defined as the color intensity difference between this object and its surrounding background. An object with low contrast means that its color is similar to the color of its surrounding background. As consequence, the object detection quality is significantly influenced. Second, a low object contrast level also decreases the discrimination of the appearance between different objects. So the quality of tracking algorithms which rely on object appearances decreases in this case. The contrast of an object can vary due to the change of its spatial location (see figure 4.8) or of time (see figure 4.9).

Let B_i be the 2D bounding box of object i which has center c_{B_i}, width w_{B_i} and height h_{B_i}.

a. At 12:08:28, the 31 - March - 2008 b. At 13:07:29, the 14 - Apr - 2008

Figure 4.9 – Illustration of object contrast variation over time : a. High contrast b. Low contrast

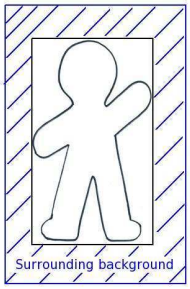


Figure 4.10 – Illustration of object surrounding background

α is a predefined value in interval $[0, 1]$, and B_i^+ is a 2D bounding box whose center $c_{B_i^+}$, width $w_{B_i^+}$ and height $h_{B_i^+}$ are respectively defined as follows :

$$c_{B_i^+} = c_{B_i} \tag{4.4}$$

$$w_{B_i^+} = w_{B_i} + \alpha min(w_{B_i}, h_{B_i}) \tag{4.5}$$

$$h_{B_i^+} = h_{B_i} + \alpha min(w_{B_i}, h_{B_i}) \tag{4.6}$$

The surrounding background of object i is defined as the area $\mathfrak{B}_i = B_i^+ \setminus B_i$ (see figure 4.10). Let H_i^R, H_i^G and H_i^B be respectively the normalized intensity histograms of object i in red, green and blue channels. Similarly, let $H_{\mathfrak{B}_i}^R$, $H_{\mathfrak{B}_i}^G$ and $H_{\mathfrak{B}_i}^B$ be respectively the normalized intensity histograms of the area $\mathfrak{B}_i$ in red, green and blue channels.

The contrast of object i, denoted c_i, is defined as follows :

$$c_i = \frac{\sum_{k=R,G,B} EMD(H_{\mathfrak{B}_i}^k, H_i^k)}{3} \tag{4.7}$$

where $EMD(H_{\mathfrak{B}_i}^k, H_i^k)$ is the Earth Mover Distance [Rubner et al., 1998] between $H_{\mathfrak{B}_i}^k$ and H_i^k. The EMD distance is selected because it is used widely in the state of the art for comparing two

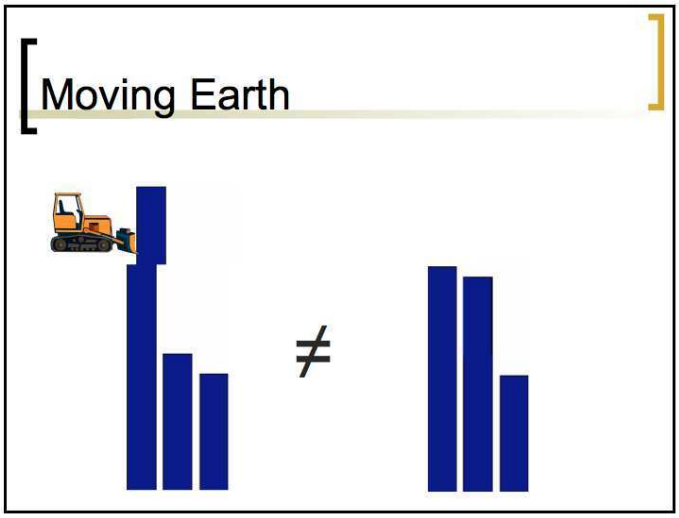


Figure 4.11 – An intuitive illustration of the transport of values from one bin to another bin in EMD (source [Peters, 2011])

histograms.

- Earth Mover's Distance (EMD)

The Earth Mover's Distance (EMD) is a distance between two distributions which reflects the minimal amount of work that must be performed to transform one distribution into the other by moving "distribution mass" around. In this section we describe how to apply this idea for comparing two histograms, denoted H_1 and H_2. Let N_1 and N_2 be the number of bins of histogram H_1 and H_2. Intuitively, the EMD measures the least amount of cost needed to transform values from every bin i of histogram H_1 into every bin j of histogram H_2 (see illustration in figure 4.11). The transport cost from bin i to bin j is defined as the distance (i.e difference) between bin i and bin j. The computation of the EMD is based on a solution to the old transportation problem [Dantzig, 1951]. This is a bipartite network flow problem which can be formalized as the following linear programming problem : each bin of H_1 is a supplier, each bin of H_2 is a consumer, and c_{ij}, called ground distance, is the cost to ship a unit of supply from bin $i \in H_1$ to bin $j \in H_2$, and is defined as follows :

$$c_{ij} = |i - j| \tag{4.8}$$

Figure 4.12 shows an example with three suppliers (corresponding to three bins of H_1) and three consumers (corresponding to three bins of H_2). We want to find a set of flows f_{ij} that minimize the overall cost.

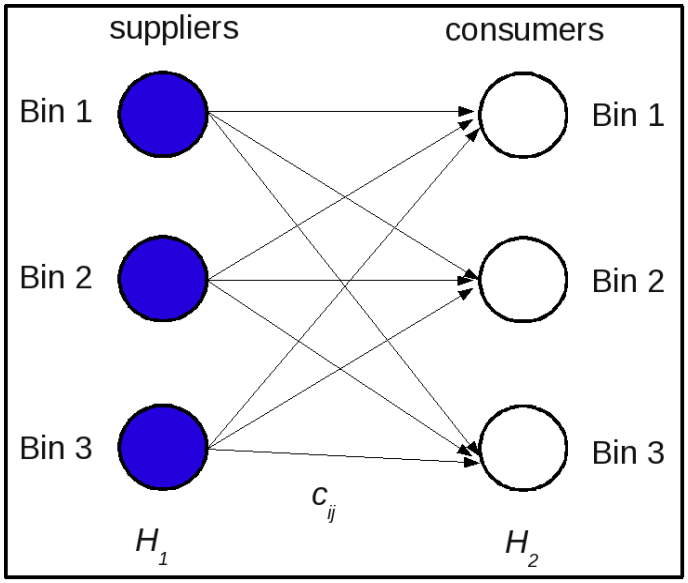


Figure 4.12 – An example of a transportation problem with three suppliers (corresponding to three bins of histogram H_1) and three consumers (corresponding to three bins of histogram H_2).

$$\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} c_{ij}f_{ij} \tag{4.9}$$

Let $H_1(i)$ be the value of bin i in histogram H_1 ; f_{ij} satisfies the following constraints :

$$f_{ij} \geq 0 \quad \forall i \in [1, N_1],\ \forall j \in [1, N_2] \tag{4.10}$$

$$\sum_{i=1}^{N_1} f_{ij} = H_2(j)\ ,\ j \in H_2 \tag{4.11}$$

$$\sum_{j=1}^{N_2} f_{ij} \leq H_1(i)\ ,\ i \in H_1 \tag{4.12}$$

Constraint 4.10 allows shipping of supplies from a supplier to a consumer and not inverse direction. Constraint 4.11 forces the consumers to fill up all of their capacities and constraint 4.12 limits the supply that a supplier can send to its total amount.

A feasibility condition is that the total demand does not exceed the total supply :

$$\sum_{j=1}^{N_2} H_2(j) \leq \sum_{i=1}^{N_1} H_1(i) \tag{4.13}$$

When the total value resulting from all of the bins of the two considered histograms are not equal, the smaller plays role as a consumer (i.e. H_2) in order to satisfy the feasibility condition

4.13. In this work, as we use the normalized histograms, the total value of all bins of each histogram is equal to 1 :

$$\sum_{i=1}^{N_1} H_1(j) \;=\; \sum_{j=1}^{N_2} H_2(j) \;=\; 1 \tag{4.14}$$

Therefore, the order of histograms in this EMD distance is not important. Once the transportation problem is solved, and we have found the optimal flow $\mathcal{F}^*$, the EMD between two histograms H_1 and H_2 is defined as :

$$EMD(H_1,\ H_2) = \frac{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} c_{ij}\mathcal{F}^*_{ij}}{\sum_{j=1}^{N_2} H_2(j)} = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} c_{ij}\mathcal{F}^*_{ij} \tag{4.15}$$

Using formula 4.15, we compute the $EMD(H^k_{\mathfrak{B}_i},\ H^k_i)$ for each color channel $k \in R,\ G,\ B$. The contrast of an object i is defined as in formula 4.7. The contrast value at t, denoted $\bar{c}_t$, is defined as the mean value of all object contrast at this instant :

$$\bar{c}_t = \frac{\sum_{i=1}^{n_t} c^i_t}{n_t} \tag{4.16}$$

where c^i_t is the contrast of object i at instant t, n_t is the number of mobile objects at instant t.

4.2.4 Contrast Variance of Mobile Objects

When different object contrast levels exist in the scene (see figure 4.13), a mean value cannot represent correctly the contrast of all objects in the scene. Therefore we define the variance of object contrasts at instant t, denoted $\hat{c}_t$, as their standard deviation value :

$$\hat{c}_t = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(c^i_t \;-\; \bar{c}_t)^2} \tag{4.17}$$

where c^i_t is the contrast value of object i at t (computed in formula 4.7), $\bar{c}_t$ is the mean value of all object contrasts at instant t (computed in formula 4.16).

4.2.5 2D Area of Mobile Objects

2D area of an object is defined as the number of pixels within its 2D bounding box. Therefore, this feature also characterizes the reliability of the object appearance for the tracking

Figure 4.13 – Illustration of the contrast difference between objects at an instant

Figure 4.14 – Illustration of variation of object areas over time : a. At 10 :12 :58 : Large object areas b. At 10 :13 :30 : Small object areas

Figure 4.15 – Illustration of the area difference between objects at an instant

process. Greater the object area is, higher the object appearance reliability is. Figure 4.14 illustrates the variation of 2D areas of objects in a same camera view.

The 2D area feature value at t, denoted $\bar{a}_t$, is defined as the mean value of the 2D areas of mobile objects at instant t :

$$\bar{a}_t = \frac{\sum_{i=1}^{n_t} a_t^i}{n_t} \tag{4.18}$$

where a_t^i is the 2D area of object i at instant t, n_t is the number of mobile objects at instant t.

4.2.6 2D Area Variance of Mobile Objects

When many objects exist in the scene (see figure 4.15), a mean value cannot represent correctly the area of all objects in the scene. Therefore we define the variance of object 2D areas at instant t, denoted $\hat{a}_t$, as their standard deviation value :

$$\hat{a}_t = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_t^i - \bar{a}_t)^2} \tag{4.19}$$

where a_t^i is the 2D area value of object i at t, $\bar{a}_t$ is the mean value of all object 2D areas at instant t (computed in formula 4.18).

4.3 Context Segmentation and Code-book Modeling

4.3.1 Context Segmentation

The objective of the learning phase is to determine the best satisfactory parameter values for each context. A long video usually has a varied context. As we have analysed above, the contextual variation can influence significantly the tracking quality. Therefore it is not correct enough to keep the same parameter values all along video. In order to solve this problem, we propose an algorithm to segment the training videos in consecutive chunks whose context has to be stable enough.

The minimal temporal length of a chunk, denoted l, should not be too low to decrease the processing time. This value should not be too high to ensure a stable enough context. The proposed context segmentation algorithm is done as follows.

1. We segment the training videos in a set of parts of l frames. The last part can have a temporal length lower than l. Each video part is supposed to have a stable enough context.
2. The contextual feature values of the video first part is represented by a context code-book model (see more details in section 4.3.2).
3. From the second video part, we compute the context difference between the current part and the context code-book model of the previous part (see more details in section 4.3.3). If their distance is lower than a threshold Th_1 (e.g. 0.5), the context code-book model is updated with the current video part. Otherwise, a new context code-book model is created to represent the context of the current video part.

At the end of the context segmentation algorithm, a training video is divided into a set of chunks (of different temporal length) that are corresponding to the obtained context code-book models. Each chunk has a stable enough context.

There are two open problems : How to represent a video context as a model of code-books ? and how to compute the distance between a context code-book model and a context. The following sections answer these two questions.

4.3.2 Code-book Modeling

During the tracking process, low frequent feature values can take an important role for tuning tracking parameters. For example, when mobile object density is high in few frames, the tracking quality can decrease significantly. Therefore, we decide to use a code-book model [Kim et al., 2004] to represent the values of contextual features because this model can estimate complex and low-frequence distributions. In our approach, each contextual feature is represented by a code-book, called feature code-book and denoted cb^k, $k = 1..6$. So a video

context is represented by a set of six feature code-books, called context code-book model and denoted CB, $CB = \{cb^k,\ k = 1..6\}$ (see illustration in figure 4.16). A feature code-book includes a set of code-words which describe the values of this feature. The number of code-words depends on the diversity of feature values.

4.3.2.1 Definition of Code-word

A code-word contains the intensity and temporal information of a contextual feature. A feature code-book can have many code-words. A code-word i of code-book k ($k = 1..6$), denoted cw_i^k, is defined as follows :

$$cw_i^k = \{\overline{\mu_i^k},\ m_i^k,\ M_i^k,\ f_i^k,\ \lambda_i^k,\ q_i^k\} \tag{4.20}$$

where

- $\overline{\mu_i^k}$ is the mean of the feature values belonging to this code-word.
- m_i^k, M_i^k are the minimal and maximal feature values belonging to this word.
- f_i^k is the number of frames in which the feature values belong to this word.
- λ_i^k is the maximum negative run-length (MNRL) defined as the longest interval during the training period in which the code-word has not been activated by any feature values (a code-word is activated if an incoming feature value belongs to this code-word).
- q_i^k is the last access time instant that the code-word has been activated.

4.3.2.2 Algorithm for Updating Code-word

The training phase for updating code-word works as follows :

- At the beginning, the code-book cb^k of a context feature k is empty.
- For each value μ_t^k of a contextual feature k computed at time t, verify if μ_t^k activates any code-word in cb^k. μ_t^k activates code-word cw_i^k if both conditions are satisfied :
  + μ_t^k is in range $[0.7 \times m_i^k,\ 1.3 \times M_i^k]$.
  + The distance between μ_t^k and cw_i^k is smaller than a threshold ϵ_3. This distance is defined as follows :

$$dist(\mu_t^k,\ cw_i^k) = 1 - \frac{min(\mu_t^k,\ \overline{\mu_i^k})}{max(\mu_t^k,\ \overline{\mu_i^k})} \tag{4.21}$$

  where $\overline{\mu_i^k}$ is the mean value of code-word cw_i^k (presented in section 4.3.2.1).
- If cb^k is empty or if there is no code-word activated, create a new code-word and insert it into cb^k. The values of this new code-word is computed as follows :

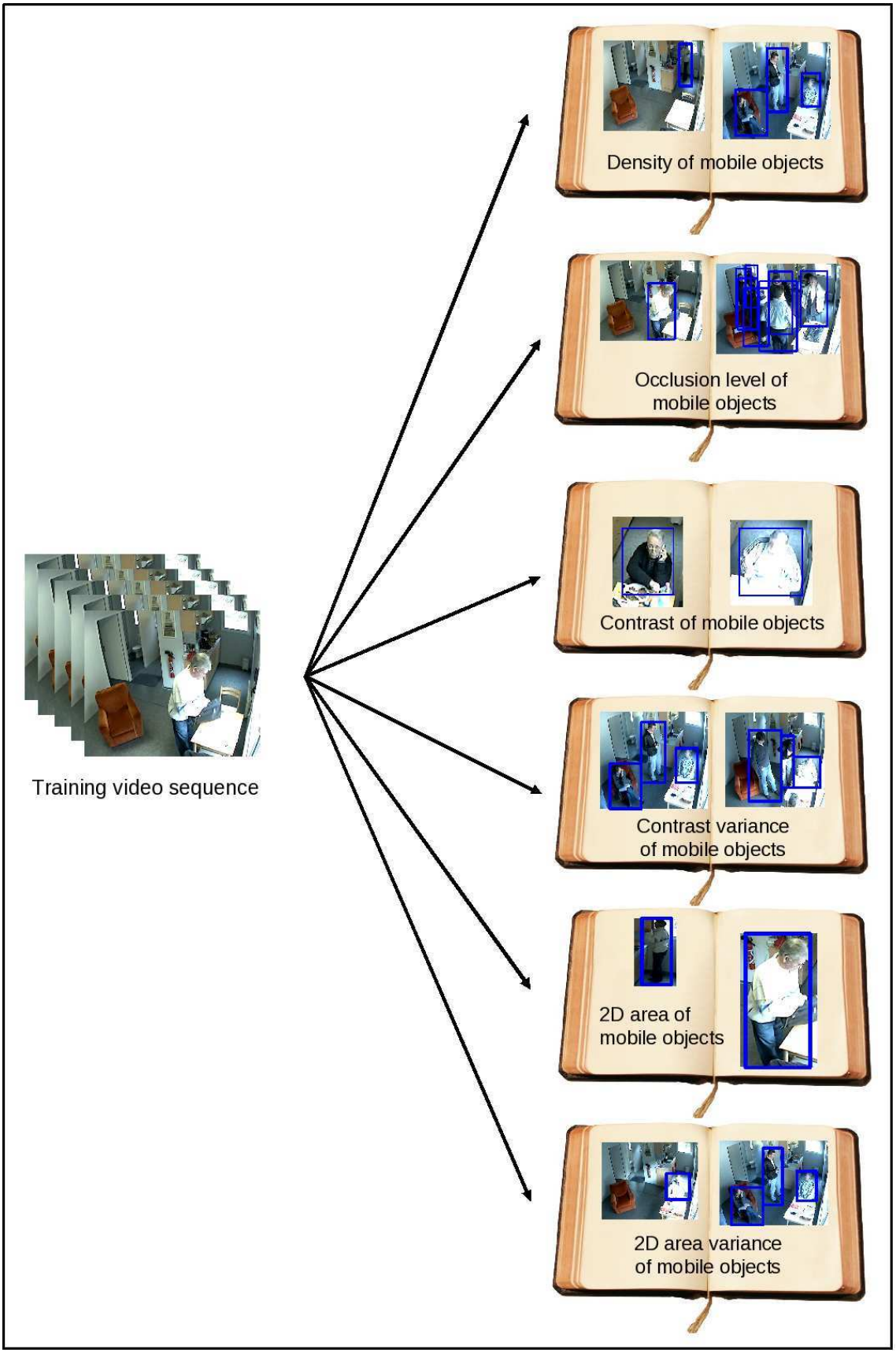


Figure 4.16 – Illustration of the code-book modeling for a training video chunk

$$\overline{\mu_i^k} = \mu_t^k \quad (4.22)$$
$$m_i^k = \mu_t^k \quad (4.23)$$
$$M_i^k = \mu_t^k \quad (4.24)$$
$$f_i = 1 \quad (4.25)$$
$$\lambda_i = 1 \quad (4.26)$$
$$q_i = t \quad (4.27)$$

– If μ_t^k activates cw_i^k, this code-word is updated with the value of μ_t^k :

$$\overline{\mu_i^k} = \frac{\overline{\mu_i^k} \times f_i + \mu_t^k}{f_i + 1} \quad (4.28)$$
$$m_i^k = min(m_i^k,\ \mu_t^k) \quad (4.29)$$
$$M_i^k = max(M_i^k,\ \mu_t^k) \quad (4.30)$$
$$f_i = f_i + 1 \quad (4.31)$$
$$q_i = t \quad (4.32)$$

As the temporal order of the contextual feature values within one code-word is not important, $\overline{\mu_i^k}$ in formula 4.28 is updated with the mean of all feature values belonging to this code-word.

– λ_i is kept unchanged.
– For the other code-words which are not activated, increase the value of λ :

$$\forall j \neq i : \lambda_j = \lambda_j + 1 \quad (4.33)$$

– At the end, for a code-word cw_i, the algorithm sets $\lambda_i = max(\lambda_i,\ \mathcal{N} - q_i)$ where $\mathcal{N}$ is the number of frames in which mobile objects appear.

The code-words whose value f_i is lower than a threshold, are eliminated because they are corresponding to very low frequence feature values.

4.3.3 Context Distance

The context distance is defined to compute the distance between a context c and a context code-book model $CB = \{cb^k,\ k = 1..6\}$.

The context c of a video chunk (of $\mathfrak{L}$ frames) is represented by a set of six values : the density, occlusion level of mobile objects, their contrast with regard to the surrounding background, their contrast variance, their 2D area and their 2D area variance over time. For each context

```
function ContextDist(c, CB, £)

Input : context code-book model CB, context c, £ (number of frames of context c)
Output : context distance between cluster CB and context c

countTotal = 0;
For each code-book cb^k in CB (k = 1..6)
   count = 0;
   For each value μ_t^k of context c
      For each codeword cw_i^k in code-book cb^k
         if (dist(μ_t^k, cw_i^k) < 0.5) {
            count++;
            break;
         }
   if (count / £ < 0.5) return 1;
   countTotal += count;

return ( 1 − countTotal/(£ ∗ 6) )
```

Table 4.1 – Function for computing the distance between a context code-book CB and a video context c

feature k ($k = 1..6$), the contextual feature value at instant t is denoted μ_t^k. For each such value, we consider whether it matches any code-word of the corresponding feature of the feature code-book cb^k. The distance between context c and code-book cb^k is expressed by the number of times where matching code-words are found. This distance is normalized in the interval $[0,\ 1]$.

Table 4.1 presents the algorithm that computes the distance between a context c and a context code-book model CB. The distance $dist(\mu_t^k,\ cw_i^k)$ is defined as in formula 4.21.

4.4 Tracking Parameter Optimization

The objective of the tracking parameter optimization step is to find the values of the control parameters which ensure the best possible tracking quality for each video context chunk. This quality is greater or equal to the threshold ϵ presented in hypothesis 3, section 1.2.2. These parameters are called "best satisfactory parameters". The optimization task is done using the manually annotated trajectories.

This step receives as input the annotated detected objects, a tracking algorithm, a video chunk and a set of control parameters for this tracker $\mathcal{P} = \{p_1,\ p_2, ...,\ p_r\}$. The annotated objects are used as the object detection result. This step gives as output the best satisfactory parameter values. The information of a control parameter i includes its name, value range

$[p_i^{min},\ p_i^{max}]$ and step value, denoted s_i. Based on the number of control parameters and their step values, the size of the search space S is computed as follows :

$$S = \Pi_{i=1}^{r} \frac{p_i^{max} - p_i^{min}}{s_i} \tag{4.34}$$

Depending on the size of the search space of control parameters, we can select a suitable optimization algorithm. In order to represent the reliability of the found parameters, we associate them with two values. The first one is the number of frames of the training video chunk in which mobile objects appear (called "number of training frames"). The second one is a $F\text{-}Score$ value representing the tracking quality of the considered context chunk for the found tracking parameter values. The best satisfactory parameter values, their reliability values and the contextual code-book model corresponding to this training video are stored into a temporary learned database.

In the following sections, we first present the definition of the optimization problem and how to apply the optimization process for finding the best satisfactory values of control parameters. Then we present respectively three optimization algorithms : exhaustive search , genetic and Adaboost algorithms. These algorithms can be used in different cases depending on the size of the search space and the nature of variables (i.e. control parameters).

4.4.1 Optimization Problem

An optimization problem can be represented in the following way. Given a function h mapping a set $X \subseteq \mathbb{R}^n$ to the set of real values $\mathbb{R}$ as follows :

$$h : X \rightarrow \mathbb{R} \tag{4.35}$$

The objective is to find an element $x^* \in X$ such that :

$$\forall x \in \mathbb{R}^n,\ x \neq x^* : h(x^*) \geq h(x) \text{ for the maximization optimization problem} \tag{4.36}$$

or

$$\forall x \in \mathbb{R}^n,\ x \neq x^* : h(x^*) \leq h(x) \text{ for the minimization optimization problem} \tag{4.37}$$

where x is called variable ; x^* is called global solution ; X is called the search space ; and the function h is called the objective function.

In this work, variables x are control parameters. The objective function is defined as the function representing the tracking performance. This function is presented in detail in the following section. Our goal is to find the control parameter values of tracking algorithm for which the tracker performance gets the highest possible value.

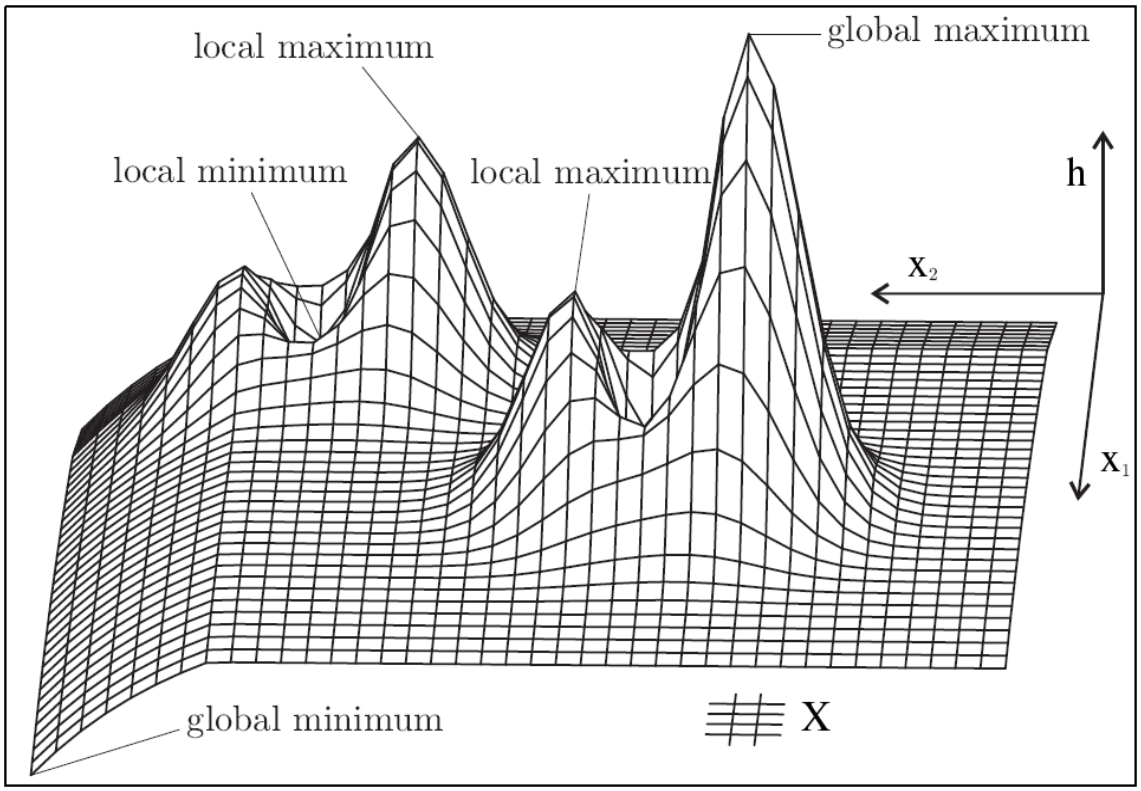


Figure 4.17 – Global and local optimal values of a two-dimensional function (source [T.Weise, 2009])

4.4.2 Objective Function

In the tracking parameter optimization, the objective function is defined as a function mapping parameter values $p \in \mathcal{P}$ to a non-negative real value (representing the tracking performance) as follows :

$$f : \mathcal{P} \rightarrow \mathbb{R}^{+} \tag{4.38}$$

Given parameter values $p \in \mathcal{P}$, we run the tracking algorithm to get the output trajectories. First, some evaluation metrics are defined using trajectories in ground-truth data and in the tracker output as follows

- True Positive (denoted TP) is the number of correct tracked trajectories.
- False Negative (denoted FN) is the number of lost trajectories.
- False Positive (denoted FP) is the number of output erroneous trajectories.
- Sensitivity (denoted S) is the number of trajectories that are successfully tracked divided by ground-truth data :

$$S = \frac{TP}{TP + FN} \tag{4.39}$$

- Precision (denoted P) is the number of trajectories that are successfully tracked divided by number of output trajectories.

$$P = \frac{TP}{TP + FP} \tag{4.40}$$

The tracking performance is defined as the F-$Score$ value in the interval $[0,\ 1]$ as follows :

$$\text{F-}Score = \frac{2 \times P \times S}{P\ +\ S} \tag{4.41}$$

The objective function h is set to the value of F-Score : $h(p) = \text{F-}Score$. We find that the objective function has the two following properties :

- This function has no explicit mathematical form and so is non differentiable.
- Only function evaluations are possible.

With these properties, it is difficult to find the global solution when the search space of parameters is large. Therefore instead of finding the global solution, we search for the best satisfactory parameters which ensure the tracking performance greater or equal to the threshold ϵ presented in hypothesis 3, section 1.2.2.

4.4.3 Optimization Algorithms

There exist many optimization algorithms, for example exhaustive search, particle swarm, genetic, simplex, greedy algorithms. Depending on the size of the parameter search space

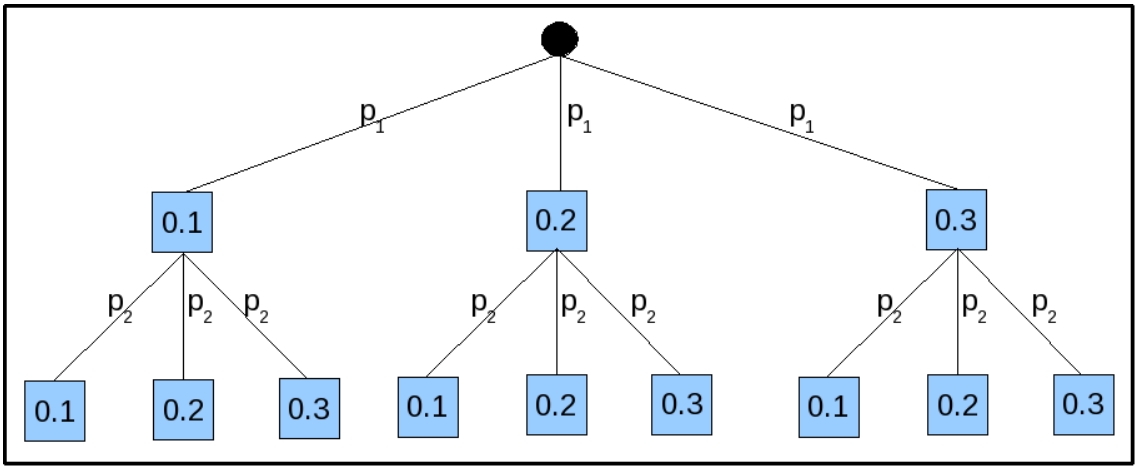


Figure 4.18 – Tree representing the search space of the exhaustive search algorithm

and the nature of parameters for each tracking algorithm, we can select a suitable optimization algorithm. If the searching space of the control parameters is small, an exhaustive search [Nievergelt, 2000] can be used to scan all the values of these parameters. Otherwise, we can use a genetic algorithm [Goldberg, 1989] for searching the satisfactory values. In some cases, an optimization problem can be converted to a classification problem whose objective is to optimize the weights of weak classifiers. In this case, an Adaboost algorithm [Freund and Schapire, 1997] can be used to determine the best values of these weights. In the following sections, these three optimization algorithms are respectively presented.

4.4.3.1 Exhaustive Search Algorithm

The principle of this search technique is to consider all possible combinations of parameters to find the best values. If the range value of a parameter is not discrete, its searching space is indefinite. Therefore, the step value required as input allows to transform the indefinite search space to a definite search space.

For example, we have two find the best satisfactory values of two parameters p_1 and p_2. The range value of p_1 and p_2 is $[0.1,\ 0.3]$, and their step value is equal to 0.1. Their search space can be represented as in figure 4.18. In this problem, we have to consider nine cases to find the best parameter values. The best parameters are only validated if the tracking performance when using these parameters is greater or equal to the threshold ϵ presented in hypothesis 3, section 1.2.2.

0.2	0.1	0	0.15	0.3	0.2	0	0	0.05

Figure 4.19 – Illustration of an individual representing a set of nine parameter values

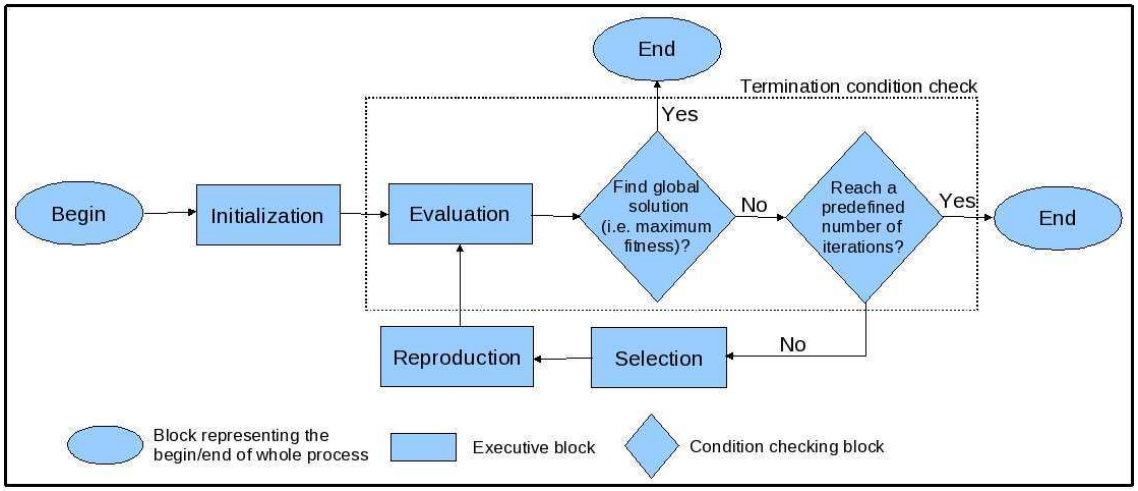


Figure 4.20 – The Flow chart of a genetic algorithm

4.4.3.2 Genetic Algorithm

When the search space of parameters is too large, the exhaustive search algorithm is not efficient due to its high complexity. In this case, a genetic algorithm can be performed. Genetic algorithm (GA) [Goldberg, 1989] is a search heuristic that imitates the process of natural evolution. This heuristic is used to generate useful solutions for optimization and search problems. Genetic algorithms belong to the class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

For a generic algorithm, we need to define the notion of "individual" representing the variable values. In the parameter optimization problem, an individual is defined as a set of parameter values p_i, $i = 1..r$ where r is the number of considered parameters (i.e. control parameters). Figure 4.19 illustrates an "individual" representing a set of nine parameter values.

The objective of this genetic algorithm is to find the best satisfactory parameters which ensure a good enough tracking quality. This algorithm includes four steps : initialization, selection, reproduction and termination. Figure 4.20 presents the flow chart of a genetic algorithm.

1. **Initialization** : We create randomly the first generation of individuals consisting of a great number individuals (e.g. 5000 individuals). Each individual is associated to a fitness value (i.e. quality) which is defined as the F-Score value (see section 4.4.2) of the tracking

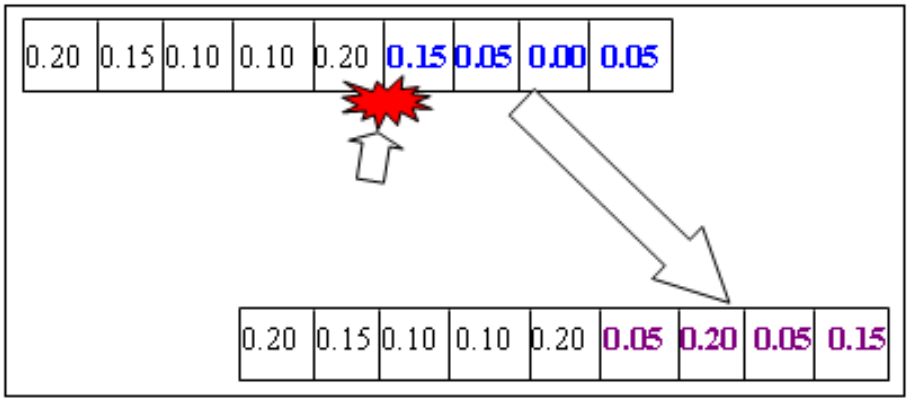


Figure 4.21 – Illustration of the mutation operator in a genetic algorithm

performance corresponding to this individual.

2. **Selection** : For each generation, a predefined proportion of the best individuals (i.e. individuals with the highest fitness) are selected to reproduce new generation.

3. **Reproduction** : For each generation, some genetic operators such as mutation, cross-over (see their descriptions below) are performed to create a new generation. Using these genetic operators, "child" individuals are created which typically shares many of the characteristics of their "parents". The process continues until a new population of appropriate size is generated. For each new individual, we compute its fitness value.
   - **Mutation operator** : This operator takes a "parent" individual as input. A position i, $i = 1..r$ is selected randomly. The parameter values from position i until r locating in this individual change to new random values (see figure 4.21).
   - **Cross-over operator** : This operator requires two "parent" individuals as input. A position i, $i = 1..r$ is selected by random. The parameter values from position i until r locating in each individual are permuted (see figure 4.22). At the end, we have two "child" individuals.

4. **Termination** : This generational process is repeated until a termination condition is reached :
   - There exists an individual whose fitness value is equal to 1 (i.e. find global solution).
   - A fixed number of generations has been reached. The best individual is verified to check if it reaches the objective (i.e. the tracking performance when using these parameters is greater or equal the threshold ϵ presented in hypothesis 3, section 1.2.2). If this verification is satisfied, this individual is given as output of the optimization process. Otherwise, the optimization has no solution.

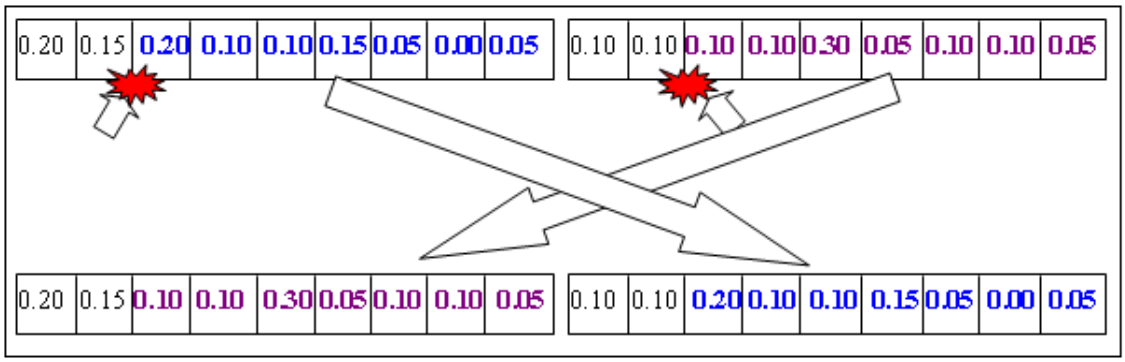


Figure 4.22 – Illustration of the cross-over operator in a genetic algorithm

4.4.3.3 Adaboost Algorithm

A tracking algorithm can play the role as a strong classifier including a set of weak classifiers. The weak classifier weights (i.e. importances) are in general the important parameters. In this case, an Adaboost algorithm can be used to optimize these weight values. This algorithm takes as input the training samples S_i, $i = 1..N$, their labels, weak classifiers $\hat{w}^j$, $j = 1..r$ and gives as output the weight values w_j, $j = 1..r$, of the weak classifiers.

The loss function for Adaboost algorithm at iteration k for each weak classifier $\hat{w}^j$ is defined as :

$$\epsilon_k^j = \sum_{i=1}^{N} D_k(i) max(0, -y_i h_j(i)) \tag{4.42}$$

where y_i is the label of training sample s_i, $h_j(i)$ is its result classification given by weak classifier $\hat{w}^j$ and $D_k(i)$ is its weigh at iteration k.

At each iteration k, the goal is to find $\hat{w}^j$ whose loss function ϵ_k^j is minimal. h_j and ϵ_k^j (corresponding to value j found) are denoted $\hat{h}_k$ and $\hat{\epsilon}_k$. The weight of this weak classifier at iteration k, denoted $\hat{\alpha}_k$, is computed as follows :

$$\hat{\alpha}_k = \frac{1}{2} ln \frac{1 - \hat{\epsilon}_k}{\hat{\epsilon}_k} \tag{4.43}$$

We then update the weight of samples s_i :

$$D_{k+1}(i) = \begin{cases} 1/N & , \quad \text{if } k = 0 \\ \frac{D_k(i) e^{(-\hat{\alpha}_k y_i \hat{h}_k(i))}}{A_z}, & \text{otherwise} \end{cases} \tag{4.44}$$

where A_z is a normalization factor so that $\sum_i^N D_{k+1}(i) = 1$.

The equation 4.44 to compute the distribution D_{k+1} is constructed by knowing that :

$$-\hat{\alpha}_k y_i \hat{h}_k(i) \begin{cases} < 0, & \text{if } y_i = \hat{h}_k(i) \\ > 0, & \text{if } y_i \neq \hat{h}_k(i) \end{cases} \tag{4.45}$$

Thus after selecting an optimal classifier $\hat{h}_k$ for the distribution D_k, the samples i that the classifier $\hat{h}_k$ identified correctly are weighted less and those that are identified incorrectly are weighted more. Therefore, when the algorithm learns the classifiers with the sample weights D_{k+1}, it will select a classifier that better identifies those samples that the previous classifier missed.

The Adaboost algorithm stops when $\hat{\epsilon}_k \geq 0.5$ or a predefined number of iterations K is done. At the end of the Adaboost algorithm, we obtain a list of weight values of selective weak classifiers $\hat{\alpha}_k$, $k = 1..K'$ where K' is the number of iterations done ($K' \leq K$).

The weight values w_j for each weak classifier $\hat{w}^j$ ($j = 1..r$, r is the number of weak classifiers) is computed as follows :

$$w_j = \sum_{k=1}^{K'} \hat{\alpha}'_k(j) \tag{4.46}$$

and $\hat{\alpha}'_k$ is defined as :

$$\hat{\alpha}'_k(j) = \begin{cases} \hat{\alpha}_k & \text{, if weak classifier } j \text{ is selected at iteration } k \\ 0 & \text{, otherwise} \end{cases} \tag{4.47}$$

A strong classifier $\mathcal{S}$ is defined as a weighted combination of weak classifiers as follows :

$$\mathcal{S} = \frac{\sum_{j=1}^{r} w_j h_j}{\sum_{j=1}^{r} w_j} \tag{4.48}$$

The found weight values of weak classifiers are only validated if the tracking performance when using this learned strong classifier is greater or equal the threshold ϵ presented in hypothesis 3, section 1.2.2.

An example of the Adaboost application for parameter optimization problem is described in section 6.1.

4.5 Clustering

In some cases, two similar contexts can have different best satisfactory parameter values because the optimization algorithm only finds local optimal solutions. Moreover, the context of a video sequence is not sufficient for playing the role of a key value for determining the best satisfactory tracking parameter values. A clustering step is thus necessary to group similar contexts and to compute the best satisfactory parameter values for the context clusters. The clustering

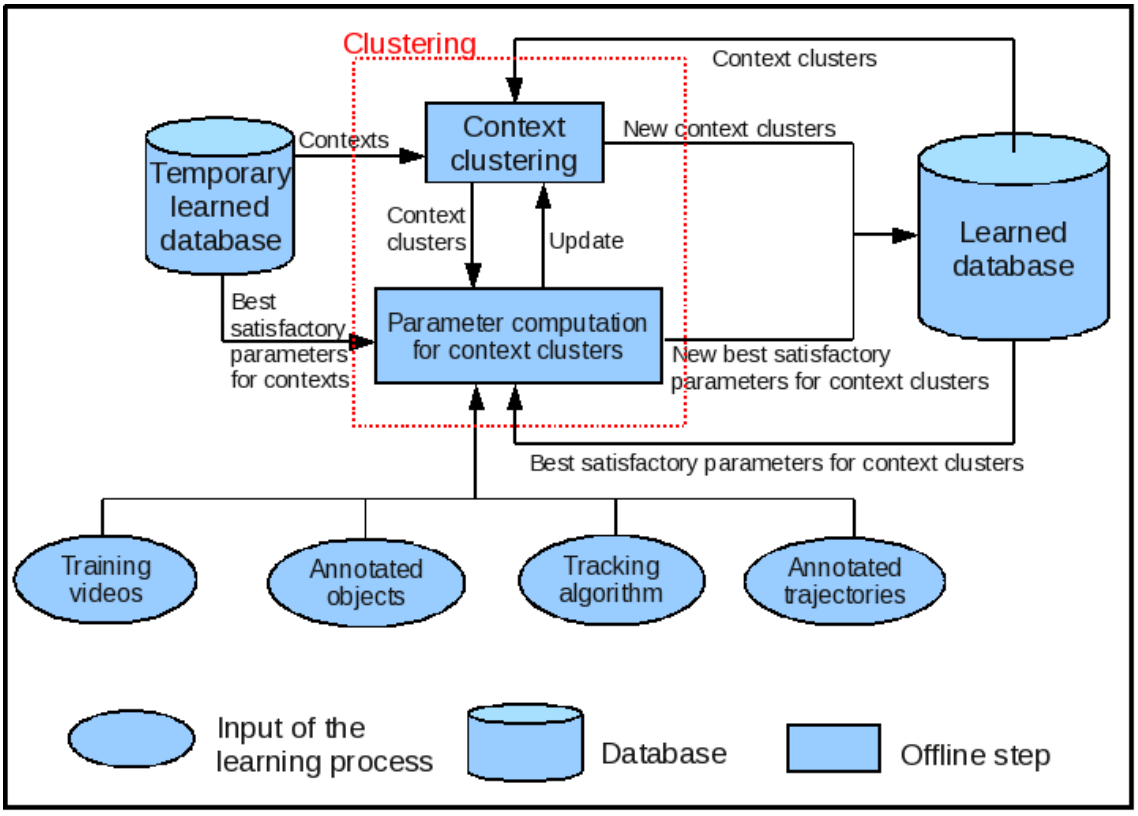


Figure 4.23 – Illustration of the clustering step

step includes two sub-steps : context clustering and parameter computation for context clusters (see figure 4.23).

The clustering step is done at the end of each learning session. This step takes as input the training videos, the annotated objects, tracking algorithm and annotated trajectories. It also requires the data stored in the temporary learned database and in the learned database resulting from the previous learning sessions. These data include a set of contexts or context clusters associated to their best tracking parameter values and the reliability values of these parameters. The data from the learned database is only required as input for the second learning session and the followings. This step gives as output the context clusters which are associated with their best satisfactory parameter values.

In the following, we describe the first learning session when there are only contexts (no context clusters) to cluster. The process of the next learning sessions (with context clusters) are completely similar because the data structure of a context cluster and the one of a context are the same.

```
Input : set of contexts C, diameter threshold d
Output : set of context clusters 𝔈

Procedure Qt_Clust(C, d)

if (|C| <= 1)    output C    // Base case
else
   For each i ∈ C
     set flag = true;
     set A_i = {i};    // A_i is the cluster started by i
     while ((flag == true) && (A_i ≠ C))
       find j ∈ (C \ A_i) such that diameter (A_i ∪ j) is minimum
       if (diameter(A_i ∪ j) > d)
         set flag = false;
       else set A_i = A_i ∪ j;    // Add j to cluster A_i
   identify set 𝔈 ∈ {A_1, A_2, ..., A_|C|} with maximum cardinality
   output 𝔈

   call QT_Clust(C \ 𝔈, d)
```

Table 4.2 – Pseudo-code of algorithm QT clustering (source [Heyer et al., 1999])

4.5.1 Context Clustering

In this work, we decide to use the Quality Threshold Clustering (QT clustering) algorithm [Heyer et al., 1999] for this step due to the following three reasons. First, only clusters that pass a user-defined quality threshold will be returned. Second, this algorithm does not require the number of clusters as input. Third, all possible clusters are considered. However, a diameter threshold is needed to consider whether two contexts can be grouped. This threshold can be estimated by defining the distance metric value between two contexts in the interval $[0,\ 1]$. Table 4.2 describes the pseudo-code of the QT clustering algorithm.

The distance between a context and a context cluster is defined as the complete linkage (i.e. the maximum distance from the context to any context of the cluster) [Everitt et al., 2001] and normalized in the interval $[0,\ 1]$. So the open problem on the application of the above QT-clustering algorithm for the context clustering problem is how to compute the distance between two context code-book models. In the following, we address this issue.

- Context Code-book Model Distance

In order to compute the distance between two context code-books, each code-book of a context is transformed into a histogram whose bin i corresponds to feature value $\overline{\mu_i^k}$ of code-

word i belonging to code-book k, and value of bin i is defined as f_i^k/N where N is the number of training frames of the code-book, f_i^k is the number of frames in which the code-word i is activated (see formula 4.20).

The distance between two code-books is defined as the Earth Mover Distance between the two corresponding histograms in which the ground distance c_{ij} (in formula 4.8) is defined as $|\overline{\mu_i^k} - \overline{\mu_j^k}|$. The distance between two contexts is defined as the mean value of the six distances between the six code-book pairs.

4.5.2 Parameter Computation for Context Clusters

The objective of the "Parameter Computation for Context Clusters" step is to compute the best satisfactory parameter values for the context clusters. This sub-step includes two stages : "Parameter Computation" and "Parameter Verification".

4.5.2.1 Parameter Computation

Once contexts are clustered, all of the code-words of these contexts become the code-words of the created cluster. The satisfactory tracking parameters of cluster j, denoted $\vec{p}^j$ can be computed as follows :

$$\vec{p}^j = \frac{\sum_{i=1}^{|\mathcal{C}_j|} \vec{p}_i \times w^i}{\sum_{i=1}^{|\mathcal{C}_j|} w^i} \tag{4.49}$$

where $|\mathcal{C}_j|$ is the number of contexts belonging to cluster j, $\vec{p}_i$ is the best satisfactory parameter values of context i belonging to this cluster, w^i is the weight (i.e. importance) of parameters $\vec{p}_i$ and is defined in function of two reliability values of $\vec{p}_i$: number of training frames N_i and F-Score$_i$:

$$w^i = \frac{\frac{N_i}{N^j} + \text{F-Score}_i}{2} \tag{4.50}$$

where N^j is the total number of training frames of all contexts belonging to context cluster j.

The values representing the reliability of context cluster j is also defined as follows :

- The number of training frames : N^j

- The tracking quality for these parameters of cluster j, denoted $\overline{\text{F-Score}}^j$, is estimated as follows :

$$\overline{\text{F-Score}}^j = \frac{\sum_{i=1}^{|\mathcal{C}_j|} \text{F-Score}(v_i, \vec{p}_i) N_i}{N^j} \tag{4.51}$$

where F-Score$(v_i, \vec{p}_i)$ is the F-Score for video v_i when using the learned satisfactory parameters $\vec{p}_i$ (computed in formula 4.41).

4.5.2.2 Parameter Verification

The objective of the parameter verification stage is to check whether the parameters for each context cluster resulting from section 4.5.2.1 are satisfactory. In other word, the tracking quality for these parameters has to be verified for all videos of the cluster.

For each cluster, this stage takes all training videos belonging to this cluster and computes the tracking performance with the parameters resulting from section 4.5.2.1. If the obtained tracking performance is greater or equal the one computed by its own satisfactory parameters, these parameters are considered "verified". Otherwise, videos which do not satisfy this condition, are removed from this cluster. They are then stored separately in the learned database. The context cluster and its best satisfactory parameters are re-computed and re-verified.

Finally, at the end of the clustering process, we obtain a set of context clusters represented similarly as a context : a context model of six code-books including code-words, best satisfactory tracking parameter values, number of training frames and tracking quality score $\overline{\text{F-Score}}$.

Figure 4.24 illustrates a record of the learned database.

4.6 Summary

In this chapter, we have presented the different steps of the offline learning phase including contextual feature extraction, context segmentation and code-book modeling, tracking parameter optimization, clustering (including context clustering and parameter computation for context clusters). This phase gives as output a learned database which contains the best satisfactory parameter values of the tracking algorithm for context clusters. These clusters have different values of density, occlusion level of mobile objects, contrasts with regard to the surrounding background, contrast variance, 2D areas and 2D area variance.

The next chapter describes how to use this learned database for tuning online tracking parameters to adapt the tracker performance to the variation of context.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<learned_database version="2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="learned_database.xsd" >

<record id_record = "1">

	<context_cluster number_training_frames="300">
		<code_book id_cb= "1" name = "density_objects">
»			<code_work id_cw ="1" mean = "0.07" min = "0.06" max = "0.11" frequence = "216" />
»			<code_work id_cw ="2" mean = "0.15" min = "0.12" max = "0.21" frequence = "84" />
		</code_book>

		<code_book id_cb= "2" name = "occlusion_level">
»			<code_work id_cw ="1" mean = "0.00" min = "0.00" max = "0.00" frequence = "246" />
»			<code_work id_cw ="2" mean = "0.17" min = "0.12" max = "0.21" frequence = "15" />
»			<code_work id_cw ="3" mean = "0.28" min = "0.23" max = "0.34" frequence = "16" />
		</code_book>

		<code_book id_cb= "3" name = "object_contrast">
»			<code_work id_cw ="1" mean = "0.29" min = "0.20" max = "0.39" frequence = "300" />
		</code_book>

		<code_book id_cb= "4" name = "object_contrast_variance">
»			<code_work id_cw ="1" mean = "0.08" min = "0.05" max = "0.10" frequence = "131" />
»			<code_work id_cw ="2" mean = "0.11" min = "0.10" max = "0.18" frequence = "127" />
		</code_book>

		<code_book id_cb= "5" name = "object_2D_area">
»			<code_work id_cw ="1" mean = "3170.12" min = "2168.67" max = "4149" frequence = "278" />
		</code_book>

		<code_book id_cb= "6" name = "object_2D_area_variance">
»			<code_work id_cw ="1" mean = "2971.74" min = "2122.70" max = "4221.43" frequence = "91" />
»			<code_work id_cw ="2" mean = "2020.10" min = "1618.41" max = "2633.30" frequence = "129" />
»			<code_work id_cw ="3" mean = "4715.35" min = "4287.90" max = "5122.99" frequence = "80" />
		</code_book>
	</context_cluster>

	<best_satisfactory_parameters number_control_parameters = "6" F-score = "0.9" >
		<parameter_value value = "0.00"  />
		<parameter_value value = "0.21"  />
		<parameter_value value = "0.45"  />
		<parameter_value value = "0.00"  />
		<parameter_value value = "0.00"  />
		<parameter_value value = "0.34"  />
	</best_satisfactory_parameters>

</record>

<record id_record = "2">
....
</record>

</learned_database>
```

Figure 4.24 – A record of the learned database represented by XML

5

ONLINE PARAMETER TUNING

In this chapter, we describe the proposed controller which aims at tuning online the tracking parameter values for obtaining the satisfactory tracking performance. This controller includes two stages : initial parameter configuration and parameter adaptation. The objective of the first stage is to set values for control parameters (i.e. parameters of tracking algorithm) at the first frames and for parameters of the proposed controller. The second stage is responsible for tuning and learning the control parameters over time to adapt them to the tracking context variation.

5.1 Initial Parameter Configuration

5.1.1 Control Parameters

Control parameters are the parameters of a tracking algorithm which are considered in the control process (i.e. to look for the best satisfactory values in the learning phase and to be tuned in the online phase). At the first frames, as the controller does not have enough information related to the tracking context, it cannot tune the control parameters. These values are set by the user preference values or by default values. Therefore, for the first frames, the tracking quality can be poor. This period lasts until when the tuning of the parameter starts to be activated (e.g at the 50^{th} frame). This duration is in general quite short compared to the size of a normal video sequence (of thousands frames), so a low tracking quality within this period is acceptable.

5.1.2 Controller Parameters

The proposed controller has no important parameters for tuning. This controller has two important values. The first value is a threshold Th_1 representing the maximal distance between

a context and its cluster (see sections 5.2.1.1 and 5.2.1.2). The second value is a number of frames l for which context is computed to find the suitable tracking parameters (see sections 5.2.1.1, 5.2.1.2, 5.2.2.1 and 5.2.2.2). These two values have already been determined in the offline learning phase (in section 4.3.1).

In the online tracking evaluation algorithm (section 5.2.2.2), there exists some thresholds for tuning (Th_6, Th_7 in formula 5.13). These thresholds can be computed using a learning algorithm or estimated by experience. They are quite independent on tracking contexts and can be set empirically. We will discuss them in detail in section 5.2.2.2.

5.2 Parameter Adaptation

In the parameter adaptation, we propose two approaches called “context-based parameter adaptation” and “evaluation-based parameter adaptation”. While the first approach only relies on the information of the current video context to tune the tracking parameters (see figure 5.1), the second approach adds an online tracking evaluation task and a clustering task to support the online control process (see figure 5.3).

5.2.1 Context-based Parameter Adaptation Approach

The goal of the context-based parameter adaptation approach is to detect changes of tracking context and to tune parameter values for coping with these changes. This approach takes as input the video stream, the list of detected objects at every frame, the learned database and gives as output the adaptive tracking parameter values for every new context detected in the video stream (see figure 5.1).

In the following sections, we describe the steps of this approach including the context detection and parameter tuning steps.

5.2.1.1 Context Detection

An open problem is to detect online the variation of the tracking context. As presented in the previous chapter, the tracking context defined in this work is represented by a set of six features : density, occlusion of mobile objects appearing within the considered video sequence, their contrast with regard to the surrounding background, their contrast variance, their 2D area and their 2D area variance. These values are computed from the result of the object detection task. In complex cases such as object occlusions, strong or low illumination intensity, the detection quality can decrease significantly. Moreover, in some cases, due to the mobile object locations, some wrong detections can happen within a small number of frames. Figure 5.2 illustrates such case. This figure represents the object detection algorithm output

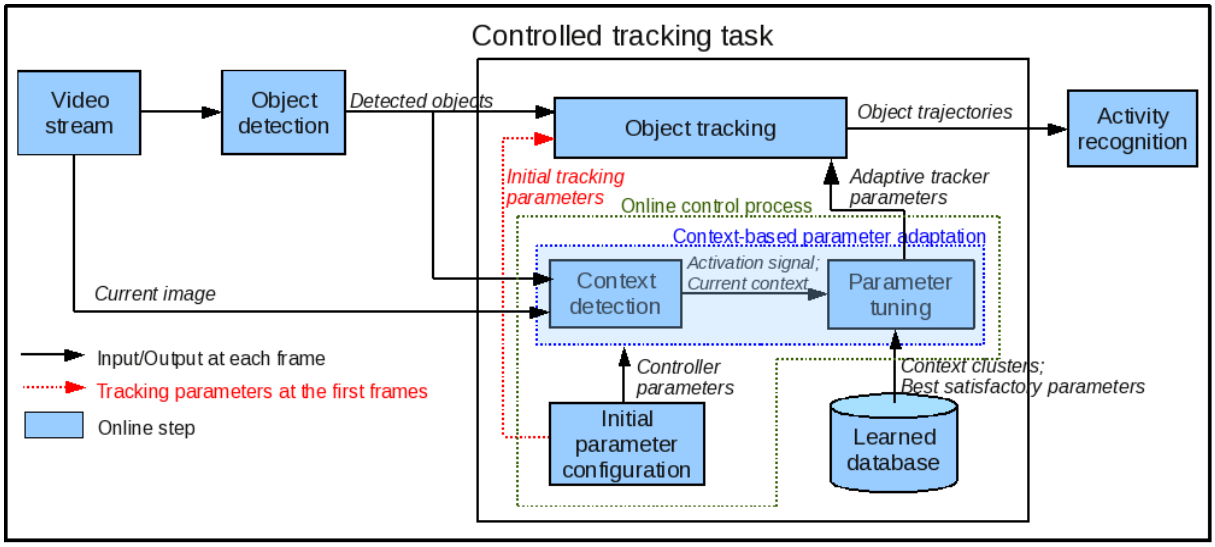


Figure 5.1 – Illustration of the controlled tracking task with the context-based parameter adaptation approach

of [Corvee and Bremond, 2010]. On the left image, at frame 279, the two persons on the left are wrongly detected as a large person because the head of the top person and the foot of the bottom person are considered as belonging to a same person. On the right image, at frame 335, due to a change in the relative position of these two persons, they are now correctly detected. Therefore, in order to detect the context at current time, we need to collect values of the contextual features in a large enough number of frames (for example 50 frames). However if this value is too large, the contextual variation is slowly detected and thus decreases the speed of the parameter adaptation.

This step takes as input for every frame the list of the current detected objects and the processing image.

For each video chunk of l frames, we compute the values of the contextual features. A contextual change is detected when the context of the current video chunk does not belong to the context cluster (clusters are learned in the offline phase) of the previous video chunk. In order to ensure the coherence between the learning phase and the testing phase, we use the same distance defined in the learning phase (section 4.3.3) to perform this classification. If this distance is lower than a threshold Th_1, this context is considered as belonging to the context cluster. The threshold Th_1 (presented in section 4.3.1) represents the maximal distance between a context and its cluster.

In the following section, we recall the definition of the context distance.

- Context Distance

Figure 5.2 – Illustration of the influence of the object detection quality (produced by approach [Corvee and Bremond, 2010]) due to object location : **a. Left image :** at frame 279, the two persons on the left are wrongly detected because the head of the top person and the foot of the bottom person are considered as belonging to a same person **b. Right image :** at frame 335, the relative position of these two persons is changed, and they are correctly detected.

The context distance is defined to compute the distance between a context c and a context code-book model $CB = \{cb^k,\ k = 1..6\}$.

The context c of a video chunk (of l frames) is represented by a set of six values : the density, occlusion level of mobile objects, their contrast with regard to the surrounding background, their contrast variance, their 2D area and their 2D area variance over time. For each context feature k ($k = 1..6$), the contextual feature value at instant t is denoted μ_t^k. For each such value, we consider whether it matches any code-word of the corresponding feature of the feature code-book cb^k. The distance between context c and code-book cb^k is expressed by the number of times where matching code-words are found. This distance is normalized in the interval $[0,\ 1]$.

Table 5.1 presents the algorithm that computes the distance between a context c and a context code-book model CB. The distance $dist(v_t^k,\ cw_i^k)$ is defined as in formula 4.21.

5.2.1.2 Parameter Adaptation

The parameter adaptation process receives as input the activation signal and the current context from the "context detection" and gives as output adaptive tracking parameter values. When this process receives an activation signal, it looks for the cluster in the learned database the current context belongs to. Let $\mathfrak{D}$ represent the learned database, a context c of a video chunk of l frames belongs to a cluster C_i if both conditions are satisfied :

$$contextDist(c,\ C_i,\ l) < Th_1 \tag{5.1}$$

```
function contextDist(c, CB, l)

Input : context code-book model CB, context c, l (number of frames of context c)
Output : context distance between cluster CB and context c

countTotal = 0;
For each code-book cb^k in CB (k = 1..6)
   count = 0;
   For each value μ_l^k of context c
      For each codeword cw_i^k in code-book cb^k
         if (dist(μ_l^k, cw_i^k) < 0.5) {
            count++;
            break;
         }
   if (count / l < 0.5) return 1;
   countTotal += count;

return ( 1 − countTotal/(l ∗ 6) )
```

Table 5.1 – Function for computing the distance between a context code-book CB and a video context c

$$\begin{aligned} &\forall C_j \in \mathfrak{D}, j \neq i : \\ &contextDist(c, C_i, l) \leq contextDist(c, C_j, l) \end{aligned} \tag{5.2}$$

where Th_1 is defined in section 5.2.1.1. The function $contextDist(c, C_i, l)$ is defined in table 5.1. If we can find a such context cluster C_i, the current tracking parameters are set to the values of the best satisfactory tracking parameters associated with C_i. Otherwise, the tracking algorithm parameters do not change.

5.2.2 Evaluation-based Parameter Adaptation Approach

The evaluation-based parameter adaptation approach takes as input a video stream, a list of detected objects at every frame, the learned database, the tracking result, a list of scene exit zones and the tracking parameter values at every frame. The learned database is the result of the offline learning process but in this approach, this base can be updated online with an unsupervised learning process. The output of this approach is the set of parameter values corresponding to the video contexts detected over time and the updated learned database (see figure 5.3).

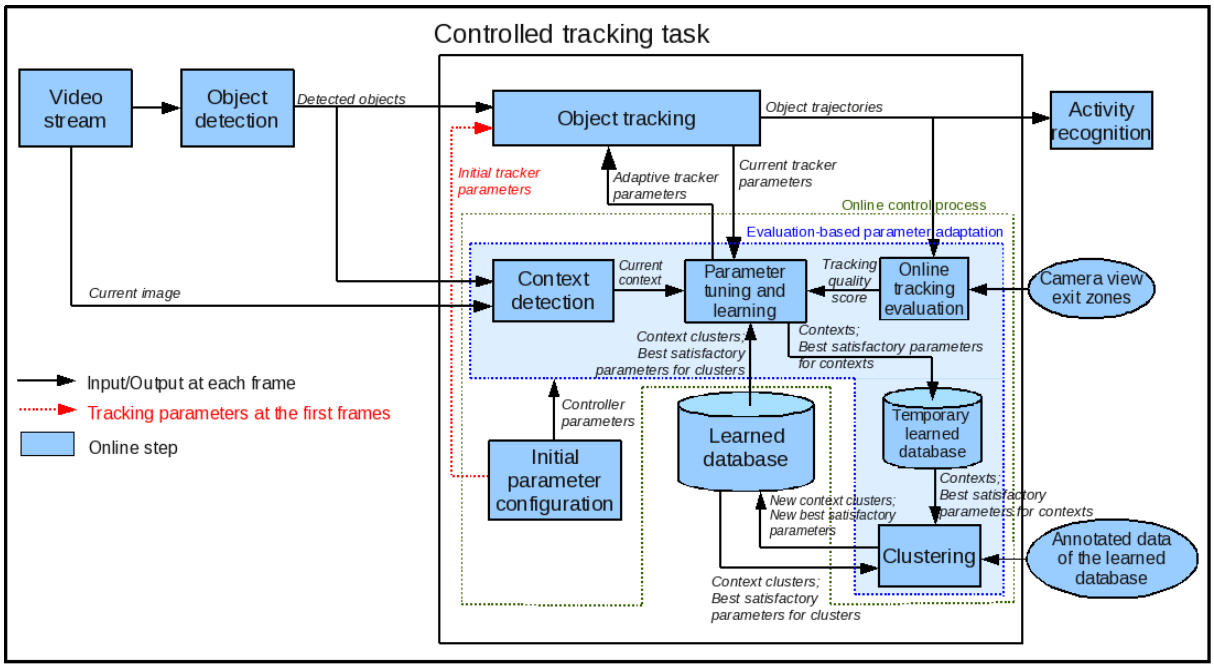


Figure 5.3 – Illustration of the controlled tracking task with the evaluation-based parameter adaptation approach

In the following sections, we describe the approach as well as the execution steps including context detection, online tracking evaluation, parameter tuning and learning, and clustering.

5.2.2.1 Context Detection

The definition of the context detection step is similar to the one defined in the context-based parameter adaptation approach (see section 5.2.1.1 for more details).

This process takes as input for every frame a list of the current detected objects and the current image. For each video chunk of denoted l frames, we compute the values of the contextual features. A contextual change is detected when the context of the current video chunk does not belong to the context cluster (clusters are learned in the offline phase) of the previous video chunk.

5.2.2.2 Online Tracking Evaluation

The objective of the online tracking evaluation step is to evaluate online the tracking algorithm performance without ground-truth data. This process receives as input the object trajectories, the exit zones of the camera view and gives as output for every l frames (l is presented

in section 5.2.1.1) a score corresponding to the tracking quality which is normalized in the interval $[0,\ 1]$. This score is high (e.g. greater than 0.5) when the considered tracking quality is good, and is low (e.g. lower than 0.5) when the tracker fails.

In order to reach this objective, we aim to extract trajectory features which best discriminate good trajectories from erroneous trajectories. For each feature, we define a method to compute a score in the interval $[0,\ 1]$ to determine whether the mobile object is correctly tracked or not. The quality of a trajectory is estimated by the combination of these scores. Seven evaluation features, indexed from 1 to 7, are proposed. These evaluation features are divided into two groups : "trajectory features" and "instantaneous features". While "trajectory features" are features that can only be computed over time once, "instantaneous features" must be computed for each frame during the object tracking time. The two groups share the same mechanism for decreasing their scores (in function of detected errors). However the increasing mechanisms are different. In order to estimate the correctness of a trajectory, we need to consider all information of this trajectory (i.e. from the first frame to the end frame). For "trajectory features", we can decide whether the object trajectories are correct because they are finished. However for ""instantaneous features" we cannot. Therefore, the conditions for increasing the scores corresponding to the "instantaneous features" group need to be more strict than the ones corresponding to the "trajectory features" group.

An online tracking evaluation score is calculated by combining these feature evaluation scores to estimate the quality of the tracking algorithm.

- Evaluation Features for Tracking Algorithm Performance

At the first frame, as we do not know whether the tracking quality is good or not, the evaluation scores for all features are initialized with value of 0.5 (the average score).

+ Trajectory Features

There are two trajectory features in this group : temporal length and exit zone (indexed 1 and 2). The score of a feature i ($i = 1, 2$) in this group is calculated at time instant t ($t > 0$) as follows :

$$S_t^i = \begin{cases} 0 & \text{if } S_{t-1}^i + \gamma^i a_t^i - \gamma^i b_t^i < 0 \\ S_{t-1}^i + \gamma^i a_t^i - \gamma^i b_t^i & \text{if } 0 < S_{t-1}^i + \gamma^i a_t^i - \gamma^i b_t^i < 1 \\ 1 & \text{if } S_{t-1}^i + \gamma^i a_t^i - \gamma^i b_t^i > 1 \end{cases} \tag{5.3}$$

where a_t^i and b_t^i are respectively the number of mobile objects that are correctly and wrongly tracked according to the feature i at time instant t ; γ^i is the cooling factor for increasing and decreasing the score of feature i.

Figure 5.4 – Illustration of a noisy trajectory and a correct trajectory

Each feature has its own computation method for the two values a_t^i and b_t^i. The method which determines these values is described in each trajectory feature section. The first and third line of the above formula enforce the trajectory feature scores to be in the interval $[0,\ 1]$.

1. Temporal length : The temporal length is defined as the number of frames for which a mobile object exists in the scene. If an object appears only in a scene for a very short period of time, this object is likely to be a noise (e.g. due to segmentation errors). Figure 5.4 illustrates a person trajectory and a noisy trajectory due to object detection error.

The score at time instant t of this feature is calculated using the formula 5.3 (with $i = 1$) where a_t^1 is the number of mobile objects satisfying the two conditions (correct length) :

– These mobile objects are no more tracked after time instant t.

– Their temporal lengths are greater or equal to a predefined threshold Th_2.

and b_t^1 is the number of mobile objects satisfying the two conditions (wrong length) :

– These mobile objects are no more tracked after time instant t.

– Their temporal lengths are lower than the threshold Th_2.

The threshold Th_2 represents the minimal length (by number of frames) of an object trajectory. This value depends on the frame rate of the processing video stream. In general, an object should exist in the scene at least one second. So the value of Th_2 is set to $\mathcal{F}$ where $\mathcal{F}$ is the frame rate of the processing video stream.

2. Exit zone : An exit zone is defined as the zone where the mobile objects are supposed to leave the scene. For each scene, we determine manually the exit zones or we can learn them

Figure 5.5 – The red areas show the exit zones.

using the object trajectories [Chau et al., 2009a]. Figure 5.5 illustrates exit zones (marked by red areas) which have been manually annotated in a subway station. When an object trajectory does not end in an exit zone, we consider this object trajectory to be of poor quality.

The score S_t^2 of exit zone feature at time instant t is calculated using formula (5.3) (with $i = 2$) where a_t^2 is the number of mobile objects disappearing in an exit zone at instant t (correct exit zone) ; and b_t^2 is the number of mobile objects disappearing at a location which does not belong to the exit zones at instant t.

- Instantaneous Features

There are five features in this group (indexed from 3 to 7). As we mentioned above, in order to evaluate the tracking quality of an object trajectory, we need to consider the whole trajectory and not only at some frames. Therefore the conditions to increase the scores belonging to this group is more strict than the ones belonging to the "trajectory features" group. For the decreasing mechanism, as the proposed evaluation score aims at penalizing strictly the tracking errors, we keep the same principle to decrease the feature evaluation scores as the one of the first feature group. Therefore, the evaluation score of a feature i ($i = 3..7$) in this group is calculated at time instant t ($t > 0$) as follows :

$$S_t^i = \begin{cases} 0 & \text{if } S_{t-1}^i + \gamma_1^i \delta_{0b_t^i} - \gamma_2^i b_t^i < 0 \\ S_{t-1}^i + \gamma_1^i \delta_{0b_t^i} - \gamma_2^i b_t^i & \text{if } 0 < S_{t-1}^i + \gamma_1^i \delta_{0b_t^i} - \gamma_2^i b_t^i < 1 \\ 1 & \text{if } S_{t-1}^i + \gamma_1^i \delta_{0b_t^i} - \gamma_2^i b_t^i > 1 \end{cases} \tag{5.4}$$

where b_t^i is the number of mobile objects that are wrongly tracked according to the feature i at time instant t ; γ_1^i and γ_2^i are respectively the cooling factor for increasing and decreasing the evaluation score of feature i ; and $\delta_{0b_t^i}$ is the Kronecker delta of 0 and b_t^i :

$$\delta_{0b_t^i} = \begin{cases} 1 & \text{if } b_t^i = 0; \\ 0 & otherwise \end{cases} \tag{5.5}$$

From formula 5.4, for a feature i, we can find that the value of S_t^i only increases when there is no error related to this feature at time instant t (i.e. $b_t^i = 0$). Similar to previous features, each feature in this group has also its own computation method for the value b_t^i.

3. Shape ratio : The shape ratio of a mobile object i at time instant t, denoted s_t^i, is defined as follows : $s_t^i = \frac{W_t^i}{H_t^i}$ where W_t^i and H_t^i are its 2D bounding box width and height. If the shape of a mobile object undergoes large variations, the tracking quality becomes poor. The score S_t^3 of shape feature at time instant t is calculated using formula 5.4 (with $i = 3$) where b_t^3 is the number of mobile objects i satisfying the following condition :

$$\frac{min(s_{t-1}^i,\ s_t^i)}{max(s_{t-1}^i,\ s_t^i)} < Th_3 \tag{5.6}$$

where Th_3 is a threshold representing the minimal shape ratio similarity of a mobile object. The value of Th_3 depends on the frame rate of the video stream, the object speed, the direction of object movement with regard to the camera view and the object posture variation (i.e. the object rigidity). In all of these elements, only the video frame rate is fixed for a video stream, the other elements depend on each mobile objects appearing in the video (e.g. the speed of a running car is faster a walking person). Because this threshold is set for the whole video stream and not for each mobile object, the value of Th_3 is defined in function of the video stream frame rate $\mathcal{F}$ as follows :

$$Th_3 = \begin{cases} 0.8 & \text{if } \mathcal{F} \geq 5; \\ 0.6 & otherwise \end{cases} \tag{5.7}$$

4. Area : The area of a mobile object i at time instant t, denoted a_t^i, is defined as follows : $a_t^i = W_t^i H_t^i$ where W_t^i and H_t^i are respectively the 2D width and height of object i at time t. If there is a large variation of area, the quality of tracking algorithm is not good. The score S_t^4 of area feature at time instant t is calculated as the formula (5.4) (with $i = 4$) where b_t^4 is the number of mobile objects satisfying the following condition :

$$\frac{min(a_{t-1}^i,\ a_t^i)}{max(a_{t-1}^i,\ a_t^i)} < Th_4 \tag{5.8}$$

where Th_4 is a threshold representing the minimal area similarity of a mobile object. Similar to the threshold Th_3, the threshold Th_4 also depends on the frame rate of the video stream, the

object speed and the direction of object movement with regard to the camera view. Because this threshold is set for whole video stream and not for each mobile object, the value of Th_4 is defined as follows :

$$Th_4 = \begin{cases} 0.8 & \text{if } \mathcal{F} \geq 5; \\ 0.6 & \text{otherwise} \end{cases} \tag{5.9}$$

where $\mathcal{F}$ is the frame rate of the video stream.

5. Displacement distance : The displacement distance of a mobile object at time instant t is defined as the displacement of the object from the previous frame $t-1$ to the current one. When a mobile object is wrongly tracked, this value could increase abnormally. In order to detect that, we associate a score S_t^5 to the displacement distance feature at time instant t using formula 5.4 (with $i = 5$) where b_t^5 is the number of mobile objects having a displacement greater than a threshold Th_5 (in meter). The determination of this threshold value is based on the possible maximum speed of mobile objects and the frame rate of the analysed video. However, because the segmentation phase is done in the 2D image, a small error in this phase (due to object shadow for example) can cause a big variation in the corresponding 3D coordinate system. Therefore, the threshold value Th_5 has to take into account this potential error. In one word, we propose a formula to compute the value Th_5 as follows :

$$Th_5 = \theta \frac{V}{\mathcal{F}} \tag{5.10}$$

where V is the maximum speed of the considered mobile objects (in m/s), $\mathcal{F}$ is the frame rate of the video stream, and θ is a cooling factor ($\theta \geq 1$) taking into account the errors of the object detection and of the scene calibration.

6. Color Histogram : In this work, the color histogram of a mobile object is characterized by a normalized histogram of moving pixels inside the bounding box. We extract R, G, B values from color pixels and represent them in three histograms of b bins. Other color features (e.g. MSER) can be used but this color histogram gives good enough results. The distance between two histograms is computed using the Earth Mover Distance (presented in section 4.2.3). Greater this distance is, lower the color similarity is. An object color usually remains similar. Therefore, for each mobile object, we compute the histogram distance between their appearances in two consecutive frames. The tracking quality of an object can be poor if its color distance is large enough over time.

The evaluation score S_t^6 of the color feature at instant t is calculated by formula (5.4) (with

$i = 6$) where b_t^6 is the number of objects i whose color distance $d_t(i)$ is high enough (e.g. greater than 0.5). $d_t(i)$ is defined as follows :

$$d_t(i) = \frac{\sum_{k=R,G,B} EMD(H_{t-1}^{k,i}(i),\ H_t^{k,i})}{3} \tag{5.11}$$

where $H_{t-1}^{k,i}$ and $H_t^{k,i}$ are respectively the histogram of object i at instant $t-1$ and t in k channel ($k = R,\ G,\ B$).

7. Direction : In general, the main trajectory direction of a mobile object does not change, or changes smoothly. In other words, it does not change suddenly during the tracking of the object. So this is an important feature to detect errors of tracking algorithms. In our work, the coordinate system of the ground plane is used to calculate the mobile direction. The angle value of the movement direction is quite sensitive because the trajectory is never a complete line. In order to estimate correctly the direction of movement, we need to observe the mobile object in a long enough temporal interval Δt (e.g. $\Delta t = 30$ frames). Therefore, in this work the direction of a mobile object is represented by the angle α of the 2D vector formed by the object location at instant t and $t-\Delta t$. The value of this angle is in the interval $[0,\ 2\pi]$. In order to know whether a mobile object changes its direction, after each Δt frames we calculate $\Delta\alpha_t$ as follows :

$$\forall t,\ t \bmod \Delta t\ = 0\ ,\ \Delta\alpha_t = \begin{cases} 0 & \text{if } t < \Delta t \\ |\alpha_t - \alpha_{t-\Delta t}| & \text{if } t \geq \Delta t \end{cases} \tag{5.12}$$

Moreover, the direction of object trajectories is only considered when they move in a large enough distance. Therefore the direction of an object i is assumed "changed" if a this object satisfies the two following conditions :

$$\begin{aligned} &\Delta\alpha_t \geq Th_6 \\ &D_{t-\Delta t,t} \geq Th_7 \end{aligned} \tag{5.13}$$

where $D_{t-\Delta t,t}$ is the distance with which object i moves during interval $[t-\Delta t, t]$; Th_6 is a threshold representing the maximal direction variation of the object trajectories (e.g. $\frac{\pi}{2}$) ; Th_7 is a threshold representing the minimum 3D displacement distance of an object to consider the change of its direction (e.g. 1 m).

Let b_t^7 be the number of mobile objects satisfying the condition (5.13). A score S_t^7 of the direction feature at time instant t is calculated using formula 5.4 (with $i = 7$).

- Online Tracking Evaluation Score

Using the seven features we have described above, an online tracking evaluation score is defined to estimate online the quality of a tracking algorithm at every frame. This score is defined as a weighted combination of these features as follows :

$$ES_t = \frac{\sum_{i=1}^{7} w^i S_t^i}{\sum_{i=1}^{7} w^i} \tag{5.14}$$

where ES_t is the online tracking evaluation score at instant t ; w^i is the weight (importance) of feature i and S_t^i is the score of evaluation feature i at instant t. The values of these weights can be computed using an unsupervised learning algorithm. For example, in section 6.1.3, we present a supervised learning algorithm using Adaboost [Freund and Schapire, 1997] to compute the weights of the proposed object tracking descriptors. We can use the same idea to learn the weights of these evaluation features.

The value of the online tracking evaluation score is always in the interval $[0, 1]$ because the evaluation score values of all the features (i.e. trajectory features and instantaneous features) also vary in this interval. The tracking quality of a video chunk of l frames is defined as the mean value of all the online tracking evaluation scores of l frames belonging to this video chunk.

When the online evaluation score is satisfactory (e.g. greater than 0.5), we can say that the tracking algorithm performance is rather good. Whereas if the value of this score is not satisfactory (e.g. lower than 0.5), that means the tracker generally fails to track accurately the detected objects.

5.2.2.3 Parameter Tuning and Learning

The objective of the parameter tuning and learning step is to find the best satisfactory parameter values which are adapted to the current video context and to update the learned database using an unsupervised learning process. For each predefined number of l frames, (this number is equal to the one defined in the context detection and the online tracking evaluation steps), this process receives as input the current video context from the "context detection" step and the tracking quality score from the "online tracking evaluation" step. In the "context-based parameter adaptation" approach, the "parameter tuning" is only performed when it receives an activation signal from the "context detection" step. In this approach, the "parameter tuning and learning" step is always performed. The output of this step is adaptive tracking parameter values and the updated learned database. Figure 5.6 presents the whole process.

The blocks presented in figure 5.6 are described as follows :

- **"Tracking quality is good ?"** : If the tracking quality score is high enough (i.e. greater than 0.5), the answer of this question is "yes", otherwise it gives "no".

- **"Context has changed ?"** : The answer of this question comes from the "context detection" step. A context is considered as "changed" if it does not belong to the cluster of the context

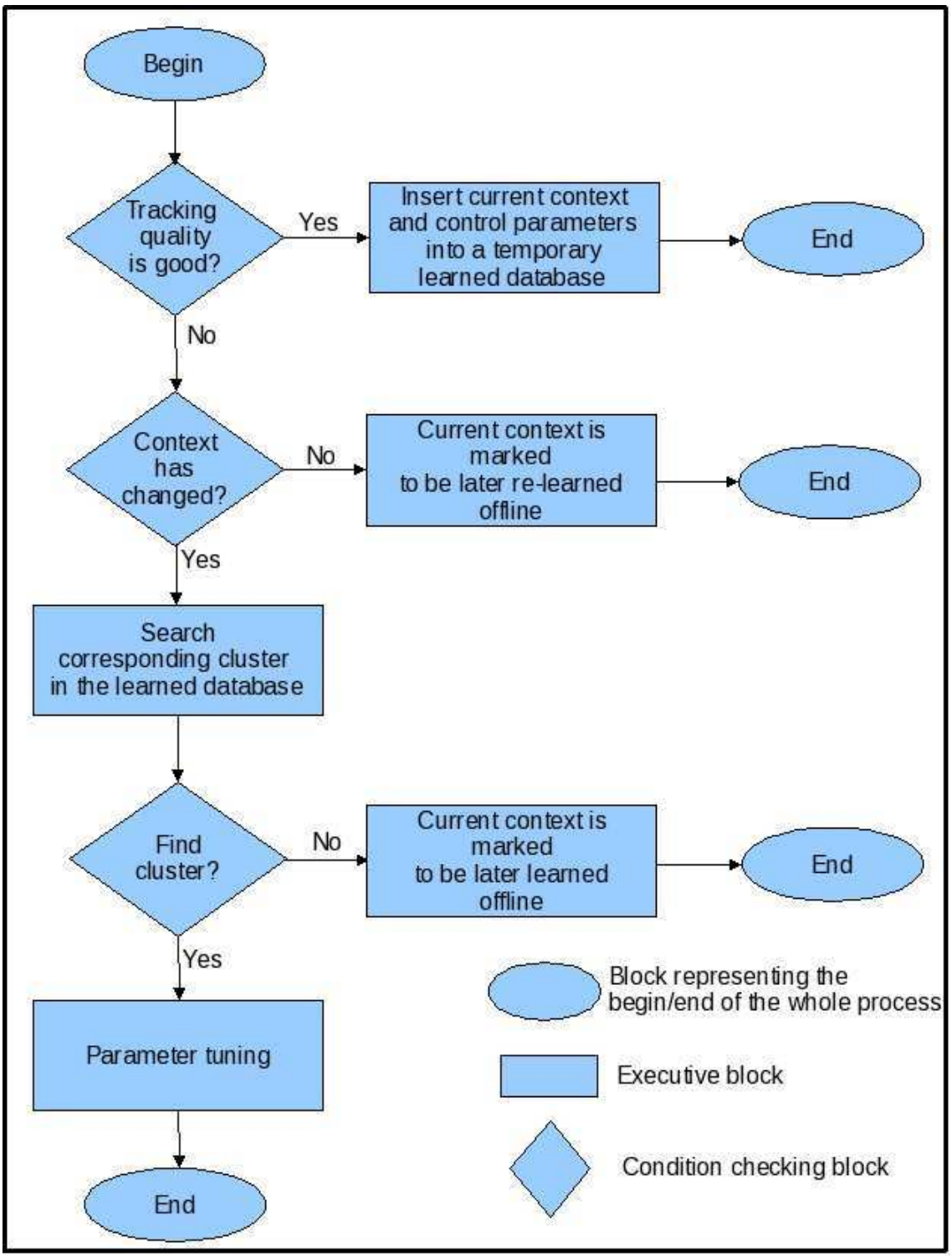


Figure 5.6 – Flow chart of the “parameter tuning and learning” step

corresponding to the previous video chunk.

- **"Search corresponding cluster in the learned database"** : We use the algorithm described in table 5.1 to compute the distance between the context of the current video chunk and a context cluster in the learned database. Then conditions presented in the formulas 5.1 and 5.2 are applied to decide to which cluster the current context belongs.

- **"Parameter tuning"** : The "parameter tuning" task is called when the best satisfactory tracking parameters are found. This task sets the found parameter values to the control parameters of the tracking algorithm.

- **"Insert current context and parameters into a temporary learned database"** : When this task is activated, it means that the current tracking quality is good enough. The current context and tracking parameters are inserted into a temporary learned database. Even if this context and these parameters are already learned before, this process is needed to increase the number of training frames (i.e. increase the reliability) of these parameters in the learned database. First, the contextual feature values are represented in the form of a model of six code-books (see section 4.3.2). Second, the code-book model and the current tracking parameters are inserted into a temporary learned database. A clustering process is performed later to merge information from this temporary learned database and the learned database.

- **"Current context is marked to be later re-learned offline"** : This task is activated if the learned optimal parameters does not give a good enough tracking quality. This can be due to one of two following causes. First, the number of training frames for learning the optimal parameters for this context was not sufficient. Second, the considered tracking algorithm cannot handle this context. The second cause is not considered because it does not fit the hypothesis 3 presented in the first chapter. In order to solve the first cause, in the further offline learning session, we could search for videos which have the same context to learn the tracking parameters. We can also use the current video chunk for the learning step.

- **"Current context is marked to be later learned offline"** : This task is activated if the current processing context is not found in the learned database. Similar to the task described above, in the next offline learning session, we could search for videos which have the same context to learn the tracking parameters. The video chunk corresponding to the current processing context could also be used for learning.

5.2.2.4 Clustering

In the evaluation-based parameter adaptation approach, a temporary learned database is created to store the detected contexts and their best satisfactory tracking parameters. Therefore a clustering process is needed to merge the learned data from this temporary database and the official database (i.e. the database resulting from the offline learning phase). However, this clustering can increase the processing time of the online control phase. Therefore, we only

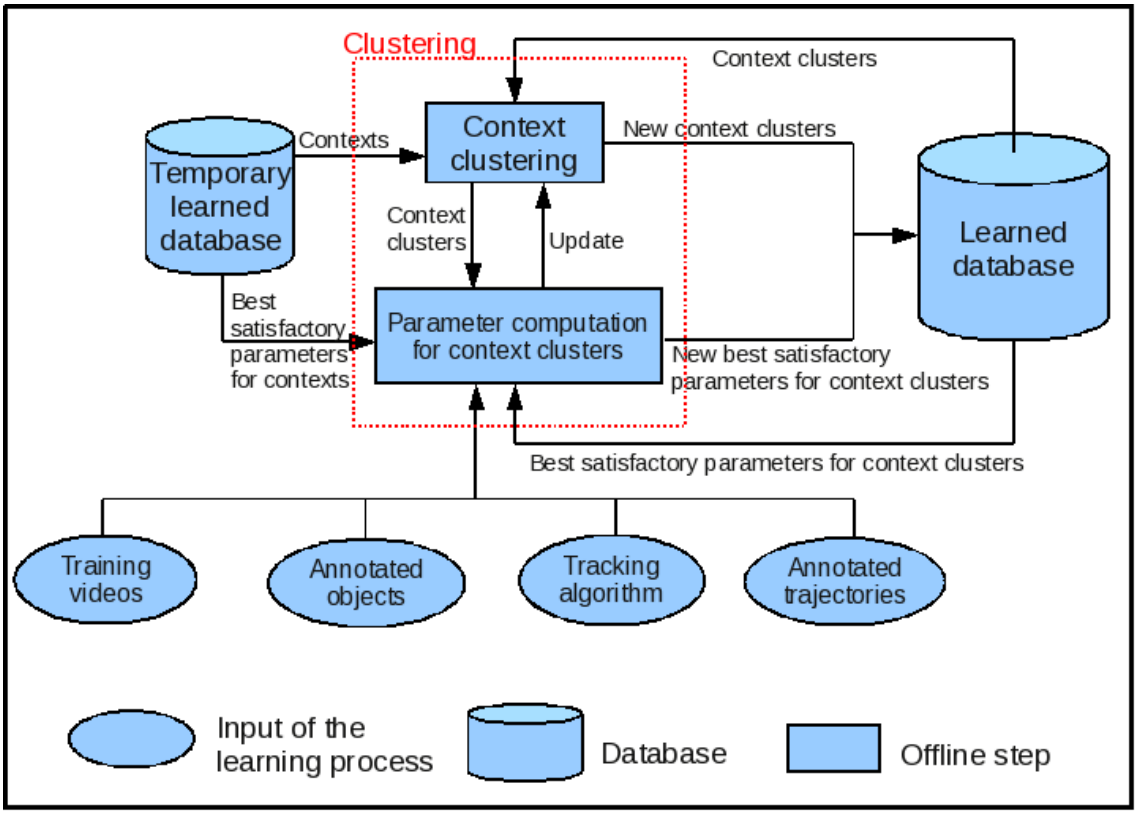


Figure 5.7 – Illustration of the clustering step

perform the clustering from time to time (e.g. every 5 minutes) or at the end of the online control phase.

This clustering step takes as input all data related to the creation of the official learned database : the training videos, the annotated objects, tracking algorithm, annotated trajectories as well as the learned context clusters and their best satisfactory parameters. It also requires the data stored in the temporary learned database. This process gives as output the updated official learned database. In general, the execution process of this clustering step is similar to the clustering process performed in the offline phase (see section 4.5). It includes two sub-steps : context clustering and parameter computation for context clusters (see figure 5.7).

- Context Clustering

This context clustering process is completely similar to the one presented in section 4.5.1. A QT-clustering algorithm is used to perform the classification of contexts. The distance between a context and a context cluster is defined as the complete linkage (i.e. the maximum distance from the context to any context of the cluster) [Everitt et al., 2001] and normalized in the interval $[0,\ 1]$.

In order to compute the distance between two context code-books, each of six feature code-books of a context is transformed into a histogram whose bin i is corresponding to feature value $\overline{\mu_i^k}$ of code-word i, and value of bin i is defined as f_i/N where N is the number of training frames of the code-book, f_i is the number of frames in which the code-word i is activated (see formula 4.20).

The distance between two code-books is defined as the Earth Mover Distance between the two corresponding histograms in which the ground distance c_{ij} (in formula 4.8) is defined as $|\overline{\mu_i^k} - \overline{\mu_j^k}|$. The distance between two contexts is defined as the mean value of the six distances between the six code-book pairs.

- Parameter Computation for Context Clusters

Similar to the "parameter computation for context clusters" task performed in the offline phase (see section 4.5.2), this process also includes two sub-tasks : "parameter computation" and "parameter verification".

+ Parameter Computation

The objective of the parameter computation sub-task is to determine the best satisfactory parameter values $\vec{p}^j$ for each context cluster j. These values are defined as a weighted combination of the best satisfactory parameters for the contexts belonging to the considered cluster (see section 4.5.2.1 for more details).

+ Parameter Verification

The objective of the parameter verification sub-task is to check whether the parameters for each context cluster resulting from the previous computation are the best. In other word, the tracking quality for these parameters has to be verified whether it is better than the quality given by the learned parameters of contexts belonging to that cluster.

For each cluster, this process takes all training videos belonging to this cluster and compute the tracking performance using the ground-truth data and the parameters $\vec{p}^j$ resulting from the "parameter computation" step. For every context belonging to the considered cluster, if the obtained tracking performance, quantified by the $F\text{-}Score$ value (see formula 4.41), is greater or equal to the one computed by its own satisfactory parameters, the parameters $\vec{p}^j$ are considered as "verified". Otherwise, contexts which are not satisfied with this condition, are removed from this cluster. They are then stored separately in the learned database. As the ground-truth data (including object detection, object trajectories) of videos which are learned in the online phase are not available, their contexts are not considered in this verification.

5.3 Summary

In this chapter, we have presented a controller whose objective is to tune and learn the tracking parameters to adapt online the tracker performance to the context variation. The controller uses the database resulting from the offline learning phase. The proposed controller has two stages : initial parameter configuration and parameter adaptation ones. For the parameter adaptation stage, two approaches are proposed. The first one, called context-based parameter adaptation, relies completely on the context detected for every video chunk to activate the parameter tuning process. The second approach, called evaluation-based parameter adaptation, relies on the detected context and on an online tracking quality evaluation to tune and learn automatic the control parameters.

The proposed controller has no important parameters for tuning. This controller has the two important values : a threshold Th_1 representing the maximal distance between a context and its cluster, and the number of frames l for which context is computed to find the suitable tracking parameters. These two values have already been determined in the the offline learning phase and they are not dependent significantly on the training and testing videos. We can use the same values for all video contexts.

Two other predefined thresholds in the online tracking evaluation algorithm (i.e. Th_6, Th_7 in formula 5.13) can be computed using a learning algorithm or estimated by user experience. They are quite independent on the tracking context. Similar to the two previous values Th_1 and l, the same values of these parameters can be used for all test videos.

All the context feature values (i.e. density, occlusion level of mobile objects, their 2D area, 2D area variance, their contrast with regard to the surrounding background and their contrast variance) are dependent on the object bounding box. The training phase is performed with the annotated objects, so a low quality of the object detection in the online phase makes decrease the correctness of the context detection. Consequently, the quality of the proposed controller can decrease significantly. So, one drawback of the proposed controller is the dependence of its performance to the object detection quality.

6

TRACKING ALGORITHMS

In the two previous chapters, we have presented a method for controlling tracking algorithms to cope with the context variations. The proposed method can handle the parameterizable trackers which rely on object appearance (e.g object size, color) and on points of interest. In order to validate this method, we present in this chapter three tracking algorithms in which the first two trackers belong to the appearance tracker category and the third one belongs to the point tracking category. The first tracker uses an object descriptor pool for tracking mobile objects. The second tracking algorithm has two stages : object tracking and global tracking. The object tracking uses a Kalman filter including three steps : estimation, measurement and correction. The global tracking stage aims at fusing trajectories belonging to a same mobile object and filtering out noisy trajectories. The third tracker relies on the tracking of KLT features located on mobile objects.

6.1 A Tunable Object Descriptor-based Tracking Algorithm Enabling Adaptation to Scene Conditions

Object appearance is widely used in the state of the art for object tracking [Zhou et al., 2006, Kwolek, 2009, Bak et al., 2009, Monari et al., 2009] but most of them cannot perform robustly in different contexts. We propose in this section a tracking algorithm which is able to perform in different scene conditions, denoted **appearance-based tracker**. First an object descriptor pool is used to compute the matching score between two detected objects. This descriptor pool includes 2D, 3D positions, 2D shape ratio, 2D area, color histogram, histogram of oriented gradient (HOG), color covariance and dominant color. In the tracking process, a temporal window is defined to establish the links between the detected objects. This enables to find the object

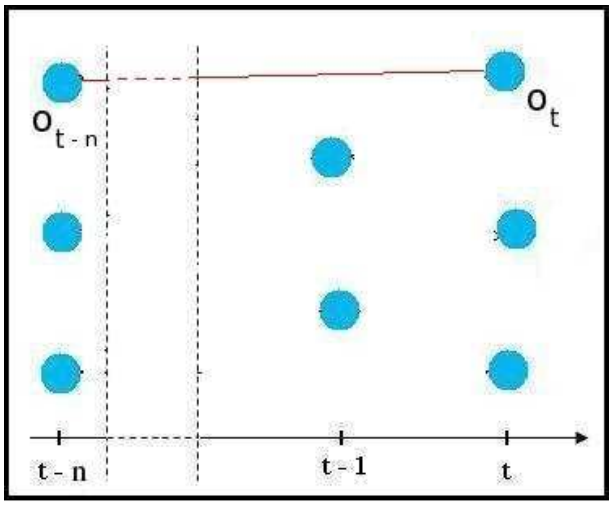


Figure 6.1 – Illustration of objects detected in a temporal window $[t-n,\ t]$

trajectories even if the objects are misdetected in some frames. A trajectory filter is proposed to remove noisy trajectories.

6.1.1 Object Descriptor Pool and Object Link Score

6.1.1.1 Object Descriptor Pool and Descriptor Similarity

The principle of the proposed tracking algorithm is based on an assumption of the coherence of the eight mobile object descriptors over time. These descriptors are 2D, 3D positions, 2D shape ratio, 2D area, color histogram, histogram of oriented gradient (HOG), color covariance and dominant color. For each descriptor i ($i = 1..8$), we define a descriptor similarity score $DS_i(o_t,\ o_{t-n})$ between an object o_t detected at t and an object o_{t-n} detected at $t-n$ ($n > 0$) (see figure 6.1). Then these descriptor similarity scores are combined together to compute a link score $LS(o_t,\ o_{t-n})$ between these two objects.

In the following sections, we describe the object descriptors and how to compute the descriptor similarity scores between two detected objects.

1. 3D Position

Depending on the object type (e.g. car, bicycle, human), the object speed cannot exceed a fixed threshold. So we can use the object 3D position to compute a similarity score between two detected objects. Let D^{3D}_{max} be the possible maximal 3D displacement of a mobile object for one frame in a video and $d^{3D}(o_t,\ o_{t-n})$ be the 3D distance of two considered objects o_t and o_{t-n}, we define a similarity score DS_1 between these two objects using the 3D position as follows :

$$DS_1(o_t,\ o_{t-n}) = max(0,\ 1 - \frac{d^{3D}(o_t,\ o_{t-n})}{nD^{3D}_{max}}) \tag{6.1}$$

The value of D^{3D}_{max} is determined according to the frame rate of the processing video stream and the object speed.

2. 2D Position

Similar to DS_1, we also define a similarity score DS_2 between two objects using their positions in the 2D image coordinate system :

$$DS_2(o_t,\ o_{t-n}) = max(0,\ 1 - \frac{d^{2D}(o_t,\ o_{t-n})}{nD^{2D}_{max}}) \tag{6.2}$$

where D^{2D}_{max} is the possible maximal 2D displacement of a mobile object for one frame ; $d^{2D}(o_t,\ o_{t-n})$ is the 2D distance between two considered objects o_t and o_{t-n}. In a 3D coordinate system, a value of D^{3D}_{max} can be set for the whole scene because it only depends on the type of physical object (e.g. car, human, bicycle) and the video stream frame rate. However, this value should not be unique in a 2D coordinate system. This threshold changes according to the distance between the considered object and the camera position. The nearer the object to the camera, the larger its 2D displacement is. Furthermore, the value D^{2D}_{max} also depends on the object movement direction. If an object moves in a perpendicular direction to the camera view, its displacement distance is higher than other directions. In order to cope with all of these situations, the threshold D^{2D}_{max} is defined in function of the 2D bounding box widths of the considered objects as follows :

$$D^{2D}_{max} = \frac{W_{o_t} + W_{o_{t-n}}}{2\beta} \tag{6.3}$$

where W_{o_t} and $W_{o_{t-n}}$ are 2D bounding box widths of object o_t and o_{t-n}; β is a parameter depending on the frame rate of the processing video stream and on the object type.

3. 2D Shape Ratio

Let W_{o_t} and H_{o_t} be the width and height of the 2D bounding box of object o_t. The 2D shape ratio of this object is defined as $s_{o_t} = W_{o_t}/H_{o_t}$. If the mobile objects are well detected, shape ratio within a temporal window does not vary much even if the lighting and contrast conditions are not good. A similarity score DS_3 between two 2D shape ratios of objects o_t and o_{t-n} is defined as follows :

$$DS_3(o_t,\ o_{t-n}) = \frac{min(s_{o_t},\ s_{o_{t-n}})}{max(s_{o_t},\ s_{o_{t-n}})} \tag{6.4}$$

4. 2D Area Similarity

The 2D area of object o_t is defined as $a_{o_t} = W_{o_t} H_{o_t}$. If the frame rate of the processing video stream is not too low (e.g. greater than 3 fps), the object 2D area varies slowly frame-to-frame when this object approaches or goes away from the camera. With other object movement direction, the change of object 2D area is less important. Similar to the 2D shape ratio, if mobile objects are well detected, object 2D area can be a useful descriptor to track objects even if the lighting and contrast conditions are not good. A similarity score DS_4 between two 2D areas of objects o_t and o_{t-n} is defined as follows :

$$DS_4(o_t,\ o_{t-n}) = \frac{min(a_{o_t},\ a_{o_{t-n}})}{max(a_{o_t},\ a_{o_{t-n}})} \tag{6.5}$$

5. Color Histogram Similarity

Object color is one of the most important appearance descriptors used in object tracking. When the scene lighting condition is good, a RGB color histogram of moving pixels is an efficient and effective descriptor for tracking object due to its rapid calculation, compared to some other color descriptors such as color covariance.

First for each object o_t, we compute a normalized histogram of b bins in the red channel, denoted $H^R_{o_t}$, where R represents the red channel, $H^R_{o_t}(i)$ ($i = 1..b$) represents the percentage of occurrence of moving pixels whose color belongs to bin i :

$$\sum_{i=1}^{b} H^R_{o_t}(i) = 1 \tag{6.6}$$

There exists several methods to comparing two histograms. In this work, we use a metric based on histogram intersection [M. J. Swain, 1991] due to its low time consuming computation. The similarity score $simil(H^R_{o_t},\ H^R_{o_{t-n}})$ between two histograms $H^R_{o_t}$, $H^R_{o_{t-n}}$ is defined as follows :

$$simil(H^R_{o_t},\ H^R_{o_{t-n}}) = \sum_{i=1}^{b} min(H^R_{o_t}(i),\ H^R_{o_{t-n}}(i)) \tag{6.7}$$

Figure 6.2 illustrates a histogram intersection. The histogram on the left and on the right represent two considered histograms, and the middle histogram illustrates the area intersecting (marked by red color) these two histograms.

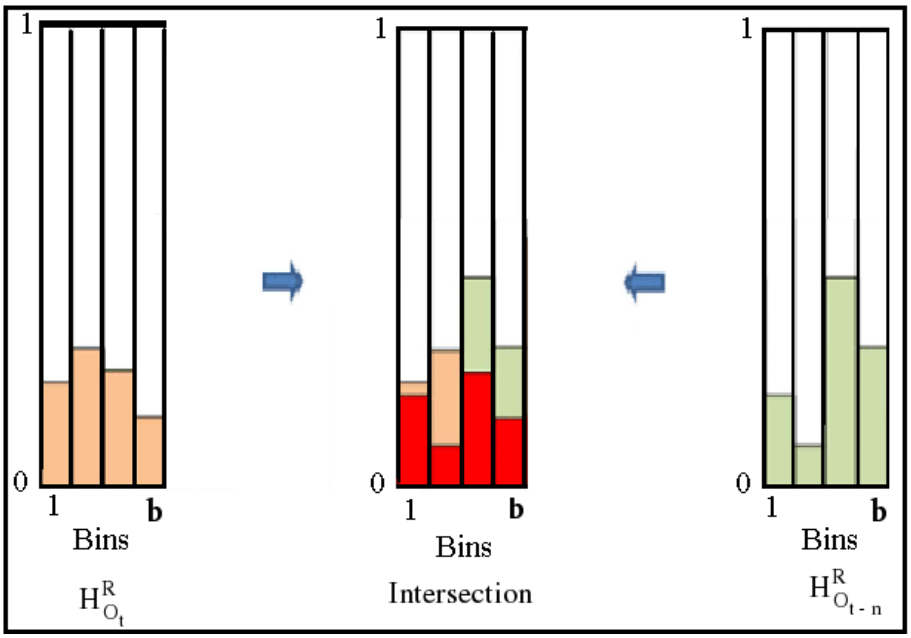


Figure 6.2 – Illustration of a histogram intersection. The intersection between left histogram and right histogram is marked by the red color in the middle histogram (adapted from [Hwang et al., 2009]).

The similarity score between two objects o_t and o_{t-n} for color histogram descriptor is defined as the mean of three histogram similarity scores corresponding to the three channels : red, green and blue as follows :

$$DS_5(o_t,\ o_{t-n}) = \frac{simil(H^R_{o_t},\ H^R_{o_{t-n}}) + simil(H^G_{o_t},\ H^G_{o_{t-n}}) + simil(H^B_{o_t},\ H^B_{o_{t-n}})}{3} \tag{6.8}$$

where $H^G_{o_t}$ and $H^B_{o_t}$ ($H^G_{o_{t-n}}$, $H^B_{o_{t-n}}$ respectively) represent the histogram of object o_t (o_{t-n} respectively) in green and blue channels.

6. Histogram of Oriented Gradients (HOG)

In the case of occlusion, the detection algorithm may fail to detect the full appearance of mobile objects. The above global descriptors are then unreliable for object tracking. In order to address this issue, we propose to use the HOG descriptor to track locally interest points on mobile objects and to compute the trajectory of these points. The HOG similarity between two objects is defined as a value proportional to the number of pairs of tracked points belonging to both objects. In [Bilinski et al., 2009], the authors present a method to track FAST points based on HOG descriptors. However the reliability of the obtained point trajectories is not addressed. In this work, we propose a method to quantify the reliability of the trajectory of each interest

point by considering the coherence of the Frame-to-Frame (F2F) distance, the direction and the HOG similarity of the points belonging to a same trajectory. We assume that the variation of these descriptors follows a Gaussian distribution.

Let $(p_1, p_2, ..., p_i)$ be the trajectory of a point. Point p_i is on the current tracked object and point p_{i-1} is on an object previously detected. We define a coherence score S_i^{dist} of F2F distance of point p_i using the probability density function of a Gaussian distribution as follows :

$$S_i^{dist} = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(d_i-\mu_i)^2}{2\sigma_i^2}} \tag{6.9}$$

where d_i is the 2D distance between p_i and p_{i-1}, μ_i and σ_i are respectively the mean and standard deviation values of the F2F distance distribution formed by the set of points $(p_1, p_2, ..., p_i)$ $(i > 1)$. They are computed as follows :

$$\mu_i = \frac{\sum_{j=2}^{i} d_j}{i-1} \tag{6.10}$$

$$\sigma_i = \sqrt{\frac{1}{i-1}\sum_{j=2}^{i}(d_j - \mu_i)^2} \tag{6.11}$$

In the same way, we compute the direction coherence score S_i^{dir} of each interest point by replacing d_i in the three above formulas by an angle characterizing the direction of point pair $< p_{i-1}, p_i >$. The HOG descriptor similarity coherence S_i^{desc} is also defined in a similar way by replacing d_i by the HOG descriptor similarity between two points p_{i-1} and p_i.

Finally for each interest point p_i on the tracked object o_t, we define a coherence score $S_i^{o_t}$ as the mean value of these three coherence scores :

$$S_i^{o_t} = \frac{S_i^{dist} + S_i^{dir} + S_i^{desc}}{3} \tag{6.12}$$

Let P be the set of interest point pairs of tracked points belonging to two considered objects o_t and o_{t-n} ; $S_i^{o_t}$ ($S_j^{o_{t-n}}$ respectively) be the coherence score of point i (j respectively) on object o_t (o_{t-n} respectively) belonging to set P. We define the similarity of HOG between these two objects as follows :

$$DS_6(o_t,\ o_{t-n}) = min(\frac{\sum_{i=1}^{|P|} S_i^{o_t}}{M_{o_t}}, \frac{\sum_{j=1}^{|P|} S_j^{o_{t-n}}}{M_{o_{t-n}}}) \tag{6.13}$$

where M_{o_t} and $M_{o_{t-n}}$ are the total number of interest points detected on objects o_t and o_{t-n}.

7. Color Covariance

Color covariance is a very useful descriptor to characterize the appearance model of an image region. In particular, the color covariance matrix enables to compare regions of different sizes and is invariant to identical shifting of color values. This becomes an advantageous property when objects are tracked under varying illumination conditions. In [Bak et al., 2010b], for a point i in a given image region R, the authors define a vector $\overrightarrow{f_i}$ including 11 sub-descriptors :

$$\overrightarrow{f_i} = \{x, y, R_{xy}, G_{xy}, B_{xy}, M^R_{xy}, O^R_{xy}, M^G_{xy}, O^G_{xy}, M^B_{xy}, O^B_{xy}\} \tag{6.14}$$

where x, y are pixel locations, R_{xy}, G_{xy}, and B_{xy} are RGB channel values at position $(x,\ y)$; M and O correspond to gradient magnitude and orientation in each channel at position (x, y). The covariance of region R is characterized by a matrix C of 11×11 :

$$C_R = \frac{1}{n-1}\sum_{i=1}^{n}(\overrightarrow{f_i} - \overrightarrow{\mu_R})(\overrightarrow{f_i} - \overrightarrow{\mu_R})^T \tag{6.15}$$

where n is the number of pixels in region R ; $\overrightarrow{\mu_R}$ is a vector of 11 dimensions representing the mean values of the 11 sub-descriptors of all points in the region R ; $\overrightarrow{f_i}$ is the sub-descriptor vector of point i, defined in formula 6.14.

We use the distance defined by [Forstner and Moonen, 1999] to compare two covariance matrices :

$$\rho(C_i, C_j) = \sqrt{\sum_{k=1}^{F} \ln^2 \lambda_k(C_i, C_j)} \tag{6.16}$$

where F is the number of considered point sub-descriptors ($F = 11$ in this case), $\lambda_k(C_i, C_j)$ is the generalized eigenvalue of C_i and C_j, determined by :

$$\lambda_k C_i x_k - C_j x_k = 0 \tag{6.17}$$

where x_k is the generalized eigenvector ($x_k \neq 0$).

In order to take into account the object spatial coherence and also to manage occlusion cases, we propose to use the spatial pyramid match kernel defined in [Grauman and Darrel, 2005]. The main idea is to divide the image region of the considered objects into a set of sub-regions. At each level k ($k \geq 0$), each of the considered objects is divided into a set of $2^k \times 2^k$ sub-regions. Then we compute the color covariance distance for each pair of corresponding sub-regions using formula 6.16 (see figure 6.3). The computation of each sub-region pair helps to evaluate the spatial structure coherence between two considered objects. In the case of occlusions, the color covariance distance between two regions corresponding to occluded parts can be very high. Therefore, we take only the lowest color covariance distances (i.e. highest similarities) at each level to compute the final color covariance distance. Let $M^k_z = \{\rho^k_1, \rho^k_2, ..., \rho^k_z\}$ be the set of

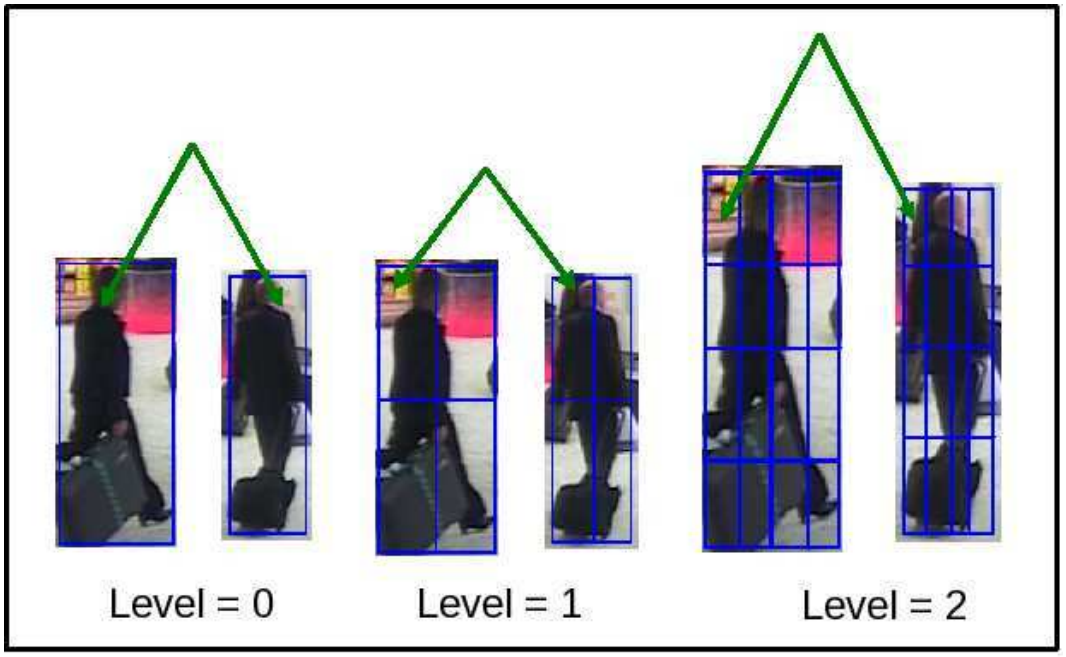


Figure 6.3 – Illustration of different levels in the spatial pyramid match kernel

the z largest distances between corresponding covariance matrices at level k. The covariance distance between two objects at level k is defined as follows :

$$D^k = \frac{\sum_{i=1}^{2^k \times 2^k} \rho(C_i^{o_t}, C_i^{o_{t-n}}) - \sum_{j=1}^{|M_z^k|} \rho_j^k}{2^k \times 2^k - |M_z^k|} \tag{6.18}$$

where $C_i^{o_t}$ and $C_i^{o_{t-n}}$ are respectively the covariance matrices of objects o_t and o_{t-n} at subregion i, $\rho(C_i^{o_t}, C_i^{o_{t-n}})$ is the covariance distance between $C_i^{o_t}$, $C_i^{o_{t-n}}$ defined in formula 6.16,

Then, the number of distances that are computed at each level are combined using a weighted sum. Distances computed at finer resolutions are weighted more highly than distances computed at coarser resolutions. So the distance of color covariance $d_{cov}(o_t,\ o_{t-n})$ between the two objects o_t and o_{t-n} is defined as follows :

$$d_{cov}(o_t, o_{t-n}) = D^L + \sum_{k=0}^{L-1} \frac{1}{2^L - 1} D^k - D^{k+1} = \frac{1}{2^L} D^0 + \sum_{k=1}^{L} \frac{1}{2^L - k + 1} D^k \tag{6.19}$$

where L is a parameter representing the maximal considered level ($L \geq 0$). We define the similarity score for color covariance descriptor between two objects o_t and o_{t-n} as follows :

$$DS_7(o_t,\ o_{t-n}) = max(0,\ 1 - \frac{d_{cov}(o_t,\ o_{t-n})}{D_{cov\_max}}) \tag{6.20}$$

where D_{cov_max} is the maximal distance for two color covariance matrices to be considered as similar.

8. Dominant Color

Dominant color descriptor (DCD) has been proposed by MPEG-7 and is extensively used for image retrieval [Yang et al., 2008]. This is a reliable color descriptor because it takes into account only the main colors of the considered image region. DCD of an image region is defined as $F = \{\{c_i, p_i\},\ i = 1..\mathcal{C}\}$ where $\mathcal{C}$ is the total number of dominant colors in the considered image region, c_i is a 3D RGB color vector, p_i is its relative occurrence percentage, with $\sum_{i=1}^{\mathcal{C}} p_i = 1$.

Let F_1 and F_2 be the DCDs of two considered image regions of detected objects. The dominant color distance between these two regions is defined using the similarity measure proposed in [Yang et al., 2008] :

$$D(F_1,\ F_2) = 1 - \sum_{i=1}^{\mathcal{C}_1} \sum_{j=1}^{\mathcal{C}_2} a_{ij} S_{ij} \tag{6.21}$$

where

$$S_{ij} = (1 - |p_i - p_j|) min(p_i, p_j) \tag{6.22}$$

$$a_{ij} = max(1 - \frac{d_{ij}}{D_{DC\_max}},\ 0) \tag{6.23}$$

where d_{ij} is the Euclidean distance between two colors, threshold D_{DC_max} is the maximal distance for two colors to be considered as similar.

Similar to the color covariance descriptor, in order to take into account the spatial coherence and also occlusion cases, we propose to use the spatial pyramid match kernel for comparing the DCD between two objects. We divide the object into sub-regions and compute the dominant color distance between corresponding region pairs using formula (6.21). A distance value is computed at each level k thanks to formula (6.18). Finally, the distance $d_{DC}(o_t,\ o_{t-n})$ between two objects o_t and o_{t-n} for dominant color is computed similarly as formula (6.19).

The DCD similarity score is defined as follows :

$$DS_8(o_t,\ o_{t-n}) = 1 - d_{DC}(o_t,\ o_{t-n}) \tag{6.24}$$

where $d_{DC}(o_t,\ o_{t-n})$ is the spatial pyramid distance of dominant colors between two considered objects.

6.1.1.2 Object Link Score

Using the eight object descriptors which are described above, we establish the links between objects detected within a temporal window. Each link is associated with a similarity score of the two corresponding objects. In order to increase the reliability of the established links, we only consider the pairs of objects whose 2D or 3D distance between themselves is low enough. Therefore, a link score $LS(o_t,\ o_{t-n})$ is defined as a weighted combination of descriptor similarity scores DS_i between two objects o_t and o_{t-n} as follows :

$$LS(o_t,\ o_{t-n}) = \begin{cases} \frac{\sum_{k=3}^{8} w_k DS_k(o_t,\ o_{t-n})}{\sum_{k=3}^{8} w_k} & \text{if } DS_1(o_t,\ o_{t-n}) > 0 \vee DS_2(o_t,\ o_{t-n}) > 0 \\ 0 & otherwise \end{cases} \tag{6.25}$$

where w_k is the weight (corresponding to its effectiveness) of descriptor k. At least one weight is not null. These weights can be learned offline for each video context (see section 6.1.3). A link between two objects is established if their link score is higher than a predefined threshold Th_8.

6.1.2 Proposed Object Tracking Process

The proposed tracking algorithm takes as input the video stream and a list of objects detected in a temporal window. The size of this temporal window (denoted T_1) is a parameter. The proposed tracker is composed of three stages. First, we compute a link score between any two detected objects appearing in the given temporal window to establish possible links (see section 6.1.2.1). Second, the object trajectories that include a set of consecutive links resulting from the previous stage, are determined (see section 6.1.2.3). Finally, a filter is applied to remove noisy trajectories (see section 6.1.2.4).

6.1.2.1 Establishment of Object Links

For each detected object pair in a given temporal window of size T_1, we compute the link score (i.e. instantaneous similarity) defined in formula 6.25. A temporary link is established between two objects when their link score is greater or equal to a predefined threshold Th_8 (see section 6.1.1.2). At the end of this stage, we obtain a weighted graph whose vertices are the detected objects in the considered temporal window and whose edges are the temporarily established links. Each edge is associated with a score representing the link score between its two vertices (see figure 6.4). For each detected object, we search its matched objects in a predefined radius and in a given temporal window to establish possible links so that even when one mobile object cannot be detected in some frames, it still can find its successor objects.

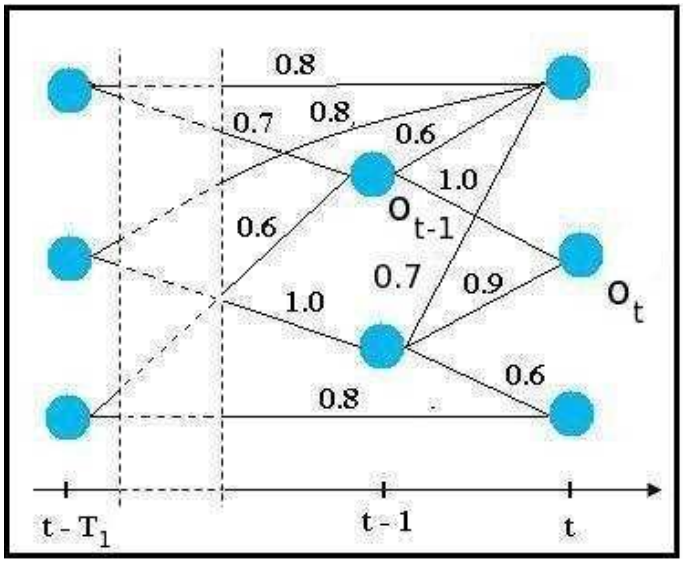


Figure 6.4 – The graph representing the established links of the detected objects in a temporal window of size T_1 frames.

6.1.2.2 Long-term Score

In the tracking task, the appearance of a tracked object usually changes over time due to occlusion, scene illumination variation or object detection error. Therefore, the link score between two detected objects (denoted o_t and o_{t-n}) is not reliable enough to decide whether they belong to the same trajectory. To solve this problem, we propose a method to compute the similarity score between an object o_t and the o_{t-n} trajectory. In order to decrease the processing time and to focus on comparing with the latest appearances of object o_{t-n}, we select the last Q detected objects belonging to o_{t-n} trajectory. Let $\mathfrak{T}_{o_{t-n}}$ be the trajectory of object o_{t-n}, Q is defined as follows :

$$Q = min(\mathcal{Q},\ T(\mathfrak{T}_{o_{t-n}})) \tag{6.26}$$

where $\mathcal{Q}$ is a parameter representing the maximal number of objects belonging to $\mathfrak{T}_{o_{t-n}}$ which are considered ; $T(\mathfrak{T}_{o_{t-n}})$ represents time length (number of frames) of $\mathfrak{T}_{o_{t-n}}$. This score is called long-term score (to distinguish with the link score between two objects which is like an instantaneous similarity). Thanks to the selection of the last Q-objects, the long-term score takes into account the latest variations of the o_{t-n} trajectory. Each descriptor has an own definition of long-term score. In the following sections, we denote $\mathcal{S}^{Q}_{o_{t-n}}$ as the set of last Q detected objects belonging to $\mathfrak{T}_{o_{t-n}}$.

- 2D Shape Ratio and 2D Area Long-term Scores

By assuming that the variation of the 2D area and shape ratio of a tracked object follows a Gaussian distribution, we can use the Gaussian probability density function (PDF) to compute this score. Also, longer the trajectory of o_{t-n} is, more reliable the PDF is. Therefore, for these two descriptors k ($k = 3$ or $k = 4$), we define a long-term score between object o_t and trajectory of o_{t-n} as follows :

$$LT_k(o_t,\ \mathfrak{T}_{o_{t-n}}) = \frac{1}{\sqrt{2\pi\sigma^2_{o_{t-n}}}} e^{-\frac{(s^k_{o_t} - \mu^k_{o_{t-n}})^2}{2(\sigma^k_{o_{t-n}})^2}} \frac{Q}{\mathcal{Q}} \tag{6.27}$$

where $s^k_{o_t}$ is the value of descriptor k for object o_t ($s^k_{o_t}$ can be 2D area or 2D shape ratio value), $\mu^k_{o_{t-n}}$ and $\sigma^k_{o_{t-n}}$ are respectively mean and standard deviation values of descriptor k of objects belonging to $\mathcal{S}^Q_{o_{t-n}}$. Longer the $\mathfrak{T}_{o_{t-n}}$ is, more reliable this long-term score is. Therefore, in the above formula, we multiply the score with $\frac{Q}{\mathcal{Q}}$.

- Color Histogram Long-term Score

First, we define a mean color histogram $\overline{H}^R_{\mathcal{S}^Q_{o_{t-n}}}$ in red channel for objects belonging to $\mathcal{S}^Q_{o_{t-n}}$. The value at bin i of this histogram is defined as follows :

$$\overline{H}^R_{\mathcal{S}^Q_{o_{t-n}}}(i) = \frac{\sum_{q=1}^{Q} H^R_{o^q}(i)}{Q} \quad , i = 1..b \tag{6.28}$$

where b represents the number of bins of the considered histograms ; $H^R_{o^q}$ ($q = 1..Q$) represents the color histogram in red channel of object q belonging to $\mathcal{S}_{o_{t-n}}$. In a similar way, we can compute the mean histograms in green and blue channels.

The long-term score of the color histogram descriptor between object o_t and trajectory $\mathfrak{T}_{o_{t-n}}$ is defined in function of the similarities between three histogram pairs $\overline{H}^k_{\mathcal{S}^Q_{o_{t-n}}}$ and $H^k_{o_t}$ ($k = R,\ G,\ B$).

We use first formula 6.7 to compute the histogram similarity in each color channel. The long-term similarity is defined as :

$$LT_5(o_t,\ \mathfrak{T}_{o_{t-n}}) = \frac{\sum_{k=R,G,B} simil(H^k_{o_t},\ \overline{H}^k_{\mathcal{S}^Q_{o_{t-n}}})}{3} \frac{Q}{\mathcal{Q}} \tag{6.29}$$

Similar to the previous long-term score, longer the $\mathfrak{T}_{o_{t-n}}$ is, more reliable this long-term score is. Therefore, in the above formula, we multiply the score with $\frac{Q}{\mathcal{Q}}$.

- Color Covariance Long-term Score

Similar to the color histogram long-term score, for a trajectory $\mathfrak{T}_{o_{t-n}}$ we compute a mean color covariance matrix for objects belonging to $\mathcal{S}^Q_{o_{t-n}}$, denoted $\overline{C}^{\mathcal{S}^Q_{o_{t-n}}}$. Let C^{o_t} be the color

covariance matrix of object o_t. The color covariance long-term score between an object o_t and trajectory $\mathfrak{T}_{o_{t-n}}$ is defined as the similarity between two covariance matrices : C^{o_t} and $\overline{C}^{\mathcal{S}^Q_{o_{t-n}}}$. We use first formula 6.16 to compute the distance between these two covariance matrices, denoted $\rho(C^{o_t},\ \overline{C}^{\mathcal{S}^Q_{o_{t-n}}})$. Then formula 6.20 is used to map this distance to a similarity score in the interval $[0,\ 1]$:

$$LT_7(o_t,\ \mathfrak{T}_{o_{t-n}}) = max(0,\ 1 - \frac{\rho(C^{o_t},\ \overline{C}^{\mathcal{S}^Q_{o_{t-n}}})}{D_{cov\_max}})\frac{Q}{\mathcal{Q}} \tag{6.30}$$

- Dominant Color Long-term Score

First, we compute the dominant color similarity score $DS_8(o_t, o^q)$ between o_t and each object o^q belonging to $\mathcal{S}^Q_{o_{t-n}}$ using formula 6.24. The long-term score between o_t and trajectory $\mathfrak{T}_{o_{t-n}}$ is defined as :

$$LT_8(o_t, \mathfrak{T}_{o_{t-n}}) = \frac{\sum_{q=1}^{Q} DS_8(o_t, o^q)}{Q}\frac{Q}{\mathcal{Q}} \tag{6.31}$$

- HOG Long-term Score

For this descriptor, the long-term score is set to the same value of the corresponding link score :

$$LT_6(o_t,\ \mathfrak{T}_{o_{t-n}}) = DS_6(o_t,\ o_{t-n}) \tag{6.32}$$

6.1.2.3 Trajectory Determination

The goal of this stage is to determine the trajectories of the mobile objects. For each object o_t detected at instant t, we consider all its matched objects o_{t-n} (i.e. objects with temporarily established links) in previous frames that do not have yet official links (i.e. trajectories) to any objects detected at the following frames. For such an object pair $(o_t,\ o_{t-n})$, we define a global score $GS(o_t,\ o_{t-n})$ as follows :

$$GS(o_t,\ o_{t-n}) = \frac{\sum_{k=3}^{8} w_k GS_k(o_t,\ o_{t-n})}{\sum_{k=3}^{8} w_k} \tag{6.33}$$

where w_k is the weight of descriptor k (presented in formula 6.25), $GS_k(o_t,\ o_{t-n})$ is the global score for descriptor k between o_t and o_{t-n}, defined in function of link score and long-term score for descriptor k :

$$GS_k(o_t,\ o_{t-n}) = (1-\beta)DS_k(o_t,\ o_{t-n}) + \beta LT_k(o_t,\ \mathfrak{T}_{o_{t-n}}) \tag{6.34}$$

where $DS_k(o_t,\ o_{t-n})$ is the similarity score for descriptor k between the two objects o_t and o_{t-n} (defined section 6.1.1.1) ; $LT_k(o_t,\ \mathfrak{T}_{o_{t-n}})$ is their long-term scores (defined in section 6.1.2.2) ;

β is the weight of long-term score. Longer the $\mathfrak{T}_{o_{t-n}}$ is, higher this reliability is. So the weight of the long-term score is computed as follows :

$$\beta = min(\frac{T(\mathfrak{T}_{o_{t-n}})}{\mathcal{Q}}, \mathfrak{W}) \tag{6.35}$$

where $T(\mathfrak{T}_{o_{t-n}})$, $\mathcal{Q}$ are presented in formula 6.26, and $\mathfrak{W}$ is the maximal expected weight for the long-term score.

The object o_{t-n} having the highest global similarity is considered as a temporary father of object o_t. After considering all objects at instant t, if more than one object get o_{t-n} as a father, the pair $(o_t,\ o_{t-n})$ corresponding to the highest value of $GS(o_t,\ o_{t-n})$ is kept, and the link between this pair is official (i.e. become officially a trajectory segment). A mobile object is no longer tracked if it cannot establish any official links in T_1 consecutive frames.

6.1.2.4 Trajectory Filtering

Noise usually appears when wrong detection or misclassification (e.g. due to low image quality) occurs. Hence a static object (e.g. a chair, a machine) or some image regions (e.g. window shadow, merged objects) can be detected as a mobile object (see figure 6.5) . However, such noise usually only appears in few frames or have no real motion. We thus use temporal and spatial filters to remove potential noises. A trajectory is considered as "finished" if its last detected object cannot establish any links with other objects in consecutive T_1 frames (see section 6.1.2.3 for more details). A "finished" trajectory $\mathfrak{T}_{o_t}$ of object o_t is classified as a noise if one of the following conditions is satisfied :

$$T(\mathfrak{T}_{o_t}) < Th_9 \tag{6.36}$$

$$d_{max}(\mathfrak{T}_{o_t}) < Th_{10} \tag{6.37}$$

where $T(\mathfrak{T}_{o_t})$ is time length of the o_t trajectory ; $d_{max}(\mathfrak{T}_{o_t})$ is the maximal 2D spatial length of the o_t trajectory ; Th_9, Th_{10} are thresholds. The threshold Th_9 represents the minimal length (in number of frames) of an object trajectory. This value depends on the frame rate of the processing video stream. In general, an object should exist in the scene at least one second. So the value of Th_9 is set to $\mathcal{F}$ where $\mathcal{F}$ is the frame rate of the processing video stream. The threshold Th_{10} represents the minimal spatial length of an object trajectory. This threshold value depends on application and the camera view.

6.1.3 Learning Descriptor Similarity Weights

Each object descriptor described above is effective for some particular contexts. The descriptors concerning size as shape ratio, area can be used when only mobile object sizes are different

Figure 6.5 – Illustration of a noisy trajectory

each other. Similarly, color histogram descriptor can be useful when the mobile object is fully detected. When the lighting condition of scene is not good, the color covariance descriptor can give a reliable similarity score between detected mobile objects. In the case of occlusions, the HOG, covariance and dominant color descriptors are still reliable. HOG descriptors whose goal is to track the interest points, can track mobile objects even when they are only partially detected. Thanks to the integration of the spatial pyramid match kernel, the color covariance and dominant color descriptors also can be useful in this situation. However the dominant color and color histogram descriptors are more sensible with illumination condition.

The open question is how can the user quantify correctly the descriptor significance for a given context ? In order to address this issue, we propose an offline supervised learning algorithm using the Adaboost algorithm [Freund and Schapire, 1997]. This learning algorithm takes as input the training videos and tracked object annotations (including object positions and bounding boxes over time) and gives as output the six object descriptor weight values w_k (k = 3..8) depending on the training video.

For each context, we select a training video sequence representative of this context. First, for each object pair (o_t, o_{t-n}) (called a training sample) in two different frames, denoted op_i ($i = 1..N$), we classify it into two classes {+1, -1} : $y_i = +1$ if the pair belongs to the same tracked object and $y_i = -1$ otherwise. This training sample labeling stage is performed using the tracked object annotations.

For each descriptor j ($j = 3..8$), we define a weak classifier for a pair op_i as follows :

$$h_j(i) = \begin{cases} +1 & , \text{ if } \forall o'_t \neq o_t,\ DS_j(o'_t,\ o_{t-n}) \leq DS_j(o_t,\ o_{t-n}) \ \wedge\ DS_j(o_t,\ o_{t-n}) \geq Th_8 \\ -1 & , \text{ otherwise} \end{cases} \tag{6.38}$$

where $DS_j(o_t,\ o_{t-n})$ is the similarity score of descriptor j (defined in section 6.1.1.1) between two objects o_t and o_{t-n}, and o'_t is an object detected at time instant t and is different from o_t ; Th_8 is a predefined threshold representing the minimal descriptor similarity considered as similar (presented in section 6.1.2.1).

The detail of the Adaboost-based learning process is presented in section 4.4.3.3. At the end of the Adaboost process, the weight values w_j for object descriptor j ($j = 3..8$) is computed using formula 4.46.

These values w_j allow to compute the link score and global score defined in formulas 6.25 and 6.33. The weight values w_j have significant influences on the tracking quality and their values are dependent on the tracking context. Therefore, we take them as the control parameters for experimenting the proposed controller. Section 7.3 presents in detail this experiment.

6.2 A Tracking Algorithm based on Kalman Filter and Global Tracking

This section presents an approach to track mobile objects based on their trajectories properties, denoted **estimator-based tracker**. The proposed approach includes two stages : tracking and global tracking. The tracker follows the steps of a Kalman filter including estimation, measurement and correction. However, the mobile object trajectories are usually fragmented because of occlusions and misdetections. Therefore, the global tracking stage aims at fusing the fragmented trajectories belonging to a same mobile object and removing the noisy trajectories.

6.2.1 Object Tracking Stage

The proposed tracker takes as input the processing video stream and a list of objects detected at each processing video frame. A tracked object at frame t is represented by a state $X_t = [x, y, l, h]$ where $(x,\ y)$ is center position, l is width and h is height of its 2D object bounding box at frame t. In the tracking process, we follow three steps of the Kalman filter : estimation, measurement and correction. The estimation step is first performed to estimate a new state of each tracked object in the current frame. The measurement step is then done to search for the best detected object similar to each tracked object in the previous frames. The state of the found object refers to as “measured state”. The correction step is finally performed to compute the “corrected state” of mobile object resulting from the “estimated state” and the “measured state”. This state is considered as the official state of the considered tracked object in the current

frame. For each detected object which does not match with any tracked object, a new tracked object with the same position and size is created.

6.2.1.1 Estimation of Object Position and Size

For each tracked object in the previous frame, the Kalman filter is applied to estimate a new state of the object in the current frame. The Kalman filter is composed of a set of recursive equations used to model and evaluate object linear movement. Let X_{t-1}^{+} be the corrected state at instant $t-1$, the estimated state at time t, denoted X_t^-, is computed as follows :

$$X_t^- = DX_{t-1}^{+} + W \tag{6.39}$$

$$P_t^- = DP_{t-1}^{+}D^T + Q^t \tag{6.40}$$

P_t^- and P_{t-1}^{+} are respectively the predicted and corrected covariances at time t and $t-1$; W is a noise matrix ; Q is the covariance of the noise W ; D is the state transition matrix which defines the relation between the state variables at time t and $t-1$; the size of matrix D is $n \times n$ where n is the considered descriptor number ($n = 4$ in this case). Note that in practice D can change with each time step, but here we assume it is constant.

6.2.1.2 Measurement

For each tracked object in the previous frame, the goal of this step is to search for the best matched object in the current frame. In a tracking task, the processing time is very important to ensure a real time system. Therefore, in this paper we propose to use a set of four descriptor similarities on 2D position, 2D shape ratio, 2D area and color histogram for computing the matching possibility between two objects. The computation of all these descriptor similarities are not time consuming and the proposed tracker can thus be executed in real time. Because all measurements are computed in the 2D space, our method does not require scene calibration information. Similar to the previous tracking algorithm, for each descriptor i ($i = 1..4$), we define a descriptor similarity score DS_i in the interval $[0,\ 1]$ to quantify the similarity between two detected objects on the descriptor i. A link score is defined as a weighted combination of these descriptor similarities. The state of detected object with the highest link score becomes the measured state.

- Descriptor Similarity

The similarity scores of all the considered object descriptors (i.e. 2D position, shape ratio, area and color histogram) are defined similarly to DS_2, DS_3, DS_4 and DS_5 in section 6.1.1.1.

+ Link Score

Using the four object descriptors we have presented above, a link score LS is defined to compute the similarity between two objects. A tracked object can have over time some size variations because of detection errors or some color variations by illumination changes, but its maximal speed cannot exceed a determined value. Therefore similar to the previous tracker, the link score in this tracker uses 2D position descriptor similarity as a filter to limit the matching objects search. Let o_t and o_{t-n} be respectively objects detect at time instant t and $t-n$, we define their link score $LS(o_t,\ o_{t-n})$ as follows :

$$LS(o_t,\ o_{t-n}) = \begin{cases} \frac{\sum_{i=2}^{4} w_i DS_i(o_t,\ o_{t-n})}{\sum_{i=2}^{4} w_i} & \text{if } DS_1(o_t,\ o_{t-n}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.41}$$

where $DS_i(o_t,\ o_{t-n})$ is the descriptor similarity score between two objects o_t and o_{t-n} ; and w_i is the weight (i.e. reliability) of descriptor i. The detected object with the highest link score value LS is selected as the matched object if :

$$LS(o_t,\ o_{t-n}) \geq Th_{11} \tag{6.42}$$

where Th_{11} is a predefined threshold. Higher the value of Th_{11} is set, more correctly the matching links are established, but a too high value of Th_{11} can make lose the matching links in some complex environment (e.g. poor lighting condition, object occlusion). The state of the matched object (including its position and its bounding box size) is called “measured state”. At a time instant t, if a tracked object cannot find its matched object, the measured state Z_t is set to 0.

6.2.1.3 Correction

Thanks to the estimated and measured states, we can update the position and size of tracked object by computing the corrected state X_t^+ as follows :

$$X_t^+ = X_t^- + K_t[Z_t - MX_t^-] \tag{6.43}$$

where M is the measurement prediction matrix, Z_t is the measured state, K_t is the Kalman gain which represents the reliability of the measurement result and is computed as follows :

$$K_t = P_t^- M^T [MP_t^- M^T + R_t]^{-1} \tag{6.44}$$

where R is the covariance matrix of noise in measurement. The corrected covariance matrix at t is also defined as follows :

$$P_t^+ = P_t^- - K_t M P_t^- \tag{6.45}$$

6.2.2 Global Tracking Stage

When there exists occlusions in a scene or the illumination is too weak or too saturated, the detection algorithms often fail to correctly detect objects. Hence these objects cannot be tracked correctly. The term "global tracking algorithm" represents a class of algorithms which aim at fusing the fragmented trajectories belonging to a same mobile object and removing the noisy trajectories. These algorithms are addressed recently in some studies [Chau et al., 2009a, Kuo et al., 2010].

In this work, we propose a global tracking process as follows. If a tracked object cannot find the corresponding detected object in the measurement step, it then turns into a "waiting state". This tracked object goes out of "waiting state" when it finds its matched object. A tracked object can turn into and goes out of "waiting state" many times during its life. This "waiting step" allows us to let a tracked object live for some frames when its matched object is not found. Mobile objects are so tracked completely even when they sometime are not detected or are detected incorrectly. This prevents the mobile object trajectories from being fragmented. However, the "waiting state" can cause an error when the corresponding mobile object goes out of the scene definitively. Therefore, we propose a rule to decide the moment when a tracked object ends its life and also to avoid maintaining for too long its "waiting state". A more reliable tracked object is kept longer in the "waiting state". In this global tracking, the tracked object reliability is directly proportional to the number of frames this object finds matched objects. The greater number of matched objects, the greater tracked object reliability is. Let ID of a frame be the order of this frame in the processed video sequence, a tracked object ends if :

$$F_l < F_c - min(N_r, T_2) \tag{6.46}$$

where F_l is the latest frame ID where this tracked object finds matched object (i.e. the frame ID before entering the "waiting state"), F_c is the current frame ID, N_r is the number of frames in which this tracked object was matched with a detected object, T_2 is a parameter to determine the maximal number of frames with a "waiting state" for a tracked object. With this method, a tracked object that finds a greater number of matched objects is kept in the "waiting state" for a longer time but its "waiting state" time never exceed T_2. Higher the value of T_2 is set, higher the probability of finding lost objects is, but this can decrease the correctness of the fusion process.

We also propose a set of rules to detect the noisy trajectories. The first two rules are similar to the ones presented in section 6.1.2.4. A temporal threshold Th'_9 and spatial threshold Th'_10 are defined to filter out noisy trajectories. A trajectory $\mathfrak{T}_{o_t}$ is considered as noise if one of two following conditions is right :

$$T(\mathfrak{T}_{o_t}) < Th'_9 \tag{6.47}$$

$$d_{max}(\mathfrak{T}_{o_t}) < Th'_{10} \tag{6.48}$$

where $T(\mathfrak{T}_{o_t})$ is the temporal length (number of frames) of the considered trajectory ("waiting state" time included) ; $d_{max}(\mathfrak{T}_{o_t})$ is the maximal spatial length of this trajectory. The Th'_9 value depends on the frame rate of the processing video stream. In general, an object should exist in the scene at least one second. So the value of Th'_9 is set to $\mathcal{F}$ where $\mathcal{F}$ is the frame rate of the processing video stream. The Th'_{10} value depends on application and the camera view.

A trajectory is composed of object locations throughout time, so a trajectory is unreliable if the corresponding tracked object cannot find enough matching objects and usually lives in the "waiting state". Therefore we define a temporal threshold than which a "waiting state" total time is greater, the corresponding trajectory is considered as a noise :

$$(\frac{T_w(\mathfrak{T}_{o_t})}{T(\mathfrak{T}_{o_t})} \geq Th_{12}) \tag{6.49}$$

where $T(\mathfrak{T}_{o_t})$ is the temporal length of the considered trajectory ("waiting state" time included) ; $T_w(\mathfrak{T}_{o_t})$ is the number of frames for which the considered tracked object is in "waiting state" during its life ; Th_{12} is a predefined threshold.

The conditions 6.47, 6.48 and 6.49 are only examined for the trajectories which end their life according to equation (6.46). If one of these conditions is satisfied, the considered trajectory is classified as a noise.

6.3 KLT Feature-based Tracker

The second mobile object tracker relies on the tracking of Kanade-Lucas-Tomasi (KLT) features [Shi and Tomasi, 1994], denoted **KLT-based tracker**. The KLT tracker takes the detected objects as input. This tracker includes three steps : KLT feature selection, KLT feature tracking and object tracking.

6.3.1 KLT Feature Selection

The objective of the KLT feature selection is to select the KLT feature point of interest located on the detected objects. The process of selection goes as follows :

1. At the first frame, compute the gradient matrix G and its minimum eigenvalue λ_{min} at every pixel of detected mobile objects. From the second frame, this computation is only done for detected objects on which the number of tracked KLT feature points is lower than a given threshold.

2. Call λ_{max} the maximum value of λ_{min} over the whole image.
3. Retain the image pixels that have a λ_{min} value larger than a percentage of λ_{max} (for example 5%).
4. From those pixels, retain the local maximum pixels. A pixel is kept if its λ_{min} value is larger than that of any other pixel in its $s \times s$ neighbourhood, where s represents the feature window size.
5. Keep the subset of those pixels so that the minimum distance between any pair of pixels is larger than a given threshold distance m (e.g. 5 pixels).

6.3.2 KLT Feature Tracking

Given the location $p_t = (x, y)$ of a pixel p at frame t, its location in the next frame $t + 1$ is :

$$p_{t+1} = Ap + d \tag{6.50}$$

where d is the displacement vector ; $A = I+D$, I is a 2×2 identity matrix, D is a deformation matrix. Call $I(p_t)$ the color intensity of image I at location p_t, we obtain the following equation :

$$J(Ap + d) = I(p) \tag{6.51}$$

where J represents the next frame of image I in the processing video.

Given two images I and J and a window in image I, tracking means determining the parameters that appear in the deformation matrix D and displacement vector d. This problem is then that of finding the A and d that minimize the dissimilarity :

$$\epsilon = \int\int_W [J(Ap + d) - I(p)]^2 w(d) \ dp \tag{6.52}$$

where W is the given feature window and $w(x)$ is a weighting function. Under pure translation, the matrix A is constrained to the identity matrix. To minimize the expression 6.52, we differentiate it with respect to the unknown entries of the deformation matrix D and the displacement vector d and set the result to zero.

6.3.3 Object Tracking

The object tracking relies on the number of matching features over time between detected objects. Let P be the number of of interest point pairs which trajectories pass through two objects o_{t-1}, detected at $t - 1$, and o_t, detected at t. We define link score between these two objects as follows :

$$LS_{KLT}(o_t, o_{t-1}) = min(\frac{P}{M_{o_{t-1}}}, \frac{P}{M_{o_t}}) \tag{6.53}$$

where $M_{o_{t-1}}$ and M_{o_t} are respectively the total number of KLT tracked points located on object o_{t-1} and o_t. The object trajectories are established as the total number of matching points between detected objects is maximal.

6.4 Summary

In this chapter, we have presented three tracking algorithms. The first algorithm includes three stages : establishment of object links, trajectory determination and trajectory filtering. The establishment of object links aims at computing the link score between an object detected at t, denoted o_t, and an other object detected at $t - n$, denoted o_{t-n}. This score is computed thanks to a descriptor pool including 2D, 3D positions, shape ratio, area, color histogram, HOG descriptor, color covariance and dominant color descriptors. The objective of the trajectory determination stage is to find object trajectories based on the object link score computed in the previous stage and a long-term similarity score. The long-term score represents the similarity between o_t and trajectory of o_{t-n}. The trajectory filtering aims at removing noisy trajectories using temporal and spatial thresholds.

The second algorithm combines the ideas of a Kalman filter and a global tracker. This algorithm includes two stages : tracking and global tracking. The tracking stage is a Kalman tracking algorithm including three steps : estimation, measurement and correction of mobile object state. The state of an object is defined as a set of its 2D position and 2D bounding box. In the measurement step, we use four descriptors including 2D position, 2D shape ratio, 2D area and color histogram to search for the best matching object from the list of detected objects. The proposed global tracking stage aims at fusing fragmented trajectories. To do that, we define a "waiting state" into which a tracked object can turn when it does not find a corresponding detected object in the measurement step. This state helps the tracked objects can be kept "living" (i.e. not to be removed) in some frames when its correspondence is not found. This proposed method is useful when there exists occlusions in the scene and in consequence, the objects are not always visible for the detection. The second role of this global tracking stage is a trajectory filtering whose the objective is to remove noisy trajectories.

The third tracker is a KLT feature-based tracker. This tracker includes three steps : KLT feature selection, KLT feature tracking and object tracking. The objective of the KLT feature selection is to detect and select the points of interest located on mobile objects. Then, the KLT feature tracking step track these points over frames by determining the pixel displacement vector and the deformation matrix. The objective of the object tracking step is to determine

object trajectories by optimizing the number of matching points between detected objects.

For the first and the second tracking algorithms, the descriptor weights w_k (used in formulas 6.25, 6.33 and 6.41) are dependent on tracking context. These weights also influence significantly the tracking quality. Therefore we can use them as the control parameters. Section 7.3 presents in detail the control method experimentation for the first tracker.

For the third tracking algorithm, two parameters which are dependent on the tracking context are the minimum distance between KLT feature points m (see section 6.3.1) and the dimension of feature window W (see equation 6.52). For example, in the case of object occlusion, the values of m and W must be low and otherwise. When object 2D area is great, the value of m must be high. Therefore we can use these two parameters as the control parameters. Section 7.3.1.2 presents in detail the control method experimentation for this tracker.

Beside these parameters, these two trackers have some parameters which are dependent on the application, the frame rate of the test video stream and the user requirement, for example the thresholds Th_8 (formula 6.25) and Th_{10} (formula 6.37). Therefore we do not consider controlling them in this work but they can be controlled by the end-user with the same mechanism.

7

EXPERIMENTATION AND VALIDATION

In this chapter, we present in the first part the experimental results of the two proposed tracking algorithms : the appearance-based and the estimator-based trackers. In the second part, the proposed control method is experimented with the two trackers : the appearance-based tracker and the KLT-based tracker. These approaches are tested with three pubic video datasets (ETISEO, TRECVid and Caviar) in which a comparison with other tracking algorithms in state of the art is presented. Two long video sequences (more than one hour) are experimented in this chapter.

7.1 Implementation

The proposed algorithms in this thesis are implemented in a platform developed by the Pulsar team which is named SUP (i.e. Scene Understanding Platform). SUP is written in C++ language. This platform provides algorithms for video interpretation systems such as image acquisition, image segmentation, object classification. SUP is also able to integrate algorithms belonging to other vision platforms such as OpenCV[1] or LTI-Lib[2]. All experiments presented in this chapter have been performed in a machine of Intel(R) Xeon(R) CPU E5430 @ 2.66GHz (4 cores) and of 4GB RAM.

[1] http ://opencv.willowgarage.com/wiki/

[2] http ://ltilib.sourceforge.net/doc/homepage/index.shtml

7.2 Tracking Algorithms

7.2.1 Evaluation Overview

The objective of the first experimentation is to prove the robustness of the two tracking algorithms presented in chapter 6.

For the first tracker experimentation, an other objective is to show the effect of object descriptor weight learning. To this end, in the first part, we test the appearance-based tracker with two complex videos (many moving people, high occlusion occurrence frequency) which are respectively provided by the Caretaker European project and the TRECVid dataset. These two videos are tested in both cases : without and with the descriptor weight learning. The tracking result of the estimator-based tracker for these two videos are also provided.

In the second part, two videos belonging to two public datasets ETISEO are experimented, and the results of the two proposed trackers are compared with other seven object tracking approaches in the state of the art.

- Dataset Presentation

In the experimentation of the two proposed tracking algorithms, we process the video sequences belonging to the following datasets and projects :

- **Caretaker project**[3] **:** The Caretaker project focuses on the extraction of a structured knowledge from large multimedia collections recorded over networks of cameras and microphones deployed in subways. The people number in the scene is quite great. Although most people in subways follow rather simple trajectories (e.g. from entry zone straight to validating ticket zone), their trajectories are hard to be accurately detected. This is due to segmentation errors, poor video quality (highly compressed data), numerous static and dynamic occlusions and 3D error measurement of far away object locations from the camera. The frame rate of the videos in this dataset is 5 fps ($frames\ per\ second$).
- **TRECVid dataset [Smeaton** et al.**, 2006] :** The TREC Video Retrieval Evaluation (TRECVid) is an international benchmarking activity to encourage research in video information retrieval by providing a large test collection. In this experimentation, we test the proposed trackers with the videos recorded in an airport where there exists frequently a high people density and occlusion level.
- **ETISEO dataset**[4] **:** The ETISEO videos are provided by the ETISEO project. This project seeks to work out a new structure contributing to an increase in the evaluation of video scene understanding ; with the active participation of industrialists and many research

[3] http ://cordis.europa.eu/ist/kct/caretaker_synopsis.htm
[4] http ://www-sop.inria.fr/orion/ETISEO/

laboratories, such as French, European and International partners. Project ETISEO focuses on the treatment and interpretation of videos involving pedestrians and (or) vehicles, indoors or outdoors, obtained from fixed cameras. In this experimentation, we test the two proposed trackers with two ETISEO videos for comparing their performances with the tracker performances available in this dataset.

7.2.2 Evaluation Metrics

There are many different methods to evaluate a tracking algorithm result and we can classify them by two principal approach types : offline evaluation using ground truth data (for example approaches presented in [Nghiem et al., 2007] and [Needham and Boyle, 2003]) or online evaluation without ground truth (for example the algorithm presented in section 5.2.2.2, or [Erdem et al., 2004], [Liu et al., 2008], [Vaswani, 2007]). In order to be able to compare our tracker performances with the other ones on the ETISEO videos, we use the tracking evaluation metrics defined in the ETISEO project [Nghiem et al., 2007] which belongs to the first approach type.

The first tracking evaluation metric M_1, which is the "tracking time" metric, measures the percentage of time during which a reference data (ground-truth data) is tracked. The match between a reference data and a tracked object is done with respect to their bounding boxes. The tracked object which has the greatest intersection time interval with the reference data will be chosen to compute metric value. This metric gives us a global overview of the performance of tracking algorithms and is computed as follows :

$$M_1 = \frac{1}{N} \sum_{i=1}^{N} \frac{T_i}{F_i} \tag{7.1}$$

where N is the number of tracked objects annotated in ground truth data ; T_i is number of frames when the reference object i is tracked (by the considered tracking algorithm) ; F_i is number of frames when the reference object i is annotated (in ground truth data).

The second metric M_2 "object ID persistence" helps to evaluate the ID persistence. It computes over the time how many tracked objects are associated to one reference object as follows :

$$M_2 = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{Frag_i} \tag{7.2}$$

where N is defined in the formula 7.1, $Frag_i$ is the number of different tracked objects (of the considered tracking algorithm) which takes the reference object i.

On the contrary, the third metric M_3 "object ID confusion" computes the number of reference object IDs per detected object as follows :

$$M_3 = \frac{1}{D} \sum_{i=1}^{T} \frac{1}{Conf_i} \tag{7.3}$$

where D is the number of tracked objects (of the considered tracker) matching with ground truth data, $Conf_i$ is the number of different reference objects which take the tracked object i (of the considered tracking algorithm).

In order to evaluate the performance of tracking algorithms, these metrics must be used together to obtain a complete evaluation. Therefore, we also define a tracking metric $\overline{M}$ which takes the average value of three tracking metrics mentioned above :

$$\overline{M} = \frac{M_1 + M_2 + M_3}{3} \tag{7.4}$$

All four metric values are in the interval [0, 1]. The higher the metric value is, the better the tracking algorithm performance gets.

7.2.3 Object Detection

In this experimentation, we use the people detection algorithm based on the HOG descriptor of the OpenCV library. So we focus the experimentation on the sequences of people movements. However the principles of the proposed tracking algorithms are not dependent on the tracked object type.

7.2.4 Experimental Tracking Parameters

7.2.4.1 Appearance-based Tracker

We present in table 7.1 the experimental parameter values of the appearance-based tracker. These values are empirically set. However they can change depending on applications and the user requirements. In this case, the Th_{10} value is quite small to avoid filtering out trajectories of mobile objects which do not move really.

7.2.4.2 Estimator-based Tracker

The experimental parameter values of the estimator-based tracker is presented in table 7.2. Similar to the previous tracker parameter values, these values are also set empirically and can change depending on application and user requirement. For example, if we want to increase the precision of the tracking algorithm, the value Th_{11} can be increased. However, this decreases the sensitivity value of the tracking performance.

Formula	Parameter	Semantic	Value
6.6	b	Number of histogram bins	32 bins
6.20	D_{cov_max}	The maximal covariance distance	1.5
6.25	Th_8	Minimal link score between two matched objects	0.3
Section 6.1.2.1	T_1	Temporal window size for establishing object temporary links	30 frames
6.26	$\mathcal{Q}$	The maximal number of objects considered for computing the long-term score	10
6.35	$\mathfrak{W}$	The maximal expected weight for computing the long-term score	0.5
6.37	Th_{10}	Spatial threshold to filter noisy trajectories	5 pixels

Table 7.1 – Experimental parameter values for the appearance-based tracker

Formula	Parameter	Semantic	Value
6.42	Th_{11}	Minimal link score between two matched objects	0.8
6.46	T_2	Minimal number of frames of a tracked object for living in the "waiting state"	20 frames
6.48	Th'_{10}	Spatial threshold to filter noisy trajectories	5 pixels
6.49	Th_{12}	Threshold representing the minimal ratio between "waiting state" length and temporal length of an object trajectory	40%

Table 7.2 – Experimental parameter values for the estimator-based tracker

7.2.5 Experimental Videos and Results

7.2.5.1 Caretaker and TRECVid videos

In this section we present the tracking results in a Caretaker and a TRECVid video sequences. For these two sequences, we experiment the appearance-based tracker in two cases : without and with the offline object descriptor weight learning. In the first case, all the weight values w_k (k = 3..8, used in formulas 6.25 and 6.33) are set equal to $1/6$. In the second case, for each testing sequence, we select a video sequence which has a similar tracking context and learn offline the object descriptor weights using annotated objects and trajectories (see section 6.1.3). We use the learned weight values for tuning the appearance-based tracker in the testing phase. We do not consider in this experimentation the online variation of tracking contexts and the online parameter tuning. These problems will be mentioned in section 7.3.

The tested Caretaker video, denoted Caretaker 1, depicts people moving in a subway station. The frame rate of this sequence is 5 fps and the length is 5 min (see image 7.1a).

For the first tracker, we have learned object descriptor weights on a sequence of 2000 frames. The training sequence is recorded by the same camera view, but four days before the testing sequence. The learning algorithm selects $w_5 = 0.5$ (color histogram descriptor) and $w_6 = 0.5$ (HOG descriptor).

The tested TRECVid sequence depicts the movements of people in an airport (see image 7.1b). It contains 5000 frames and lasts 3 min 20 sec. We have learned object descriptor weights on a sequence of 5000 frames. The training sequence is recorded by the same camera view, but one hour later the testing sequence. The learning algorithm selects $w_4 = 0.5$ (2D area) and $w_5 = 0.50$ (color histogram).

Table 7.3 presents the tracking results in three cases : the appearance-based tracker without descriptor weight learning, the appearance-based tracker with descriptor weight learning, the estimator-based tracker. We can find that with the proposed learning scheme, the tracker performance increases in both tested videos. For the Caretaker sequence 1, the $\overline{M}$ increases from 0.59 to 0.70. For the TRECVid sequence, the $\overline{M}$ increases from 0.77 to 0.82. Also, the processing time of the tracker decreases significantly because many descriptors are not used.

The two following tested videos belong to the ETISEO dataset. The first tested ETISEO video shows a building entrance, denoted ETI-VS1-BE-18-C4. It contains 1108 frames and the frame rate is 25 fps. In this sequence, there is only one person moving (see image 7.2a). We have learned object descriptor weights on a sequence of 950 frames. The learning algorithm has selected the color histogram descriptor as the unique descriptor for tracking in this context.

The second tested ETISEO video shows an underground station denoted ETI-VS1-MO-7-C1 with occlusions. The difficulty of this sequence consists in the low contrast and bad illumination. The scene depth is quite important (see image 7.2b). This video sequence contains 2282 frames

			Caretaker sequence 1	TRECVid sequence
Tracker 1	Without learning	M_1	0.62	0.60
		M_2	0.16	0.82
		M_3	0.99	0.90
		$\overline{M}$	**0.59**	**0.77**
		s	1	1
	With learning	M_1	0.47	0.70
		M_2	0.83	0.93
		M_3	0.80	0.84
		$\overline{M}$	**0.70**	**0.82**
		s	2	4
Estimator-based tracker		M_1	0.11	0.55
		M_2	0.68	0.90
		M_3	0.88	0.85
		$\overline{M}$	**0.56**	**0.77**
		s	7	10

Table 7.3 – Summary of tracking results of the two proposed trackers. s denotes the average processing speed of the tracking task (fps).

Figure 7.1 – Illustration of the two tested videos : a. Caretaker 1 b. TRECVid

and frame rate is 25 fps. We have learned descriptor weights on a sequence of 500 frames. The object color covariance is selected as the unique descriptor for tracking in this context. It is a good solution because the dominant color and HOG descriptor do not seem to be effective due to bad illumination. Also, the size descriptors (i.e. 2D area and 2D shape ratio) are not reliable because their measurements do not seem to be discriminative for far away moving people from the camera.

In these two experiments, tracker results from seven different teams (denoted by numbers) in ETISEO have been presented : 1, 8, 11, 12, 17, 22, 23. Because names of these teams are hidden, we cannot determine their tracking approaches. Table 7.4 presents performance results of the considered trackers. The tracking evaluation metrics of the proposed trackers get the highest values in most cases compared to the other teams.

7.3 Control Method

7.3.1 Context-based Parameter Adaptation Approach

7.3.1.1 Appearance-based Tracker

Evaluation Overview

The objective of this experimentation is to prove the effect and robustness of the proposed control method. To this end, we use 12 video sequences recorded in different locations to learn offline the best satisfactory tracking parameters. Because the performance of the controller is

		ETI-VS1-BE-18-C4 sequence	ETI-VS1-MO-7-C1 sequence
Appearance-based tracker with learning	M_1	0.50	0.79
	M_2	**1.00**	**1.00**
	M_3	**1.00**	**1.00**
	$\overline{M}$	**0.83**	**0.93**
Tracker 2	M_1	0.50	**0.87**
	M_2	**1.00**	0.92
	M_3	**1.00**	**1.00**
	$\overline{M}$	**0.83**	**0.93**
Team 1	M_1	0.48	0.77
	M_2	0.80	0.78
	M_3	0.83	**1.00**
	$\overline{M}$	0.70	0.85
Team 8	M_1	0.49	0.58
	M_2	0.80	0.39
	M_3	0.77	**1.00**
	$\overline{M}$	0.69	0.66
Team 11	M_1	**0.56**	0.75
	M_2	0.71	0.61
	M_3	0.77	0.75
	$\overline{M}$	0.68	0.70
Team 12	M_1	0.19	0.58
	M_2	**1.00**	0.39
	M_3	0.33	**1.00**
	$\overline{M}$	0.51	0.91
Team 17	M_1	0.17	0.80
	M_2	0.61	0.57
	M_3	0.80	0.57
	$\overline{M}$	0.53	0.65
Team 22	M_1	0.26	0.78
	M_2	0.35	0.36
	M_3	0.33	0.54
	$\overline{M}$	0.31	0.56
Team 23	M_1	0.05	0.05
	M_2	0.46	0.61
	M_3	0.39	0.42
	$\overline{M}$	0.30	0.36

Table 7.4 – Summary of tracking results for two ETISEO videos. The highest values are printed **bold**.

Figure 7.2 – Illustration of the two tested videos : a. ETI-VS1-BE-8-C4 b. ETI-VS1-MO-7-C1

dependent on the quality of the object detection, we test the online control process in two cases : with automatically detected objects and with manually annotated objects. In the first case, the controller is experimented with a video sequence of 1h 42 minutes recorded in a subway station (provided by the Caretaker project) in which there are some contextual variations. Then the proposed controller is experimented with 20 videos from the Caviar dataset to compare the tracking performance with some other state of the art approaches. Finally, an other subway video (provided by the Vanaheim project) is tested. In the second case, we select two video sequences for experimenting. The first sequence is OneStopMoveEnter2cor belonging to the Caviar dataset. The second one is the Vanaheim video which is previously experimented. All the hypotheses presented in section 2 are verified in this experimentation. Training and testing videos satisfy hypotheses 4 and 5. For all the tested videos, the tracking results are presented in two cases : without and with the controller. In the case of "without the controller", for each experimental video, the object descriptor weight values are empirically determined.

- Dataset Presentation

In the experimentation of the proposed control method, we process the video sequences belonging to the following datasets and projects :

- **ETISEO dataset, Caretaker project :** see their descriptions at section 7.2.1.
- **Gerhome project**[5] **:** The objective of the Gerhome project is to develop and try out technical solutions supporting the assistance services for enhancing autonomy of the elderly at home, by using intelligent technologies for house automation. The videos belonging

[5]http ://gerhome.cstb.fr/en/home

to this project contain most of the time one person in the scene. The frame rate of this sequence is 8 fps.

- **Caviar dataset**[6] **:** For the Caviar dataset, a number of video clips were recorded acting out the different scenarios of interest. These scenarios include people walking alone, meeting with others, window shopping, entering and exiting shops, fighting and passing out and last, but not least, leaving a package in a public place. In this experimentation, we are interested in videos recorded from the corridor view.
- **Vanaheim project**[7] **:** The aim of the Vanaheim project is to study innovative surveillance components for autonomous monitoring of complex audio/video surveillance infrastructure, such as the ones prevalent in shopping malls or underground stations. In this experimentation, we process a Vanaheim video sequence recorded in a subway station.

- Evaluation Metrics

Due to the change of the test videos, we also change the evaluation metrics in order to be able to compare the obtained tracking performances with the tracker performances existing in state of the art. In this experimentation, we select the tracking evaluation metrics used in several publications [Wu and Nevatia, 2007, Xing et al., 2009, Huang et al., 2008, Li et al., 2009, Kuo et al., 2010]. Let GT be the number of trajectories in the ground-truth of considered video sequence. These metrics are defined as follows :

- Mostly tracked trajectories (MT) : The number of trajectories that are successfully tracked for more than 80% divided by GT. This value represents the sensitivity of the tracker performance.
- Partially tracked trajectories (PT) : The number of trajectories that are tracked between 20% and 80% divided by GT.
- Mostly lost trajectories (ML) : The number of trajectories that are tracked less than 20% divided by GT.

We also define the precision of the tracker performance, denoted P, as the number of trajectories that are successfully tracked for more than 80% divided by number of output trajectories. The quality of the tracking output is quantified by the F-Score in the interval $[0,\ 1]$ as follows :

$$\text{F-Score} = \frac{2 \times \text{P} \times \text{MT}}{\text{P} + \text{MT}} \tag{7.5}$$

The F-Score is in the interval [0, 1]. Higher the F-Score value is, higher the tracking quality is.

- Training Phase

[6] http ://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

[7] http ://www.vanaheim-project.eu/

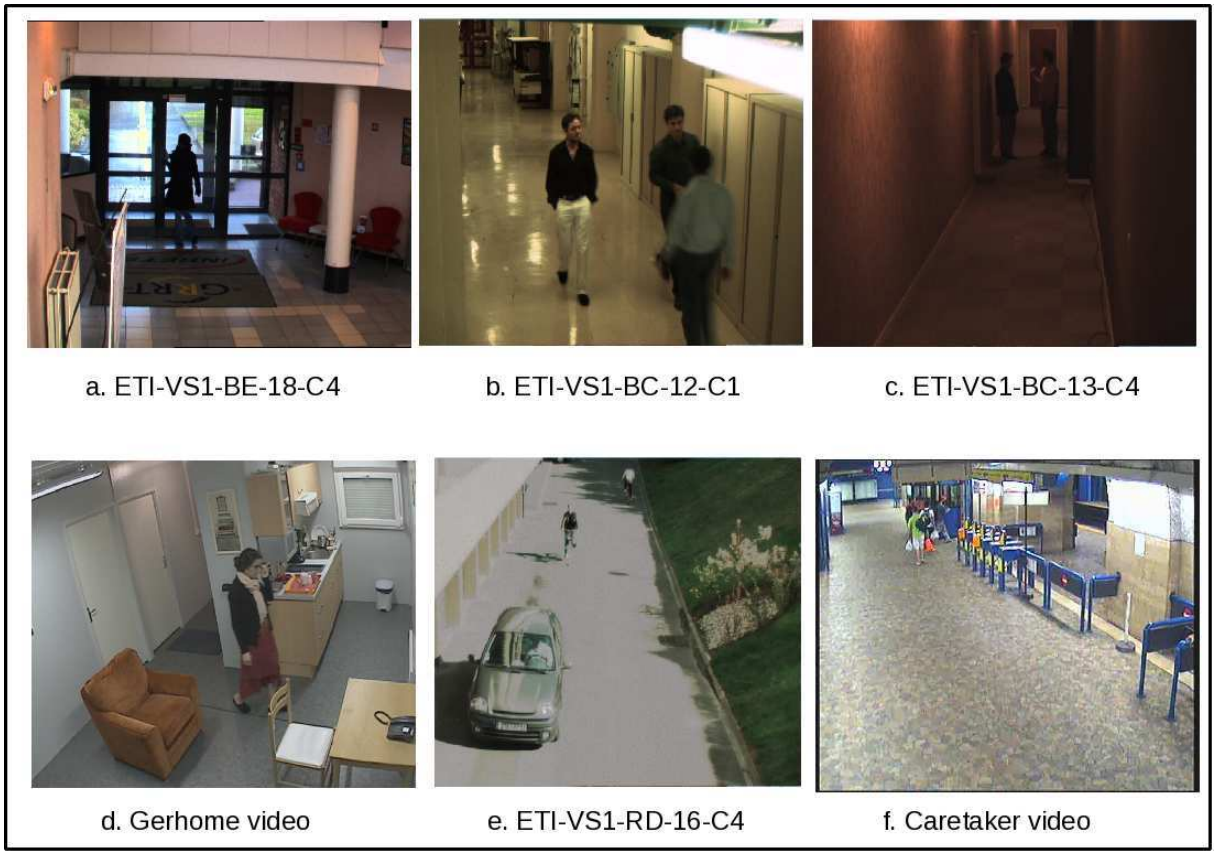


Figure 7.3 – Illustration of the training videos

In the offline learning phase, we use 12 video sequences belonging to different context types (i.e. different levels of the density and the occlusion of mobile objects as well as of their contrast with regard to the surrounding background, their contrast variance, their 2D area and their 2D area variance) to build the learning database (so the hypothesis 2 presented in section 1.2.2 is ensured) :

- 3 videos are recorded in buildings denoted ETI-VS1-BE-18-C4, ETI-VS1-BC-12-C1 and ETI-VS1-BC-13-C4 (provided by the ETISEO dataset, see figures 7.3a, 7.3b and 7.3c).
- 1 video is recorded at home (provided by the Gerhome project, see figure 7.3d).
- 1 video is recorded in a street, denoted ETI-VS1-RD-16-C4 (provided by the ETISEO project, see figure 7.3e).
- 1 video is recorded in a subway station (provided by the Caretaker European project, see figure 7.3f).
- 6 videos are recorded in a shopping center from a corridor view (provided by the Caviar dataset, see figure 7.4).

In this experimentation, we control the tracking algorithm presented in section 6.1 whose

Figure 7.4 – Illustration of the Caviar video

Notation	Semantic
w_3	2D shape ratio descriptor weight
w_4	2D area descriptor weight
w_5	Color histogram descriptor weight
w_6	HOG descriptor weight
w_7	Color covariance descriptor weight
w_8	Dominant color descriptor weight

Table 7.5 – Semantic of the notations of the object descriptor weights

principle is to use a pool of six appearance descriptors (2D shape ratio, size, color histogram, color covariance, dominant color and HOG descriptor) to track mobile objects. The six weights (i.e. importance) of the object descriptor similarities (see formulas 6.25 and 6.33) are selected as the control parameters for experimenting the proposed controller. Table 7.5 reminds the semantic of these parameters. These parameters have a strong influence on the tracking performance, so the hypothesis 1 (presented in section 1.2.2) is ensured. Because the HOG descriptor computation is too time consuming, its weight (denoted w_6) is removed from the list of control parameters. It means that we do not consider to activate this descriptor during the object tracking process. In the tracking parameter optimization, we use the Adaboost algorithm to learn the control parameters (i.e. object descriptor weights) for each context (see section 6.1.3).

Each training video is segmented automatically into a set of context chunks (of different temporal length). The number of context chunks is dependent on the contextual variation of this training video. Figure 7.5 presents the context segmentation result of the sequence ThreePastShop2cor belonging to the Caviar dataset. The values of object 2D areas are divided by

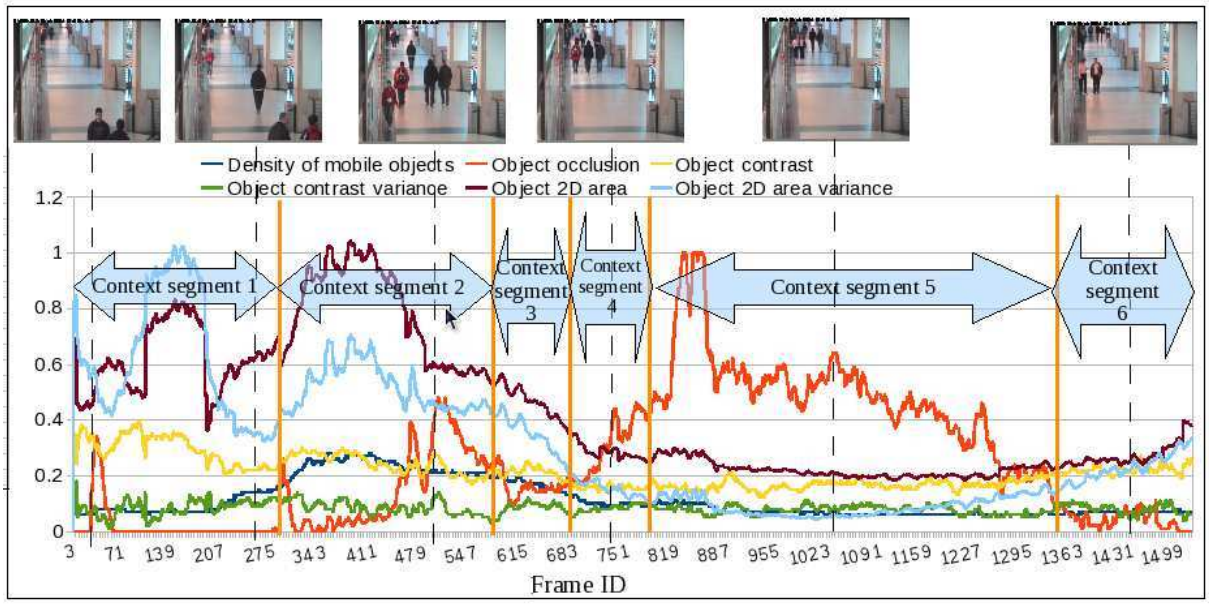


Figure 7.5 – Context segmentation of the sequence ThreePastShop2cor (belonging to the Caviar dataset). The context segments are separated by the vertical orange lines. The control parameters are then learned for each context segment.

a constant value for normalizing them with the other values. The context of this sequence is divided automatically into six context segments. For each context segment, we learn the best satisfactory control parameter values. Table 7.6 presents the learned parameter values for the context segments.

The first context segment is from frame 1 to frame 300. The learned tracking parameters for this context are $w_4 = 0.21$, $w_5 = 0.46$, $w_8 = 0.33$. In this context segment, the object occlusion level is very low. Therefore the color histogram is selected as the most important object descriptor for tracking mobile objects. The object 2D area variance is quite high, it means that the object 2D areas are very different. There exist at the same time objects of high and low areas. So the 2D area is also selected as an object descriptor for tracking. With the existence of objects whose 2D areas are high, the use of dominant color descriptor is reasonable because this descriptor discriminates well the large mobile objects.

In the second context segment (from frame 301 to 600), the density and the occlusion level of mobile objects increase, therefore the dominant color descriptor weight is higher than the one in previous context segment. The similarity score between object dominant color descriptors is combined with the pyramid matching kernel (see description of this descriptor at section 6.1.1.1), therefore it can manage the object occlusion cases.

For the two context segments 3 and 4, the density of mobile objects decreases but their occlusion level is still high. Therefore the dominant color descriptor weight is still selected as

Context segment	From frame	To frame	Learned control parameter values				
			w_3	w_4	w_5	w_7	w_8
1	1	300	0	0.21	0.46	0	0.33
2	301	600	0	0.22	0	0.01	0.77
3	601	700	0	0	0	0	1
4	701	800	0	0	0.17	0	0.83
5	801	1350	0	0	0.34	0.33	0.33
6	1351	1510	0	0.20	0.20	0.20	0.40

Table 7.6 – Learned control parameter values for the sequence ThreePastShop2cor (belonging to Caviar dataset). w_6 is not considered in this experimentation.

the most important descriptor for object tracking. In the context segment 3, the weight value of this descriptor is equal to 1, and in the context segment 4, this value is 0.83.

In the context segment 5, the object 2D areas decrease significantly. While the dominant colors between small objects (i.e. low resolution objects) might be similar, the color histogram descriptor is reliable because this descriptor takes into account all colors belonging to objects. Therefore, in this context segment, the weight of the dominant color descriptor decreases from 0.83 to 0.33, and the color histogram descriptor reliability increases from 0 to 0.33. The color covariance descriptor is used for solving the occlusion cases which occur frequently in this context segment.

In the last context segment, the object area variance is high (approximately equal to the object 2D area values), therefore the object 2D area descriptor is selected with the weight $w_2 = 0.2$.

After segmenting the 12 training videos, we obtain 58 contexts. By applying the clustering process (with the diameter threshold $d = 0.3$, see the algorithm description in table 4.2), 21 context clusters are created. Table 7.7 presents the learned control parameters for each cluster. The shape ratio descriptor is never selected in these context clusters because this descriptor cannot well discriminate the mobile objects. In the cases where there are only few mobile objects and no object occlusion occurrences (e.g. clusters 10, 11, 14, 15, 16, 18, 19), many object descriptors such as color covariance, dominant color, color histogram can be selected for tracking objects. In this case, the color histogram descriptor is selected because this descriptor is more efficient in term of processing time.

- Online Control Phase

+ Controller Experimentation with Automatically Detected Objects

1. Caretaker Video

Context Cluster ID	Parameters				
	w_3	w_4	w_5	w_7	w_8
1	0	0.20	0.14	0.10	0.56
2	0	0.21	0.45	0	0.34
3	0	0.08	0.05	0.12	0.75
4	0	0.12	0.17	0.03	0.68
5	0	0.12	0.16	0.11	0.61
6	0	0.11	0.19	0.07	0.62
7	0	0	0.66	0	0.34
8	0	0.15	0.15	0	0.69
9	0	0.14	0.16	0.17	0.52
10	0	0	1	0	0
11	0	0	1	0	0
12	0	0.05	0.86	0	0.09
13	0	0.14	0.39	0.17	0.3
14	0	0	1	0	0
15	0	0	1	0	0
16	0	0	1	0	0
17	0	1	0	0	0
18	0	0	1	0	0
19	0	0	1	0	0
20	0	0.01	0	0.13	0.86
21	0	0.1	0	0.15	0.75

Table 7.7 – Result of the learning phase. 21 context cluster are created. HOG descriptor (corresponding to w_6) is not considered because its computation is time consuming

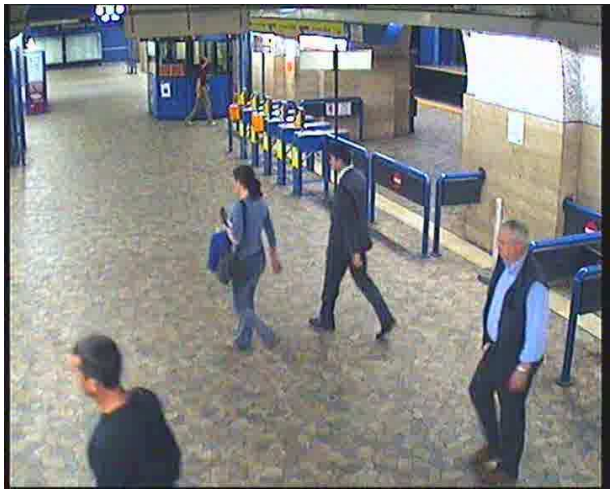


Figure 7.6 – Illustration of the tested Caretaker video 2

The first tested video sequence, denoted Caretaker 2, belongs to the Caretaker project whose video camera is installed in a subway station in Rome. The frame rate of this sequence is 5 fps and the length is 1 hour 42 minutes (see figure 7.6). We use an object detection algorithm given by the Keeneo company (becoming now a part of Digital Barriers plc.)[8]. The graph in figure 7.7 presents the variation of contextual feature values and of the detected contexts in the Caretaker sequence 2 from frame 2950 to frame 3200. The values of object 2D areas are divided by a constant value for normalizing them with the other values. From frame 2950 to 3100, the area and area variance values of objects are very small most of the time (see the brown and light blue curves). It means that the object 2D areas are mostly small. The context of this video chunk belongs to cluster 12. In this cluster, the color histogram is selected as the most important object descriptor for tracking mobile objects ($w_5 = 0.86$). This parameter tuning result is reasonable because compared to the other considered object descriptors, the color histogram descriptor is quite reliable for discriminating and tracking objects of low resolution (i.e. low 2D area). From frame 3101 to 3200, a larger object appears, the context belongs to cluster 9. In this context cluster, the dominant color descriptor weight is the most important ($w_8 = 0.52$). This parameter tuning is very reasonable because for objects of high resolution, their appearances are well visible. The dominant colors can be correctly computed.

Similarly, the graph in figure 7.8 illustrates the video chunk from frame 3775 to 3850. The mobile objects have high resolution, and the context of this video chunk belongs to cluster 9 in which the dominant color descriptor takes the most important role for object tracking.

Table 7.8 presents the tracking results for the Caretaker sequence 2 in two cases : without and with the proposed controller. We find that the proposed controller helps to increase the

[8]http ://www.digitalbarriers.com/

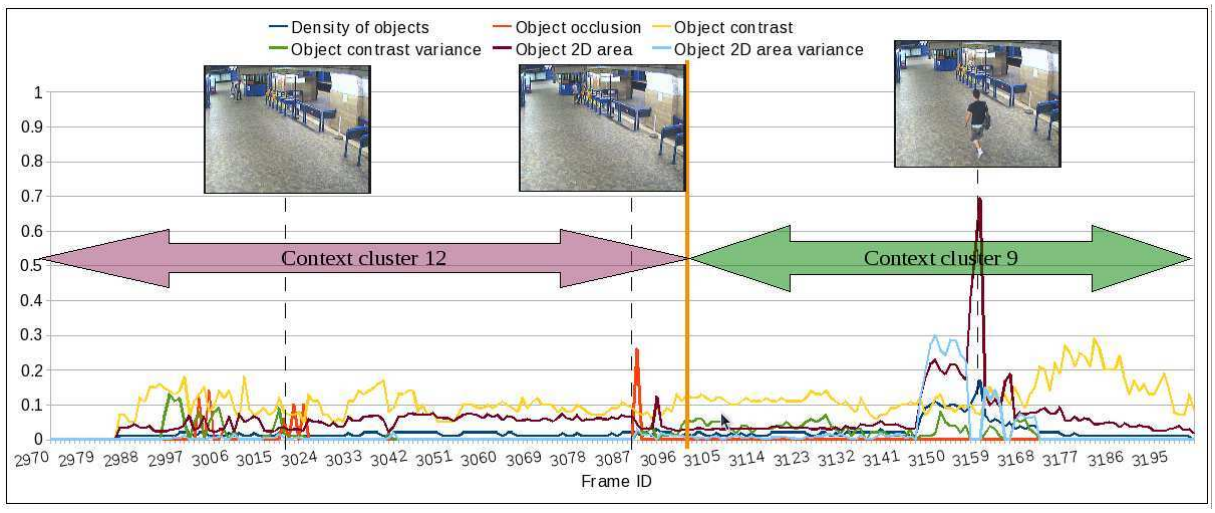


Figure 7.7 – Variations of the contextual feature values and of the detected contexts in the Caretaker sequence 2 from frame 2950 to frame 3350. The values of object 2D areas are divided by a constant value for normalizing them with the other values. For the context cluster 12, the color histogram descriptor is selected as the most important one. This descriptor can discriminate correctly small objects. For the context cluster 9, the dominant color descriptor is selected as the most important one. This descriptor can discriminate correctly large objects.

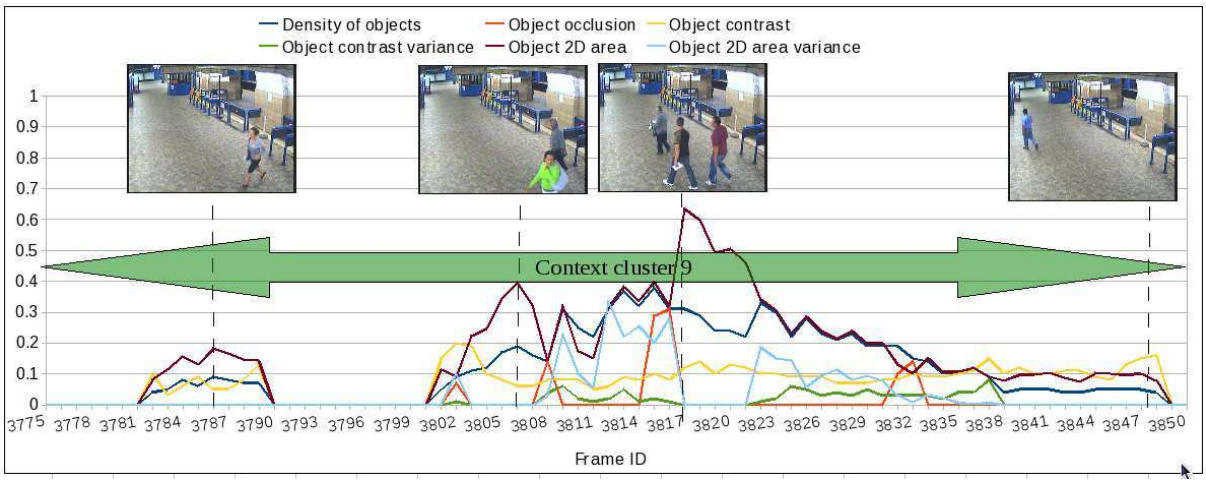


Figure 7.8 – Variations of contextual feature values and of the detected contexts in the Caretaker sequence 2 from frame 3775 to frame 3850. The values of object 2D areas are divided by a constant value for normalizing them with the other values. For the context cluster 9, the dominant color descriptor is selected as the most important one. This descriptor can discriminate correctly large objects and manage object occlusions.

Method	#GT	MT (%)	PT (%)	ML (%)
Appearance-based tracker without the proposed controller	178	61.24	14.04	24.72
Appearance-based tracker with the proposed controller	**178**	**62.92**	**13.48**	**23.60**

Table 7.8 – Tracking results on the Caretaker sequence 2. The proposed controller improves the tracking performance.

Figure 7.9 – Illustration of a tested Caviar video, denoted OneStopMoveEnter2cor

value of MT (i.e. sensitivity) from 61.24% to 62.92% and to decrease the value of PT and of ML (representing false negative). The value of PT decreases from 14.04% to 13.48%. The value of PT decreases from 24.72% to 23.60%. Although this improvement is not too significant, this experiment shows that the proposed controller can help to increase the tracking quality.

2. Caviar Dataset

As presented above, the processing Caviar videos includes 26 sequences. We use 6 sequences for training and the other 20 sequences are tested in the online control phase. For these twenty videos, a HOG-based algorithm is used for detecting people [Corvee and Bremond, 2010].

In table 7.10 at rows 2 and 3, we can find the tracking result for the sequence OneStopMoveEnter2cor belonging to this dataset. With the controller, the precision value increases from 47.06% to 47.37% and the MT value (i.e. sensitivity) increases from 72.73% to 81.82%. Figure 7.9 illustrates this video sequence.

Table 7.9 presents the tracking results on 20 tested videos of the proposed approach in two cases : without and with the proposed controller. We can see that the proposed controller increases significantly the performance of the appearance-based tracker. The MT value increases from 78.3% to 84.6%. The PT value decreases from 16.5% to 10.3%, and the ML value de-

creases from 5.2% to 5.1%. We also present in this table the tracking results from some trackers in state of the art [Wu and Nevatia, 2007, Xing et al., 2009, Huang et al., 2008, Li et al., 2009, Kuo et al., 2010].

The authors in [Wu and Nevatia, 2007], describe a human detection and tracking method based on body part detection. Body part detectors are learned by boosting edgelet feature based weak classifiers. Both the trajectory initialization and termination are based on the evidence collected from the detection responses. Most of the time, data association works to track the objects, while a meanshift tracker fills in the gaps between data association.

In [Xing et al., 2009], the authors present a detection-based object tracking method including three stages. In the first stage, called local stage, a particle filter with observer selection that could deal with partial object occlusion is used to generate a set of reliable tracklets (i.e. short tracks for further analysis). In the second stage, called global stage, the detection responses which are collected from a temporal sliding window to deal with full object occlusion cases, are used to generate a set of potential tracklets. In the last stage, the tracklets generated in the two previous stages are associated by a Hungarian algorithm on a modified pairwise tracklets association cost matrix to get the global optimal association.

The authors in [Huang et al., 2008] present a detection-based three-level hierarchical association approach to track objects in crowded environments from a single camera. At the low level, reliable tracklets are generated by linking detection responses based on conservative affinity constraints. At the middle level, these tracklets are further associated to form longer tracklets based on affinity measures. The association is formulated as a MAP problem and solved by the Hungarian algorithm. At the high level, using the already computed tracklets, the authors estimate entries, exits and scene objects and refine their final trajectories.

In [Li et al., 2009], the authors propose a learning-based hierarchical approach of multi-target tracking from a single camera by progressively associating detection responses into object trajectories. This approach selects automatically suitable features among various features (e.g. motion, appearance, trajectory length) and corresponding non-parametric models, then combines them to maximize the discriminative power on training data using a HybridBoost algorithm.

The authors in [Kuo et al., 2010] propose an algorithm for learning a discriminative appearance model for different mobile objects. First training samples are collected online from object tracklets within a time sliding window based on some spatial-temporal constraints. Then an Adaboost-based learning process is performed to find the most discriminative object local descriptors.

These trackers get quite good performance in this Caviar dataset. Our obtained tracking result with the proposed controller is equal to the best one of these state of the art trackers.

Method	#GT	MT (%)	PT (%)	ML (%)
[Wu and Nevatia, 2007]	140	75.7	17.9	6.4
[Xing et al., 2009]	140	84.3	12.1	3.6
[Huang et al., 2008]	143	78.3	14.7	7.0
[Li et al., 2009]	143	84.6	14.0	1.4
[Kuo et al., 2010]	143	84.6	14.7	0.7
Appearance-based tracker without the proposed controller	140	78.3	16.5	5.2
Appearance-based tracker with the proposed controller	**140**	**84.6**	**10.3**	**5.1**

Table 7.9 – Tracking results on the Caviar dataset. The proposed controller improves significantly the tracking performance.

3. Vanaheim Video

The third tested video sequence belongs to the Vanaheim project (see figure 7.10). This sequence contains 36006 frames and lasts 2h. For this sequence, the proposed controller improves the precision and the sensitivity of the algorithm. The precision value increases from 52.50% to 56.10% and the MT value (i.e. sensitivity) increases from 55.26% to 60.53%. A result summary for this sequence can be found in table 7.10 (at rows 4 and 5).

In all the testing video sequences, the online processing time increases only slightly (less than 10%) when the controller is used.

+ Controller Experimentation with Manually Annotated Objects

All the context feature values (i.e. density, occlusion level of mobile objects, their contrast with regard to the surrounding background and their contrast variance, their 2D area and their 2D area variance) are dependent on the object bounding boxes. The training phase is performed with the annotated objects, so a low quality of the object detection in the online phase makes decrease the correctness of the context detection. Consequently, the quality of the proposed controller can decrease significantly. So, one drawback of the proposed controller is the dependence of its performance to the object detection quality. In this section, we take the objects which are manually annotated as the result of the object detection process for experimenting the controller. This experiment helps to evaluate more correctly the proposed controller performance because we avoid the errors caused by the object detection task. We test two video sequences which are experimented previously to compare have a meaningful comparison. The first one is the OneStopMoveEnter2cor sequence belonging to the Caviar dataset. The second one is the Vanaheim video.

Figure 7.10 – Illustration of the tested Vanaheim video

1. OneStopMoveEnter2cor sequence

Graph in figure 7.11 represents the variation of contextual feature values and the detected contexts in the OneStopMoveEnter2cor sequence (belonging to the Caviar dataset). The context of this sequence is detected as 6 segments.

From frame 1 to 650, there are very few people. The value of the density feature is very low (see the blue curve). No occlusion occurs (see the red line). The context of this video chunk is considered as belonging to context cluster 9. The dominant color descriptor is selected as the most important descriptor for tracking objects in this context with the weight equal 0.52.

From frame 651 to frame 800, some occlusions occur (see the red curve), the context of this video chunk belongs to context cluster 4. In this cluster, the weight of the dominant color descriptor increases up to 0.68. This parameter tuning is reasonable because the dominant color descriptor can manage occlusion cases.

From frame 801 to frame 1000, the object occlusion level decreases. While the large person approaches to the camera position, four small persons go further the camera position. The areas between these five persons are very discriminant (see the light blue curve). The context of this chunk belongs to cluster 1. In this cluster, compared to the previous detected cluster, the weight of 2D area descriptor is more important (increase from 0.12 to 0.20). The weight of dominant color descriptor decreases from 0.68 to 0.55 (due to the decrease of the object occlusion level).

From frame 1001 to frame 2000, the 2D area variance of objects decreases significantly and the occlusion occurs at some instants (e.g. fro frame 1279 to frame 1354 or from frame 1729 to frame 1804), the context of this video chunk belongs to cluster 5. In this context cluster, the weight of dominant color descriptor increases up to 0.6 and the 2D area descriptor weight decreases to 0.12.

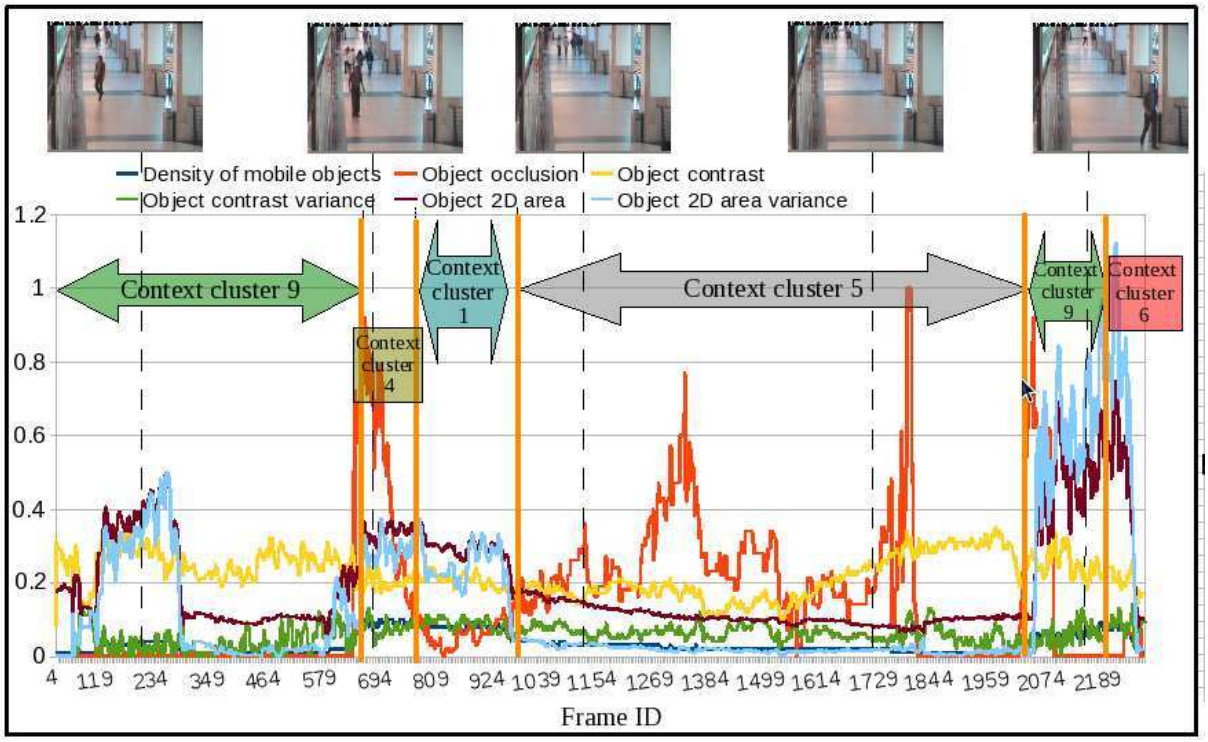


Figure 7.11 – Variation of contextual feature values and of contexts detected in the OneStopMoveEnter2cor sequence (belonging to the Caviar dataset). The values of 2D area is divided by a constant value for normalizing them with the other values. Many contextual variations are detected in this video sequence.

From frame 2001 to frame 2200, the video chunk belongs to cluster 9. In this cluster, the weight of color covariance descriptor is 0.17. Compared to the other object descriptors (e.g. dominant color, color histogram), this descriptor is reliable for tracking small object with occlusion occurrences.

The context of the last video chunk (from frame 2201 to 2237) belongs to context cluster 6. In this video chunk, there is mostly no object occlusion. The color covariance descriptor weight decreases to 0.07.

This video sequence has a lot of context variations. For this sequence, the proposed controller improves significantly the object tracking performance. The MT value increases from 72.73% to 90.91%. The PT value decreases from 27.27% to 9.09%. The precision value increases from 50% to 55.56%. The F-Score increases from 0.59 to 0.69. We can find the tracking result in both cases : without and with the proposed controller in table 7.10 at rows 6 and 7.

2. Vanaheim Video

The second tested video sequence belongs to the Vanaheim project. This sequence is similar

to the Vanaheim video which is presented in the previous experiment. It contains 36006 frames and lasts 2h. For this sequence, the proposed controller improves the precision of the object tracking performance. The precision value when using the controller increases from 92.68% to 100%. The MT value cannot be improved because without the controller, this value is already equal to 100%. The F-Score increases from 0.96 to 1. We can find the tracking result in both cases : without and with the proposed controller in table 7.10 at rows 8 and 9.

Table 7.10 summarizes the obtained tracking results on the OneStopMoveEnter2cor and Vanaheim sequences in two cases : using automatically detected objects and using manually annotated objects. For each case and for each sequence, the tracking results is presented in two parts : without and with the proposed controller. $\mathfrak{O}$ denotes the number of output object trajectories given by the tracking task.

For the OneStopMoveEnter2cor sequence, in the second case (i.e. using manually annotated objects), the controller increases the MT value by 18.18% and increases the precision by 5.56%. In the first case (i.e. using automatically detected objects), these increase values are respectively only 9.09% and 0.31%.

For the Vanaheim sequence, in the second case, the controller increases the precision by 7.32%. In the first case, this increase value is only 3.6%.

From this analysis, we conclude that the improvement of the tracking performance using controller is more significant on manually annotated objects than on automatic detected objects. It means that the object detection quality influences the controller performance. Therefore in order to increase the tracking quality, the controller should also be able to control the object detection task.

7.3.1.2 KLT-based Tracker

For the KLT-based tracker, as analysed above, the minimum distance between KLT feature points $\mathfrak{m}$ (see section 6.3.1) and the dimension of feature window W (see equation 6.52) are selected for experimenting the proposed control method. We have tested this tracker in some videos belonging to the Caviar dataset. Six videos are selected for training and three videos are used for testing.

Training Phase

In the tracking parameter optimization, due to the low number of control parameters, we use the enumerative search to learn their satisfactory values for each context. The minimum distance $\mathfrak{m}$ can get the values 3, 5, 7, 9. The feature window size can get the values 5, 10, 15. After segmenting the six training videos, we obtain 23 contexts. By applying the clustering

	Sequence	Method	#GT	MT(%)	PT(%)	ML(%)	$\mathfrak{O}$	P(%)	F-Score
Using automatically detected objects	OneStopMoveEnter2cor	Without the proposed controller	11	72.73	18.18	9.09	17	47.06	0.57
		With the proposed controller	**11**	**81.82**	**18.18**	**0**	**19**	**47.37**	**0.6**
	Vanaheim	Without the proposed controller	38	55.26	31.58	13.16	40	52.5	0.54
		With the proposed controller	**38**	**60.53**	**36.84**	**2.63**	**41**	**56.1**	**0.58**
Using manually annotated objects	OneStopMoveEnter2cor	Without the proposed controller	11	72.73	27.27	0	16	50.00	0.59
		With the proposed controller	**11**	**90.91**	**9.09**	**0**	**18**	**55.56**	**0.69**
	Vanaheim	Without the proposed controller	38	100	0	0	41	92.68	0.96
		With the proposed controller	**38**	**100**	**0**	**0**	**38**	**100**	**1**

Table 7.10 – Tracking results for OneStopMoveEnter2cor and Vanaheim video sequences in two cases : using detected objects and using annotated objects. $\mathfrak{O}$ denotes the number of object trajectories given by the tracking task. The controller improves the tracking performance more significantly in the second case.

Context Cluster ID	Parameters	
	m	W
1	5	5
2	5	7
3	5	5
4	5	5
5	10	9
6	10	7
7	5	5
8	5	3

Table 7.11 – Result of the training phase of control parameters of the KLT tracker

process (with the diameter threshold $d = 0.3$, see the algorithm description in table 4.2), eight context clusters are created. Table 7.11 presents the learned control parameters for each cluster. Parameter m represents the minimum distance between KLT features and W represents the dimension of the KLT feature window.

Testing Phase

Table 7.12 presents the tracking results for some Caviar videos in both cases : without and with the proposed controller. We find that in these testing videos, the proposed controller increases significantly the tracking performance. This improvement helps the KLT tracker to obtain a better quality than the one of the tracker [Chau et al., 2011b] for two sequences OneStopEnter1cor and OneStopNoEnter2cor. For sequence TwoEnterShop2cor, the KLT tracker is as good as [Chau et al., 2011b].

7.3.2 Evaluation-based Parameter Adaptation Approach

In this section, we experiment the online tracking evaluation algorithm.

7.3.2.1 Online Tracking Evaluation

In order to validate the performance of the proposed online tracking evaluation algorithm (presented at section 5.2.2.2), we compare the output of this algorithm with a F-Score representing the tracking quality compared to the ground-truth data. As the online tracking evaluation score is computed at every frame, this F-Score is also defined at every frame. Given the output of a tracking algorithm and the ground-truth data of the processing video sequence, at a time instant t ($t > 1$), we define the following values :

Sequence	Method	GT	MT (%)	PT (%)	ML (%)
OneStopEnter1cor	Approach of [Chau et al., 2011b]	5	80	20	0
	KLT-based tracker	5	80	20	0
	KLT-based tracker **with the proposed controller**	5	100	0	0
OneStopNoEnter2cor	Approach of [Chau et al., 2011b]	5	60	40	0
	KLT-based tracker	5	60	40	0
	KLT-based tracker **with the proposed controller**	5	80	20	0
TwoEnterShop2cor	Approach of [Chau et al., 2011b]	12	67	25	8
	KLT-based tracker	12	42	0	58
	KLT-based tracker **with the proposed controller**	12	67	25	8

Table 7.12 – KLT-based tracker performance on some Caviar videos in two cases : without and with the proposed controller

- True Positive (TP_t) : the number of correct matches between objects detected at $t-1$ and t.
- False Positive (FP_t) : the number of incorrect matches between objects detected at $t-1$ and t.
- False Negative (FN_t) : the number of missing matches between objects detected at $t-1$ and t.

Using these values, the precision and sensitivity values at t, denoted P_t and S_t, are computed as follows :

$$P_t = \frac{TP_t}{TP_t + FP_t} \tag{7.6}$$

$$S_t = \frac{TP_t}{TP_t + FN_t} \tag{7.7}$$

The instantaneous F-Score at t, denoted FS_t^i, is determined as following :

$$FS_t^i = \frac{2P_tS_t}{P_t + S_t} \tag{7.8}$$

It is important to note that the values of this F-Score and of the proposed online evaluation score do not behave similarly : while the instantaneous F-Score at instant t is only a score computed between instant t and $t-1$, the proposed online evaluation score at instant t takes into account the tracking quality computed on several frames. Therefore, we define an accumulated F-Score at t, denoted FS_t, as following :

$$FS_t = \frac{\sum_{k=1}^{t} FS_k^i}{t} \tag{7.9}$$

Higher this value is, better the tracking quality is. This score is called F-Score in the rest of the thesis.

The variation of the online tracking evaluation score has to be proportional to the F-Score value. In other word, when the F-Score increases (respectively decreases), the online evaluation score increases (respectively decreases) too. In order to verify that, at each instant t we calculate the evaluation assessment value EA_t as follows :

$$EA_t = \begin{cases} 0 & \text{if } (ES_t - ES_{t-1})(FS_t - FS_{t-1}) < 0 \\ 1 - |ES_t - ES_{t-1}| & \text{if } FS_t = FS_{t-1} \\ 1 - |FS_t - FS_{t-1}| & \text{if } ES_t = ES_{t-1} \\ 1 & \text{if } (ES_t - ES_{t-1})(FS_t - FS_{t-1}) > 0 \end{cases} \tag{7.10}$$

where ES_t and FS_t are respectively the online tracking evaluation score (defined in equation 5.14) and the F-Score at t (defined in equation 7.9).

The evaluation assessment can be considered as an evaluation of the online score at each time instant. The value of this metric is in the interval $[0, 1]$. Higher this value is, higher the performance of the online tracking evaluation score is.

In our experimentation, because the number of frames of a sequence is quite great, we only extract some interesting video chunks to analyse the similarity between the evaluation results of our algorithm and the F-Score computed at the same time instant. A global result can be found in table 7.14. The proposed online evaluation algorithm and the supervised evaluation algorithm (i.e. the F-Score computation) are tested and compared with the results obtained by two tracking algorithms. These two trackers are selected in the sate of the art approaches to prove better the robustness of the proposed method. The first tracking algorithm uses a frame to frame technique described in [Avanzi et al., 2005], denoted **F2F tracker**. The second tracker is developed by the Keeneo company, denoted **Keeneo tracker**. The experimentation phase is performed with two different real video sequences (see figure 7.12 and 7.16). The parameters values used in the experimentation are set as follows :

- $Th_6 = \frac{\pi}{2}$ (in formula 5.13)
- $Th_7 = 1m$ (in formula 5.13)

The first video sequence is provided by the Gerhome project. This video contains one person in the scene (see figure 7.12). The frame rate of this sequence is 8 frames per second and the length is 5 minutes.

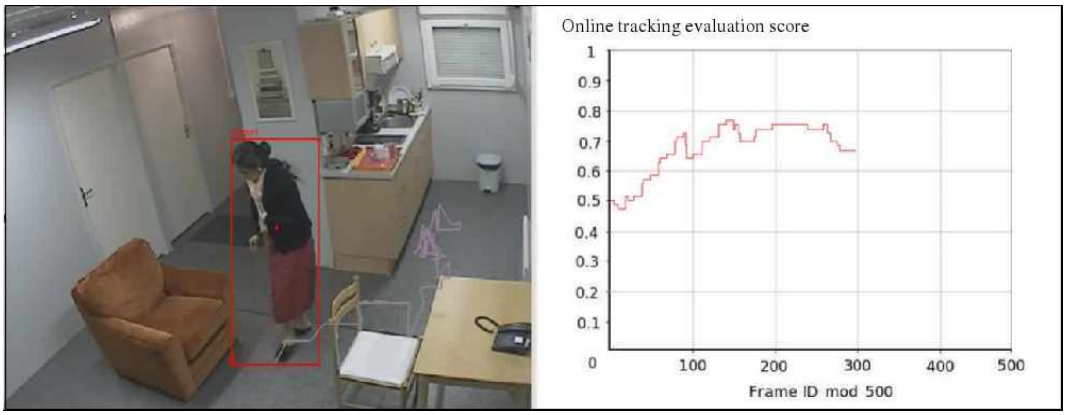


Figure 7.12 – Online tracking evaluation of the Gerhome sequence. In the left image, the person movement is tracked by the tracking algorithm 3. The right image shows the online evaluation score of the F2F tracker.

In this online evaluation algorithm, γ^i ($i = 1..2$, presented in formula 5.3), γ_1^i and γ_2^i ($i = 3..7$, presented in formula 5.4) are the cooling factors for increasing and decreasing the score of the evaluation feature i. In order to understand their roles, we compute the online tracking evaluation score with different value sets of γ.

For the last five features ($i = 3..7$), the values of γ_1^i are empirically set much lower than γ_2^i because they are not symmetric. For example, a mobile object whose only 30% time is wrongly tracked, can be induced as an incorrect tracked object. Figure 7.13 represents the online tracking evaluation score values of the F2F tracker for the Gerhome sequence with three different γ value sets (see table 7.13). We can see that the variation of the online tracking evaluation score values are very similar in all three cases. They increase or decrease simultaneously. However higher the value of γ is, faster the variation of the online score is. In other word, the reaction of the online score is proportional to the values of γ. We select the γ values of second test to continue the remaining experimentation.

Parameter	$\gamma^i \ (i = 1, 2)$	$\gamma_1^i \ (i = 3..7)$	$\gamma_2^i \ (i = 3..7)$
Test 1	0.3	0.015	0.3
Test 2	0.1	0.005	0.1
Test 3	0.05	0.0025	0.05

Table 7.13 – The different values of γ used for testing (value of i denotes the evaluation feature index)

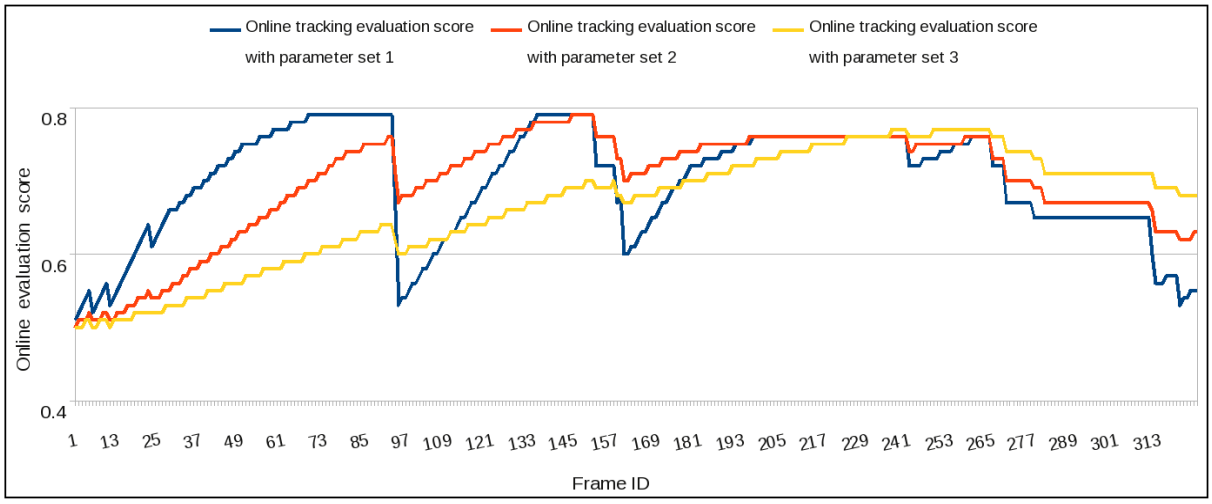


Figure 7.13 – Online tracking evaluation scores with different values of γ. The blue line corresponds to the values of γ in test 1, the red line corresponding to the values of γ in test 2, the yellow line corresponding to the values of γ in test 3. The variations of the online tracking evaluation score values are very similar in all three cases. Higher the value of γ is, faster the variation of the online score is.

The graph in figure 7.14 represents the online tracking evaluation score (red line), the F-Score (red dotted line) of the F2F tracker and the corresponding evaluation assessment (blue line) from frame 1 to frame 350. We remark that when the F-Score decreases (from frame 151 to 163 and from frame 265 to 283 for example), the online evaluation score goes down

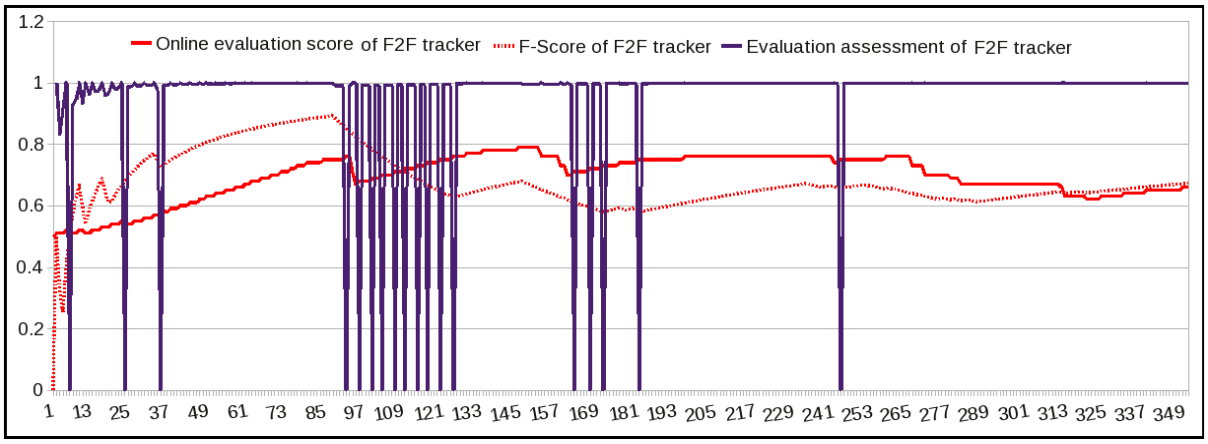


Figure 7.14 – Online evaluation score (red line), F-Score (red dotted line) and evaluation assessment of the F2F tracker for the Gerhome sequence. The evaluation assessment values are high in most of the time.

too. When the F-Score value increases, the online evaluation score generally increases too (for example from frame 37 to frame 90 or from frame 133 to frame 150).

When the online tracking evaluation score and the F-Score are coherent (i.e. increase or decrease simultaneously), the evaluation assessment value is high (e.g. from frame 37 to 85, or from frame 193 to frame 241) and otherwise it is low. For example, from frame 97 or 130, while the F-Score decreases, the online evaluation score increases. As consequence, the evaluation assessment value is very low at these frames. However in most of the time, this value is very high.

The top graph in figure 7.15 presents the online evaluation scores of the F2F tracker (red line) and of the Keeneo tracker (green line). The bottom graph in figure 7.15 shows the F-Scores of the F2F tracker (red dotted line) and of the Keeneo tracker (green dotted line). We can see that the online evaluation score values of the F2F tracker are mostly higher than the ones of the Keeneo tracker. The F-Scores of these two algorithms also indicate this order : the red dotted line is always above the dotted green line.

The second tested video concerns the movements of people in a subway station. This sequence is the Caretaker sequence 1 which is presented in section 7.2.5.1 (see figure 7.16). In this video sequence, the people number in the scene is much greater than the previous sequence. People occlusion occurs quite frequently. The frame rate of this sequence is 5 frames per second and the length is 3 min 20 sec.

The top graph in figure 7.17 presents the online evaluation scores of the F2F tracker (red line) and of the Keeneo tracker (green line) from frame 1 to frame 525 of the Caretaker sequence 1. The bottom graph in figure 7.17 shows the F-Scores of the F2F tracker (red dotted line) and of the Keeneo tracker (green dotted line). We can see that the online evaluation score

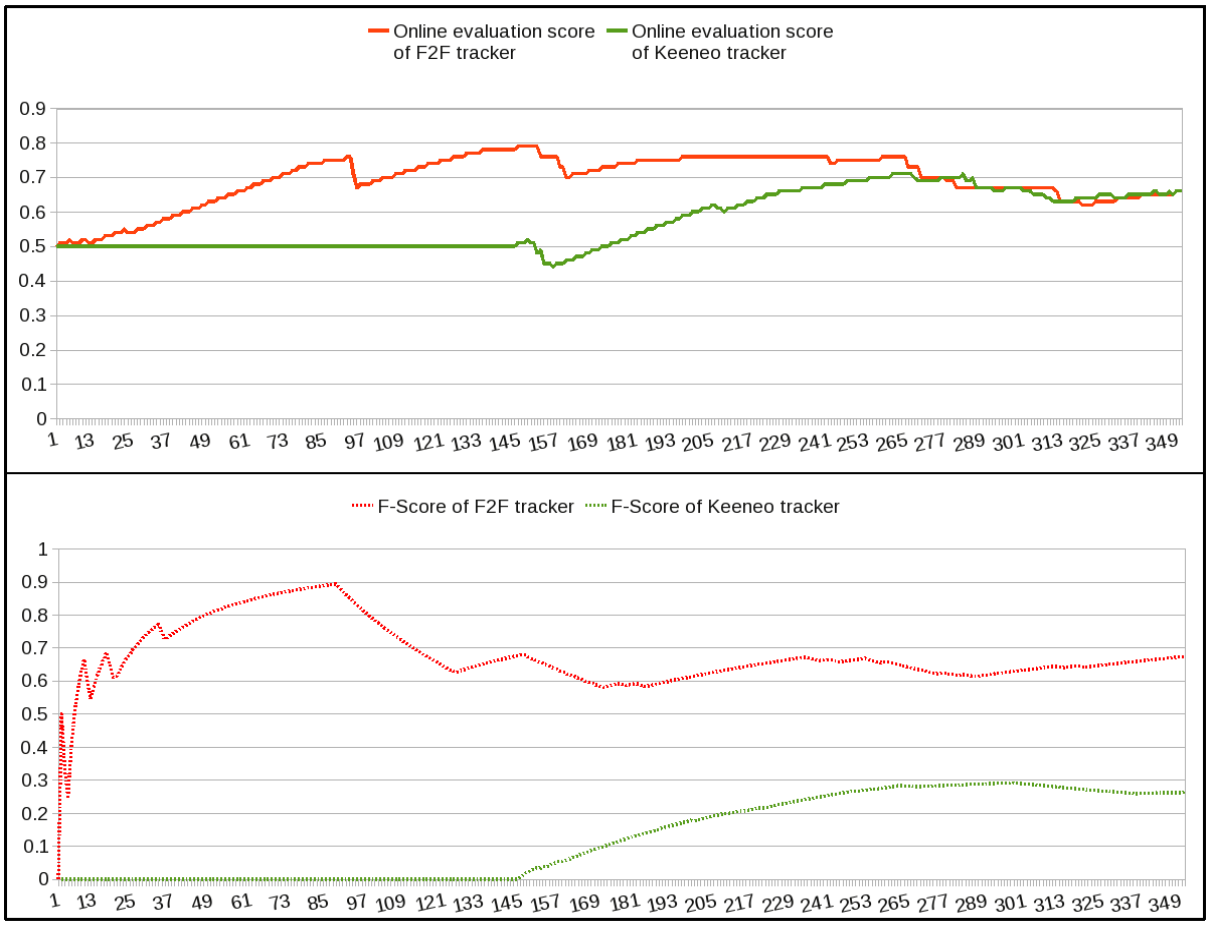


Figure 7.15 – The top graph presents the online evaluation scores of the F2F tracker (red line) and of the Keeneo tracker (green line). The bottom graph shows the F-Scores of the F2F tracker (red dotted line) and of the Keeneo tracker (green dotted line). The online evaluation score values of the F2F tracker are mostly higher than the ones of the Keeneo tracker. The F-Scores of these two algorithms also indicate this order.

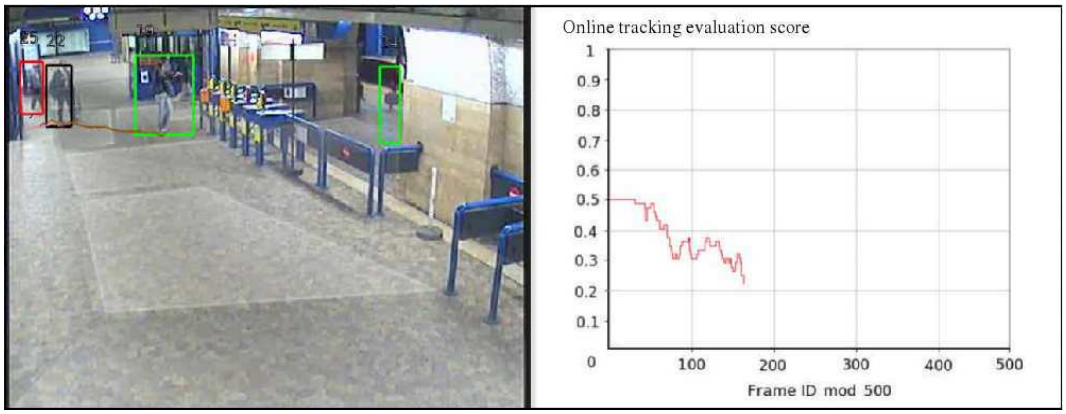


Figure 7.16 – Online tracking evaluation of the Caretaker sequence 1

values of the F2F tracker are lower than the ones of the Keeneo tracker. The F-Scores of these two algorithms also indicate this order : the red dotted line is always under the dotted green line.

A result summary of this experiment can be found in table 7.14. In this table, we present the mean values of online tracking evaluation score, F-Score and evaluation assessment (denoted Eval. asses.) values of each tracking algorithm for whole testing sequence. Some conclusions are given :

- If the F-Score of the F2F tracker is greater (respectively lower) than the F-Score of the Keeneo tracker, the online tracking evaluation score of the F2F tracker is also greater (respectively lower).

- If the F-Score of a tracking algorithm is high (e.g. greater than 0.5) (respectively low) then the online tracking evaluation score of this tracking algorithm is also high (respectively low).

- The value of evaluation assessment is very high in all cases.

		F2F tracker	Keeneo tracker
Gerhome video	Online evaluation score	0.67	0.63
	F-Score	0.84	0.51
	Eval. asses.	0.93	0.94
Caretaker video 1	Online evaluation score	0.22	0.39
	F-Score	0.26	0.35
	Eval. asses.	0.77	0.78

Table 7.14 – Summary of the evaluation results

In conclusion, the results of the proposed online evaluation method are compatible with the output of the offline evaluation method using ground-truth data. This experimentation validates

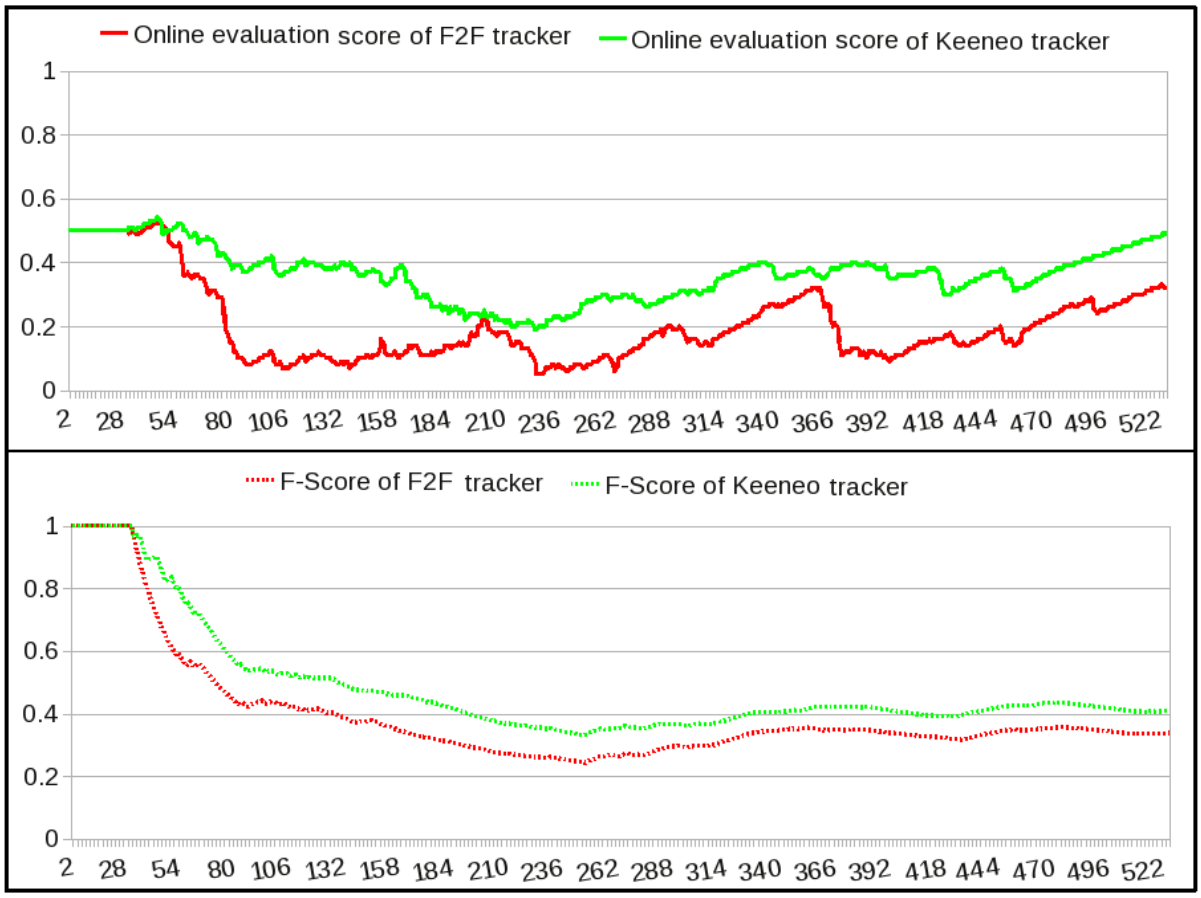


Figure 7.17 – The top graph presents the online evaluation scores of the F2F tracker (red line) and of the Keeneo tracker (green line) from frame 1 to frame 525 of the Caretaker sequence 1. The bottom graph shows the F-Scores of the F2F tracker (red dotted line) and of the Keeneo tracker (green dotted line). The online evaluation score values of the F2F tracker are lower than the ones of the Keeneo tracker. The F-Scores of these two algorithms also indicate this order.

the performance of the proposed evaluation algorithm.

7.4 Conclusion

In this chapter, we have presented the experimental results of the two proposed tracking algorithms and of the proposed control method. For the tracker experimentation, we have tested some complex videos which have frequently object occlusion, low object contrast, low object resolution. The performance of these two tracking algorithms have been compared with the other seven tracker on two videos belonging to the ETISEO dataset. The comparison result shows that the two proposed trackers have better performance than these seven trackers for

most of the evaluation metrics.

The proposed control method includes two approaches : context-based parameter adaptation and evaluation-based parameter adaptation. For the first approach, we have experimented two long video sequences (belonging to the Caretaker and Vanaheim projects) and the Caviar public dataset. The experimental results show that the proposed controller improves the tracking quality. The obtained tracking result is as good as the ones in the state of the art. The tracking quality is high enough so that object trajectories can be used for action recognition.

The performance of the proposed controller is dependent on the object detection quality and the variation of the tracking contexts. For video sequences with good object detection and strong context variations, the proposed controller improves significantly the object tracking performance.

For the second approach, we experiment the online tracking evaluation algorithm. The result of this algorithm is compared with the evaluation metrics using ground-truth data. The results show that the output of the proposed online evaluation algorithm is coherent with the supervised evaluation (i.e. using ground-truth data).

8

CONCLUSION AND FUTURE WORK

In this chapter, we will present in the first part the conclusion of this manuscript. We start by summarizing the contributions of the manuscript. Then, the limitations in this study will be analysed. In the second part, we will discuss about the future work which includes short-term and long-term perspectives.

8.1 Conclusion

In this thesis, we present a new control approach for mobile object tracking. More precisely in order to cope with the tracking context variations, this approach learns how to tune the parameters of object appearance-based tracking algorithms. The tracking context of a video sequence is defined as a set of six features : density of mobile objects, their occlusion level, their contrast with regard to the background, their contrast variance, their 2D area and their 2D area variance.

In an offline supervised learning phase, we learn the best satisfactory tracking parameters for context clusters. In the online control phase, two approaches are proposed. In the first one, once a context change is detected, the tracking parameters are tuned using the learned values. In the second approach, the parameter tuning is performed when the context changes and the tracking quality (computed by an online evaluation algorithm) is not good enough. An online learning process enables to update the context/parameter relations.

The approach has been experimented on long, complex videos and on three public video datasets (ETISEO, Caviar and TRECVid).

8.1.1 Contributions

This thesis brings the five following contributions over the state of the art :

8.1.1.1 A New Video Classification Method for Learning Offline Tracking Parameters

In the state of the art, many approaches have been proposed to improve the tracking quality, but most of them either use prior scene information [Chau et al., 2009a] or are only specific for their tracking algorithms [Kuo et al., 2010, Santner et al., 2010]. Therefore these approaches cannot be generic enough for different tracking algorithms or video scenes. In this work, we define the notion of "context" of a video chunk that includes a set of six feature values influencing the tracking quality. These six contextual features are : the density, the occlusion level of mobile objects appearing in this video sequence, their contrast with regard to the surrounding background, their contrast variance, their 2D area and their 2D area variance. Each contextual feature is represented by a code-book model whose codewords describe the values of the corresponding feature. Thanks to this context notion, we can divide a video sequence into chunks of uniform feature values. Video chunks are classified by clustering their contextual features to create context clusters. The best satisfactory parameter values are learned for each context cluster. Therefore, the proposed control method can be applied for any scene type and for several tracking algorithms whose quality is influenced by the defined contextual features.

8.1.1.2 An Online Tracking Evaluation Algorithm

In order to tune and learn online the tracking parameters, an online tracking quality evaluation is necessary. Therefore we propose in this thesis an algorithm to evaluate online the performance of tracking algorithms. In this work, the trajectory confidence is defined in function of seven evaluation features. These evaluation features are classified into two groups : trajectory and instantaneous features. The trajectory features include exit zone and temporal length. In the instantaneous feature group, we define the five following features : shape ratio, area, displacement distance, color histogram and direction. This algorithm gives as output a score estimating online the tracking performance over time. The advantages of the approach over the existing state of the art approaches are : (1) little prior knowledge is required (only exit zones in the camera view are required), (2) the method can be applied in complex scenes containing several mobile objects.

8.1.1.3 A Method to Tune and Learn Online the Tracking Parameters

While parameter tuning for static image applications or for object detection in videos has been largely studied in the state of the art [Martin and Thonnat, 2008, Bhanu and Das, 1995,

Nghiem, 2009], online parameter tuning for the tracking task is not really addressed. In this thesis, we propose a controller which is able to tune and learn online the tracking parameters thanks to a learned database resulting from a supervised or an unsupervised learning phase. This controller includes two stages : initial parameter configuration and parameter adaptation. The objective of the first stage is to set values for control parameters at the first frames. The second stage is responsible for tuning the control parameters over time to cope with the tracking context variation. For the second stage, we propose two approaches. The first one, called context-based parameter adaptation, relies completely on the context detected for every video chunk to activate the parameter tuning process. The second approach, called evaluation-based parameter adaptation, relies on the detected context and on an online tracking quality evaluation to decide whether to activate the parameter tuning process. This approach defines also an unsupervised learning process to learn online the satisfactory tracking parameters.

8.1.1.4 A Tunable Object Descriptor-based Tracking Algorithm Enabling Adaptation to Scene Conditions

Object appearance is widely used in the state of the art for object tracking [Zhou et al., 2006, Kwolek, 2009, Bak et al., 2009, Monari et al., 2009] but most of them cannot perform robustly in different contexts. We propose in this manuscript a tracking algorithm which is effective in different scene contexts. First an object descriptor pool is used to compute the matching score between two detected objects. This pool includes 2D, 3D positions, 2D sizes, color histogram, histogram of oriented gradient (HOG), color covariance and dominant color descriptors. In the tracking process, a temporal window is defined to establish the matching links between the detected objects. This enables to find the object trajectories even if the objects are misdetected in some frames. A trajectory filter is defined to remove trajectories considered as noise. The object descriptor weights have a strong influence on the tracker quality. We propose to use an Adaboost algorithm in which each descriptor plays the role as a weak classifier to learn offline their values for each tracking context. The descriptor weights can be tuned online to cope with the contextual variation. This tracking algorithm brings some contributions over the state of the art trackers : (1) a robust tracking algorithm based on an object descriptor pool, (2) a new method to quantify the reliability of HOG descriptor, (4) a combination of color covariance and dominant color descriptors with a spatial pyramid kernel to manage the case of object occlusion.

8.1.1.5 A Tracking Algorithm based on Kalman Filter and Global Tracking

In this work, we propose an algorithm to track mobile objects based on their trajectory properties. The proposed tracker includes two stages : tracking and global tracking. The tracking

stage follows the steps of a Kalman filter including estimation, measurement and correction. First for each tracked object, its state including position and 2D bounding box is estimated by a Kalman filter. Second, in the measurement step, this tracked object searches for the best matching object based on four descriptors : 2D position, 2D area, 2D shape ratio and color histogram. Third, the best matching object and its estimated state are combined to update the position and 2D bounding box sizes of the tracked object. However, the mobile object trajectories are usually fragmented because of occlusions and misdetections. Therefore, the global tracking stage aims at fusing the fragmented trajectories belonging to the same mobile object and removing the noisy trajectories. The advantages of our approach over the existing state of the art ones are : (1) no prior knowledge information is required (e.g. no calibration and no contextual models are needed), (2) the tracker can be effective in different scene conditions : single/several mobile objects, weak/strong illumination, indoor/outdoor scenes, (3) a global tracking stage is defined to improve the object tracking performance.

8.1.2 Limitations

There are some limitations in the proposed contributions. We divide these limitations into two groups : theoretical and experimental ones.

8.1.2.1 Theoretical Limitations

- **The performance of the proposed controller is dependent on the quality of the object detection task :** All the context feature values (i.e. density, occlusion level of mobile objects, their 2D area, 2D area variance, their contrast with regard to the surrounding background and their contrast variance) are defined in function of the object bounding boxes. The training phase is performed with annotated objects, so a low quality of object detection in the online phase decreases the correctness of the context detection. Consequently, the quality of the proposed controller can decrease significantly. So, the proposed controller performance is dependent on the object detection quality.
- **The contextual features are not accurate enough to determine the tracking parameter values :** The six proposed contextual features are quite general. This makes the relation between a context and its best satisfactory tracking parameters becoming weak. It means that a context might have different best satisfactory tracking parameter values. Although we have proposed a clustering task in which the learned tracking parameter values are verified, we need a large training video dataset to obtain a reliable learned database. We miss a mathematical model to verify the correctness of the proposed contextual features and to find the new ones.
- **The proposed approach limits itself at controlling the tracking task :** In fact in many

cases, the tracking errors are caused principally by the errors of the tasks at lower levels such as object detection, image segmentation. The tracking control cannot solve these errors.

8.1.2.2 Experimental Limitations

- **Some tracking contexts are not tested :** Although experimentations have been performed with many video scenes of different contexts, some context types are not yet considered such as the change of object contrast over time. Moreover, some other video types are not addressed in the experimentation such as outside videos or videos of vehicles.
- **Experiments on the evaluation-based parameter adaptation approach are not performed entirely :** In this work, we have presented two approaches for controlling tracking algorithms. While the first approach (context-based parameter adaptation) has been completely validated experimentally, the experimentation of the second one (evaluation-based parameter adaptation) is only performed for the online object tracking evaluation algorithm. Moreover, this experimentation is only done for few video sequences.

8.2 Future Work

This research topic opens many interesting future work. In this section, we propose short-term and long-term perspectives as follows.

8.2.1 Short-term Perspectives

- **Add contextual features :** As mentioned above, the proposed contextual features might not be sufficient enough to determine the tracking parameter values. Some features can be added to better describe the context such as the color variance of mobile objects or the complexity of object trajectories.
- **Improve the online tracking evaluation algorithm** : This algorithm plays an important role during the online control process. It can help to detect tracking errors and classify these errors. This information is very useful to find a suitable tracking control strategy. The proposed online tracking evaluation algorithm is still simple and is quite dependent on the detection quality. We can improve it by adding more evaluation features (e.g. object color covariance, HOG, contour) and combine them with other techniques such as reverse tracking [SanMiguel et al., 2010].

8.2.2 Long-term Perspectives

- **Define a mathematical model to verify the contextual features :** In this work, the contextual features are found thanks to an experimental analysis. This study contains potential risks and is not rigorous. The existence of a mathematical model can remove these limitations. This mathematical model can also help to identify new contextual features which cannot be detected intuitively.
- **Switch online trackers :** In this work, we study the parameter tuning of a tracking algorithm. However the quality of a tracking algorithm is dependent on the video context. A tracker can only perform well in some contexts. Therefore in many cases, the parameter tuning cannot improve the tracking performance. If the controller is able to switch the trackers during the object tracking process, the tracking performance can increase significantly.
- **Control tracker over scene space :** In the proposed controller, the tracking parameters are tuned over time but not over space. In fact, at a same moment, the distribution of mobile objects (e.g occlusion level, density) as well as their properties (e.g. color, size) can be different over space. In that case, we need a parameter tuning mechanism over scene. It means that the tracking parameters change over the tracked objects, their location.
- **Quantify the reliability of execution steps :** All steps belonging to the proposed control method (e.g. context detection, satisfactory control parameter learning) can contain errors and uncertainties. If the outputs of these steps are associated with values representing their reliability, we can estimate the reliability of the controller, detect to which step errors belong and propose effective improvements.
- **Define contextual features independent of the object detection task :** As analysed above, one of the drawbacks of the proposed controller is the dependence of its performance on the object detection quality. This drawback limits significantly the capability of the proposed controller. If the contextual features are less dependent on the algorithm output, the controller performance will increase.
- **Evaluate and control online the algorithms belonging to lower levels** : The origin of an error in the tracking process can be in lower levels such as object detection and image segmentation. Therefore if we only consider or focus on the tracking task, it will not solve all errors. We need to propose methods to repair the errors detected at these tasks.

RÉSUMÉ SUBSTANTIEL

1. Introduction

De nos jours, la vidéo surveillance est utilisée dans un grand nombre d'endroits : au parking, au musée, à l'hôpital, à la gare ... La vidéo surveillance nous permet d'observer des positions différentes et sous différents angles simultanément et de reconnaître rapidement les événements d'intérêt qui se passent dans la scène. Cependant, comment un agent de sécurité peut-il analyser en temps réel des dizaines d'écrans de contrôle avec un risque d'erreur minimal pendant des heures ? D'autre part, l'observation des écrans pendant une longue période est un travail ennuyeux. Une solution pour résoudre ces problèmes est d'appliquer la surveillance vidéo intelligente. La vidéo intelligente est un nouveau domaine qui étudie ce qui se passe dans une vidéo. Ce terme exprime un axe de recherche assez large, qui est appliqué dans de différents domaines : la robotique, la médecine et la sécurité par exemple. Beaucoup de recherches sont investies depuis une vingtaine années. Un système d'interprétation de vidéo pour la reconnaissance d'activités comprend généralement les tâches suivantes : acquisition d'images, détection d'objets, classification d'objets, suivie d'objets et reconnaissance d'activités. Ce manuscrit présente des études pour le suivi d'objets mobiles.

La qualité d'un algorithme de suivi est influencée par les tâches réalisées au couches plus basses (la détection d'objets, classification d'objets) et par certaines caractéristiques de vidéos comme la complexité des mouvements des objets, l'intensité d'illumination, contraste d'objet par rapport son image de fond et niveau d'occultation des objets. C'est pour cela que la performance d'un algorithme de suivi est dépendante de la scène considérée. En particulier, pour une séquence longue ou il y a une variation fréquente de ces caractéristiques, la qualité de

suivi peut décroître considérablement. Deux questions ouvertes : 1. Comment on peut suivre le mouvement des objets mobiles dans ces cas complexes ? 2. Comment l'utilisateur peut régler les paramètres de suivi pour obtenir une meilleure qualité de suivi ?

1.1. Objectif et Hypothèses

1.1.1. Objectif

Dans ce manuscrit de thèse, nous proposons une méthode pour contrôler des algorithmes de suivi. Cette méthode est capable d'adapter en ligne le processus de suivi au cours de variations contextuelles d'une séquence longue. La méthode proposée doit satisfaire ces trois propriétés suivantes : être **générique, flexible** et **intelligente**.

Le terme «générique» signifie que notre méthode peut gérer des algorithmes de suivi différentes. Dans ce travail, notre objectif est de contrôler des algorithmes de suivi basés sur l'apparence ou sur des points d'intérêts sur des objets. Ces algorithmes de suivi sont choisis parce que leurs approches sont largement étudiées dans l'état de l'art. Le terme «flexible» implique que la structure de l'algorithme de contrôle proposée peut être appliquée pour contrôler d'autres catégories d'algorithme de suivi (par exemple le suivi basé sur le contour de l'objet). Le terme «intelligente» signifie que cette approche a moins besoin d'interaction humaine que les autres méthodes de contrôle dans l'état de l'art.

1.1.2. Hypothèses

La méthode de contrôle présentée dans ce manuscrit se fait avec les cinq hypothèses suivantes :

1. Les algorithmes de suivi considérés ont au moins un paramètre réglable qui influence significativement la performance de suivi.
2. Il existe un ensemble de vidéos d'apprentissage qui représente les différents contextes vidéo rencontrés en surveillance vidéo.
3. Il existe un certain nombre de contextes qui ont un impact sur la qualité de suivi.
4. La vidéo d'entrée est générée par une caméra monoculaire.

1.2. Contributions

Par rapport à l'état de l'art, cette thèse apporte des cinq contributions significatives parmi lesquelles, les trois premières contributions relient à la méthode de contrôle, les deux dernières concernent le suivi des objets.

1. Une nouvelle méthode de classification pour apprendre hors ligne des paramètres de suivi.
2. Un algorithme d'évaluation de suivi en ligne (publié à ICDP 2009)
3. Une nouvelle approche pour régler et apprendre en ligne des paramètres de suivi (publié à ICIP 2012).
4. Un algorithme de suivi d'objets basé sur des descripteurs d'apparence pouvant s'adapter aux différentes conditions de la scène (publié à ICDP 2011 et au workshop de ICVS 2011).
5. Un algorithme de suivi basé sur le filtre de Kalman et le suivi global (publié à VISAPP 2011)

Nous avons testé la méthode de contrôle et des algorithmes de suivi proposés sur les trois datasets de vidéos publiques : Caviar, ETISEO, TRECVid et aussi sur des vidéos longues et complexes appartenant à des projets : Gerhome, Caretaker et Vanaheim.

2. Les Chapitres du Manuscrit

Cette thèse présente une nouvelle approche pour contrôler des algorithmes de suivi d'objets mobiles. Plus précisément, afin de s'adapter aux variations contextuelles de suivi, cette approche apprend à régler les paramètres des algorithmes de suivi d'objets basés sur l'apparence. Le contexte de suivi d'une vidéo est défini par un ensemble de caractéristiques : la densité des objets mobiles, leur niveau d'occultation, leur contraste et leur surface 2D. Dans une phase d'apprentissage supervisée, des valeurs de paramètres sont déterminées pour chaque séquence d'apprentissage. Puis ces séquences sont classifiées en groupant leurs caractéristiques contextuelles. A chaque contexte sont associées les valeurs de paramètres apprises. Dans la phase de contrôle en ligne, deux approches sont proposées. Pour la première approche, quand le contexte change, les paramètres sont réglés en utilisant les valeurs apprises. Pour la deuxième, le réglage des paramètres est réalisé quand le contexte change et quand la qualité de suivi (estimée par notre algorithme d'évaluation en ligne) n'est pas assez bonne. Un processus d'apprentissage

en-ligne met à jour les relations contextes/paramètres. L'approche a été expérimentée avec de longues, complexes vidéos ainsi qu'avec plusieurs ensembles de vidéos publiques.

Le manuscrit est organisé comme suit.

Le chapitre 1 présente l'introduction, l'objectif, les hypothèses et aussi les contributions du manuscrit de thèse.

Le chapitre 2 présente tout d'abord un état de l'art dans le domaine du suivi d'objet. Les algorithmes de suivi sont classifiés en trois catégories : le suivi des points d'intérêt, le suivi d'apparence des objets et le suivi de silhouette des objets. Deuxièmement, nous décrivons un état de l'art des méthodes de contrôle dont le but est d'adapter les tâches de vision aux variations contextuelles d'images ou de séquences vidéo. Ces méthodes sont divisées en trois catégories différentes : méthodes basées hors ligne, celles basées en ligne et celles hybrides (combinant les deux méthodes précédentes).

Le chapitre 3 présente une description générale de la méthode de contrôle proposée comprenant deux phases : l'apprentissage hors ligne et le contrôle en ligne. Nous décrivons le processus d'exécution étape par étape dans les deux phases (hors ligne et en ligne).

Le chapitre 4 décrit en détail les étapes d'exécution de la phase d'apprentissage hors ligne de la méthode proposée. L'objectif de cette phase d'apprentissage est de créer ou de mettre à jour une base de données d'apprentissage qui est utilisée pour contrôler un algorithme de suivi. Cette base contient les valeurs de paramètres les mieux adaptées de l'algorithme de suivi pour des conditions variées de la scène en fonction de la densité, du niveau d'occultation d'objets mobiles, du contraste de l'objet par rapport à son image de fond, de leur variance de contraste, de leur surface 2D et de la variance de surface 2D. Les valeurs des paramètres apprises sont associées à deux valeurs représentant leur fiabilité.

Cette phase d'apprentissage prend comme entrée les vidéos, des objets annotés (y compris leurs positions 2D et leurs tailles des boîte encadrées 2D), leurs trajectoires annotées, un algorithme de suivi et ses paramètres contrôlables. Le terme "ses paramètres contrôlables" réfère aux paramètres qui sont pris en compte dans le processus de contrôle (c-à-d être considérées pour obtenir les meilleurs valeurs durant la phase d'apprentissage et d'être réglées dans la phase en ligne). Pour chaque paramètre contrôlable, cette phase nécessite un nom, une plage de valeur et un "pas de valeur". Le "pas de valeur" d'un paramètre est défini comme la variation minimale qui peut provoquer une variation significative de la qualité de suivi. Cette valeur permet d'éviter un parcours trop exhaustif de l'espace du paramètre lors de la recherche de

sa valeur donnant la qualité satisfaisante de suivi. Dans ce travail, nous ne considérons que les paramètres numériques, cependant, la méthode proposée peut être appliquée aussi à des paramètres symboliques.

À la fin de la phase d'apprentissage, une base de données apprises est créé (si c'est la première session d'apprentissage) ou est mise à jour (si ce n'est pas le cas). Une session d'apprentissage peut traiter de nombreuses séquences vidéo.

Le chapitre commence par une étude de l'influence de certaines caractéristiques contextuelles à la performance du processus de suivi d'objet. Ensuite, nous décrivons les pas d'exécution du processus d'apprentissage hors ligne, y compris l'extraction des caractéristiques contextuelles, la segmentation de contexte et la modélisation de code-book, l'optimisation des paramètres de suivi, le groupement (y compris le groupement contextuel et le calcul des paramètres de suivi pour les groupes de contextes).

Le chapitre 5 décrit le processus de contrôle en ligne. L'objectif de ce processus est de régler et d'apprendre de façon non supervisée les paramètres de suivi pour adapter la tâche de suivi aux variations du contexte en utilisant la base de données apprises. Il comprend deux étapes : la configuration initiale des paramètres et l'adaptation des paramètres. L'objectif de la première étape consiste à déterminer des valeurs pour les paramètres contrôlables (c-à-d paramètres de l'algorithme de suivi) dans les premières trames de la séquence vidéo testée et des paramètres du contrôleur. La deuxième étape a pour tâche de régler et d'apprendre des paramètres contrôlables au cours du temps pour les adapter à la variation du contexte de suivi. Pour l'étape d'adaptation des paramètres, nous présentons deux approches appelées : adaptation des paramètres basée sur le contexte et adaptation des paramètres basée sur l'évaluation.

Pour l'approche basée sur le contexte, l'objectif est de détecter les changements de contextes et de régler en ligne les paramètres contrôlables de suivi pour les adapter à ces changements. Cette approche prend comme entrée le flux vidéo testé, une liste des objets détectés à chaque trame, la base de données apprise et donne comme sortie les valeurs des paramètres les mieux adaptées pour chaque nouveau contexte détecté. L'approche comprend donc deux sous-étapes : détection de contexte et adaptation des paramètres.

Pour l'approche basée sur l'évaluation, l'objectif est de régler et d'apprendre en ligne les paramètres contrôlables en se basant sur les changements du contexte et sur la qualité du processus de suivi. Généralement, le principe de cette approche est similaire à l'approche précédente. Cependant, nous ajoutons dans cette approche une méthode pour estimer en ligne

la qualité de suivi. Cette approche comprend donc quatre sous-étapes : détection de contexte, évaluation en ligne de qualité de suivi, réglage et apprentissage des paramètres, groupement de contextes et de paramètres.

Le chapitre 6 présente trois algorithmes de suivi utilisés pour tester le contrôleur proposé. Le premier algorithme de suivi est basé sur un ensemble de huit descripteurs d'objets : position 3D, position 2D, ratio de forme, surface 2D, histogramme de couleur, histogramme de gradient orienté, couleur covariance et couleur dominante. Ce traqueur utilise un filtre de trajectoire pour éliminer les trajectoires erronées. Le deuxième algorithme de suivi utilise un filtre de Kalman associé à un ensemble de quatre descripteurs d'apparence d'objets. Un suivi global est associé à ce traqueur pour suivre des objets mobiles quand la détection des objets n'est pas assez bonne et aussi pour supprimer des trajectoires erronées. Le troisième algorithme de suivi est basé sur le suivi des points d'intérêt de Kanade-Lucas-Tomasi (KLT).

Le chapitre 7 est consacré à l'expérimentation et la validation des méthodes proposées. Nous présentons les résultats de la tâche de suivi dans ces deux cas : avec et sans le processus de contrôle. L'expérimentation de la méthode de contrôle est présentée en deux phases : apprentissage et test. Pour la phase d'apprentissage, nous utilisons des séquences vidéo différentes (extraites d'ensembles de données publiques ETISEO, Caviar et des projets Gerhome, Caretaker) pour créer la base de données apprises. Pour la phase de test, les séquences de données publiques Caviar et des projets Européens Vanaheim, Caretaker sont utilisées pour expérimenter l'approche proposée. Une comparaison avec les traqueurs de l'état de l'art est également présentée pour mettre en évidence la performance des méthodes proposées. Les résultats expérimentaux ont montré que le contrôleur proposé améliore significativement la qualité de suivi. La performance de suivi obtenue en utilisant la méthode de contrôle proposée est meilleure que celle de certaines algorithmes dans l'état de l'art.

Le chapitre 8 présente les conclusions finales et les limitations des contributions de la thèse. Nous discutons également des perspectives à court terme et à long terme de cette étude.

3. Limitations

Il existe quelques limitations dans ce travail. Nous les divisons en deux groupes : les limitations théoriques et celles expérimentales.

3.1. Limitations Théoriques

- **La performance du contrôleur proposé est dépendante de la qualité de la tâche de détection :** Les valeurs des caractéristiques contextuelles (densité, le niveau de l'occultation des objets mobiles, leurs surfaces 2D, les variances de surface 2D, leurs contrastes par rapport à l'image de fond et leurs variances de contraste) sont définies en fonction de boites encadrées des objets. La phase d'apprentissage est réalisée avec des objets annotés. Une mauvaise qualité de détection d'objets durant la phase en ligne diminue la performance de la détection de contexte. En conséquence, la qualité du contrôleur proposé peut diminuer considérablement.

- **Les caractéristiques contextuelles ne sont pas suffisamment précises pour déterminer les valeurs des paramètres de suivi :** Les six caractéristiques contextuelles proposées sont assez générales. Cela fait la relation entre un contexte et ses paramètres de suivi satisfaisants devenir faible. Cela signifie que le contexte pourrait avoir des valeurs satisfaisantes différentes des paramètres de suivi. Bien que nous avions proposé un processus de groupement dans lequel les valeurs apprises des paramètres de suivi sont vérifiées, nous avons besoin d'un plus grand nombre de vidéos d'apprentissage pour obtenir une base de données apprise plus fiable. Il nous manque un modèle mathématique pour vérifier l'exactitude des caractéristiques contextuelles proposées et aussi de nous permettre de trouver de nouvelles caractéristiques contextuelles.

- **L'approche proposée se limite à contrôler la tâche de suivi :** En fait, dans de nombreux cas, les erreurs de suivi sont causées principalement par les erreurs des tâches aux niveaux inférieurs tels que la détection d'objet, la segmentation d'images. Le contrôle de suivi ne peut pas résoudre directement ces erreurs.

3.2 Limitations Expérimentales

- **Certains contextes de suivi ne sont pas testés :** Bien que les expérimentations ont été réalisées avec des scènes de vidéo provenant de nombreux contextes différents, certains types de contexte ne sont pas encore testés, par exemple les contextes dans lesquels il y a un changement de contraste de l'objet au cours du temps. Certains autres types de vidéos ne sont pas abordés dans l'expérimentation comme des vidéos pour les scènes extérieures.

- **L'approche d'adaptation des paramètres basée sur l'évaluation manque d'expérimentation :** Dans ce travail, nous avons présenté deux approches pour le contrôle des algorithmes

de suivi. Alors que la première approche (adaptation des paramètres basée sur le contexte) a été entièrement validé expérimentalement, l'expérimentation de la seconde (adaptation des paramètres basée sur l'évaluation) n'est effectuée que pour l'algorithme d'évaluation en ligne de l'objet suivi.

4. Perspectives

4.1. Perspectives à Court Terme

- **Ajouter des caractéristiques contextuelles :** Certaines caractéristiques peuvent être ajoutées afin de mieux décrire le contexte comme la discrimination de la couleur entre des objets mobiles ou la complexité des trajectoires des objets.

- **Améliorer l'algorithme d'évaluation de suivi en ligne :** Cet algorithme joue un rôle important au cours du processus de contrôle en ligne. Il peut aider à détecter les erreurs de suivi et de classifier ces erreurs. Cette information est très utile pour trouver une stratégie de contrôle appropriée pour le suivi. Nous pouvons améliorer cet algorithme en ajoutant des critères d'évaluation (par exemple en ajoutant un covariance des couleurs des objets, des HOG et des contours) et les combiner avec d'autres techniques telles que le suivi inverse.

4.2. Perspectives à Long Terme

- **Définir un modèle mathématique afin de vérifier les caractéristiques contextuelles :** Dans ce travail, les caractéristiques contextuelles sont trouvées grâce à une analyse expérimentale. Cette étude contient des risques potentiels et nécessite plus d'investissement. L'existence d'un modèle mathématique peut supprimer ces limitations. Ce modèle mathématique peut aussi aider à identifier de nouvelle caractéristiques contextuelles qui ne peuvent pas être détectées de manière intuitive.

- **Changer en ligne les traqueurs :** Dans ce travail, nous étudions le réglage des paramètres d'un algorithme de suivi. Cependant la qualité d'un algorithme de suivi est dépendante du contexte de vidéo. Un traqueur peut être adapté à certains contextes seulement. Par conséquent, dans de nombreux cas, le réglage des paramètres de suivi ne peut pas améliorer les performances de suivi. Si le contrôleur est capable de changer les traqueurs au cours du processus de suivi d'objets, la qualité du suivi peut augmenter de manière significative.

- **Contrôler des traqueurs en fonction des objets suivis :** Dans le contrôleur proposé, les paramètres de suivi sont réglés au cours du temps, mais pas à travers de l'espace. En fait, à un même moment, la distribution des objets mobiles (par exemple leur niveau d'occultation, leur densité), ainsi que leurs propriétés (par exemple la couleur, taille) peut être différentes dans l'espace. Dans ce cas, nous avons besoin d'un mécanisme de réglage des paramètres au cours de locations spatiales. Cela signifie que les paramètres de suivi sont changés au cours des objets suivis et de leur positions.

- **Quantifier la fiabilité des étapes d'exécution :** Des étapes d'exécution de la méthode proposée (détection contexte, apprentissage des valeurs satisfaisantes des paramètres contrôlables) peuvent indure des erreurs et des incertitudes. Si les sorties de ces étapes sont associées aux valeurs représentant leur fiabilité, nous pouvons estimer la fiabilité du contrôleur, détecter les étapes auxquelles il existe des incertitudes, et proposer des améliorations efficaces.

- **Définir les caractéristiques contextuelles indépendantes de la détection des objets mobiles :** Si les caractéristiques contextuelles sont moins dépendantes de la sortie de l'algorithme de détection des objets, la performance du contrôleur va être augmentée.

- **Évaluer et contrôler en ligne les algorithmes appartenant aux couches inférieures :** L'origine d'une erreur du processus de suivi peut se trouver dans des couches inférieures telles que la détection d'objet ou la segmentation de l'image. Par conséquent, si nous considérons uniquement la tâche de suivi, nous ne pouvons pas résoudre toutes les erreurs. Nous avons besoin de proposer des méthodes pour réparer les erreurs détectées à ces tâches.

Les publications réalisées pendant la thèse :

1. D. P. Chau, F. Bremond and M. Thonnat, Online Evaluation of Tracking Algorithm Performance, in The International Conference on Imaging for Crime Detection and Prevention (ICDP), London, UK, 3rd December 2009.
2. D. P. Chau, F. Bremond, M. Thonnat and E. Corvee, Robust mobile object tracking based on multiple feature similarity and trajectory filtering, in The International Conference on Computer Vision Theory and Applications (VISAPP), Algarve, Portugal, March 5-7, 2011.
3. S. Zaidenberg, B. Boulay, C. Garate, D. P. Chau, E. Corvee and F. Bremond, Group interaction and group tracking for video-surveillance in underground railway stations, in The International Workshop on Behaviour Analysis and Video Understanding, in conjunction with International Conference on Computer Vision Systems (ICVS) , Sophia Antipolis,

France, September 23, 2011.

4. D. P. Chau, F. Bremond and M. Thonnat, A multi-feature tracking algorithm enabling adaptation to context variations, in The International Conference on Imaging for Crime Detection and Prevention (ICDP), London, UK, 3 - 4 November 2011.
5. S. Bak, D. P. Chau, J. Badie, E. Corvee, F. Bremond and M. Thonnat, Multi-target Tracking by Discriminative Analysis on Riemannian Manifold, in The IEEE International Conference on Image Processing (ICIP), Florida, USA, September 30 - October 3, 2012.

BIBLIOGRAPHY

[Aggarwal and Cai, 1997] Aggarwal, J. K. and Cai, Q. (1997). Human motion analysis : a review. In The Nonrigid and Articulated Motion Workshop pp. 90–102,. 35

[Almanza-Ojeda, 2011] Almanza-Ojeda, D. L. (2011). Détection et suivi d'objets mobiles perçus depuis un capteur visuel embarqué. In PhD thesis, University of Toulouse. 35

[Almeida et al., 2005] Almeida, A., Almeida, J. and Araujo, R. (2005). Real-time tracking of multiple moving objects using particle filters and probabilistic data association. In Automatika pp. 39–48,. 41

[Avanzi et al., 2005] Avanzi, A., Bremond, F., Tornieri, C. and Thonnat, M. (2005). Design and Assessment of an Intelligent Activity Monitoring Platform. In The EURASIP Journal on Applied Signal Processing, special issue in "Advances in Intelligent Vision Systems : Methods and Applications", vol. 14 pp. 2359–2374,. 185

[Baiget et al., 2009] Baiget, P., Sommerlade, E., Reid, I. and Gonzalez, J. (2009). Finding Prototypes to Estimate Trajectory Development in Outdoor Scenarios. In The 1st International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with The British Machine Vison Conference (BMVC), Leeds, UK. 37

[Bak et al., 2012] Bak, S., Chau, D. P., Badie, J., Corvee, E., Bremond, F. and Thonnat, M. (2012). Multi-target tracking by discriminative analysis on Riemannian Manifold. In The International Conference on Image Processing (ICIP). 31

[Bak et al., 2010a] Bak, S., Corvee, E., Bremond, F. and Thonnat, M. (2010a). Person Reindentification using Haar-based and DCD-based Signature. In The 2nd Workshop on Activity Monitoring by Multi-Camera Surveillance Systems (AMMCSS), in conjunction with The International Conference on Advanced Video and Signal-Based Surveillance (AVSS), Boston, USA. 15, 44, 45

[Bak et al., 2010b] Bak, S., Corvee, E., Bremond, F. and Thonnat, M. (2010b). Person Reidentification Using Spatial Covariance Regions of Human Body Parts. In The International Conference on Advanced Video and Signal-Based Surveillance (AVSS), Boston, USA. 139

[Bak et al., 2009] Bak, S., Sundaram, S., Bremond, F. and Thonnat, M. (2009). Fusion of motion segmentation with online adaptive neural classifier for robust tracking. In The Interna-

tional Conference on Computer Vision Theory and Applications (VISAPP), Lisboa, Portugal. 31, 64, 133, 195

[Bar-Shalom and Fortmann, 1988] Bar-Shalom, Y. and Fortmann, T. (1988). Tracking and Data Association. In The Academic Press. 41

[Beymer and Konolige, 1999] Beymer, D. and Konolige, K. (1999). Real-time tracking of multiple people using continuous detection. In The Frame-Rate Workshop, in conjunction with The IEEE International Conference on Computer Vision (ICCV). 40

[Bhanu and Das, 1995] Bhanu, B. and Das, S. S. L. (1995). Adaptive image segmentation using genetic and hybrid search methods. In The IEEE Transactions on Aerospace and Electronic Systems pp. 1268–1291,. 31, 54, 195

[Bilinski et al., 2009] Bilinski, P., Bremond, F. and Kaaniche, M. (2009). Multiple object tracking with occlusions using HOG descriptors and multi resolution images. In The International Conference on Imaging for Crime Detection and Prevention (ICDP), London, UK. 15, 37, 38, 39, 40, 81, 137

[Broida and Chellappa, 1986] Broida, T. and Chellappa, R. (1986). Estimation of object motion parameters from noisy images. In The IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 90–99,. 40

[Brookner, 1998] Brookner, E. (1998). Tracking and Kalman Filtering Made Easy. Publisher : John Wiley & Sons Inc. ISBN 13 : 9780471184072, ISBN 10 : 0471184071. 40

[Caporossi et al., 2004] Caporossi, A., Hall, D., Reignier, P. and Crowley, J. L. (2004). Robust Visual Tracking from Dynamic Control of Processing. In The Workshop on Performance Evaluation for tracking and Surveillance (PETS), in the conjunction with The European Conference on Computer Vision (ECCV), Prague, Czech. 53, 63

[Chau et al., 2009a] Chau, D. P., Bremond, F., Corvee, E. and Thonnat, M. (2009a). Repairing people trajectories based on point clustering. In The International Conference on Computer Vision Theory and Applications (VISAPP), Lisboa, Portugal. 30, 121, 151, 194

[Chau et al., 2009b] Chau, D. P., Bremond, F. and Thonnat, M. (2009b). Online evaluation of tracking algorithm performance. In The International Conference on Imaging for Crime Detection and Prevention (ICDP), London, UK. 31

[Chau et al., 2011a] Chau, D. P., Bremond, F. and Thonnat, M. (2011a). A multi-feature tracking algorithm enabling adaptation to context variations. In The International Conference on Imaging for Crime Detection and Prevention (ICDP), London, UK. 31

[Chau et al., 2011b] Chau, D. P., Bremond, F., Thonnat, M. and Corvee, E. (2011b). Robust mobile object tracking based on multiple feature similarity and trajectory filtering. In The

International Conference on Computer Vision Theory and Applications (VISAPP), Algarve, Portugal. 32, 182, 183

[Colombo et al., 2008] Colombo, A., Orwell, J. and Velastin, S. (2008). Color Constancy Techniques for Re-Recognition of Pedestrians from Multiple Surveillance Cameras. In Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications (M2SFA2). 44

[Comaniciu and Meer, 1997] Comaniciu, D. and Meer, P. (1997). Robust analysis of feature spaces : color image segmentation. In The IEEE conference on Computer Vision and Pattern Recognition (CVPR) pp. 750–755,. 43

[Corvee and Bremond, 2009] Corvee, E. and Bremond, F. (2009). Combining face detection and people tracking in video surveillance. In The International Conference on Imaging for Crime Detection and Prevention (ICDP), London, UK. 44

[Corvee and Bremond, 2010] Corvee, E. and Bremond, F. (2010). Body parts detection for people tracking using trees of Histogram of Oriented Gradient descriptors. In The International Conference on Advanced Video and Signal-Based Surveillance (AVSS). 17, 115, 116, 175

[Crowley and Reignier, 2003] Crowley, J. L. and Reignier, P. (2003). An architecture for context aware observation of human activity. In The Workshop on Computer Vision System Control Architectures (VSCA). 54, 55

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In The International Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA pp. 886–893,. 35, 38, 55

[Dantzig, 1951] Dantzig, G. B. (1951). Application of the simplexmethod to a transportation problem. In Activity Analysis of Production and Allocation pp. 359 – 373,. 89

[Erdem et al., 2004] Erdem, C. E., Tekalp, A. and Sankur, B. (2004). Metrics for performance evaluation of video object segmentation and tracking without ground-truth. In The IEEE International Conference on Image Processing (ICIP), Greece. 61, 159

[Erdem et al., 2003] Erdem, C. E., Tekalp, M. and Sankur, B. (2003). Video object tracking with feedback of performance measures. In The IEEE Transactions on Circuits and Systems for Video Technology pp. 310 – 324,. 49

[Everitt et al., 2001] Everitt, B. S., Landau, S. and Leese, M. (2001). In book : Cluster Analysis (Fourth ed.), London : Arnold. ISBN 0-340-76119-9. 108, 128

[Forstner and Moonen, 1999] Forstner, W. and Moonen, B. (1999). A metric for covariance matrices. In Quo vadis geodesia... ?, Festschrift for Erik W.Grafarend on the occasion of his 60th birthday, TR Dept. of Geodesy and Geoinformatics, Stuttgart University. 139

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. In The Journal of Computer and System Sciences pp. 522–536,. 44, 70, 102, 125, 147

[Georis et al., 2007] Georis, B., Bremond, F. and Thonnat, M. (2007). Real-time Control of Video Surveillance Systems with Program Supervision Techniques. In The Journal of Machine Vision and Applications pp. 189–205,. 16, 56, 57, 59, 63, 65

[Goldberg, 1989] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning (1st ed.). Addison-Wesley Longman Publishing Co., Inc.,. Boston, MA, USA. 51, 54, 70, 102, 103

[Grabner and Bischof, 2006] Grabner, H. and Bischof, H. (2006). Online Boosting and Vision. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1 pp. 260–267,. 42

[Grauman and Darrel, 2005] Grauman, K. and Darrel, T. (2005). The pyramid match kernel : Discriminative classification with sets of image features. In The International Conference on Computer Vision (ICCV), Beijing, China. 139

[Hall, 2006] Hall, D. (2006). Automatic parameter regulation of perceptual system. In The Image and Vision Computing pp. 870–881,. 16, 50, 51, 63

[Heyer et al., 1999] Heyer, L. J., Kruglyak, S. and Yooseph, S. (1999). Exploring expression data : Identification and analysis of coexpressed genes. In The Journal of Genome Research pp. 1106–1115,. 21, 108

[Huang et al., 2008] Huang, C., Wu, B. and Nevatia, R. (2008). Robust Object Tracking by Hierarchical Association of Detection Responses. In The European Conference on Computer Vision (ECCV). 167, 176, 177

[Hwang et al., 2009] Hwang, A. D., Higgins, E. C. and Pomplun, M. (2009). A model of top-down attentional control during visual search in complex scenes. In The Journal of Vision. 17, 137

[Johnson, 1998] Johnson, R. (1998). Tutorial : A Brief Summarization of the Kalman Filter. In " ". 15, 40

[Kang et al., 2004] Kang, J., Cohen, I. and Medioni, G. (2004). Object Reacquisition Using Invariant Appearance Model. In The International Conference on Pattern Recognition (ICPR). 15, 48

[Khoudour et al., 2001] Khoudour, L., Hindmarsh, J., Aubert, D., Velastin, S. and Heath, C. (2001). Enhancing security management in public transport using automatic incident detection. In : L. Sucharov, C. Brebbia (eds.) Urban Transport VII : Proceedings of the 7th

International Conference on Urban Transport and the Environment for the 21st Century, WIT Press, Southampton, United Kingdom pp. 619–628,. 27

[Kim et al., 2004] Kim, K., Chalidabhongse, T., Harwood, D. and Davis, L. (2004). Background modeling and subtraction by codebook construction. In The International Conference on Image Processing (ICIP), Singapore. 69, 94

[Kuo et al., 2010] Kuo, C., Huang, C. and Nevatia, R. (2010). Multi-target tracking by online learned discriminative appearance models. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA. 16, 30, 49, 54, 55, 63, 151, 167, 176, 177, 194

[Kwolek, 2009] Kwolek, B. (2009). Object tracking via multi-region covariance and particle swarm optimization. In The Sixth IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS), Genova, Italy. 31, 63, 64, 133, 195

[Li et al., 2009] Li, Y., Huang, C. and Nevatia, R. (2009). Learning to Associate : HybridBoosted Multi-Target Tracker for Crowded Scene. In The International Conference on Computer Vision and Pattern Recognition (CVPR). 167, 176, 177

[Liu et al., 2008] Liu, R., Li, S., Yuan, X. and He, R. (2008). Online Determination of Track Loss Using Template Inverse Matching. In The International Workshop on Visual Surveillance (VS). 159

[M. J. Swain, 1991] M. J. Swain, D. H. B. (1991). Color indexing. In The International Journal of Computer Vision pp. 11 – 32,. 136

[MacQueen, 1967] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. In The 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press pp. 281–297,. 50

[Maggio et al., 2007] Maggio, E., Smerladi, F. and Cavallaro, A. (2007). Adaptive Multifeature Tracking in a Particle Filtering Framework. In The IEEE Transactions on Circuits and Systems for Video Technology pp. 1348–1359,. 64

[Martin and Thonnat, 2008] Martin, V. and Thonnat, M. (2008). A Cognitive Vision Approach to Image Segmentation. In Tools in Artificial Intelligence pp. 265–294,. 31, 50, 63, 195

[Moisan et al., 1995] Moisan, S., Shekhar, C. and Thonnat, M. (1995). Real-time perception program supervision for vehicule driving assistance. In The International Conference on Recent Advances in Mechatronics (ICRAM), Istambul, Turkey. 50

[Monari et al., 2009] Monari, E., Maerker, J. and Kroschel, K. (2009). A Robust and Efficient Approach for Human Tracking in Multi-Camera Systems. In The International Conference on Advanced Video and Signal-Based Surveillance (AVSS). 15, 31, 43, 44, 133, 195

[Motamed, 2006] Motamed, C. (2006). Motion detection and tracking using belief indicators for an automatic visualsurveillance system. In The Journal of Image and Vision Computing pp. 1192–1201,. 35

[Needham and Boyle, 2003] Needham, C. J. and Boyle, R. D. (2003). Performance evaluation metrics and statistics for positional tracker evaluation. In The International Conference on Computer Vision Systems (ICVS), Graz, Austria. 159

[Nghiem, 2009] Nghiem, A. T. (2009). Adaptive algorithms for background estimation to detect moving objects in videos. In PhD thesis, University of Nice Sophia Antipolis, France. 31, 56, 58, 63, 65, 195

[Nghiem et al., 2007] Nghiem, A. T., Bremond, F., Thonnat, M. and Valentin., V. (2007). Etiseo, performance evaluation for video surveillance systems. In The IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS), London, UK. 159

[Nievergelt, 2000] Nievergelt, J. (2000). Exhaustive Search, Combinatorial Optimization and Enumeration : Exploring the Potential of Raw Computing Power. In The 27th Conference on Current Trends in Theory and Practice of Informatics. 102

[Ojala et al., 2002] Ojala, T., Pietikainen, M. and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. In The IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 971–987,. 41

[O'Rourke and Badler, 1980] O'Rourke, J. and Badler, N. I. (1980). Model-based image analysis of human motion using constraint propagation. In The IEEE Transaction on Pattern Analysis and Machine Intelligence pp. 522–536,. 63

[Peters, 2011] Peters, F. (2011). http ://aiatwvu.blogspot.com/2011/04/histogramsand-emd.html. 16, 89

[R. Vincent and Thonnat, 1994] R. Vincent, S. M. and Thonnat, M. (1994). Learning has a means to refine a knowledge-based system. In The Third Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop, Hatoyama, Japon. 53

[Robert, 2009] Robert, K. (2009). Night-Time Traffic surveillance : A robust framework for multi-vehicle detection, classification, and tracking. In The Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). 15, 40, 41

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for highspeed corner detection. In The European Conference on Computer Vision (ECCV). 38, 81

[Rubner et al., 1998] Rubner, Y., Tomasi, C. and Guibas, L. (1998). A metric for distributions with applications to image databases. In The IEEE International Conference on Computer Vision (ICCV), Bombay, India. 44, 88

[Russell and Norvig, 2003] Russell, S. J. and Norvig, P. (2003). Artificial Intelligence : A Modern Approach. In The Prentice Hall Series in Artificial Intelligence. 54

[SanMiguel et al., 2010] SanMiguel, J., Cavallaro, A. and Martinez, J. (2010). Adaptive online performance evaluation of video trackers. In The IEEE Transactions on Image Processing. 197

[Santner et al., 2010] Santner, J., Leistner, C., Saffari, A., Pock, T. and Bischof, H. (2010). PROST : Parallel Robust Online Simple Tracking. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA. 30, 49, 54, 56, 63, 194

[Scovanner and Tappen, 2009] Scovanner, P. and Tappen, M. (2009). Learning Pedestrian Dynamics from the Real World. In The International Conference on Computer Vision (ICCV). 15, 37, 38

[securite-surveillance.com, 2009] securite-surveillance.com (2009). Video surveillance : La ville de Mexico signe avec Thales, http ://www.securite-surveillance.com/blog/index.php/video-surveillance-la-ville-de-mexico-signe-avec-thales/. 15, 26

[Shekhar et al., 1999] Shekhar, C., Moisan, S., Vincent, R., Burlina, P. and Chellappa, R. (1999). Knowledge-based Control of Vision Systems. In The Image and Vision Computing pp. 667–683,. 16, 27, 50, 52, 53, 63

[Sherrah, 2010] Sherrah, J. (2010). Learning to Adapt : A Method for Automatic Tuning of Algorithm Parameters. In The 12th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS), Sydney, Australia pp. 414–425,. 50, 52, 63

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good Features to Track. In The International Conference on Computer Vision and Pattern Recognition (CVPR). 152

[Smeaton et al., 2006] Smeaton, A., Over, P. and Kraaij, W. (2006). Evaluation campaigns and trecvid. In MIR'06 : Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval. 32, 158

[Snidaro et al., 2008] Snidaro, L., Visentini, I. and Foresti, G. (2008). Dynamic models for people detection and tracking. In The 8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS) pp. 29–35,. 15, 41, 42, 64

[Souded et al., 2011] Souded, M., Giulieri, L. and Bremond, F. (2011). An Object Tracking in Particle Filtering and Data Association Framework, Using SIFT Features. In The International Conference on Imaging for Crime Detection and Prevention Conference (ICDP). 62

[Thonnat et al., 1999] Thonnat, M., Moisan, S. and Crubezy, M. (1999). Experience in Integrating Image Processing Programs. In The 1st International Conference on Vision Systems (ICVS), Lecture Notes in Computer Science, Spain. 27, 50, 63

[Torkan and Behrad, 2010] Torkan, S. and Behrad, A. (2010). A New Contour Based Tracking Algorithm Using Improved Greedy Snake. In The 18th Iranian Conference on Electrical Engineering (ICEE). 46

[Tuzel et al., 2006] Tuzel, O., Porikli, F. and P.Meer (2006). Region Covariance : A Fast Descriptor for Detection and Classification. In The European Conference on Computer Vision (ECCV) pp. 589–600,. 55

[T.Weise, 2009] T.Weise (2009). In e-book : Global Optimization Algorithms - Theory and Application. 17, 100

[Vaswani, 2007] Vaswani, N. (2007). Additive Change Detection in Nonlinear Systems With Unknown Change Parameters. In The IEEE Transactions on Signal Processing, issue 3, volume 55 pp. 859–872,. 159

[Viola and Jones, 2003] Viola, P. and Jones, M. (2003). Rapid object detection using a boosted cascade of simple features. In The Proceeding of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 511–518,. 35, 41

[Wikipedia, 2011a] Wikipedia (2011a). CIE color space : http ://en.wikipedia.org/wiki/Tristimulus. 43

[Wikipedia, 2011b] Wikipedia (2011b). Deterministic system : http ://en.wikipedia.org/wiki/Deterministic_system. 37

[Wu and Nevatia, 2007] Wu, B. and Nevatia, R. (2007). Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors. In The International Journal of Computer Vision (IJCV) *75*, 247–266. 167, 176, 177

[Wu et al., 2007] Wu, H., Sankaranarayanan, A. and Chellappa, R. (2007). In Situ Evaluation of Tracking Algorithms Using Time Reversed Chains. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 61

[Wu and Zheng, 2004] Wu, H. and Zheng, Q. (2004). Self-evaluation for video tracking systems. In The 24th Army Science Conference. 61

[Xing et al., 2009] Xing, J., Ai, H. and Lao, S. (2009). Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In The International Conference on Computer Vision and Pattern Recognition (CVPR). 167, 176, 177

[Xu and Ahuja, 2002] Xu, N. and Ahuja, N. (2002). Object contour tracking using graph cuts based active contours. In The International Conference on Image Processing (ICIP). 15, 46, 47

[Yang et al., 2005] Yang, C., Duraiswami, R. and Davis, L. (2005). Fast multiple object tracking via a hierarchical particle filter. In The IEEE International Conference on Computer Vision (ICCV) pp. 212–219,. 62

[Yang et al., 2008] Yang, N., Chang, W., Kuo, C. and Li, T. (2008). A fast mpeg-7 dominant color extraction with new similarity measure for image retrieval. The Journal of Visual Communication and Image Representation *19*, 92–105. 141

[Yilmaz et al., 2006] Yilmaz, A., Javed, O. and Shah, M. (2006). Object tracking : A survey. In The ACM Computing Surveys (CSUR). 15, 35, 36, 37, 41, 45, 49

[Yilmaz and Shah, 2004] Yilmaz, A. and Shah, M. (2004). Contour Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras. In The IEEE Transactions on Pattern Analysis and Machine Intelligence. 49

[Yong et al., 2005] Yong, X., Feng, D., Rongchun, Z. and Petrou, M. (2005). Learning-based algorithm selection for image segmentation. In The Journal of Pattern Recognition Letters pp. 1059–1068,. 50, 52, 63

[Zaidenberg et al., 2011] Zaidenberg, S., Boulay, B., Garate, C., Chau, D. P., Corvee, E. and Bremond, F. (2011). Group interaction and group tracking for video-surveillance in underground railway stations. In The International Workshop on Behaviour Analysis and Video Understanding, in conjunction with The International Conference on Computer Vision Systems (ICVS) , Sophia Antipolis, France. 31

[Zhou et al., 2006] Zhou, Y., Hu, B. and Zhang, J. (2006). Occlusion detection and Tracking Method based on Bayesian decision theory. In The Lecture Notes in Computer Science, Volume 4319/2006 pp. 474–482,. 31, 42, 133, 195