Online Parameter Tuning for Object Tracking Algorithms

Duc Phu Chau, Monique Thonnat, François Bremond, Etienne Corvee

Online Parameter Tuning for Object Tracking Algorithms

Duc Phu CHAU, Monique THONNAT, François BREMOND and Etienne CORVEE
{Duc-Phu.Chau, Monique.Thonnat, Francois.Bremond, Etienne.Corvee}@inria.fr

STARS team, INRIA Sophia Antipolis - Méditerranée
2004 route des Lucioles, 06560 Valbonne, France

Abstract

Object tracking quality usually depends on video scene conditions (e.g. illumination, density of objects, object occlusion level). In order to overcome this limitation, this article presents a new control approach to adapt the object tracking process to the scene condition variations. More precisely, this approach learns how to tune the tracker parameters to cope with the tracking context variations. The tracking context, or 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 surrounding background, their contrast variance, their 2D area and their 2D area variance. In an offline phase, training video sequences are classified by clustering their contextual features. Each context cluster is then associated to satisfactory tracking parameters. In the online control phase, once a context change is detected, the tracking parameters are tuned using the learned values. The approach has been experimented with three different tracking algorithms and on long, complex video datasets. This article brings two significant contributions: (1) a classification method of video sequences to learn offline tracking parameters, (2) a new method to tune online tracking parameters using tracking context.

Keywords:
Object tracking, online parameter tuning, controller, self-adaptation, machine learning

1. Introduction

Mobile object tracking plays an important role in an increasing number of computer vision applications (e.g. home care, sport scene analysis and visual surveillance). The object trajectories are useful for activity recognition, learning of interest zones or paths in a scene and detection of events of interest. Unfortunately the tracking quality depends on many factors: 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 frequency of mobile objects. In particular, for a long video sequence (i.e. several hours) in which the variations of these properties happen frequently, the tracking quality is still an issue. The problems we focus on 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?

In order to answer these two questions, we propose in this article a new method for controlling tracking algorithms. 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 video sequence 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 algorithm categories. In this work, our objective is to control tracking algorithms which rely on object appearance or 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 adapted for handling other tracking algorithm category (e.g. object silhouette-based tracking). The term "**intelligent**" means that this approach requires less human interaction than the control methods in the state of the art.

1.1. Hypotheses

The control method presented in this manuscript relies on the two following hypotheses:

1. The considered tracking algorithms have at least one tunable parameter which influences significantly the tracking quality.
2. There exists a number of contexts which have an impact on the tracking quality. Let g be a function mapping a video v_i to its context. For a tracking algorithm $\mathfrak{T}$, we suppose that there exists a function $f_{\mathfrak{T}}$ mapping a video context to satisfactory tracking parameter values (i.e. parameter values for which the tracking quality is greater than a predefined threshold $\mathfrak{s}$):

$$\forall v_i, \exists f : |Q(\mathcal{O}_{\mathfrak{T}}(f_{\mathfrak{T}} \circ g(v_i)), G_{v_i})| > \mathfrak{s} \quad (1)$$

where G_{v_i} represents the tracking ground-truth data of video v_i; $\mathcal{O}_{\mathfrak{T}}(.)$ represents the

output of tracker $\mathfrak{T}$ corresponding to given parameter values; $Q(\mathcal{O}_{\mathfrak{T}}(.), G_{v_i})$ represents the quality of tracker $\mathfrak{T}$ output compared to the tracking ground-truth data of video v_i.
Let ϵ_1, ϵ_2 be predefined thresholds. The function f is assumed to satisfy the following property if the temporal lengths of v_1 and v_2 are short enough (lower than 50 frames):

$$\forall v_1, v_2 : if\ |g(v_1) - g(v_2)| \ < \epsilon_1 \tag{2}$$

$$=> |Q(\mathcal{O}_{\mathfrak{T}}(f_{\mathfrak{T}} \circ g(v_1)), G_{v_1}) \ - Q(\mathcal{O}_{\mathfrak{T}}(f_{\mathfrak{T}} \circ g(v_2)), G_{v_2})| \ < \epsilon_2 \tag{3}$$

This hypothesis means that if the contexts of two videos v_1 and v_2 are close enough, the tracking performances for v_1 and v_2 corresponding to their satisfactory tracking parameter values are also close enough.
The hypothesis 2 is given for two objectives. First, we can compute the satisfactory tracking parameter values for a video context cluster using satisfactory parameters of contexts (see section 3.4.2). Second, the satisfactory tracking parameters for context clusters can be used for tuning online the tracking parameters (see section 4.2).

1.2. Article Organization

This article is organized as follows. Section 2 presents a state of the art on control methods. Section 3, entitled "learning phase", details a scheme to learn satisfactory tracking parameters for each video context cluster. Section 4 describes the online parameter tuning process. Section 5 is dedicated to the experimentation and validation of the proposed method. Section 6 presents concluding remarks as well as future work.

2. State of the Art

Many approaches have been proposed to track mobile objects in a scene [Yilmaz et al., 2006]. Depending on taxonomy criteria, the trackers can be classified into different categories. Figure 1 presents a taxonomy example (the red ellipses mark the tracker categories controlled by the proposed method). However the quality of tracking algorithms always depends on scene properties such as: mobile object density, contrast intensity, scene depth and object size. The selection of a tracking algorithm for an unknown scene becomes a hard task. Even when the tracker has already been determined, it is difficult to tune online its parameters to get a high performance.

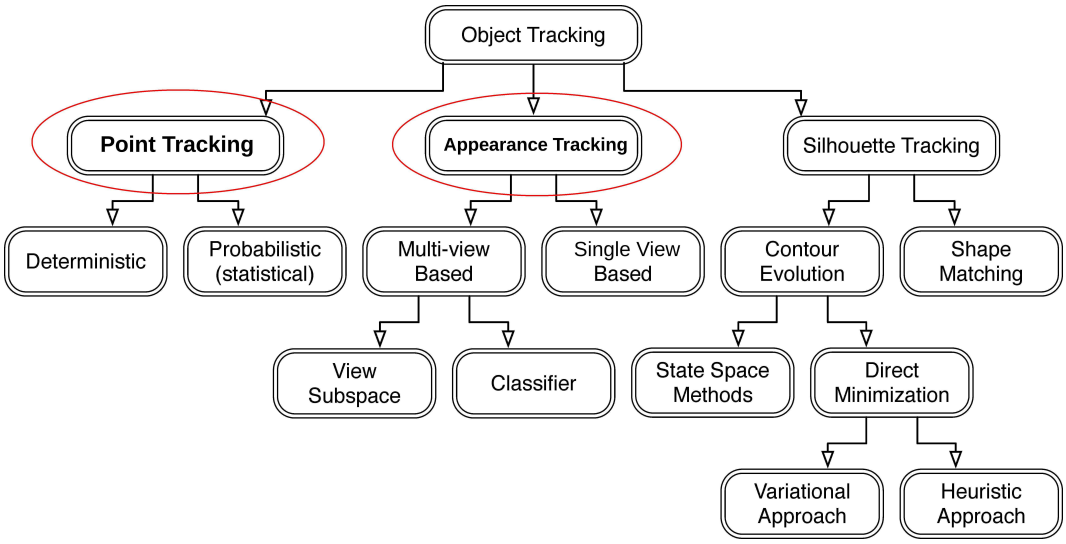

Figure 1: Taxonomy of tracking methods (adapted from [Yilmaz et al., 2006]). The red ellipses mark the tracker categories controlled by our method.

The idea about an automatic control to adapt the performance of a system to the problem of scene variations has already been studied. However some methods limit their studies to static image and not to video processing. For example 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 authors in [Georis et al., 2007] present a controlled video understanding system based on a knowledge base. The system is composed of three main components in which the control component performs several steps for managing all the online processes of the system (e.g program execution and automatic parameter tuning). Different rules are defined in this component based on user goal, contextual information and evaluation results. However their approach does not address directly the tracking task.

Some methods have addressed the tracking parameter tuning, however their approaches require too strong hypotheses and expert knowledge. For example, the author in [Sherrah, 2010] proposes an approach to tune automatically tracking algorithm parameters. In this approach, the

tracker quality is represented as a function of tuned parameters. The author supposes that this function has no local optimal solutions. Using this hypothesis, for each parameter and a training video, the author determines its optimal value thanks to expert knowledge. Then the parameter tendency (i.e. increase or decrease) for converging to the optimal value is learned in function of the tracker input and output. This learned parameter tendency is used in the online phase to tune automatically the corresponding parameter to improve the tracking performance. In [Caporossi et al., 2004], the authors compare the tracker results with corresponding ground-truth data to determine the importance of each parameter for each context and to exploit the influence of each parameter variation on tracker performance. The authors suppose that parameter variations are independent. This is a strict hypothesis because the parameters are usually dependent on each other. In [Chau et al., 2011a], the authors propose a tracking algorithm whose parameters can be learned offline for each tracking context. However the authors suppose that the context within a video sequence is fixed over time. Moreover, the tracking context is manually selected.

Some approaches have been proposed to decrease the need of expert knowledge [Hall, 2006, Santner et al., 2010], however they are expensive in term of processing time and their performance are dependent on an automatic tracking evaluation. For example, in [Hall, 2006], the author proposes two strategies to regulate the parameters for improving the tracking quality. In the first strategy, the parameter values are determined using an enumerative search. In the second strategy, a genetic algorithm is used to search for the best parameter values. This approach does not require human supervision and parameter knowledge for controlling its tracker. However, it is computationally expensive because of the parameter optimization performed in the online phase. Moreover, this approach requires an online tracking evaluation (without ground-truth data) to verify the performance of the tracker when using the found parameters. This can decrease the approach performance. 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 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. The approach is interesting but the authors do not mention how to estimate online the tracker quality. Also, the execution of three trackers in

parallel is very expensive in terms of processing time.

2.1. Discussion

As analyzed above, many approaches whose objective is to control the tracking process have been studied in state of the art. These methods have the following issues.

The first issue relates to the context notion. While some methods study context for static image applications [Thonnat et al., 1999], to our knowledge, there are no approach which proposes a formal definition for object tracking context.

The second issue is about the generic level of the control methods. Some approaches need too strong hypotheses on the relation between the tracking quality and tracking parameters [Sherrah, 2010] or on the independence between tracking parameters [Caporossi et al., 2004]. Some other methods require expert knowledge [Thonnat et al., 1999, Chau et al., 2011a] for building knowledge base or for tuning parameters. These requirements reduce the genericity of these approaches.

The third issue pertains to the feasibility of these studies. Some approaches are expensive in term of processing time [Hall, 2006, Santner et al., 2010].

In this article, we propose a control method for object tracking algorithms addressing these issues. In this article, the control trackers belong to "Appearance tracking" or "Point tracking" (see figure 1). These tracker categories are selected because they are the most popular ones and are largely studied in the state of the art. Our proposed method includes two phases: an offline learning phase and an online parameter tuning. The next sections present in detail the steps of these two phases.

3. Offline Learning Phase

The objective of the learning phase is to create a database which supports the control process of a tracking algorithm. This database contains satisfactory parameter values of the tracking algorithm for various scene conditions.

This phase takes as input training video sequences, annotated objects, annotated trajectories, a tracking algorithm including its control parameters. The term "control parameters" refers to parameters which are considered in the control process (i.e. to look for satisfactory values in the learning phase and to be tuned in the online phase). In this work we consider only numerical

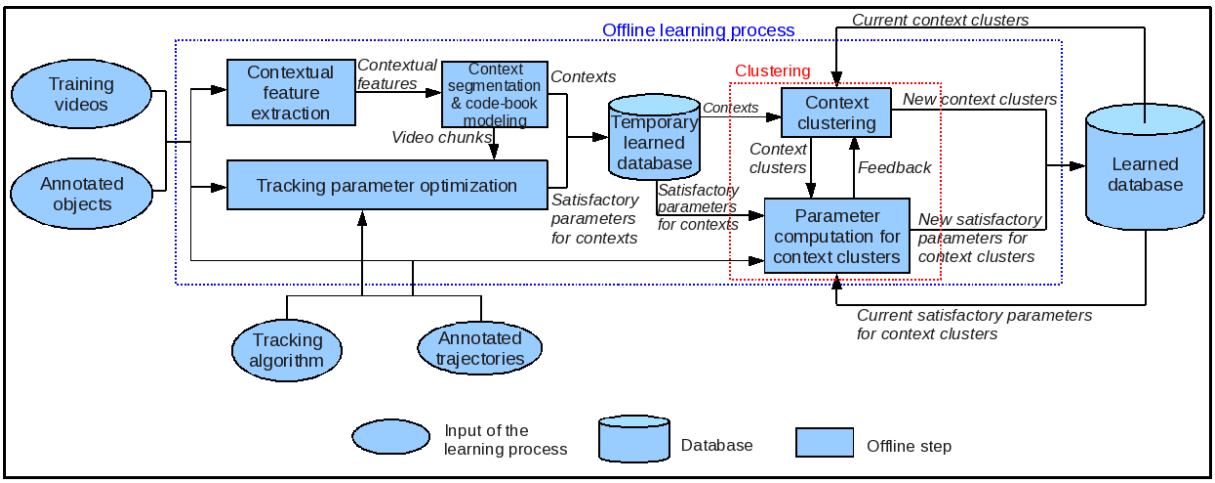

Figure 2: The offline learning scheme

parameters, however the proposed method can be applied also on 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. Figure 2 presents the proposed scheme for building and updating the learned database.

The notion of "context" (or "tracking context") in this work represents elements in the videos which influence the tracking quality. More precisely, a 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 surrounding background, their contrast variance, their 2D area and their 2D area variance. For each training video, we extract these contextual features from annotated objects and then use them to segment the training video in a set of consecutive chunks. Each video chunk has a stable context. The context of a video chunk is represented by a set of six code-books (corresponding to six features). An optimization process is performed to determine satisfactory tracking parameter values for the video chunks. These parameter values and the set of code-books are inserted into a temporary learned database. After processing all training videos, we cluster these contexts and then compute satisfactory tracking parameter values for context clusters.

In the following, we describe the four steps of the offline learning process: (1) contextual feature extraction, (2) context segmentation and code-book modeling, (3) tracking parameter optimization and (4) clustering (composed of two sub-steps: context clustering and parameter computation for context clusters).

3.1. Contextual Feature Extraction

For each training video, the context feature values are computed for every frame.

3.1.1. Density of Mobile Objects

The density of mobile objects influences significantly the tracking quality. A high density of objects may lead to a decrease of object detection and tracking performance. The density of mobile objects at instant t is defined by the sum of all object areas over the 2D camera view.

3.1.2. Occlusion Level of Mobile Objects

The occlusion level of mobile objects is the main element which influences the tracking quality. An occlusion occurrence makes the object appearance partially or completely not visible. It decreases the object detection and tracking performance. In particular, the variation of object occlusion level over time is even more challenging because the coherence of object appearance changes significantly. Given two objects i, j at instant t of respectively 2D areas a_t^i and a_t^j, we compute their occlusion level based on their area overlap as follows:

$$ol_t^k = \frac{a_t^{ij}}{min(a_t^i, a_t^j)} \tag{4}$$

where k denotes the index value of this occlusion in the set of occlusions occurring 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 a predefined threshold. Let $\mathfrak{N}_t$ be the number of object occlusion occurrences at instant t, ol_t^k is the occlusion level of case k ($k = 1..\mathfrak{N}_t$). The occlusion level of mobile objects in a scene at instant t, denoted o_t, is defined as follows:

$$o_t = min(\frac{\sum_{k=1}^{\mathfrak{N}_t} ol_t^k \times 2}{n_t},\ 1) \tag{5}$$

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.

3.1.3. Contrast of Mobile Objects

The contrast of an object is defined as the color intensity difference between this object and its surrounding background. Let $B_i = \{C_i,\ W_i,\ H_i\}$ be the 2D bounding box of object i where C_i, W_i and H_i are respectively its 2D center, width and height. We define an extra bounding box of object i: $B_i^+ = \{C_i, W_i + \gamma\mathcal{M}_i,\ H_i + \gamma\mathcal{M}_i\}$ where $\mathcal{M}_i = min(W_i,\ H_i)$, γ is a

Figure 3: Illustration of the object contrast variation over space (a.) and over time (b. and c.)

predefined value in interval $[0,\ 1]$. The surrounding background of object i is defined as the area $\mathfrak{B}_i = B_i^+ \setminus B_i$.

An object with low contrast reduces first the object detection quality. Second, this 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 3a.) or over time (see figure 3b. and c.). The contrast of mobile objects at instant t is defined as the mean value of the contrasts of objects at instant t.

3.1.4. Contrast Variance of Mobile Objects

When different object contrast levels exist in the scene (see figure 3a.), 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 as their standard deviation value:

$$\hat{c}_t = \sqrt{\frac{1}{n_t}\sum_{i=1}^{n}(c_t^i - \bar{c}_t)^2} \tag{6}$$

where c_t^i is the contrast value of object i at t, $\bar{c}_t$ is the mean value of all object contrasts at t.

3.1.5. 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 process. Greater the object area is, higher the object appearance reliability is. The 2D area feature value at t is defined as the mean value of the 2D areas of mobile objects at instant t.

3.1.6. 2D Area Variance of Mobile Objects

Similar to the contrast feature, we define the variance of object 2D areas at instant t as their standard deviation value.

3.2. Context Segmentation and Code-book Modeling

3.2.1. Context Segmentation

The contextual variation of a video sequence influences significantly the tracking quality. Therefore it is not optimal to keep the same parameter values for a long video. In order to solve this issue, we propose an algorithm to segment a training video in consecutive chunks, each chunk is defined as having a stable context (i.e. the values of a same context feature in each chunk are close to each other). This algorithm is described as follows.

1. The training video is segmented in parts of l frames. The last part can have a temporal length lower than l. The value of l should be low enough (e.g. 50 frames) so that each video part has a stable enough context.
2. The contextual feature values of the first part is represented by a context code-book model (see more details in section 3.2.2).
3. From the second video part, we compute the context distance between the current part and the context code-book model of the previous part (see more details in section 3.2.3). 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. The higher Th_1 value, less stable the obtained context code-book models are.

At the end of the context segmentation algorithm, the training video is divided into a set of chunks (of different temporal lengths) corresponding to the obtained context code-book models. There are two open problems: How to represent a video context with a code-book model? and how to compute the distance between a context code-book model and a context. The following sections answer these two questions.

3.2.2. Code-book Modeling

During the tracking process, low frequent feature values play an important role for tuning tracking parameters. For example, when mobile object density is high in few frames, the track-

ing 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-frequency 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 $CB, CB\ = \{cb^k,\ k = 1..6\}$. A feature code-book is composed of a set of code-words describing the values of this feature. The number of code-words depends on the diversity of feature values.

Code-word definition: A code-word represents the values and their frequencies of a contextual feature. A code-word i of code-book k ($k = 1..6$), denoted cw_i^k, is defined as follows:

$$cw_i^k = \{\overline{\mu_i^k},\ m_i^k,\ M_i^k,\ f_i^k\} \tag{7}$$

where $\overline{\mu_i^k}$ is the mean of the feature values belonging to this code-word; m_i^k and M_i^k are the minimal and maximal feature values belonging to this word; f_i^k is the number of frames when the feature values belong to this word.

Algorithm for Updating Code-word:

- At the beginning, the code-book cb^k of a context feature k is empty.

- For a value μ_t^k of a contextual feature k computed at time t, verify if μ_t^k activates any code-word in code-book 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:

$$distance(\mu_t^k,\ cw_i^k)\ =\ 1\ -\ \frac{min(\mu_t^k,\ \overline{\mu_i^k})}{max(\mu_t^k,\ \overline{\mu_i^k})} \tag{8}$$

where $\overline{\mu_i^k}$ is the mean value of code-word cw_i^k.

- If cb^k is empty or if there is no code-word activated, create a new code-word and insert it into cb^k.

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

The code-words whose value f_i is lower than a threshold, are eliminated because they are corresponding to too low frequency feature values. The "contextual feature extraction" and

function $contextDistance(c,\ CB,\ l)$

Input: context code-book model CB, context c, l (number of frames of context c)

Output: context distance between code-book model 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 at time t

        For each codeword cw_i^k in code-book cb^k

            if ($distance(\mu_t^k,\ cw_i^k) < \epsilon_3$) { $count$++; break; }

    if ($count$ / $l < 0.5$) return 1;

    $countTotal\ += count$;

return ($1 - countTotal/(l * 6)$)

Table 1: Function for computing the distance between a context code-book CB and a video context c

"code-book modeling" steps of a video chunk plays the role of the function g (mapping a video sequence to its context) presented in hypothesis 2 (section 1).

3.2.3. Context Distance

This section presents how to compute the distance between a context c and a context code-book model $CB\ = \{cb^k,\ k\ = 1..6\}$. This distance is defined as a function of sub-distances between context c and code-books cb^k. This sub-distance is expressed by the number of times where matching code-words are found. Table 1 presents the algorithm to compute the context distance in which the function $distance(\mu_t^k,\ cw_i^k)$ is defined as in formula (8).

3.3. Tracking Parameter Optimization

The objective of the tracking parameter optimization task is to find the values of the control parameters which ensure the best possible tracking quality for each video chunk. This quality has to be greater than the threshold $\mathfrak{s}$ presented in hypothesis 2, section 1.1. These parameters are called "satisfactory parameters".

This task takes as input annotated objects, annotated trajectories, a tracking algorithm, a video chunk and control parameters for the considered tracker. The annotated objects are used

as object detection results. This task provides as output satisfactory parameter values. For each control parameter, its name, value range and step value are needed. The step value of a parameter is defined as the minimal variation which causes a significant change on the tracking quality. This value helps to avoid scanning the entire parameter space when searching for its satisfactory values.

Depending on the search space size and the nature of the control parameters, we can select suitable optimization algorithm. If the control parameter space is small, an exhaustive search [Nievergelt, 2000] or an enumerative search can be used to scan the values of these parameters. Otherwise, we can use a genetic algorithm [Goldberg, 1989] for searching 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, the Adaboost algorithm [Freund & Schapire, 1997] can be used to determine the best values of these weights (see example in [Chau et al., 2011a]). More than one optimization algorithm can be performed if the search space or the nature of the control parameters are different.

In order to represent the reliability of the found parameter values, we associate them to 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-Score* value representing the tracking quality of the considered video chunk while using the found tracking parameter values. Satisfactory parameter values, their reliability values and the context code-book model corresponding to this video chunk are stored into a temporary learned database.

3.4. Clustering

The clustering step is done at the end of each learning session when the temporary learned database contains the processing results of all training videos. In some cases, two similar contexts can have different satisfactory parameter values because optimization algorithm only finds local optimal solutions. Moreover, the context of a video sequence is not sufficient for determining the best satisfactory tracking parameter values. A clustering step is thus necessary to group similar contexts and to compute satisfactory parameter values for the context clusters. The clustering step is composed of two sub-steps: context clustering and parameter computation for context clusters (see figure 4).

This step takes as input the training videos, the annotated objects, tracking algorithm and

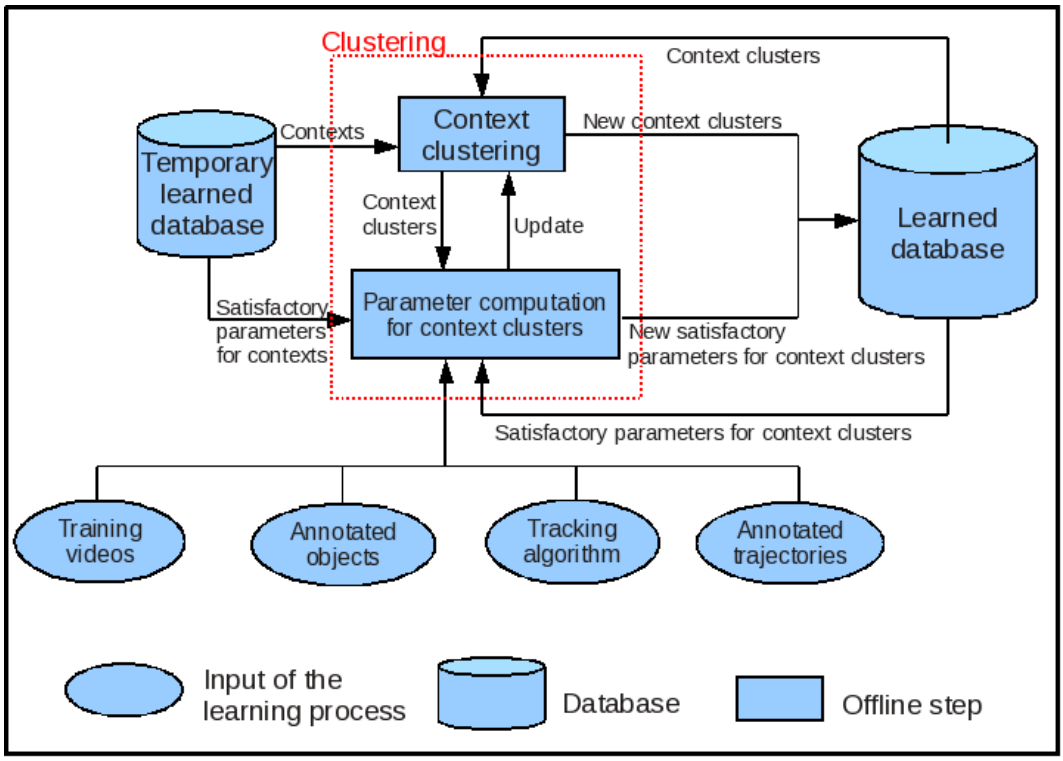

Figure 4: The clustering step

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 satisfactory tracking parameter values and the reliability values of these parameters. This step gives as output the new context clusters which are associated to their satisfactory parameter values and the reliability values of these parameters.

3.4.1. Context Clustering

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

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] to ensure a high reliability for the clustering process. In the following, we present how to compute the distance between two context code-book models.

- Context Code-book Model Distance

In order to compute the distance between two context code-book models CB_α and CB_β, each feature code-book cb^k ($k = 1..6$) of a context is transformed into a histogram whose bin i corresponds to feature value $\overline{\mu_i^k}$ of code-word i, 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 code-word i is activated (see section 3.2.2).

The distance between two feature code-books is defined as the Earth Mover Distance between the two corresponding histograms in which the ground distance between bins i and j is defined as $|\overline{\mu_i^k} - \overline{\mu_j^k}|$. The distance between two context code-book models is defined as the mean value of the six distances between the six feature code-books.

3.4.2. Parameter Computation for Context Clusters

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

- Parameter Computation

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

$$\vec{p}^j = \frac{\sum_{i=1}^{\Theta_j} \vec{p}_i \times w^i}{\sum_{i=1}^{\Theta_j} w^i} \tag{9}$$

where Θ_j is the number of contexts belonging to cluster j, $\vec{p}_i$ is satisfactory parameter values of context i belonging to this cluster, w^i is the weight of parameters $\vec{p}_i$ and is defined in function of the two reliability values of $\vec{p}_i$: number of training frames N_i and *F-Score*$_i$:

$$w_i = \frac{N_i/N^j + \textit{F-Score}_i}{2} \tag{10}$$

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

The reliability of context cluster j is also represented by two values: number of training frames N^j and a tracking quality score defined as a weighted combination of *F-Score*$_i$.

- Parameter Verification:

The objective of the parameter verification stage is to check whether the parameters of context clusters resulting from the previous stage ("Parameter Computation") are satisfactory. For each cluster, this stage takes all training videos belonging to this cluster and computes the tracking performance with the parameters resulting from the previous stage. For each training video, if the obtained tracking performance is greater or equal to the one computed by its own satisfactory parameters, these parameters are considered "verified". Otherwise, this video is removed from the considered cluster. It is then stored separately in the learned database. The context cluster and its satisfactory parameters are re-computed and re-verified.

At the end of the clustering process, we obtain in the learned database a set of context clusters represented similarly as a context: a context model of six code-books associated to satisfactory tracking parameter values, number of training frames and tracking quality score.

3.5. Training Phase Cost

The training phase cost represents the time needed for the training phase. This cost depends on the costs from the contextual feature computation, code-book modeling, tracking parameter optimization and clustering. The contextual features are low computational using 2D bounding box features. The code-book modeling and clustering tasks are also not expensive in terms of processing time. Therefore, the training phase cost mostly depends on the complexity of the tracking parameter optimization task. More precisely, it depends on the number of control parameters and their search space size. The cost reduction is twofold. First, we only control parameters which significantly influence the tracking quality. Second, we select a suitable optimization algorithm in function of the search space size and of the nature of control parameters as analyzed in the tracking parameter optimization task (see section 3.3).

We should note that the training phase requires annotated objects and trajectories as input. This can be done using public annotated datasets (e.g ETISEO, Caviar) or a graphical tool (e.g. Viper[1]).

[1] http://viper-toolkit.sourceforge.net/

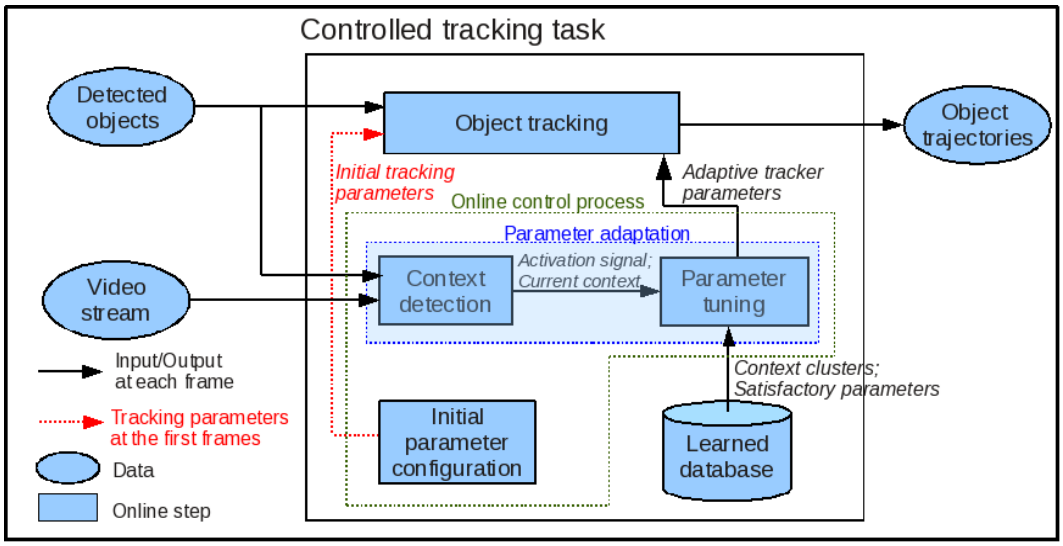

Figure 5: The controlled tracking task

4. Online Parameter Tuning

In this section, we describe the proposed controller which aims at tuning online the tracking parameter values for obtaining satisfactory tracking performance. The parameter adaptation task 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). In the following sections, we describe the two steps of this task: the context detection and parameter tuning steps.

4.1. Context Detection

An open problem is to detect online the variation of the tracking context. The contextual features 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. Also, in some cases, due to the mobile object locations, some wrong detections can happen within a small number of frames. Figure 6 illustrates such case. Therefore, in order to detect the context at current time, we need to collect the values of the contextual features in a large enough number of 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 image. For each video chunk of l frames, we compute the values of the contextual features.

Figure 6: The influence of the object detection quality due to object location (output of [Corvee & Bremond, 2010]): **a. Left image:** at frame 279, the two persons on the left are wrongly detected **b. Right image:** at frame 335, the relative position of these two persons is changed, and they are correctly detected.

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 3.2.3) to perform this classification. If this distance is lower than threshold Th_1, this context is considered as belonging to the context cluster. Otherwise, the "Parameter adaptation" task is activated.

4.2. Parameter Adaptation

The parameter adaptation process takes as input the current context from the "context detection" task, an activation signal 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:

$$contextDistance(c,\ C_i,\ l) \quad < \quad Th_1 \tag{11}$$

$$\forall C_j \in \mathfrak{D}, j \neq i: \quad contextDistance(c,\ C_i,\ l) \quad \leq \quad contextDistance(c,\ C_j,\ l) \tag{12}$$

where Th_1 is defined in section 3.2.1. The function $contextDistance(c,\ C_i,\ l)$ is defined in table 1. The expression (11) represents the condition (2) of hypothesis 2 (section 1.1). If a such context cluster C_i is found, then according to this hypothesis, the satisfactory tracking parameters associated with C_i are good enough for parameterizing the tracking of the current

video chunk. Otherwise, the tracking algorithm parameters do not change, the current video chunk is marked to be learned offline later.

4.3. Processing Time

During the online phase, the processing time depends on the context detection and parameter tuning tasks. The context detection task is fast because the computation of context features and context distance are not time consuming. The parameter tuning task complexity is low because it depends linearly on the number of clusters belonging to the offline learned database.

5. Qualitative Comparison with State of the Art Approaches

In the literature, we mention two articles [Caporossi et al., 2004] and [Sherrah, 2010] which have proposed parameter tuning approaches for object tracking algorithm. In this section, we present a qualitative comparison between both approaches and the proposed one. Our comparison relies on the following criteria:

- **Online execution:** As object tracking plays an important role in camera surveillance as well as other online applications, this criterion is very important for the object tracking approaches. The approach [Caporossi et al., 2004] needs ground-truth data to analyze offline the influence of tracking parameters for tracking quality. This approach cannot be done online whereas our proposed approach and [Sherrah, 2010] can be performed online.

- **Parameter tuning should be applicable to a large variety of tracking algorithms:** For this criterion, we consider two sub-criteria as follows.

  - **Requirement of tracking parameter independence:** The approaches from [Caporossi et al., 2004] and [Sherrah, 2010] require that the tracking algorithm have independent tracking parameters. During the training phase, both approaches optimize independently the tracker parameters. The proposed approach does not require this independence.

  - **Requirement of unimodality for the tracking performance on parameters:** This means the performance of the tracking algorithm has a unimodal distribution (i.e. no

	Approach [Caporossi et al., 2004]	Approach [Sherrah, 2010]	**Proposed approach**	**Ideal approach**
Online execution	No	Yes	**Yes**	**Yes**
Requirement of tracking parameter independence	Yes	Yes	**No**	**No**
Requirement of unimodality for the tracking quality on parameters	-	Yes	**No**	**No**
Requirement of ground-truth data in the training	Yes	Yes (and with expert knowledge)	**Yes**	**No**

Table 2: Qualitative comparison of the proposed approach with [Caporossi et al., 2004] and [Sherrah, 2010]

local optimum) in function of each parameter. The approach [Sherrah, 2010] needs this hypothesis to simplify the search of the best parameter value. Its parameter tuning method is inspired by the first derivative computation of the tracking performance function. Our proposed approach does not need this hypothesis as we use more generic and global optimization techniques (e.g. exhaustive search, enumerate search).

- **Requirement of ground-truth data in the training phase:** All the three approaches require ground-truth data to train offline the tracking parameters. However in the training phase of Sherrah [2010], the best tracking parameter value is determined by hand, and this needs expert knowledge.

Table 2 summarizes the qualitative comparison of the proposed approach with the approaches [Caporossi et al., 2004] and [Sherrah, 2010]. The column **"ideal approach"** shows the expectations of an ideal approach. We find that our proposed parameter control approach is more generic and practical than the approaches [Caporossi et al., 2004] and [Sherrah, 2010].

6. Experimentation and Validation

The objective of this experimentation is to measure the effect and robustness of the proposed control method. We experiment this method with three object trackers: an appearance tracker [Chau et al., 2011a], a tracker based on KLT [Shi & Tomasi, 1994] and a tracker based on Surf [Bay et al., 2008].

6.1. Parameter Setting and Object Detection Algorithm

The proposed control method has three predefined parameters. The distance threshold Th_1 to decide whether two contexts are close enough (sections 3.2.1 and 4.2) is set to 0.5. The minimum number of frames l of a context segment (sections 3.2.1 and 4.1) is set to 50 frames. The diameter threshold d in QT clustering algorithm (section 3.4.1) is set to 0.3. All these values are unchanged for all experiments presented in this article.

A HOG-based algorithm is used for detecting people [Corvee & Bremond, 2010] in videos.

6.2. Evaluation Metrics

In this experimentation, we select the tracking evaluation metrics used in several publications [Xing et al., 2009, Li et al., 2009, Kuo et al., 2010]. Let GT be the number of trajectories in the ground-truth of the test video. 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.

- 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.

6.3. Appearance Tracker Control

The appearance tracker [Chau et al., 2011a] takes as input a video stream and a list of objects detected in a predefined temporal window. The similarity of a pair of detected objects is defined as a weighted combination of five descriptor similarities on 2D area, 2D shape ratio, RGB color histogram, color covariance and dominant color. An object pair with the highest similarity is considered as belonging to a same object trajectory.

For this tracker, six parameters are selected for testing the proposed control method. The first five parameters are the object descriptor weights w_k (k = 1..5). These parameters depend on the tracking context and have a significant effect on the tracking quality. For the dominant color descriptor described in [Chau et al., 2011a], the number of dominant colors is required as an input parameter. This parameter is also influenced by the tracking context (for example the smaller object, the higher the number of dominant colors should be). Therefore we use these six parameters as control parameters so that hypothesis 1 (section 1.1) is ensured. Clearly, the number of dominant colors is only controlled when the weight of dominant color descriptor is not null. As the performance of the controller depends on the object detection quality, the online control process is tested in two cases: with automatically detected objects and with manually annotated objects.

6.3.1. Training Phase

In the learning phase, we use 12 video sequences belonging to different context types (i.e. different levels of density and 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). These videos belong to three public datasets (ETISEO[2], Caviar[3] and Gerhome[4]), to the European Caretaker project[5], and are recorded in different places (see examples in figure 7). The annotated data of object 2D bounding boxes in the videos from Caviar and ETISEO datasets are available on their websites.

Each training video is segmented automatically in a set of context segments (of different temporal lengths). The number of context segments depends on the contextual variation of the training video. Figure 8 presents the context segmentation result of sequence ThreePastShop2cor belonging to the Caviar dataset. The values of object 2D area and 2D area variance are normalized for displaying. The context of this sequence is divided automatically into six context segments. For each context segment, satisfactory control parameter values are learned.

In the tracking parameter optimization process, we use firstly an Adaboost algorithm to learn

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

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

[4]http://gerhome.cstb.fr/en/

[5]http://cordis.europa.eu/ist/kct/caretaker_synopsis.htm

Figure 7: Illustration of some training videos

the object descriptor weights (e.g. dominant color weight) for each context segment because each object descriptor similarity can be considered as a weak classifier for linking two objects detected within a temporal window. Secondly, we search the best number of dominant colors (denoted $\mathfrak{C}$) in context segments when the dominant color descriptor is used. We suppose the value range of $\mathfrak{C}$ is from 2 to 7 colors. An exhaustive search is performed to find its best value. Table 3 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 $w_2 = 0.21$, $w_3 = 0.46$, $w_5 = 0.33$, $w_1 = w_4 = 0$, $\mathfrak{C} = 2$. In this context segment, the object occlusion level is very low. 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 there exist at the same time objects of large and small 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 large mobile objects. In the second context segment (from frame 301 to 600), the density and the occlusion level of mobile objects increase. The dominant color descriptor weight is higher than the one in previous context segment because this descriptor integrated with the

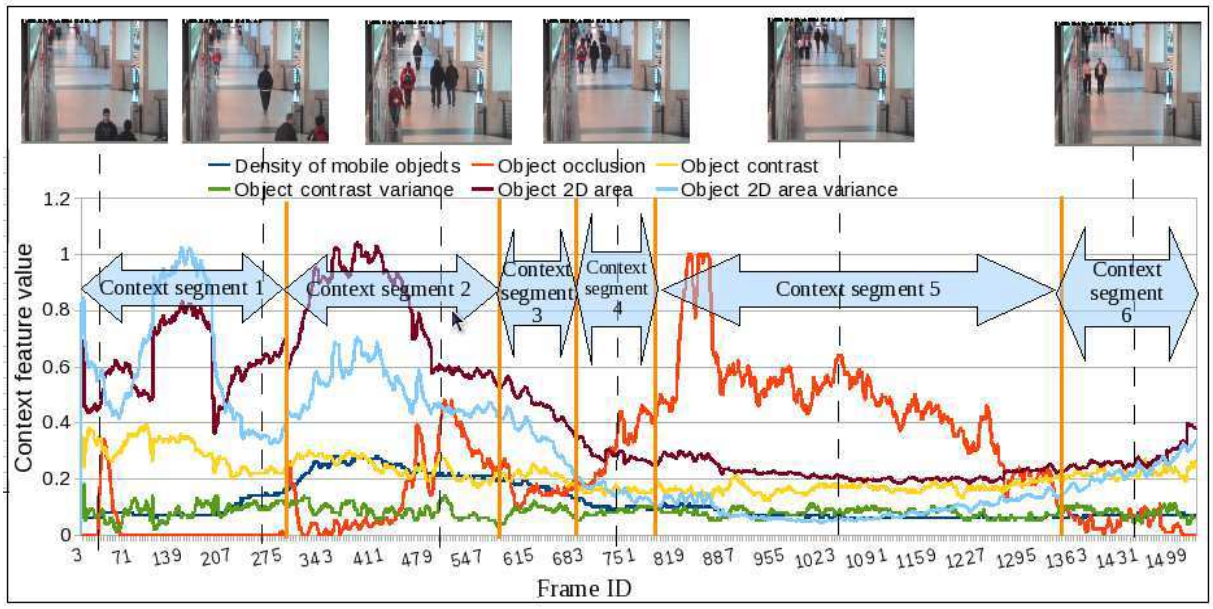

Figure 8: Context segmentation of sequence ThreePastShop2cor (belonging to Caviar dataset). The context segments are separated by the vertical orange lines. The control parameters are then learned for each context segment.

spatial pyramid kernel can manage the object occlusion cases (see [Chau et al., 2011a] for more details). For context segments 3 and 4, the dominant color descriptor weight is still selected as the most important descriptor for object tracking. In context segment 4, the objects are smaller, so the number of dominant color descriptor increases from 2 to 3 to better discriminate the objects. In context segment 5, the value of object 2D area decreases significantly. While the dominant colors between small objects might be similar, the color histogram descriptor is reliable because this descriptor takes into account all pixels 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 0.34. The color covariance descriptor is also used for solving the occlusion cases which occur frequently in this context segment. In the last context segment, the object 2D area variance increases, therefore the object 2D area descriptor is selected again with the weight $w_2 = 0.2$.

After segmenting the 12 training videos, we obtain 58 contexts. By applying the clustering process, 21 context clusters are created. Table 4 presents the learned control parameters for each cluster. The shape ratio descriptor is defined as the ratio between object 2D width and height. This descriptor is never selected in the context clusters because it cannot well discriminate the mobile objects in these training videos.

Context segment	From frame	To frame	Learned control parameter values					
			w_1 (Shape ratio)	w_2 (2D area)	w_3 (RGB color histogram)	w_4 (Color covariance)	w_5 (Dominant color)	$\mathfrak{C}$ (Number of dominant colors)
1	1	300	0	0.21	**0.46**	0	0.33	2
2	301	600	0	0.22	0	0.01	**0.77**	2
3	601	700	0	0	0	0	**1**	2
4	701	800	0	0	0.17	0	**0.83**	3
5	801	1350	0	0	**0.34**	0.33	0.33	2
6	1351	1510	0	0.20	0.20	0.20	**0.40**	2

Table 3: Learned control parameter values for the sequence ThreePastShop2cor (belonging to Caviar dataset). The most important object descriptor weights are printed bold.

The cost of this training phase mainly depends on the tracking parameter optimization time. This phases requires about 8 hours for 60.24 minutes of training videos corresponding to 18071 frames and 165 mobile objects. This is done with an Intel(R) Xeon(R) CPU E5430 @ 2.66GHz (4 cores) and of 4GB RAM.

6.3.2. Testing Phase

- Controller Experimentation with Automatically Detected Objects

1. Caviar Dataset

The processing Caviar videos have 26 sequences in which 6 sequences belong to our training video set. The other 20 sequences including 143 mobile objects are used for testing. The proposed controller is experimented in two cases to show its robustness. In the first case, only five object descriptor weights are considered for tuning; the number of dominant colors $\mathfrak{C}$ is set by default to 2. In the second case, all the six parameters are considered for tuning. Table 5 presents the tracking results of the proposed approach and of some recent trackers from the state of the art. In the first case, the proposed controller increases significantly the performance of the appearance tracker. The MT value increases from 78.3% to 84.6% and the ML value decreases from 5.2% to 5.1%. In the second case, when the parameter $\mathfrak{C}$ is also tuned by the controller, the tracking performance continues to be improved. The MT value increases from

Context cluster Id	Learned Control Parameter Values					
	w_1 (Shape ratio)	w_2 (2D area)	w_3 (RGB color histogram)	w_4 (Color covariance)	w_5 (Dominant color)	$\mathfrak{C}$ (Number of dominant colors)
1	0	0.20	0.14	0.10	**0.56**	2
2	0	0.21	**0.45**	0	0.34	2
3	0	0.08	0.05	0.12	**0.75**	2
4	0	0.12	0.17	0.03	**0.68**	2
5	0	0.12	0.16	0.11	**0.61**	2
6	0	0.11	0.19	0.07	**0.62**	2
7	0	0	**0.66**	0	0.34	3
8	0	0.15	0.15	0	**0.69**	2
9	0	0.14	0.16	0.17	**0.52**	2
10	0	0	**1**	0	0	
11	0	0	**1**	0	0	
12	0	0.05	**0.86**	0	0.09	3
13	0	0.14	**0.39**	0.17	0.3	4
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**	2
21	0	0.1	0	0.15	**0.75**	2

Table 4: Result of the training phase for the appearance tracker. 21 context cluster are created. The most important object descriptor weights are in bold.

Method	MT (%)	PT (%)	ML (%)
[Wu & Nevatia, 2007]	75.7	17.9	6.4
[Huang et al., 2008]	78.3	14.7	7.0
[Xing et al., 2009]	84.3	12.1	3.6
[Li et al., 2009]	84.6	14.0	1.4
[Kuo et al., 2010]	84.6	14.7	**0.7**
Appearance Tracker [Chau et al., 2011a] without the proposed controller	78.3	16.5	5.2
Appearance Tracker [Chau et al., 2011a] with the control of object descriptor weights	84.6	10.3	5.1
Appearance Tracker [Chau et al., 2011a] with the control of object descriptor weights and number of dominant colors	**85.7**	**11.3**	**3.0**

Table 5: Tracking results on the Caviar dataset. MT: Mostly tracked trajectories, higher is better; PT: Partially tracked trajectories; ML: Mostly lost trajectories, lower is better. The proposed controller improves significantly the tracking performance. The best values are printed in red.

84.6% to 85.7% and the ML value decreases from 5.1% to 3.0%. We obtain the best MT value compared to state of the art trackers.

In the rest of the article, we only present the results of the controller while tuning all the six parameters (i.e w_i with $i = 1..5$ and $\mathfrak{C}$).

2. PETS Video

The video of the second test belongs to the PETS dataset 2009. PETS videos are not used for learning. We select the sequence S2_L1, camera view 1, time 12.34 for testing because this sequence is experimented in several state of the art trackers. This sequence has 794 frames, contains 21 mobile objects and several occlusion cases (see figure 9a.).

In this test, we use the tracking evaluation metrics presented in [Kasturi et al., 2009] to compare with other tracking algorithms. The first metric is ATA (Average Tracking Accuracy) which computes the average accurate tracking time per object. The second metric is MOTP

a. b.

Figure 9: Illustration of two test videos: a. PETS and b. Vanaheim

(Multiple Object Tracking Precision) which is calculated from the spatio-temporal overlap between the ground-truth trajectories and the algorithm's output trajectories. The third metric is MOTA (Multiple Object Tracking Accuracy) which penalizes the number of missed detection, false positives and switches in the output trajectory for a given reference ground-truth trajectory. All the three metrics are normalized in the interval $[0,\ 1]$. The higher these metrics, the better the tracking quality is.

For this sequence, the controller selects the parameters associated to context cluster 6 for tracking. The dominant color descriptor is selected as the most important descriptor for tracked objects because this descriptor can well handle the object occlusion cases. With the proposed controller, the tracking result increases significantly. Table 6 presents the metric results of the proposed approach and of different trackers from the state of the art. The metric $\overline{M}$ represents the average value of the three metrics. With the proposed controller, we obtain the best values in metrics ATA, MOTP and $\overline{M}$. The MOTA value of our approach (0.75) gets the second rank due to some missed detection.

3. Vanaheim Video

The video of the third test belongs to the European Vanaheim project (see figure 9b). Vanaheim videos are not used for learning. The test sequence contains 36006 frames and lasts 2 hours. Table 7 presents the performance of the proposed approach and three recent trackers from state of the art.

For this sequence, the proposed controller improves the performance of the tracker [Chau et al., 2011a]. The MT value increases from 55.26% to 60.53%. The ML value decreases significantly from 13.16% to 2.63%. The tracking result with the proposed controller gets the

Method	ATA	MOTP	MOTA	$\overline{M}$
[Arsic et al., 2009]	0.02	0.46	0.41	0.30
[Berclaz et al., 2009]	0.14	0.50	0.56	0.40
[Breitenstein et al., 2009]	0.30	0.60	0.74	0.55
[Ge & Collins, 2009]	0.04	0.46	0.65	0.38
[Alahi et al., 2009]	0.04	0.53	0.61	0.39
[Conte et al., 2010]	0.09	0.64	**0.83**	0.52
Appearance Tracker [Chau et al., 2011a] without the proposed controller	0.26	0.63	0.62	0.50
Appearance Tracker [Chau et al., 2011a] with the proposed controller	**0.31**	**0.69**	**0.75**	**0.58**

Table 6: Tracking results on the sequence S2.L1, camera view 1, sequence time 12.34. The proposed controller improves significantly the tracking performance. The best values are printed in red.

best quality among the trackers presented in table 7.

- Controller Experimentation with Manually Annotated Objects

All the six context feature values depend on the object bounding boxes. The training phase is performed with annotated objects, so a low quality object detection in the online phase decreases the quality of the context detection. So, one drawback of the proposed controller is the dependence of its performance on the object detection quality. In this section, manually annotated objects are used for testing the controller. This experiment helps to better evaluate the proposed controller performance because the errors of the object detection task are eliminated. We test two video sequences. The first one is the OneStopMoveEnter2cor sequence belonging to the Caviar dataset. The second one is the Vanaheim video experimented previously.

Table 8 summarizes the obtained tracking results (without and with the controller) on these two sequences in two cases: using automatically detected objects and using manually annotated objects. For the OneStopMoveEnter2cor sequence, the controller increases the MT value by 18.18% (from 72.73% to to 90.91%) in the second case and only by 9.09% (from 72.73% to 81.82%) in the first case. For the Vanaheim sequence, in the second case, the controller

Method	#GT	MT (%)	PT (%)	ML (%)
[Chau et al., 2011b]	38	10.53	13.16	76.31
[Souded et al., 2011]	38	44.74	42.11	13.15
Appearance Tracker [Chau et al., 2011a] without the proposed controller	38	55.26	31.58	13.16
Appearance Tracker [Chau et al., 2011a] with the proposed controller	**38**	**60.53**	**36.84**	**2.63**

Table 7: Tracking results on the Vanaheim video. #GT denotes the number of ground-truth trajectories. The proposed controller improves significantly the tracking performance. The best values are printed in red.

	Sequence	Method	#GT	MT(%)	PT(%)	ML(%)
Using **automatically** detected objects	OneStopMove-Enter2cor	Without the proposed controller	11	72.73	18.18	9.09
		With the proposed controller	**11**	**81.82**	**18.18**	**0**
	Vanaheim	Without the proposed controller	38	55.26	31.58	13.16
		With the proposed controller	**38**	**60.53**	**36.84**	**2.63**
Using **manually** annotated objects	OneStopMove-Enter2cor	Without the proposed controller	11	72.73	27.27	**0**
		With the proposed controller	**11**	**90.91**	**9.09**	**0**
	Vanaheim	Without the proposed controller	38	92.47	7.53	**0**
		With the proposed controller	**38**	**100**	**0**	**0**

Table 8: Results of the appearance tracker for the OneStopMoveEnter2cor and Vanaheim video sequences in two cases: using detected objects and using annotated objects. The controller improves the tracking performance more significantly in the second case. Best values are in red.

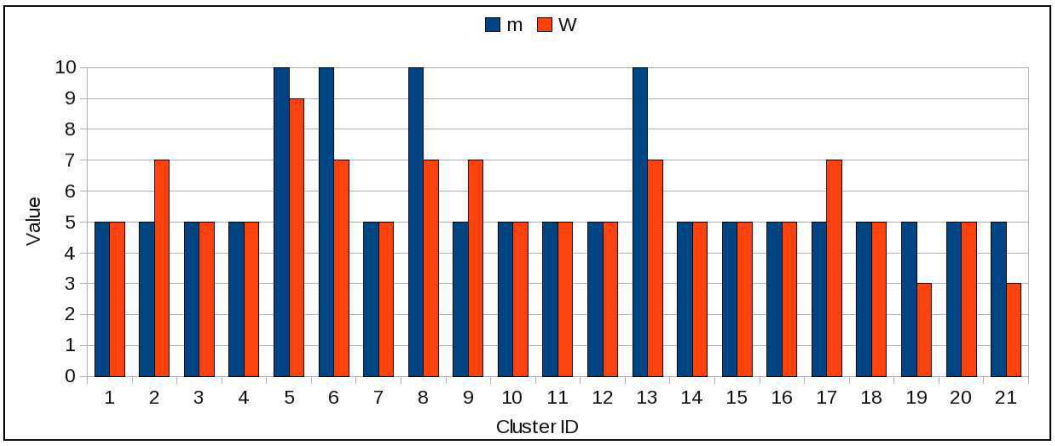

Figure 10: Result of the training phase of control parameters of the KLT tracker

increases the MT value by 7.53% (from 92.47% to 100%) compared to 5.27% in the first case.

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 controller performance is proportional to the object detection quality.

6.4. KLT Tracker Control

The KLT tracker relies on the tracking of Kanade-Lucas-Tomasi (KLT) features [Shi & Tomasi, 1994]. The KLT tracker takes detected objects as input. The object tracking relies on the number of matching KLT features over time between the detected objects. For the KLT tracker, we find two parameters depending on the tracking context: the minimum distance between KLT feature points m and the size of feature window W (see the definition of W at formula (3) of [Shi & Tomasi, 1994]). For example, in the case of object occlusion, the values of m should be low to detect a high enough number of KLT features for each object. When object 2D area is large, the values of m and W should be high to take into account whole object. Therefore these two parameters are selected for experimenting the proposed control approach so that hypothesis 1 (section 1.1) is ensured. We train the controller for this tracker on the same 12 training video sequences presented in section 6.3.1. The 20 Caviar videos (not belonging to the training sequences) are used for testing.

6.4.1. Training Phase

We suppose that the minimum distance m can get the values 3, 5, 7, 9 pixels and the feature window size W can get the values 5, 10, 15 pixels. In the tracking parameter optimization,

Method	MT (%)	PT (%)	ML (%)
KLT Tracker without the proposed controller	74.4	13.4	12.2
KLT Tracker with the proposed controller	**80.0**	**13.3**	**6.7**

Table 9: Tracking results on the Caviar dataset. Our controller improves significantly the tracking performance.

due to the small space of control parameters, we use an enumerative search to learn satisfactory parameter values for each context. Figure 10 presents the learned control parameter values for each context cluster.

6.4.2. Testing Phase

Table 9 presents the tracking results for 20 test Caviar videos in both cases: without and with the proposed controller. In the first case, the values of m and W are set by default to 5. While using the proposed controller, the tracking performance is increased significantly. The MT value increases by 5.6% (from 74.4% to 80%) and the ML value decreases from 12.2% to 6.7%. Compared to the improvement of the MT value for the appearance tracker which is 7.4% (from 78.3% to 85.7%, see table 5), the controller performance for the KLT tracker is less significant because fewer parameters are controlled and these parameters influence less the tracking quality. Also, they depend less on the tracking context.

6.5. Surf Tracker Control

We train the controller for this tracker on the same 12 training video sequences presented in section 6.3.1. The two videos belonging to PETS dataset[6] and TUD dataset [Andriluka et al., 2010] are used for testing. These two datasets are not used in the training phase.

The Surf tracker relies on the tracking of Surf (Speeded Up Robust Features) [Bay et al., 2008]. Similar to the KLT tracker, the Surf tracker takes detected objects as input. The object tracking relies on the number of matching Surf features over time between the detected objects. For the Surf tracker, we consider two parameters:

- Hessian threshold h: This is a threshold for the key point detector. Only features, whose hessian is larger than Hessian threshold are retained by the detector. Therefore, the larger

[6]http://www.cvg.rdg.ac.uk/PETS2013/a.html

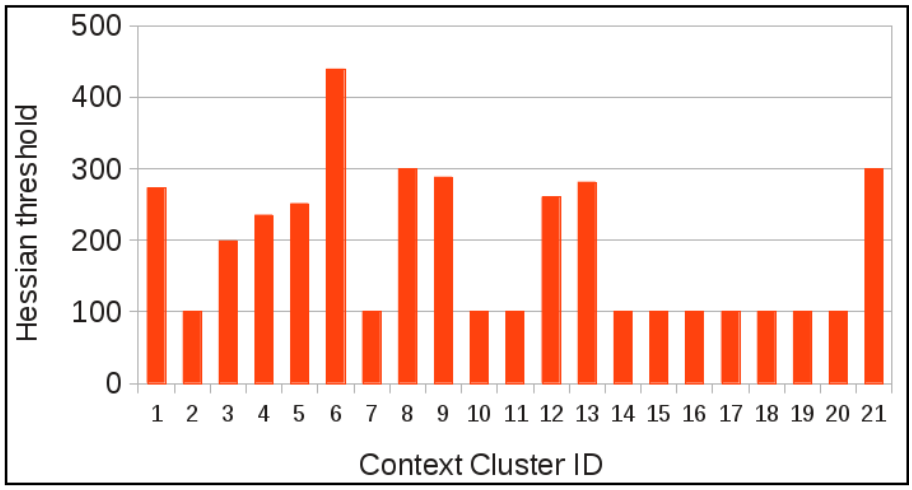

Figure 11: Training results of the Hessian threshold parameter for 21 context clusters

the value, the less key points are detected.

- Number of octave layers n: The number of images within each octave of a Gaussian pyramid.

6.5.1. Training Phase

We suppose that the Hessian threshold h can get the values 100, 300, 500 and the number of octave layers n can get the values 2, 4, 6. In the tracking parameter optimization, due to the small space of control parameters, we use an enumerative search to learn satisfactory parameter values for each context.

Similar to the training phases of the previous trackers, 21 context clusters are created. We compute then satisfactory tracking parameters for each context cluster. Figures 11 and 12 present respectively the training results of the parameters of Hessian threshold and the number of octave layers for 21 context clusters. For each context cluster, satisfactory tracking parameters are defined as weighted combinations of the ones of contexts belonging to that cluster. Therefore the learned values of control parameters can be different from the values which are initially determined. For example, the learned hessian threshold value of context cluster 3 is 200; the learned number of octave layers of context cluster 6 is 5. From cluster 1 to 9, the learned pamrameter values are quite different each other. This means that these two control parameters are influenced by the tracking context.

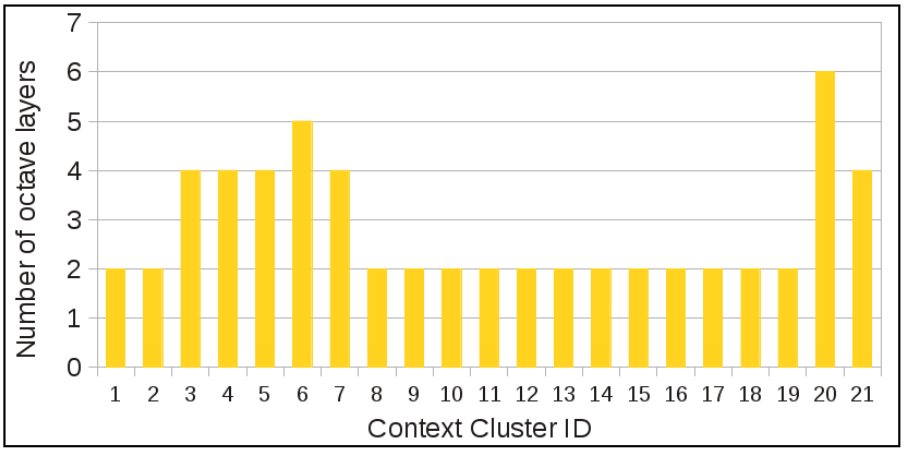

Figure 12: Training results of the parameter of number of octave layers for 21 context clusters

6.5.2. Testing Phase

In the testing phase, when the controller is not used, the value of hessian threshold is set to 100, and the value of number of octave layers is set to 2. These two values are selected because they are determined as the satisfactory values for many context clusters in the training phase.

1. PETS Video

This PETS video is also the one tested at section 6.3.2. Illustration of this video is presented at figure 9a. Table 10 presents the metric results of the proposed approach and of different trackers from the state of the art. While using the proposed controller, the tracking result increases significantly. The value of MOTA increases 0.80 to 0.86; the value of MOTP increases 0.66 to 0.69; and the value of $\overline{M}$ increases 0.73 to 0.78. The obtained values are the best compared to the ones presented in the table.

2. TUD dataset

For the TUD dataset, we select the TUD-Stadtmitte sequence for testing. This video contains only 179 frames and 10 objects but it is very challenging due to heavy and frequent object occlusions. For this sequence, the controller selects context cluster 13 in which parameters $h = 281$, $n = 2$ are used for parameterizing the tracking process. With such high value of h, the number of detected Surf points is small. In this tracker, we take the detected objects as input and compute Surf points in corresponding 2D bounding boxes. In the case of high occlusion level as in this video, object bounding boxes may contain a part of other objects. A low number

Methods	MOTA	MOTP	$\overline{M}$
[Berclaz et al., 2011]	0.80	0.58	0.69
[Shitrit et al., 2011]	0.81	0.58	0.70
[Henriques et al., 2011]	0.85	**0.69**	0.77
Surf tracker [Bay et al., 2008] without the proposed controller	0.80	0.66	0.73
Surf tracker [Bay et al., 2008] with the proposed controller	**0.86**	**0.69**	**0.78**

Table 10: Tracking results on the PETS sequence S2.L1, camera view 1, time 12.34. MOTA: Multiple Object Tracking Accuracy; MOTP: Multiple Object Tracking Precision (higher is better). The best values are printed in **red**.

Methods	MT(%)	PT(%)	ML(%)
[Kuo & Nevatia, 2011]	60.0	30.0	**10.0**
[Andriyenko & Schindler, 2011]	60.0	30.0	**10.0**
Surf tracker [Bay et al., 2008] without the proposed controller	50.0	10.0	40.0
Surf tracker [Bay et al., 2008] with the proposed controller	**70.0**	**10.0**	**20.0**

Table 11: Tracking results for the TUD-Stadtmitte sequence. MT: Mostly tracked trajectories, higher is better; PT: Partially tracked trajectories; ML: Mostly lost trajectories, lower is better. The best values are printed in **red**.

of detected Surf points helps to decrease the distribution of these points on different objects. The tracking quality is then better.

Figures 13 to 16 illustrate the tracking output in two cases: without controller (figures 13 and 14) and with the proposed controller (figures 15 and 16). While there is a ID switch between two persons (marked my arrows) in the first case, this error is solved in the second case. Table 11 presents the tracking results of the proposed approach and three recent trackers from the state of the art. While using the proposed controller, the MT value increases significantly 50% to 70%. Also the obtained MT value is the best compared to these three trackers.

In all the testing video sequences and for three trackers, the online processing time increases only slightly (less than 10%) when the controller is used.

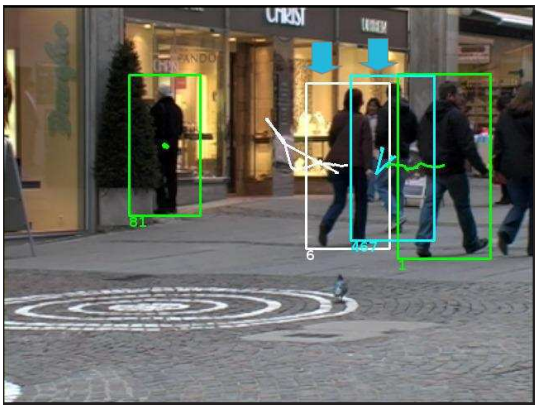

Figure 13: **Frame 51, without controller**: Persons 6 and 467 are tracked correctly before their occlusion

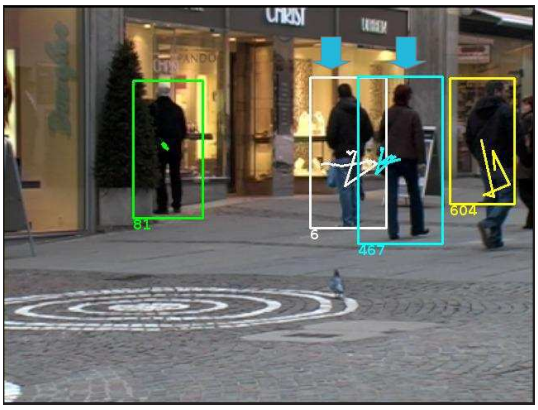

Figure 14: **Frame 70, without controller**: Persons 6 and 467 switch their ids after their occlusion

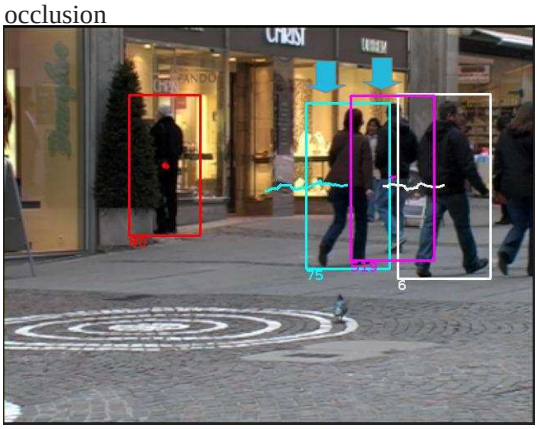

Figure 15: **Frame 51, with the proposed controller**: Persons 75 and 519 are tracked correctly before their occlusion

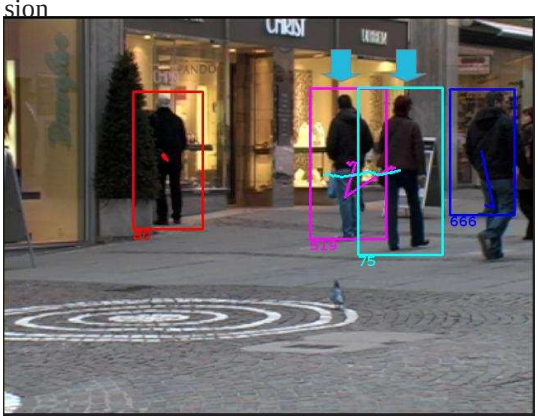

Figure 16: **Frame 70, with the proposed controller**: Persons 75 and 519 are still tracked correctly after their occlusion

7. Conclusion

In this article, we have presented a new control approach for object tracking which is generic, flexible and intelligent. More precisely in order to cope with tracking context variations, this approach learns how to tune the parameters of 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 surrounding background, their contrast variance, their 2D area and their 2D area variance. In an offline phase, we learn satisfactory tracking parameters for context clusters. In the online control phase, once a context change is detected, the tracking parameters are tuned using the learned values. This method is able to control trackers belonging to two different categories (appearance tracking and point tracking). Moreover, other tracker category can still be controlled by adapting the context notion to the tracker principle (for example to control silhouette-based trackers, we can add the object rigidity feature to the context). The training and testing phases are not time consuming. The proposed approach has been experimented with three trackers on a long, complex video and on three public datasets (Caviar, PETS and TUD). The experimental results show a significant improvement of the performances while using the proposed controller.

In future work, we will extend the context notion which should be independent from the object detection quality. Also, the proposed control approach should be able to interact with the object detection task to improve the detection quality. An online mechanism for updating the learned database is also necessary to increase the performance of the proposed approach.

References

Alahi, A., Jacques, L., Boursier, Y., & Vandergheynst, P. (2009). Sparsity-driven people localization algorithm: Evaluation in crowded scenes environments. In *The International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), in conjunction with CVPR*.

Andriluka, M., Roth, S., & Schiele, B. (2010). Monocular 3d pose estimation and tracking by detection. In CVPR.

Andriyenko, A., & Schindler, K. (2011). Multi-target tracking by continuous energy minimization. In CVPR.

Arsic, D., Lyutskanov, A., Rigoll, G., & Kwolek, B. (2009). Multi camera person tracking applying a graph-cuts based foreground segmentation in a homography framework. In *The International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), in conjunction with CVPR*.

Bay, H., Ess, A., Tuytelaars, T., & Gool, L. (2008). Surf: Speeded up robust features. *In The Journal of Computer Vision and Image Understanding (CVIU), 110*, 346–359.

Berclaz, J., Fleuret, F., & Fua, P. (2009). Multiple object tracking using flow linear programming. In *The International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), in conjunction with CVPR*.

Berclaz, J., Fleuret, F., Turetken, E., & Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *TPAMI*, *33*, 1806–1819.

Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., & Gool, L. V. (2009). Markovian tracking-by-detection from a single, uncalibrated camera. In *The International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), in conjunction with CVPR*.

Caporossi, A., Hall, D., Reignier, P., & 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 ECCV, Prague, Czech*.

Chau, D. P., Bremond, F., & 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*.

Chau, D. P., Bremond, F., Thonnat, M., & 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*.

Conte, D., Foggia, P., Percannella, G., & Vento, M. (2010). Performance evaluation of a people tracking system on pets2009 database. In *The International Conference on Advanced Video and Signal Based Surveillance (AVSS)*.

Corvee, E., & 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)*.

Everitt, B. S., Landau, S., & Leese, M. (2001). In *book: Cluster Analysis (Fourth ed.), London: Arnold. ISBN 0-340-76119-9*.

Freund, Y., & 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).

Ge, W., & Collins, R. T. (2009). Evaluation of sampling-based pedestrian detection for crowd counting. In *The International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), in conjunction with CVPR*.

Georis, B., Bremond, F., & 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).

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning (1st ed.). Addison-Wesley Longman Publishing Co., Inc.,. Boston, MA, USA.

Hall, D. (2006). Automatic parameter regulation of perceptual system. In *The Journal of Image and Vision Computing* (pp. 870–881).

Henriques, J. F., Caseiro, R., & Batista, J. (2011). Globally optimal solution to multi-object tracking with merged measurements. In ICCV.

Heyer, L. J., Kruglyak, S., & Yooseph, S. (1999). Exploring expression data: Identification and analysis of coexpressed genes. In *The Journal of Genome Research* (pp. 1106–1115).

Huang, C., Wu, B., & Nevatia, R. (2008). Robust object tracking by hierarchical association of detection responses. In *The European Conference on Computer Vision (ECCV)*.

Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., Boonstra, M., Korzhova, V., & Zhang, J. (2009). Framework for Performance Evaluation of Face, Text,

and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol . In *The IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (pp. 319–336).

Kim, K., Chalidabhongse, T., Harwood, D., & Davis, L. (2004). Background modeling and subtraction by codebook construction. In *The International Conference on Image Processing (ICIP), Singapore*.

Kuo, C., & Nevatia, R. (2011). How does person identity recognition help multi-person tracking? In CVPR.

Kuo, C. H., Huang, C., & Nevatia, R. (2010). Multi-target tracking by online learned discriminative appearance models. In *The International Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA*.

Li, Y., Huang, C., & Nevatia, R. (2009). Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *The International Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nievergelt, J. (2000). Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power. In *The Conference on Current Trends in Theory and Practice of Informatics*.

Santner, J., Leistner, C., Saffari, A., Pock, T., & Bischof, H. (2010). PROST: Parallel Robust Online Simple Tracking. In *The International Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA*.

Sherrah, J. (2010). Learning to Adapt: A Method for Automatic Tuning of Algorithm Parameters. In *The International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS), Sydney, Australia* (pp. 414–425).

Shi, J., & Tomasi, C. (1994). Good features to track. In *The International Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shitrit, H. B., Berclaz, J., Fleuret, F., & Fua, P. (2011). Tracking multiple people under global appearance constraints. In ICCV.

Souded, M., Giulieri, L., & 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 (ICDP), London, UK*.

Thonnat, M., Moisan, S., & Crubezy, M. (1999). Experience in Integrating Image Processing Programs. In *ICVS, Lecture Notes in Computer Science, Spain*.

Wu, B., & 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*, *75*, 247–266.

Xing, J., Ai, H., & 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)*.

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. In *The ACM Computing Surveys (CSUR)*.