UNIVERSITÀ DEGLI STUDI DI MILANO

SCUOLA DI DOTTORATO IN INFORMATICA

DIPARTIMENTO DI INFORMATICA

CORSO DI DOTTORATO - CICLO XXV

TESI DI DOTTORATO DI RICERCA

BTLD$^+$: A Bayesian Approach to Tracking Learning Detection by Parts

INF/01

Candidato
Giorgio Gemignani

Tutor
Prof. Alfredo Petrosino

Coordinatore del Dottorato
Prof. Ernesto Damiani

ANNO ACCADEMICO 2012/2013

This thesis is dedicated to my family
For their endless love, support and encouragement

Acknowledgments

First of all, I have to thank myself. To my madness in beginning this amazing adventure.
I would like to express my gratitude to my parents, Domenico and Francesca for their understanding, encouraging as well personal guidance. Thanks to my sister Sara and her boyfriend Enrico, for supporting me in this last critical year. A very special acknowledgment goes to my girlfriend Maria, who loved and supported me during the final, critical months of my dissertation, and made me feel like anything was possible. I love you, Mary. I would also like to acknowledge her parents Giuseppe and Alessandra, as well as her brother Edo, who treated me like family.
I would like to sincerely thank my supervisor, Prof. Alfredo Petrosino, for his guidance and support throughout this study, and especially for his confidence in me. I would also like to thank Dr. Ferone for helping me. His comments and questions were very beneficial in my completion of the manuscript. I learned from his insight a lot.
During these years, I have collaborated with people of *CVPRLab* (Computer Vision and Pattern Recognition Laboratory), of University of Naples "Parthenope", with whom I shared funny and joyful moments, and to whom I wish to extend my warmest thanks. Thank you Mario , Alessia and Ihsan.
To all my friends, thank you for your understanding and encouragement in my many,many moments of crisis. Your friendship makes my life a wonderful experience. I cannot list all the names here, but you are always on my mind.

Tracking objects of interest in video streams, referred in computer vision literature as *visual tracking*, is a challenging task involving image perception and object understanding capabilities. Visual trackers rely on an appearance model, i.e. an internal representation of the target appearance, learned by extracting highly discriminative visual informations characterizing the target. Target localization is formulated as the problem of searching for the most similar image region to representation encoded into the appearance model.
Despite extensive research on such topic, it remains extremely challenging and far from being solved. Indeed, several factors such as illumination variation, partial occlusion, shape deformation, and camera motion generate high variability in object appearance. Learning such variability is the main issue addressed by visual trackers.
In the last decades, significant advances have been achieved in tracking known classes of objects such as people and cars. For such objects, a large amounts of hand-labeled data is available and supervised learning methods have demonstrated impressive results in capturing the complete data variability. However, in many practical applications only partially labeled or unlabeled data is available. Building an appearance model by supervised learning methods requires hand-labeling for such unlabeled data, a very annoying and time consuming process. To overcame this task, researchers are moving towards semi-supervised learning techniques. With such methodologies a learner builds the target appearance model from a minimal set of labeled data. Model adaption is performed by exploring the large amount of unlabeled data revealed during the tracking process. Such scenario has been recently proposed to the attention of computer vision scientists through the definition of the challenge of "*robust long-term visual tracking of an unknown object with minimum prior information*", where labeled data are available *only at the first frame of the sequence.*
The contribution proposed in this thesis focuses on this particular instance of the visual tracking problem, referred as *Adaptive Ap-*

pearance Tracking. We proposed different approaches based on the Tracking Learning Detection (*TLD*) decomposition proposed in [55]. *TLD* decomposes visual tracking into three components, namely the *tracker*, the *learner* and *detector*. The *tracker* and the *detector* are two competitive processes for target localization based on complementary sources of informations. The former searches for local features between consecutive frames in order to localize the target; the latter exploits an on-line appearance model to detect confident hypothesis over the entire image. The *learner* selects the final solution among the provided hypothesis. It updates the target appearance model, if necessary, reinitialize the tracker and bootstraps the detector's appearance model. In particular, we investigated different approaches to enforce the *TLD* stability. First, we replaced the tracker component with a novel one based on *mcmc particle filtering*; afterwards, we proposed a robust appearance modeling component able to characterize deformable objects in static images; after all, we integrated a modeling component able to integrate local visual features learning into the whole approach, lying to a couple layered representation of the target appearance.

Keywords: computer vision, machine learning, on-line learning, boosting, feature selection, object detection, object tracking, background modeling,particle filter, mcmc, drifting, supervision.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The proliferation of high-powered computers, the availability of high quality and inexpensive video cameras has generated a great deal of interest in development of intelligent machines able to understand and react to their surrounding environment using visual perception as humans. The research discipline involved with development of mathematical models and algorithms able to understand digital images and videos is referred as computer vision [76] and represent the scientific field investigated in this thesis.
Among the huge set of problems investigated by computer vision scientists, *visual tracking* or *video tracking*, represents a challenging task. It formalizes the problem of localizing an object of interest in video stream. We investigated on a particular instance of *visual tracking* problem defined as the task of *localizing an unknown object in unconstrained video stream.* Our objective is to enable real-time, accurate as well as robust tracking.
This chapter provides an overview of the entire thesis. Section 1.1 introduces the problem. Section 1.2 presents the main challenges that have to be tackled. Section 1.3 introduces possible applications of our research. Section 1.4 introduces the contributions made in the thesis. Section 1.5 outlines the rest of the thesis.

1.1 Adaptive Visual Tracking

In its elementary form, *visual tracking* is formulated as the problem of estimating the location X^t of an object of interest at time t given a sequence of images $I_1, \cdots, I_t$ extracted from a video stream.
To estimate object trajectory, visual trackers learn an internal representation of the target appearance, the *appearance model*, by extracting highly discriminative visual features characterizing the target in each incoming video frame. This model is exploited to evaluate image regions similarity and estimate the most confident target location in the current frame.
In [54], *single target trackers* are classified into two broad categories namely, *short term trackers* (*STT*) and *long term trackers* (*LTT*). The former refers to standard tracking approaches such as [96] that try to solve frame to frame correspondences assuming no complete occlusion or disappearance of the tracked object between consecutive frames; the latter refers to methods copying with sequences of possibly infinite length, affected by frame cuts, fast camera movements and object temporary disappearance from the scene. In [55] a new challenge for single target trackers is formulated as "*long-term on-line tracking with minimum prior information*". In this scenario, a tracker learns an appearance model by continuously adapting itself to new observed data and exploiting only information from the past. Minimum prior information underlines that the labeled data necessary to learn the appearance model, are provided manually by the user *only at the first frame of the sequence.* Such formulation poses fundamental questions on how continuously adapt the visual model describing the target. Model adaptation introduces several challenges, such as the need for simultaneous fulfillment of the contradicting goals of rapid learning and stable memory referred in [45] as the *stability-plasticity* dilemma. Furthermore, on-line evaluation of new data samples becomes a critical issue since detection and learning changes in pose and scale or varying illumination condition

is an essential feature for a stable tracker. These conditions generate an high variability into the data, namely the "*intra class variability*". The appearance model should be flexible enough capture such variability. At the same time it should be plastic to discard wrong measurements generated by tracking failures. This issue is referred as the "*wrong sample selection*" problem [93]. *Adaptive Appearance Trackers* (*AAT*), [55, 77, 118, 104, 6, 105] relying on models able to learn changing imaging conditions, represent the new generation of visual trackers designed to solve the challenges involved with the long term tracking task.

In the last decades, supervised statistical machine learning techniques [12] have demonstrated impressive results in learning complex object representation for a wide range of object classes such as car or people. Given a set of training samples, the objective of a supervised machine learning algorithm is to train a model able to predict the labels of samples even if they have not been observed during the training stage. Such capability is referred in machine learning literature as *generalization*.

In practice, both the training samples and their corresponding labels are provided by a human labeler in advance and are assumed to be enough representative to capture the whole intraclass variability. The learners are thus called supervised methods and when enough and proper training samples exist, these approaches can obtain very high recognition and classification performances.

Supervised learning algorithms are *batch* or *off-line* learning methods, because the learning algorithm analyzes all samples simultaneously and repeatedly process the entire training set until a certain stopping criterion is met; i.e., the training error has fallen under a certain threshold or the maximum number of iterations has been reached. The main limitation with such approaches is related to the data availability, that should be given in advance. In contrast to *off-line* methods, there exist *incremental* or *on-line* approaches where the data is usually not provided at once but arrives sequentially.

Both incremental and on-line methods process only one sample at a time and then update the model.
The main difference between the two is that in *on-line* learning each sample is discarded after an update, in incremental learning not. On-line learning is always incremental, whereas, incremental learning can be done either on-line or off-line.
The lack of sufficient labeled training data and the demand of real time localization make supervised learning methods not completely reliable for the long term visual tracking task and have led to recent attentions towards the investigation of unsupervised and semi-supervised on-line learning methods.
Unsupervised learning methods try to find an interesting natural structure in the data using training samples without their corresponding labels, i.e., unlabeled data. Although unsupervised learning in principle is convenient because human labeling can be fully avoided, it can be far from reaching the discriminative performance of supervised learning algorithms.
In contrast, semi-supervised learning tries to exploit both, a reasonably small amount of labeled data and a large amount of unlabeled data. Basically it combines the benefits of both supervised learning because highly discriminative classifiers are learned and unsupervised learning because it holds the potential to reasonably exploit a massive amount of unlabeled data. Semi-supervised approach represents the best learning methodology to design the long term tracking task. It naturally fits our problem and defines the specific field of research investigated in this thesis to build the target appearance model.
Despite several advances has been reached by adaptive visual trackers, long term tracking stability is nowadays an open problem far from been solved as stated by organizers of the visual object tracking workshop *VOT2013*[1]. The absence of a definitive solution to long

[1] *VOT2013*, is a workshop, which will be held in conjunction with the *International Conference on Computer Vision*, *ICCV2013*, where researchers are invited to participate in a first challenge focusing on single object visual tracking. Its aim is to provide a common platform for comparison, analysis and discussion of existing as well as new single object trackers.

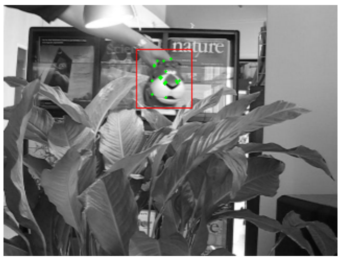
term visual tracking is one the most important aspect motivating the studies proposed in this thesis.

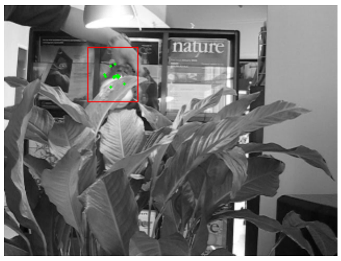
1.2 Challenges

Given a video stream depicting various moving objects and a bounding box enclosing an object of interest indicated by the user, our objective is to automatically discover object's position or reporting its absence in every successive frame. The object may be occluded or disappears from the camera field of view for an unknown length of time, reappearing at certain point and at any possible location. Frame cut in video stream and high noised images are also possible. In other words, our goal is to assigns consistent labels to the tracked object over the different frames of a video.
In this scenario, the construction of an appearance model able to continuously adapt to changes of appearance and at the same time, robust to wrong measurements generated by object having similar appearance, represent a fundamental issue in order to perform accurate visual tracking. Considering our tracking task, where an unknown object is marked in the first frame, a powerful tracking approach is to learn a binary classifier with the marked patch as positive sample and the surrounding patches as negatives, respectively. Then, the tracking task can be performed by using this classifier to re-detect the marked object in the subsequent frames. This approach is referred in literature as *tracking-by-detection* ([93]) and represents the most popular paradigm exploited to design adaptive appearance models. In order to make the classifiers adaptive towards rapid appearance and illumination changes, typically the classifier updates itself on the re-detected object, according to hand-designed update rules. Such approach yield highly fast and accurate trackers because the object has only to be discriminated versus its local background and the model complexity can be kept very low. Nevertheless, one problem

that all of these methods have in common is that *slight errors during the self-updating process can easily accumulate* and finally lead to failure of tracking, i.e., the object is lost. This is condition is referred to as the *drifting* problem. The main reason for drifting in *tracking-by-detection* approaches lies in the fact that supervised learners are abused for an in principle unsupervised learning task. In fact, as previously described, labeled data is only available at the first frame when the object is marked. In all subsequent frames the tracker performs autonomously, without getting any additional labels for the data it is observing.
Hence, the learner over time has to be able to exploit both labeled data and unlabeled data, which is a natural semi-supervised learning problem. Following this observation, in the recent years, novel online semi-supervised learning algorithms have applied to make the tracker less sensitive to the risk of including wrong hypothesis while updating the target model[55, 40, 41, 101].
Evaluating on-line and without ground truth the quality of the estimated target location, so as to adjust accordingly its contribution to model update, is main challenge associated with long term visual tracking. Additionally, the appearance of other objects and of the background may be similar to the appearance of the target and therefore may interfere with its observation. In such a case, image features extracted from non-target image areas may be difficult to discriminate from the features that we expect the target to generate. This phenomenon is referred as *clutter*.
Furthermore, there are several external events that interfere during the tracking, increasing the difficulty of the overall process.

Occlusion and disappearance of the object. During tracking, the object may be occluded or disappear from the camera field of view for an unknown length of time, reappearing at certain point and at any location. Occlusions can be classified in to two categories: *partial* occlusions and *total* occlusions.


d


e

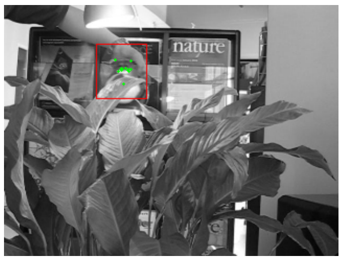

f

Figure 1.1: **Partial occlusion**. As the puppet starts moving behind the leafs, a partial occlusion is verified. (**a-b-c**) depict respectively frames 4, 58, 108 from the *Tiger1* sequence from *MILBOOST Dataset*[6]

The former affects only a small portion of the target area and are generally verified in cluttered scenes where other moving objects obscure the view of a target; in fig. 1.1, a partial occlusion of the puppet is depicted. The latter, involve the complete object disappearance and are typically verified when target is moving behind a static object, such as a column or wall. To address this challenge, a long term tracker should integrate a mechanism to control each level of occlusion. Local feature based representations, encoding information for a small region of the target, increase the robustness of a video tracker to partial occlusions. Higher-level reasoning about typical motion behaviors and preexisting occlusion patterns are generally used to deal with total occlusions.

Camera motion and viewpoint changes. The object of interest may change its appearance according to its position relative to the camera view. This events stresses the ability of the learning method to capture intraclass variability. When only a single sample from the initial frame is available, the problem becomes even more critical. Indeed, the initial object appearance can be very dissimilar from the other appearances observed during tracking. In fig. 1.2 an example of this challenge is showed.

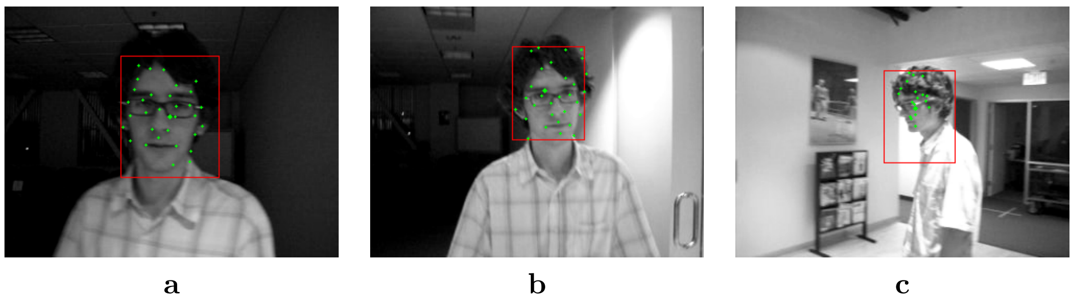


Figure 1.2: **Camera motion and viewpoint changes**. Camera and target motion inevitably affects target appearance state. (**a**- **b**-**c**) depicts respectively frames 1, 86, 156 from the *David* sequence from *MILBOOST Dataset*[6]

Ambient illumination. The direction, intensity and color of the external light sources corrupts the appearance of the target. Moreover, changes in global illumination are often a challenge in both outdoor (i.e. clouds obscure the sun) and indoor scenes(the light turns off, as in fig. 1.3).

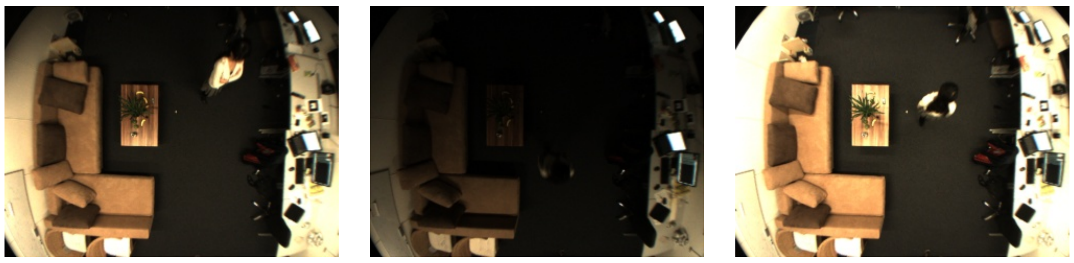


Figure 1.3: **Changes of light conditions**. Adversary light conditions are possible. (**a**-**b**-**c**) depicts respectively frames 3, 100, 115 of the *Night scenario* sequence from *KSERA Dataset* [116].

Noise. The image acquisition process is affected by a certain degree of noise, depending on the quality of the sensor. Additionally, motion blurring corrupts visual features representing the object appearance. In fig. 1.4 is depicted a scenario with low-resolution images. A long-term tracker should be robust to noise sources that may affect input data.

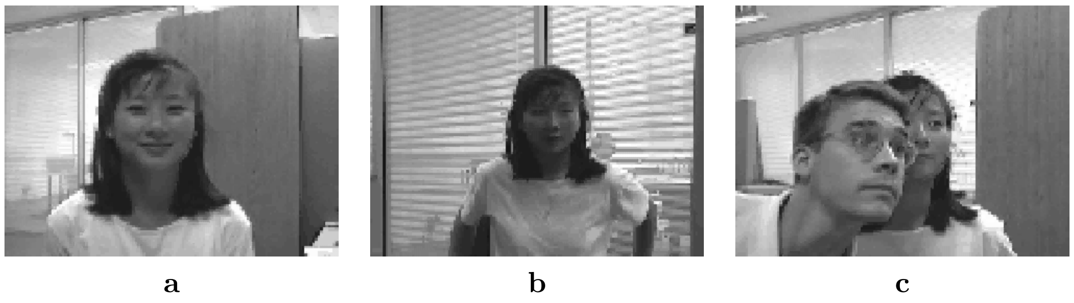


Figure 1.4: **Low sensor quality and motion blur**. Low qualities instances of the appearance model have to be learned. (**a**-**b**-**c**) depicts respectively frames 1, 100, 654 from the *Girl* sequence from *TLD Dataset* [55]

Another important aspect to be taken in account is the limited processing required by interactive applications. A tracker should run at full frame rate, limiting the complexity of algorithms that have to be extremely efficient.

1.3 Applications

Visual trackers can support several tasks in daily life scenarios. They provide low level information to huge number of higher level applications ranging from medical to video surveillance.
Their use can increase productivity in several context due to the reduced amount of manual work that is necessary to complete a task and to enable natural interaction with machines.
Video tracking is a useful tool used in automated video surveillance for security, business intelligence assisted living applications. In surveillance systems, tracking can be used either as a forensic tool or as a processing stage prior to algorithms that analyze suspicious behaviors [3] . Complex video surveillance system are built on combination of video-trackers and behavior analysis models, realizing an affective tool for security events analysis.

Smart surveillance systems pervade a variety of different indoor and outdoor environments such as roads, airports, railway stations, ports, public and private buildings such as schools and banks. Examples of video effective surveillance systems are Smart Surveillance System [50] developed by IBM and VisioWave Intelligent Video Platform [30] from General Electrics.
Another interesting application domain for video-trackers is retail intelligence [51], where tracking results can be used to learn a customers profile. Indeed, tracking the position of customer in a shop, combined with information from the point of sales provide information to learn a behavioural model describing customers preferences, how they interact with products depending on their location, and what items they buy. Such information helps the marketing team in different context such as products placement in the retail space or advertisement content selection.
Ambient assisted living (AAL) is another emerging area of research that exploit visual trackers to build intelligent environment able to contribute in increasing human comfort and simplify people daily activities. In [116] a socially assistive system is built to support some activities of daily life as well as health care needs of an elderly person, specifically persons suffering from chronic obstructive pulmonary Disease. Object localization is based on visual tracking from ceiling mounted camera.
Recent video analysis applications are built on the top of trackers results to produce enhanced visualization of sport events [100]. Trackers are effective to estimate the position of players in the field in order to gather statistics about a game (e.g. a football match). Statistics and enhanced visualizations aid the commentators, coaches and supporters in highlighting team tactics and player performance.
Recently video tracking has been increasingly used by medical systems to aid expert's diagnosis and speed up the clinical analysis task: in[83] automated algorithms track the ventricular motion in ultrasound images; in [83] an on-line learning based feature track-

ing method for accurate estimation and tracking of dynamic tissue deformation is proposed; authors in [115] estimates the trajectories of Escherichia coli bacteria from in vivo phase-contrast microscopy videos; moreover, video tracking has been applied to automatically estimate the position of particular instruments such as needles [29] and [80] during surgery.
Video tracking is also pervading the human-machine interaction domain providing new interfaces of communication with machines. Interactive games are common example where tracking technology is used to capture gestures that are converted into gaming actions [35]. Webcams are already shipped with tracking software that localizes and follows the face of a user for on-desk video conferencing [38].
In summary ability to robustly track an arbitrary object with minimal training stage is of great interest with possible applications in several scenarios such as gaming, advertisement, medical, education, tourism and military, providing another reason to motivate our research in this field.

1.4 Contribution

The main contribution provided in this thesis relies on the definition of novel effective approaches that bring long term visual tracking closer to a solution by letting the tracker adaption to changing environment conditions. In particular, we proposed to adapt the most crucial component of the visual tracker: the appearance model. Such adaptation is performed on-line, frame-by-frame as the object is localized in the image.
We propose a novel approach based on Tracking Learning Detection (*TLD*), a state of the art paradigm that decomposes visual tracking into three components, named the *tracker*, the *learning* and *detector*. The tracker localizes the target between consecutive frames searching for a fixed set of local visual features in a neighbor of previous estimated location. The detector maintains on-line an internal ap-

pearance model and explore the whole image to provide confident hypothesis on the target location. Exploiting the space-time structure in the video sequence, the learning component corrects detector's errors and build a robust appearance model. By combination of such independent processes, an effective appearance model is learned on-line from a minimal amount of labeled samples.

In this thesis, we investigated novel approaches to stabilize the appearance model adaption exploiting the *TLD* decomposition.

In particular we redesigned the tracker component to reduce its instability under critical conditions such as occlusions and resembling background. We proposed a novel approach based on Bayesian optimal filtering that performs feature selection and target state estimation in a joint fashion. In contrast to the baseline method, we introduced a filtering stage to select high discriminative features according to the maximum a posterior solution. Experiments have demonstrated the efficacy of the proposed approach while tracking non deformable objects, such as car and faces. However critical limitations have been observed on fast deformable objects such as humans. Indeed with such objects, local features becomes extremely variable and easily confused with surrounding background.

To overcame this limitation, we investigated a solution based on motion detection. Exploiting motion classification provided by robust background subtraction, we stabilize the tracker performances. We evaluated the proposed solution in an ambient assisted living scenario, where humans are observed from a ceiling mounted camera. Experiments demonstrated how motion classification refines on local feature selection. Unfortunately this strategy is feasible only for static cameras where static background can be statistically modeled.

To overcame this limitation, a novel local feature selection approach has been proposed. We designed a learning component able to discriminate among target local features. The proposed appearance model, in conjunction with the old one, defines a couple-layered representation of the object that is robust to the wrong sample selection problem. As demonstrated in experimental session, by combination of local feature and global features informations, we a more stable and effective long term tracker is obtained.

1.5 Thesis outline

The thesis is organized in two main parts. In the first part, to better contextualize our contributions, we first introduce a general formulation of the long term tracking problem. We reviewed the state of the art approaches to adaptive appearance modeling, analyzing in detail the*Tracking Learning Detection* approach. In the second Part we describe our approaches to adaptive appearance tracking.
More in detail, the parts are organized as follows:

Part I:

- in chapter 2, we briefly introduce an abstract view of the long term visual tracking task, analyzing the main subtask involved in its design: the state estimation problem and the appearance modeling problem. State estimation is formulated as bayesian inference process. A brief analysis of *mcmc particle filter* is presented. Appearance modeling is analyzed from a machine learning perspective. We provide a compounded review on state of the art approaches adopted to build object visual representations.
- In chapter 3, we analyze in detail the *Tracking Learning Detection* paradigm, pointing out strength and weakness that motivated the direction of research followed in this thesis.

Part II:

- In chapter 4 we propose a bayesian approach to tracking learning detection (*BTLD*). We designed a novel tracker component able to improve *TLD* performances under critical conditions such as occlusions and resembling background. Our approach exploits *bayesian optimal filtering* for local feature selection and target state estimation. We designed a particle filtering algorithm for parameter inference and propose a solution that enables accurate and efficient tracking. The performance and the long-term stability are demonstrated and evaluated on a set of challenging video sequences usually employed to test tracking algorithms.

- In chapter 5, we extended the *BTLD* approach for tracking fast deformable objects in static images. We integrated a motion detection component based on robust background subtraction able to exchange information with both the tracker and the detector component. Exploiting motion classification we redesigned the appearance model, proposing a novel one based on color histogram representation.

- In chapter 6, we formulate our novel appearance modeling component that combine global appearance representation with part based representation, building a couple-layered representation of the object. We model target appearance as of collections of positive and negative samples representing respectively the complete target appearance and informative subregions (*the parts*) belonging to it. As demonstrated in experimental session, by combination of local feature and global features informations we achieve a more stable and effective modeling component with superior long term tracking capabilities.

- In chapter 7 , we give a conclusion by summarizing open questions. Furthermore, ongoing work is briefly presented which gives some ideas to overcome the limitations shown before.

Chapter 2

Related Work

Long-term visual tracking is a complex task that combines into an unified process object appearance modeling and object localization. Appearance modeling is the process of building and adapting a visual representation of the target. Machine learning and statistical pattern recognition provide theories and algorithms to effectively deal with such task. Localization is the process of estimating the state of the object, indirectly observed through noisy measurements, between consecutive frames relying on temporal coherence in the video. Optimal filtering theory, defines state estimation as a recursive inference problem and provides a rigorous formulation to estimate optimal or approximate solutions when dealing with object localization.
To give an overview of the most relevant problems involved with visual tracker design, the chapter is split into four parts. Section 2.1 formulate visual tracking from an abstract point of view, decomposing it into two main problems: the state estimation and the target appearance representation and modeling. In section 2.2, we briefly introduce the theory of time varying dynamical system and describe the baseline techniques used in this thesis to approach the state estimation. In section 2.3, we review the state of art approaches to object appearance modeling. We first analyze the visual representations from a feature-construction viewpoint, categorizing visual representations into local and global features based. Then, we review

statistical modeling schemes adopted for the model construction and adaption. In section 2.4 we conclude by a discussion on the open problems related to the adaptive appearance modeling.

2.1 Adaptive Visual Tracking: a general overview

Let $\mathbf{I}^T = \{I_t\}_{t=1}^T$ be an image sequence and $\mathbf{x}^t \in \mathcal{X} \subset \mathcal{R}^d$ a variable encoding the target state at time t. $\mathcal{X}$ is referred as the state space. According to the application requirements, $\mathbf{x}^t$ encodes position, dimensions and optionally, other geometric or kinematic informations such as scale, aspect ratio, orientation with respect to image axes and velocity. A long term tracker estimates the target trajectory $\mathbf{X}^T = \{\mathbf{x}^t \in \mathcal{X}\}_{t=1}^T$ given an instance of the overall target appearance model $\mathcal{A}^t$ up to time t.

According to the strategy adopted to build the appearance model,

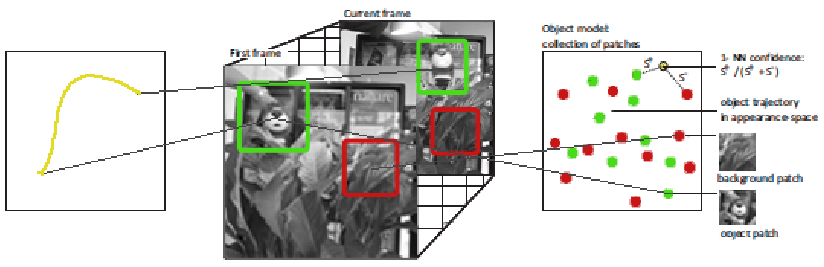


Figure 2.1: **The visual tracking problem decomposition**. On the left the estimated object trajectory $\mathbf{X}^T$. On the right the observation space $\mathcal{A}$

visual trackers can be classified in two broad categories namely *non-adaptive trackers* and *adaptive trackers*. The former learn $\mathcal{A}^t$ off-line and keep it unchanged throughout the entire sequence, while the latter update the model $\mathcal{A}^t$ over time by estimating $\mathbf{a}^{t+1}$, the appearance state at time $t+1$. Such updating process defines the main challenge of adaptive trackers.

Following the unified model proposed in [93], the processing pipeline of an adaptive appearance tracker can be decomposed into three

phases namely *sampling* and *labeling* stage, the *feature extraction* and *refinement* stage and the *model estimate* and *update* stage.
Within the *sampling* and *labeling* stage, several state hypothesis $\mathbf{x}_i^t$ are drown from $\mathcal{X}$ according to some criteria. A measurement step is performed in order to provide a confidence $\mathbf{c}_i^t$, measuring how such hypothesis are agree compatible with $\mathcal{A}^t$.
The process that project the image space $\mathcal{I}$ into the observation space $\mathcal{A}$ is referred to as *feature extraction.* Depending on the appearance modeling scheme, $\mathbf{c}_i^t$ represent the likelihood of a particle in a generative appearance method [22], the margin of a classifier in a discriminative appearance approach [40] or a combination of both in an hybrid method [55].
During the *refinement stage*, high confident hypotheses may be pruned to remove outliers and noisy measurement. In addiction a re-sampling stage can be performed in order to provide new discriminative features to update stage.
Given the refined features, the current appearance model state $\mathbf{a}^{t+1}$ is estimated. A merging step, fuses the previous model $\mathbf{A}^t$ with $\mathbf{a}^{t+1}$, yielding the updated appearance model $\mathcal{A}^{t+1}$ for the next tracking iteration. Such procedure, referred to as the *tracking loop* is repeated for all the frames in the video stream as depicted in fig 2.1.
This abstract description of the tracking problem, highlights four fundamental questions related to the design of a visual tracking system:

1. **How to model the state space?** The choice of a representation encoding the target state, referred in literature as the *target representation problem* [73] depends on the object of interest. In visual tracking, it should be descriptive enough to discriminate cluttered scenes and false targets (object with similar shape), while allowing a certain degree of flexibility to capture changes in pose and scale. In sec. 2.3.1 a short analysis of the most common target state representation is presented.

2. **How to characterize the observation space?** The observation space associated with a certain state can be defined by several informations. Low-level features exploit color or gradient informations; mid-level features exploit informations such as edges or contours; high level features extract interest points over the target surface. We briefly review the most popular feature description adopted in visual tracking in 2.3.2.

3. **How to build the appearance model?** The choice of a method to model visual appearance over time is referred in literature as the *appearance modeling* problem. Pattern recognition and machine learning concentrates on how to build effective mathematical models for object identification using statistical learning techniques.

4. **How to solve the *state estimation* problem?** By exploiting recursively information from the feature extraction step and from the already available state estimates the trajectory should be derived. This task links different instances of the same object over time and has to compensate for occlusions, clutter, and local and global illumination changes.

Designing an effective visual tracker involves a joint solution for all the aforementioned issues. The strategy adopted to approach and combine such problems each other, plays a decisive role in the robustness and efficiency of the tracker. The importance of each subproblem depends on the application requirements: for example, face tracking in a crowded scene relies more on target representation than on target dynamics [11], while in aerial video surveillance, the target motion and the ego-motion of the camera are the more important components [78].
In addiction, strategies to control the trajectory state over time, according to the specific domain need to be defined. For example in a human video surveillance scenario a target can be occluded by other people or moving out of the field of view of the camera leading to a

fragmentation of the trajectory. An effective tracker should be able to relocate the target and reconstruct its global trajectory. Moreover, when a new target appears in the scene (*target birth* [93]), the tracker must initialize a new trajectory. A target birth usually happens at specific entry areas (e.g. doors), at the image boundaries, in the far-field of the camera (when the size of the projection onto the image plane increases and the target becomes visible), or when a target spawns from another target (e.g. a driver parking a car and then stepping out). Similarly, a trajectory must be terminated (*target death* [93]) when the target leaves the field of view of the camera, or disappears at a distance or inside another object. In addition to the above, it is desirable to terminate a trajectory when the tracking performance is expected to degrade under a predefined level, thus generating a track loss condition. Another important issue is related to the computational complexity, that for real-time applications, should be as low as possible in order to perform other higher-level tasks such as recognition, trajectory interpretation and reasoning.
From an architectural design point of view, the sub-problems are generally implemented into two components, namely the *Target representation module* and the *state estimation module*. The former implement a a bottom-up process which has to cope with the changes in the appearance of the target and of its surrounding background, dealing with the *feature extraction* and the *appearance modeling* problems.
The latter implement a top-down process dealing with the dynamics of the tracked object, learning of scene priors, evaluation of different hypotheses and track management process. Following this architectural decomposition, in the next sections, we briefly review the most popular and effective approaches characterizing such building blocks in state of the art visual trackers.

2.2 State Estimation

The demand for updating the state as new uncertainty measurements becomes available, naturally lead to represent target state as the state of a discrete-time dynamical system.
Optimal filtering refers to the theory that deal with estimating the state of a time-varying system, which is indirectly observed through noisy measurements. The state of the system refers to the collection of dynamic variables such as position, velocities and accelerations or orientation and rotational motion parameters, which describe the physical state of the system. As will be pointed out in section 2.3.1, several state representation have been studied, from a bounding box [4, 55] to $3D$ articulated models [114].
The noise in the measurements refers to a noise in the sense that the measurements are uncertain, i.e even if we knew the true system state the measurements would not be deterministic functions of the state, but would have certain distribution of possible values.
The time evolution of the state is modeled as a dynamic system, which is perturbed by a certain process noise. This noise is used for modeling the uncertainties in the system dynamics even if in most cases the system is not truly stochastic, but the stochasticity is only used for representing the model uncertainties.
In this thesis, optimal filtering is reviewed in terms of bayesian inference, and both the classical and recent filtering algorithms are described using the same bayesian notation and formalism.
The probabilistic formulation provides a rigorous and general framework for dynamic state estimation applied to the context of visual tracking. From a bayesian perspective, all algorithms are treated as approximations to certain probability distributions or their parameters describing both the uncertainties in the models and the physical randomness.
Following the classical formulation of recursive Bayesian inference, we briefly introduce the theory of probabilistic filtering for state es-

timation of a dynamic system.

Definition 2.2.1 (State space model [56]**).** *A discrete-time state space model or probabilistic non-linear filtering model is a recursively defined probabilistic model of the form*

$$\begin{aligned} x^t &\approx p(x^t|x^{t-1}) \\ a^t &\approx p(a^t|x^t) \end{aligned} \tag{2.1}$$

where:

- $x^t \in \mathcal{X} \subset \mathbb{R}^m$ *is the state of the system at time step* t.
- $a^t \in \mathcal{A} \subset \mathbb{R}^n$ *is the measurement at time step* t.
- $p(x^t|x^{t-1})$ *is the dynamic model, which models the stochastic dynamics of the system. The dynamic model can be a probability density, a counting measure or combination of them. Depending on the state* x^t *it is continuous, discrete or hybrid.*
- $p(a^t|x^t)$ *is the measurement model, defining the distribution of the observations given the state.*

The model is assumed to be *markovian.* Such property implies constraints on the time dependency for the state and the observation variable simplifying the inference problem. Such dependencies are formulated in the following properties:

Property 2.2.1 (Markovian property of states [56]**).** *The states* $\{x^t : t = 1, 2, \ldots, T\}$ *form a markov sequence (or markov chain when the state is discrete). This markov property means that* x^t *and actually the whole future* $x^{t+1}, x^{t+2}, \ldots$ *given* x^{t-1} *is independent from anything that has happened in the past:*

$$p(x^t|x^{1:t-1}, a^{1:t-1}) = p(x^t|x^{t-1}) \tag{2.2}$$

and again, the past is independent of the future given the present:

$$p(x^{t-1}|x^{t:T}, a^{t:T}) = p(x^{t-1}|x^t) \tag{2.3}$$

Property 2.2.2 (Conditional independence of observations [56]). *The observation a^t given the state x^t is conditionally independent from the observation and state histories:*

$$p(a^t|x^{1:t}, a^{1:t-1}) = p(a^t|x^t) \tag{2.4}$$

The filtering model can be equivalently expressed as an *hidden markov chain* or a directed graphical model whose representation is depicted in figure 2.2

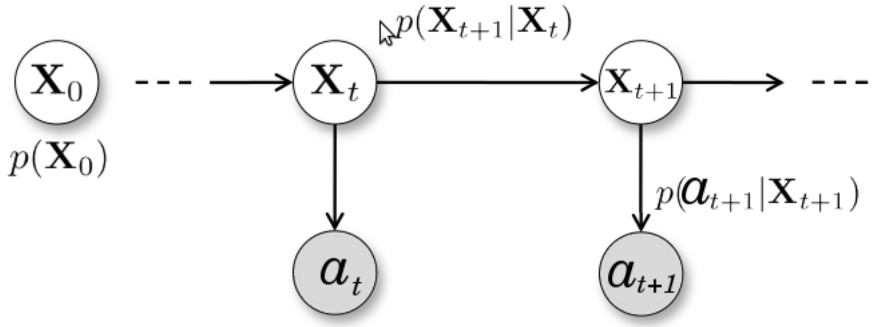


Figure 2.2: The directed graph of the Filtering Model. The link x^t and x^{t-1} express the markov property while the link between a^t and x^t express the conditional independence property of measurements

The purpose of optimal filtering is to compute the marginal posterior distribution of the state x^t on the time step t given the history of the measurements up to the time step t, referred in literature as the *filtering distribution*:

$$p(x^t|a^{1:t}) \tag{2.5}$$

The fundamental equations leading to filtering distribution estimate are provided by the following theorem:

Theorem 2.2.1 (Bayesian optimal filtering equations [56]**).** *The recursive equations for computing the predictive distribution $p(x^t|a^{1:t-1})$ and the filtering distribution $p(x^t|a^{1:t})$ on the time step t are given by the following Bayesian filtering steps:*

1. ***Initialization**. The recursion starts from the prior distribution $p(x^0)$.*
2. ***Prediction**. The predictive distribution of the state x^t on time step t given the dynamic model can be computed by the **Chapman-Kolmogorov** equation*

$$p(x^t|a^{1:t-1}) = \int p(x^t|x^{t-1})p(x^{t-1}|a^{1:t-1})dx^{t-1} \tag{2.6}$$

3. ***Update**. Given the measurement a^t on time step t the posterior distribution of the state x^t can be computed by the bayes rule*

$$p(x^t|a^{1:t}) = \frac{1}{Z^t}p(a^t|x^t)p(x^t|a^{1:t-1}) \tag{2.7}$$

where the normalization constant Z^t is given as

$$Z^t = \int p(a^t|x^t)p(x^t|a^{1:t-1})dx^t \tag{2.8}$$

If some of the components of the state are discrete, the corresponding integrals are replaced with summations.

The solution of **Chapman-Kolmogorov** equation is analytically solvable only in few cases. A notable one is when the dynamic and measurements models are linear gaussian. In this situation, the optimal solution is provided by the *Kalman filter* [56], that exploits

the well behaved properties of gaussian function under integration to derive the closed form solution to the filtering problem.
The estimated filtering distribution, provided by the *Kalman filter* is still a gaussian, whose parameters are updated according to linear operators [56]. The filtering and association techniques discussed above have been applied in computer vision for various tracking scenarios [16] When the functions and are nonlinear, by linearization the extended *kalman Filter* (*EKF*) [112] is obtained, the posterior density being still modeled as gaussian. A recent alternative to the *EKF* is the *unscented kalman filter* (*UKF*)[109] which uses a set of discretely sampled points to parameterize the mean and covariance of the posterior density. When the state space is discrete and consists of a finite number of states, Hidden markov Models (HMM) filters [86] can be applied for tracking. In [89] *EKF* is used to estimate a $3D$ object trajectory from $2D$ image motion.

2.2.1 Particle Filters

When the assumptions made by the *kalman filter* do not hold, approximate solutions to the optimal filtering problem are obtained with particle filters [88]. Particle Filtering, refers to a class of general computational approaches that replaces analytic integration defined in 2.6, 2.7,2.8 by summation over samples generated by iterative algorithms.
Problems that are intractable using analytic approaches often become treatable using some form of *particle filters* such as *importance sampling* (*IS*), *importance resampling* (*IR*) and *markov chain monte carlo* (*MCMC*), even with high-dimensional problems [66].
Given the filtering distribution $p(x^t|a^{1:t})$, an approximate estimation of the state at time t is given in the form of:

$$x^t = \int f(x^t)p(x^t|a^{1:t})dx^t \tag{2.9}$$

With *monte carlo* methods, a numerical evaluation of the integral is computed by generating N samples x_i^t from the posterior and then evaluating the sample mean:

$$\hat{x}^t = \frac{1}{N}\sum_{i=1}^{N} f(x_i^t) \tag{2.10}$$

Unfortunately, it is not possible to sample from the posterior in non gaussian-non linear case, since it has a non standard form and it is usually known only up to a proportionality constant. In practice, a density $\pi(x^t)$ approximating the posterior is available and equation 2.9 is approximated by drawing sample from $\pi(x^t)$ and weighting them accordingly,

$$\hat{x}^t = \frac{1}{N}\sum_{i=1}^{N} f(x_i^t)w(x_i^t) \quad \text{with} \quad w(x_i^t) = \frac{p(x_i^t|a^{1:t})}{\pi(x_i^t)} \tag{2.11}$$

This technique is referred as importance sampling and the pdf π is referred to as the *importance* or *proposal density.* Particle filter are based on sequential importance sampling.
The key idea is to represent the posterior by a set of random samples with associated weights, namely the *particles.* The posterior pdf is approximated as

$$p(x_i^t|a^{1:t}) = \sum_{i=1}^{N} w(x_i^t)f(x^t - x_i^t) \tag{2.12}$$

where samples are obtained at each time step from the proposal density

$$w(x_i^t) \approx \frac{p(x^t|a^{1:t})}{\pi(x^t|x^{1:t-1}, a^{1:t})} \approx w(x_i^{t-1})\frac{p(x^t|a_i^t)p(x_i^t|x_i^{t-1})}{\pi(x_i^t|x_i^{1:t-1}, a^t)} \tag{2.13}$$

and then normalized to sum to one.
The convergence of *Monte Carlo approximation* is guaranteed by *central limit theorem* (CLT) [66] and the error term is $O(N^{\frac{1}{2}})$, regardless of dimensionality of x. This invariance of dimensionality is unique to Monte Carlo methods and makes them superior to practically all other numerical methods when dimensionality of x is considerable.

In particular, the precision of the approximation depends on the independence of the samples: when the samples are correlated, the effective sample size decreases.
A problem of *SIS* algorithm is their tendency to generate a lot of particles having zero weights while only a few of them (or only one) are non-zero. This is referred to the *degeneracy problem*[66] in particle filtering literature and it limited practical applications of particle filters for long time. In order to overcame, this phenomena, re-sampling algorithms have been introduced, leading to so called sequential importance re-sampling algorithms *SIR*[66]. With *SIR* samples with low weights are eliminated, while samples with high importance weights are increased. This corresponds to computing a less accurate approximation of the posterior that concentrates on salient regions of the state space and avoids to waste computational power by propagating particles that carry on negligible contributions to the posterior approximation.
The new set of particles is generated by re-sampling with replacement L times from the cumulative sum of normalized weights of the particles. Particle filtering was first introduced in vision as the *Condensation algorithm* [52]. Based on particle filtering, in [31] is proposed an novel approach to integrate multiple cues based on concept of reliability, which is defined relative to the success of the other cues in tracking the target object. During tracking, different cues try to reach an agreement on a joint result and they adapt themselves considering the result currently agreed on. In [22, 85] particle filters are used tracks in a joint fashion a set of parts building the pictorial structure representation of the object of interest.

2.2.2 Markov Chain Monte Carlo

The development of **MCMC** is arguably the biggest advance in the computational approach to statistics. A markov chain is a stochastic process where a transition from one state to another is defined by a simple sequential procedure. The chain is started at some initial state x^0, and use a *transition function* $p(x^t|x^{t-1})$, to determine the next state, x^1 conditioned on the last observed state. Iterating such procedure a sequence of states forms the markov chain:

$$x^1 \rightarrow x^2 \rightarrow \ldots \rightarrow x^t$$

The procedure for generating a sequence of T states from a markov chain is reported in sketch:

MARKOV CHAIN GENERATION

```
0.   Set $t = 1$
1.   Generate a initial value $u$, and set $x^t = u$
3.   Repeat
3.1    $t = t + 1$
3.2    Sample a new value $u$ from the transition function $p(x^t|x^{t-1})$
3.3    Set $x^t = u$
4. Until $t = T$
```

During the chain evolution, the next state of the chain at $t + 1$ depends only on the previous state at t, giving to the whole procedure a "*memoryless*" property. This local dependency behavior is an important property when using markov chains for monte carlo integration. When initializing each markov chain, the chain will wander in state space around the starting state. Therefore, if we start a number of chains, each with different initial conditions, the chains will initially be in a state close to the starting state. This period is referred to as the *burnin*.
An important property of markov chains is that the starting state of the chain no longer affects the state of the chain after a sufficiently long sequence of transitions (assuming that certain conditions about the markov chain are met). At this point, the chain is said to reach its *steady state* and the states reflect samples from its stationary distribution.

This convergency property is quite important, because when applied to monte carlo integration, it allow us to draw samples from a distribution using a sequential procedure but where the starting state of the sequence does not affect the estimation process.
The goal of **MCMC** is to build a markov chain such that the stationary distribution of the chain is exactly the *target distribution*, i.e the distribution we are interested in sampling from.
In other words, the states sampled from some markov chain will generate samples drawn from the filtering distribution $p(x_i^t|a^{1:t})$.
The Metropolis-Hastings (*M-H*) sampler [66] represent a popular and effective method exploited to generate samples from a multivariate distribution. The goal is to sample from the target density $p(\theta)$, with $\theta \in \mathcal{R}^N$.
The Metropolis sampler creates a markov chain that produces a sequence of states,

$$\theta^1 \rightarrow \theta^2 \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \theta^t$$

where θ^t represents the state of a markov chain at iteration t. The samples from the chain, after *burnin*, reflect samples from the target distribution $p(\theta)$.
In this procedure, the first state, θ^1 is initialized to some initial value. Then, a proposal distribution $\pi(\theta^t|\theta^{t-1})$ generates a candidate point θ^* corresponding to a possible value for state at time t conditioned on the previous state of the sampler.
The next step is to either accept the proposal or reject it. The probability α of accepting the proposal is defined as:

$$\alpha = min(1, \frac{p(\theta^*)}{p(\theta^{t-1})}\frac{\pi(\theta^{t-1}|\theta^*)}{\pi(\theta^*|\theta^{t-1})} \tag{2.14}$$

To make a decision on whether accept or reject the proposal, a uniform deviate u is generated. If $u > \alpha$, the proposal is accepted and the next state is set equal to the proposal ($\theta^t = \theta^*$).
If $u \leq \alpha$, the proposal is rejected and the next state is set equal to

the old state ($\theta^t = \theta^{t-1}$). The procedure continues generating new proposals conditional on the current state of the sampler, until the sampler reaches convergence. At this point, the samples θ^t reflect samples from the target distribution $p(\theta)$.
To intuitively understand why the process leads to samples from the target distribution, note that the method will always accept a new proposal if it is more likely under the target distribution than the old state.
The choice of the proposal distribution depends on the problem at hand. A common approach consist in defining symmetric proposal distribution ($\pi(\theta = \theta^t|\theta^{t-1}) = \pi(\theta = \theta^{t-1}|\theta^t)$) such that the proposal ratio in 2.14 drops out. One important constraint for the proposal distribution is that it should cover the state space such that each potential outcome in state space has some non-zero probability under the proposal distribution old state.
Therefore, the sampler will move towards the regions of the state space where the target function has high density. However, note that if the new proposal is less likely than than the current state, it is still possible to accept this worse proposal and move toward it.
This process of always accepting a good proposal, and occasionally accepting a bad proposal ensures that the sampler explores the whole state space, by sampling from all parts of a distribution (including the tails).
According to the strategy adopted to explore multidimensional space underlying the target distribution, the M-H samplers are classified in *block-wise (BW)* or *component-wise (CW)* updating samplers.
The BW approach uses a proposal distribution having same dimensionality as the target distribution. So, when sampling from a probability distribution involving N variables, an N-dimensional proposal distribution is needed. Acceptation or rejection treats θ^* as a block, leading to high rejection rates. Another potential problem with this updating approach is related to the difficulty in defining a suitable high-dimensional proposal distributions.

In contrast, the CW approach instead of accepting or rejecting a proposal for θ involving all its components simultaneously, generates proposal for individual components of θ, once per time. This leads to a computationally simpler updating scheme where at each iteration t, an independent proposal θ_i^* for each component status θ_i^t given its previous state θ_i^{t-1} is generated. The acceptance ratio is evaluated by comparing the likelihood of $(\theta_i^*, \theta_{j\neq i}^{t-1})$ against $(\theta_i^{t-1}, \theta_{j\neq i}^{t-1})$.
The CW Metropolis Hastings sampler represent a key ingredient in the approaches proposed in this thesis and it is reported in the following algorithm.

METROPOLIS-HASTING CW

```
0.   Set $t = 1$
1.   Generate a initial value $\theta^{\mathbf{1}} = (\theta_1^1, \theta_2^1, ..., \theta_N^1)$
2.   While $t < T$
3.       For $i = 1 : N$
3.1          propose a candidate component $\theta_i^* \approx \pi(\theta_i^t | \theta_i^{t-1})$
3.2.         calculate acceptance probability:
                 $$\alpha = min(1, \frac{p(\theta_i^*, \theta_{j\neq i}^{t-1})}{p(\theta^{\mathbf{t-1}})} \frac{\pi(\theta_i^{t-1}|\theta_i^*)}{\pi(\theta_i^*|\theta_i^{t-1})})$$
3.3          generate $u$ from Uniform distribution in $[0\ ,\ 1]$
3.4          if($\alpha \leq u$)
3.5              **accept sample component**: $\theta_i^t = \theta_i^*$.
3.6          else
3.7              **reject sample component**: $\theta_i^t = \theta_i^{t-1}$.
3.8      EndFor
4.   $t++$
```

Note that in this proposal procedure, at each time only one component is varying while keeping the others constant and updated to the last accepted proposal. Therefore, the proposal of a new sample

θ_j^t is conditioned on the proposal generation for all components $i < j$.
Metropolis Hastings method has been exploited into several approaches to track single objects [58] or multiple objects [24].
In particular, in multiple target scenario [8], new problems related to the validation and association of the measurements arise [8, p.150]. Consequently, gating techniques are used to validate only measurements whose predicted probability of appearance is high.
After validation, a data association strategy is required to establish correspondences between measurements and current targets. The *Nearest Neighbor Filter* selects the closest measurement. Assuming that for any given target only one measurement is valid while the other represent random interference, i.e i.i.d. uniformly distributed random variables, the Probabilistic Data Association Filter ($PDAF$) [8] selects for each target the most probable measurement.
On the other hand, the Joint Data Association Filter (JPDAF) [8] calculates the measurement-to-target association probabilities jointly across all the targets.
A different strategy is represented by the Multiple Hypothesis Filter (*MHF*) [8] which evaluates the probability that a given target gave rise to a certain measurement sequence. The *MHF* formulation can be adapted to track the modes of the state density [21]. The data association problem for multiple target particle filtering is presented in [87].

2.3 Target Representation

As described in section 2.1, the fundamental problems dealing with the target representation process are related to the choice of:

- the State Space $\mathcal{X} \subset \mathcal{R}^n$ defining the object of interest.
- the Measurement space $\mathcal{A} \subset \mathcal{R}^m$ modeling target visual appearance.
- the approach adopted to build and adapt the appearance model while tracking.

2.3.1 State Space Modeling

As introduced in 2.1, object state defines the variables estimated during tracking, encoding the target representation.Based on the informations adopted to define the State Space $\mathcal{X}$ visual trackers can be classified into five main categories.

Points based Trackers. Small object that do not change their scale are represented by a point. The tracker estimate only translation of the object's translation using frame-to-frame tracking [96], key-point classification [61], or linear prediction [125]. Recent work is directed towards combining point based trackers with global appearance trackers leading to a multilayered representation of target appearance [19].

Geometric shapes based Trackers. Rectangular and elliptical shapes , are the most popular state representation for objects undergoing significant changes in scale, due to their low complexity[90, 55]. With such representation is possible to estimate object location, scale and in-plane rotation, while other variations are typically modeled as the changes of the object appearance.

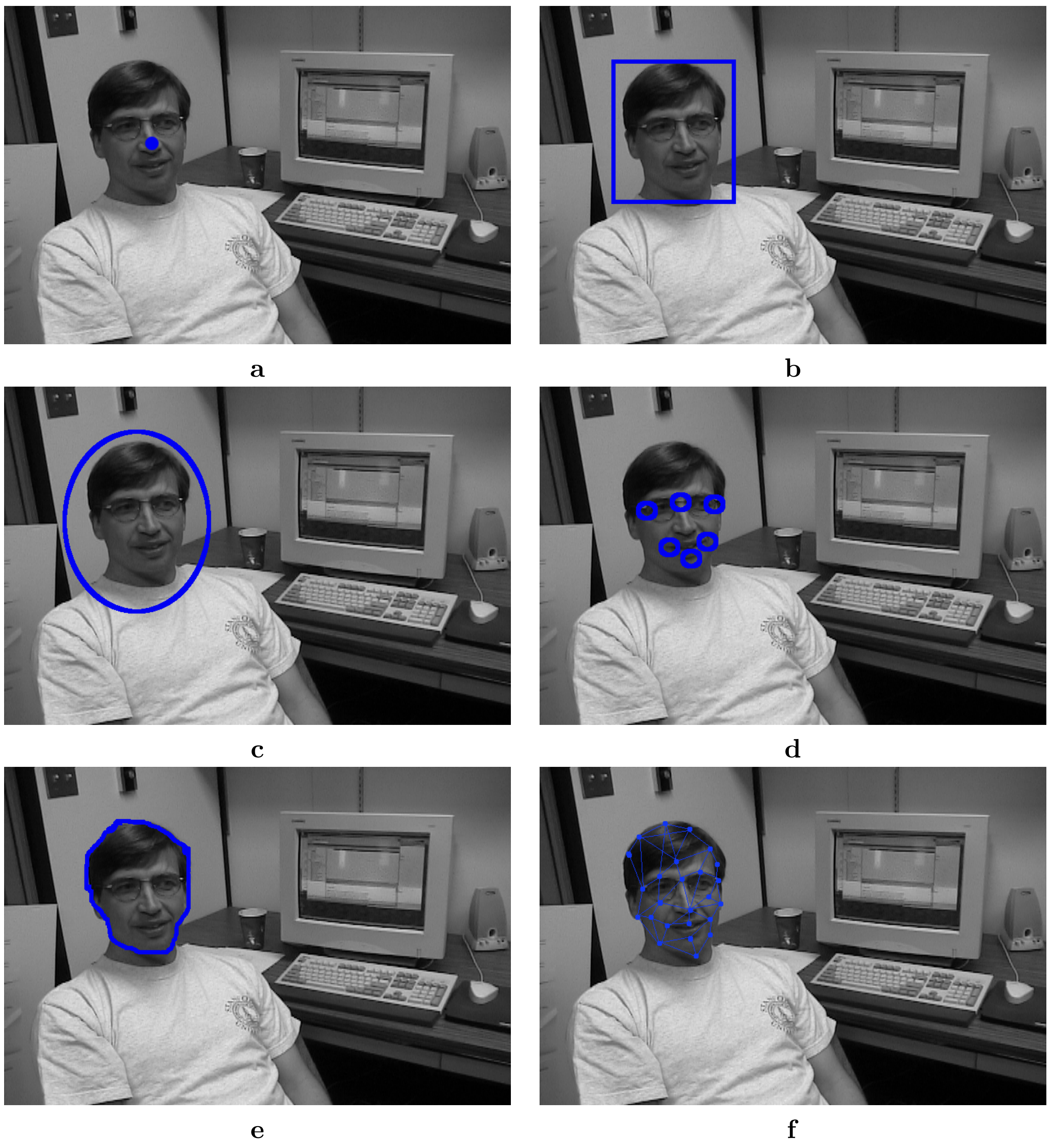


Figure 2.3: **Object States representations**: *point*(a), *rectangle*(b), *ellipse*(c),*part based representation*(d), *contour*(e),*3D articulated models*(f)

Articulated models based Trackers. Non rigid objects are defined as collection of several rigid parts, typically modeled by geometric shapes such as rectangles or ellipses. By identifying the local components and considering their inter-relationships, better detection or identification results can be achieved [22, 85, 2].

Contours based Trackers. Another suitable representation for non-rigid objects are Contours. Parametric representation of contours has been used for tracking of human heads [11] or arbitrarily complex shapes [52].

3D models based Trackers. Rigid objects, for which the geometry is known are modeled by 3D models. These models estimate location, scale and pose of the object. These methods have been applied to various objects including human faces [114].
In our approach, we represent the object state by a bounding box. This representation balances the tradeoff between the expressive power of the representation and the difficulty to reliably estimate the object motion. The major problem of this representation is related to its high sensitivity to inaccurate target localization, leading to the injection of background feature into the target appearance model.

2.3.2 Measurement Space Modeling

Selecting good features to describe object appearance plays a critical role in visual tracking. Generally, the most desirable property of a visual feature is its inherent discriminative power so that the objects can be easily distinguished in the feature space. Depending on the target properties, i.e. deformable or rigid, planar or 3d object, a variety of features has been proposed to describe the target appearance. Motion vectors, motion detection, object classification, low-level features such as pixel colors or gradients, or mid-level features such as edges and interest points are typical example of information that could be exploited to represent visual appearance (for a detailed description we remand to [73]). Based on their spatial extent visual features can be grouped into two main categories:

- **Part-wise features**. Features are extracted from small patches or even single pixels. It is relatively easy to deal with partial occlusions but these features are hard to match if the target undergoes deformation or rigid transformations such as rotations and scalings.
- **Target-wise features**. The feature represents the whole target appearance. This kind of features can typically tolerate target deformations and rigid transformations. Correct handling of occlusions represents a critical limitation of these representations.

Based on the type of visual information emplyed to describe object appearance visual features can be distinguished in four categories: color based, shape based, optical flow based and texture based.

Color. When dealing with color based representations the choice of the color space play a central role to achieve invariance to the spectral power distribution of the illuminant and the surface reflectance properties of the object. This physical phenomena affects the image formation process, leading to low effective visual representation. The *RGB* (Red, Green, Blue) color space, for example is not perceptually uniform, that is, the differences between the colors in the *RGB* space do not correspond to the color differences perceived by humans. Additionally, the *RGB* dimensions are highly correlated. In contrast, *Luv** and *Lab* are perceptually uniform color spaces, while *HSV* (Hue, Saturation, Value) is an approximately uniform color space even if hue becomes unstable near the gray axis. In general, several color spaces exist [73], each one having different properties that can affect tracking performances in relation to the scenario where the task have to be performed.

Shape. Object boundary or contour is a distinctive feature to model object shape and it is less sensitive to illumination changes compared to color features. An edge is defined by a strong discontinuity in image intensity. As consequence edge detectors are defined by operators on image gradient. Because of its simplicity and accuracy, the most popular edge detection approach is the Canny Edge detector [18]. Recently, gradient features have become very popular in pedestrian detection. In [36] is presented a contour based hierarchical chamfer matching detector for pedestrian detection. In [65] extended the global contour based models by parts decomposition and a hierarchical tree for the part templates is proposed. Ferrari et al. [33] used the network of contour segments to represent the shape of an object in order to detect object in cluttered images. Recently, statistical summarization have been very popular to exploit gradient information[69, 9, 26]. The key idea is to extract local informations characterizing regions of the image in terms of frequency of the gradient orientation. The Scale-invariant Feature Transform (*SIFT*) [69] is probably the most successful example of such approach. A patch surrounding a keypoint is split into 4×4 cells, within each cell an eight-bin histogram of gradient orientation is measured which produces a feature vector of 128 elements. In [9], *SURF* a much faster scale and rotation invariant interest point descriptor has been proposed.
In [26] the Histogram of Oriented Gradient (*HOG*) descriptor in conjunction with Support Vector Machine (*SVM*) classifier [12] demonstrated outperforming results for pedestrian detection. Later, in [124], *HOG* computational efficiency was significantly improved employing a boosted cascade of rejectors. In [74] the multi-resolution *HOG* descriptor and faster kernel SVM classifier achieved promising result for pedestrian detection. In [32] a part based deformable model based on the multi-resolution *HOG* descriptor in combination with SVM was used for pedestrian detection.

Optical Flow. Optical flow is a dense field of displacement vectors which defines the translation of each pixel in a region. It is computed using the brightness constraint, which assumes brightness constancy of corresponding pixels in consecutive frames [49]. Optical flow is commonly used as a feature in motion-based segmentation and tracking applications. Originally [71], optical flow computation was performed by assuming the flow constant in a local neighborhood of the pixel under consideration. The least squares solution of the optical flow equations defined by all the pixels in such neighborhood represent the motion of the pixel. Later on, in [13] was proposed a method dealing with the situations with varying lighting conditions to estimate optical flow.

Texture. Texture is a measure of the intensity variation in an image region, that quantifies properties such as regularity and smoothness. To extract such information, texture descriptors need complex image regions analysis that slow down the performance of the tracker. Similar to edge features, the texture features are less sensitive to illumination changes compared to color.
Several texture descriptors have been proposed in literature. Gabor wavelet [75] is probably the most studied texture feature. The Gabor filters can be considered as orientation and scale tunable edge and line detectors, and the statistics of these micro-features in a given region are often used to characterize the underlying texture information. In recent years, increasing attention was dedicated to imageś local patterns for better detection and recognition. Especially, local patterns that are binarized with an adaptive threshold provide state-of-the-art results in various application fields, such as face detection and image classification. In [102] a very efficient texture descriptor, called Local Binary Patterns (*LBP*) is proposed. The *LBP* texture operator is defined as a gray-scale invariant texture measure, derived from a general definition of texture in a local neighborhood. Its most important property are tolerance to illumination changes

and low computational cost.
Many variants of *LBP* have been recently proposed, including Local Ternary Patterns (*LTP*) [103] , multi-scale block *LBP* (*MB-LBP*) [63] and *Semantic-LBP* and *Fourier LBP* are proposed in [81]. In [123] a local binary pattern extracted from the *Gabor* filter is proposed for the task face representation and recognition. In contrast to *LBP* approach that analyzes regular neighbor structures, in [126] is proposed the *FERN* descriptor, a local pattern obtained by comparing random generated positions over the image. With such comparison invariance against constant brightness variations is achieved.

In conclusion, feature description is a crucial aspect for visual tracking as well for object detection. However, no single feature descriptor is robust and efficient enough to deal with all kinds of situations. For instance, the *HOG* descriptor focuses on edges and structures but ignores flat areas, thus fails to deal with noisy edge regions. A possible drawback of the *LBP* operator is the thresholding operation when comparing the neighboring pixels that could make it sensitive to noise. Color features represent the global information of images, which are relatively independent of the viewing angle, translation, and rotation of the objects and regions of interest. However, objects with the same color histogram may be completely different in texture, thus color histogram cannot provide enough information. How to combine various kinds of features into a coherent framework needs much more study.

2.3.3 Statistical Appearance Modeling

According to the learning scheme adopted to build and adapt the appearance model during tracking, it is possible to distinguish three main categories of appearance models: *generative, discriminative, and hybrid generative-discriminative.*
The generative appearance models aim to accurately fit the data from the object class, even if it is very difficult to verify the correctness

of the specified model. By introducing on-line update mechanisms, these approaches incrementally learn visual representations of the foreground object while ignoring the influence of the background. As a result, they often suffer from distractions caused by the background regions with appearance similar to the object class.
In contrast, discriminative appearance models formulates visual object tracking as a binary classification problem, trying to maximize the separability between the object and non-object regions.Thus, they can achieve effective and efficient predictive performances.
Nevertheless, a major limitation of the discriminative appearance models is to rely heavily on training sample selection (e.g., by self-learning or co-learning). The generative and discriminative appearance models have their own advantages and disadvantages, and to a certain extent can be considered complementary to each other . In order to exploit the benefits of both approaches and mitigate their benefits, researchers propose hybrid generative-discriminative appearance models to fuse the useful information from the generative and the discriminative models.

2.3.3.1 Generative Appearance Models

Generative methods, used to learn the appearance of an object, have been exploited to handle the variability of a target whose model is often updated on-line to adapt to appearance changes. Object localization becomes a search for the best match between an image patch and the model. Even though these methods were not designed for long-term tracking, we briefly review this category of trackers since they poses several interesting issues addressed by our approach.
Template trackers represent the first attempts to object tracking. They represent object appearance by a single template (an image patch) and exploit a similarity measure among patches in order to find the best affine transformation (translation and rotation) maximizing such measure. In order to avoid exhaustive search for the best similarity match, several technique have been proposed.For example,

assuming linear motion among consecutive frames, some strategies constrain the search in a neighborhood of the previous location, increasing efficiency of the template tracker and reducing the number of false matches generated by similar objects [71, 7, 25]. Moreover, when the target violates the motion assumption, the tracker loses the object. The most effective approaches based on local searching from the previous location can classified in two categories: *gradient-based* and *mean-shift based* methods. Gradient-based methods optimize the similarity measure using gradient descent search. Doubtless, Lucas-Kanade [71] tracker and its pyramidal implementation [120], which estimates translation of an image patch, traced the root for gradient based methods. Affine warping was later proposed in the Kanade-Lucas-Tomasi tracker [96]. Both of these approaches have been unified in the Inverse Compositional Algorithm [7]. These methods employs *ssd* as the similarity measure. Recently, the mutual information similarity has been proposed [28] demonstrating better convergence properties.

The mean-shift algorithm [25] is another approach proposed to avoid extensive search for the best matching template. The object is modeled by its color distribution and object localization is formulated as an iterative procedure for matching target distribution over the image. Using the Bhattacharyya coefficient as similarity metric, the mean shift procedure performs object localization by finding the basin of attraction of the local maxima. However, the tracker only considers color information and therefore ignores other useful information such as edge and shape, resulting in sensitivity to background clutters and occlusions. Template tracking faces a trade-off between static and adaptive tracking even if it presents several drawbacks. First of all, a single static template is not sufficient to capture the variability object's appearance . Furthermore, adaptation of the template using the template from the previous frame suffers from drift. Another limitation is its sensitivity to partial occlusions. To tackle the trade-off between static and adaptive tracking, the basic solution

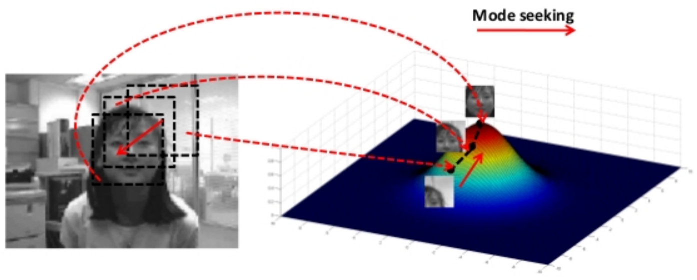


Figure 2.4: **Mean shift mode seeking**

is to adapt the template only when necessary and exploiting the previously observed templates otherwise. For this purpose evaluating the usefulness of the previously observed templates becomes a new problem to be solved [77].
Recent approaches control appearance variability as well occlusions, by modeling object appearance on multiple layers. In [53] a mixture model, named *WSL* mixture, is estimated on-line by expectation maximization algorithm to explicitly model different type appearance variability during tracking. The basic *WSL* mixture model is built on three layers, namely Wandering, Stable and Lost that model the inter-frame variations, the stable structure for all past observations, and outliers such as occluded pixels respectively. A critical point for generative approaches based on mixture models is related to the computation required by expectation maximization and the selection of the fixed number of mixing components building the distribution that effects tracker flexibility. To overcome such limitation in [46] an on-line procedure is proposed to automatically determining in linear time the number of components of the mixture model and their associated parameters including mean, covariance, and weight. To model spatial-temporal variations of the tracked objects, in [110] spatial information are integrated into the color Mixture of Gaussians leading to *SMOG*, a *Spatial-Mixture of Gaussian*. Recently, in [19] a novel coupled-layer visual model that combines the targets

global and local appearance by interlacing two layers is presented. The local layer is a set of local patches that geometrically constrain the changes in the targets appearance. This layer probabilistically adapts to the targets geometric deformation, while its structure is updated by removing and adding the local patches. The addition of these patches is constrained by the global layer that probabilistically models the targets global visual properties, such as color, shape, and apparent local motion. The global visual properties are updated during tracking using the stable patches from the local layer. By exploiting this coupled constraint paradigm between the adaptation of the global and the local layer, the method achieve a robust tracking through significant appearance changes but presents weakness on occlusions.

An interesting category of generative methods formulate object appearance modeling as a subspace learning problem. By focusing on efficient representations of the object defined over low-dimensional subspaces underlying the feature space. A first attempt to subspace learning was proposed in *EIGENTRACKING* [14] where an off-line appearance model was build in the eigenspace domain spanned by principal component analysis. To increase efficiency, incremental principal component analysis (*PCA*) algorithms have been investigated. In [62, 97] incremental robust *PCA* algorithm are investigated embedding robust analysis into the process of subspace learning.

Incremental Visual Tracking (*IVT*)[90] builds an on-line PCA-based appearance model during tracking, proving robustness to illumination and appearance variations.

Deformable objects often violates the linear manifold assumption, leading to tracker failures. Therefore, researchers attempt to employ non-linear subspace learning to capture the underlying geometric information from target samples. For the robust human tracking, a nonlinear subspace models [64, 23, 111] have been proposed using nonlinear dimension reduction techniques.

In summary research in generative trackers demonstrated that drift

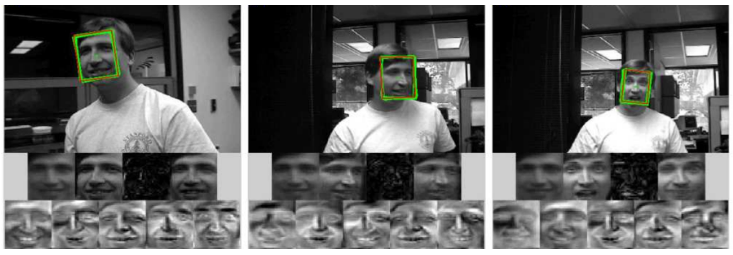


Figure 2.5: **IVT** tracker in action.

can be reduced by reusing already seen examples of the object, and that the robustness to partial occlusions can be achieved by decomposing the template into independent parts. These concepts have been exploited in our approach.
However, the most critical limitation of generative trackers is related to the lack of information coming from the surrounding environment. As a consequence, generative trackers easily drift in cluttered scenes, where different objects in the scene may look similar to the object. In order to increase the tracker robustness, discriminative methods that consider the background class in the modeling process have been investigated in the last decades.

2.3.3.2 Discriminative Appearance Models

Adaptive Discriminative Trackers (ADT) learn an appearance model focusing on distinctive features that discriminate object and its surrounding environment . This allows them to track a wide range of objects immediately after initialization, but introduces the problem of updating the classifier over time in order to handle appearance changes. According to the learning strategies employed, they can be categorized into *self-learning* based and *co-learning* based methods. Typically, the *self-learning* strategy exploits the discriminative information extracted from a single source to guide the task of object/non-object classification, while the co-learning based strategy is based on multi-source discriminative informations for object detection.

More specifically, the self-learning *ADT* first train a classifier over the data from the previous frames, and subsequently use the trained classifier to evaluate possible object regions at the current frame. After object localization, a set of so-called "positive" and "negative" samples are selected to update the classifier. These samples are labeled according to the confidence provided by the previously trained classifier. Due to tracking failures, the training samples analyzed in the tracking process may be affected by noise and hence the labels for the training samples are unreliable. As the tracking process proceeds, the tracking error may be accumulated, leading the tracker to drift.
In contrast, the *co-learning ADT* exploits semi-supervised strategies for object/non-object classification such as co-training of multiple classifiers specialized on independent set of features.
According to the learning framework they exploit, we grouped the most significant *ADT* in to three categories:

- SVM based appearance models.
- Boosting based appearance models.
- Randomized learning based appearance models.

Svm based appearance models (*SAM*). *SAM* exploits the power of max-margin learning, to build an effective appearance model with good generalization capability of distinguishing foreground and background. Informative instances of the appearance model are stored in form of support vectors for object/non-object classification, resulting in a strong discriminative power. Effective kernel selection and efficient kernel computation play an importance role when designing robust *SAM* . The first approach to *SAM*, was proposed in [4] where an off-line svm classifier able to distinguish a vehicle from the background was exploited for tracking. Unfortunately, the demand for prior training data in advance, has lead several difficulties in extending the algorithm to general object tracking task.

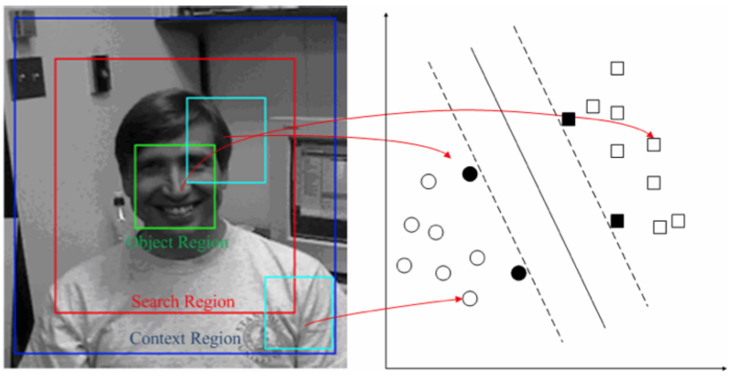


Figure 2.6: **Support Vector Tracker**. Learning an hyperplane separating the object from the surrounding background

An ensemble of linear SVM classifiers to construct a *SAM* is proposed [106]. Such classifiers are adaptively weighted according to their discriminative abilities during different periods, enhancing the robustness to large appearance variations. Recently, *STRUCK* [47] a novel approach based on structured output support vector machine, estimates directly the best transformation representing the target movement during tracking, avoiding the sample selection stage required to update the classifier. It achieves a good trade-off between plasticity and stability based on the ranking of support vectors and the use of the on-line structured output SVM framework to guide the selection of challenging samples from the background.

Boosting-based Appearance Models (*BAM*). Based on-line boosting framework[84], researchers have developed a variety of computer vision applications such as object detection [108] and visual object tracking [40]. One of the earliest work based on on-line boosting was proposed in [118] where a method to adaptively select color features best discriminating the object from the surrounding background. In [5], an adaptive ensemble of weak classifiers is learned

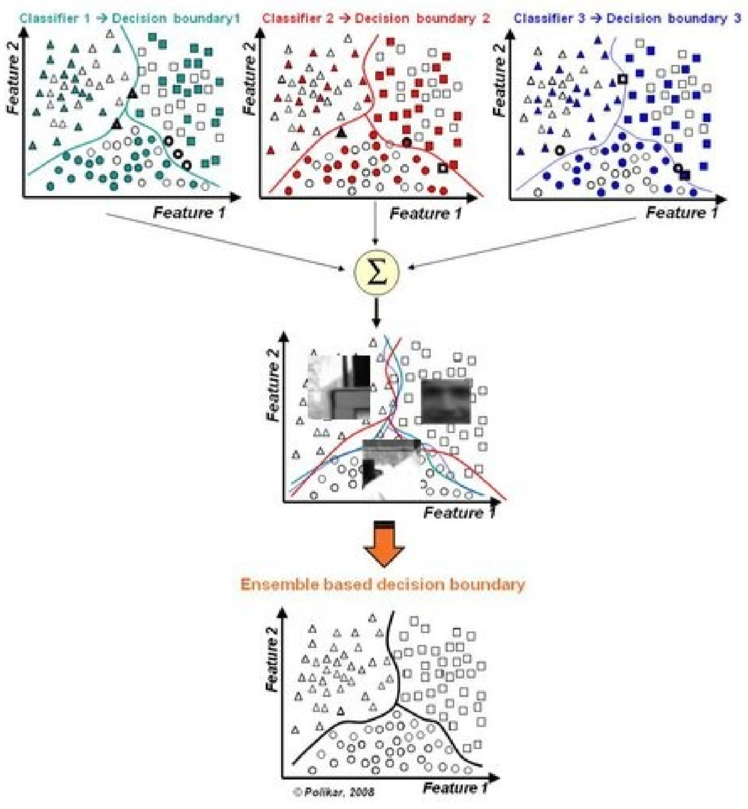

Classifier 1 → Decision boundary1
Classifier 2 → Decision boundary 2
Classifier 3 → Decision boundary 3
Feature 2
Feature 1
Σ
Ensemble based decision boundary
©Polikar, 2008

Figure 2.7: The boosting framework: by a weighted combination of weak classifiers a more stable classifier is obtained

to discriminate between pixels of the object and pixels of the background: each weak classifier is a linear hyperplane learned over an $11D$ feature space composed of R,G,B color and a histogram of gradient orientations. With the *BOOST TRACKER* [40], discriminative evaluation of each feature from a candidate feature pool and selection of the top-ranked features is exploited to improve the tracking process. To accelerate the feature selection process a gradient-based feature selection scheme is derived in [68] in order to build the appearance model. The proposed approach requires an initial set of weak classifiers to be given in advance, leading to difficulty in general object tracking.
To reduce the sensitivity to tracking failure that provide erroneous

samples during the on-line adaption semi-supervised approaches have been deeply investigated. In [41] an update stage based on two classifiers defines the *SEMIBOOST TRACKER*; the first one is used for tracking while the second one, referred to as auxiliary classifier, is used to select samples for the updating process. More in details, in the first frame, both classifiers are trained using the labeled data, while during the tracking, the auxiliary classifier remained fixed and provides soft-labels to the unlabeled patches, which are then used to update the tracking classifier. The method demonstrated reduction of drift and certain re-detection capabilities.
In *BEYONDSEMIBOOST TRACKER* [101] another auxiliary classifier is introduced in order to increase the adaptability of the system. Subsequently, in [67] the co-training strategy is used to guide on-line learning of each weak classifier in boosting instead of only the final strong classifier. The co-training strategy dynamically generates a series of unlabeled samples for progressively modifying the weak classifiers, leading to the robustness to environmental changes. It is proven that the co-training strategy can minimize the boosting error bound in theory.
Although this method exhibits high accuracy in scenarios where the object leaves the field of view completely, it still suffers for object imprecise localization during the update stage. *MIL TRACKER* [6] exploits on-line multiple instance learning to reduce sensitivity to inaccurate localization providing a stable control of the uncertainties on the selection of positive updates during tracking process. Recently, in [121] semi-supervised and multiple instance learning are combined into a coherent framework, leading to more robust results than applying both approaches separately. A critical factor for online boosting method is their poor capability in capturing the correlation between features, leading to the redundancy of selected features and the failure to compensate for the tracking error caused by other features.

Randomized learning-based appearance models (RLAM). More recently, randomized learning techniques [17, 61] have been successfully introduced into the vision community. The key idea behind randomized learning is to build a diverse classifier ensemble by performing random sample selection and random feature selection. In contrast to boosting and SVM, they are more computationally efficient, and easier to be extended for handling multi-class learning problems. Inspired by randomized learning, a variety of *RLAM* have been proposed in the field of visual object tracking, including online random forests [61], random naive Bayes classifiers [39], and *MILFOREST* [60]. In [39] a visual object tracking algorithm based on on-line random naive Bayes classifiers, with powerful real-time capability for processing long-duration video sequences is proposed. In *MILFOREST* [60] multiple instance learning is combined with randomized trees, to model hidden class labels inside target bags as random variables. In [55] an object detector built on an ensemble of ferns is exploited to localize the target during tracking.
Recently, in [122] a tracking algorithm whose appearance model is build on non-adaptive random projections is proposed. Such representation is able to preserve the structure of the image feature space of objects. A very sparse measurement matrix is adopted to efficiently extract the features for the appearance model and compress samples of foreground targets and the background. The tracking task is formulated as a binary classification via a naive Bayes classifier with on-line update in the compressed domain.
The major limitation associated to *RLDAM* based methods is their instability when dealing with deformable objects because of their random feature selection.

In summary, research in adaptive discriminative tracking has enabled tracking of objects that significantly change appearance and move in cluttered background. The speed of adaptation of the classifier plays a crucial role in these systems: it controls the impact of new appearances on the classifier, but also the speed by which the old information is forgotten. If the speed of adaptation is correctly set for a given problem, these trackers demonstrate robustness to short-term occlusion. On the other hand, if the object is occluded for long time, the tracker will eventually forget the relevant information and never recover.

2.3.3.3 Hybrid appearance models (HAM)

As pointed out in [107], the generative and the discriminative models presents advantages and disadvantages, and are to some extent complementary to each other. Achieved results in non rigid object detection and classification support the intuition that neither approach alone is sufficient for large scale object recognition, and motivated the interest of researchers in investigating new approaches to combine generative and discriminative models. As consequence, much effort has been made to propose a variety of hybrid generative-discriminative models able to combine the benefits of both the generative and the discriminative models in a more stable and flexible visual object model. In *EIGENBOOSTING* [43] an hybrid method capable of being discriminative with reconstructive abilities at the same time is proposed. In principle, *EIGENBOOSTING* aims to minimize a modified boosting error-function in which the generative information (i.e., eigenimages generated from Haarlike binary basis-functions using robust PCA) is integrated as a multiplicative prior. Authors in [99], propose to switch between discriminative and generative observation models according to targets proximity in a multi-target scenario. In [117] different generative models are aggregated by means of a weighted combination whose values are learned in each frame, by maximizing the distance to the background appearance; In [15] co-training of a short-term discriminative observation model and long-term generative one is exploited. In [70] two generative non-parametric models of target and background appearance

are used to train a discriminative tracker in each frame.
In [94] a sophisticated tracking system called *PROST* is proposed. It achieves top performance with a smart combination of three trackers: template matching based on normalized cross correlation, mean shift optical flow [113], and on-line random forests to predict the target location. Following a decompositional approach, authors in [55] decompose the long-term tracking task into three interacting sub-tasks, *Tracking Learning and Detection* (*TLD*), performed by three independent components. The *tracker* is a *STT* component that follows the target exploiting optical flow on local feature points lying on a regular grid generated at each tracking iteration inside the target bounding box. The *detector* localizes all appearances that have been observed so far and if necessary, corrects (re-initialize) the tracker. The *integrator* selects hypothesis coming from the aforementioned components and update the global appearance model defined by a set of patches. During the update stage, it also estimates detector errors and updates it to avoid these errors in the future, by ***pn*** *online learning* paradigm [55].
In conclusion, several advances have been achieved in designing hybrid methods able to combine generative models with discriminative ones, but defining an effective strategy to exploit such integration is still an open problem to be solved.

2.4 Discussion

In conclusion, appearance model adaptation introduces several challenges that have to be solved in order to limit the chances of drift:

- **Stability/Plasticity Dilemma** [45]. The simultaneous requirement for rapid learning and stable memory. This is a common problem of all on-line adaptive systems.
- **Robust integration of new target model samples**. The inclusion of new information from the current frame in the target

model has to be designed to be robust to the presence of outliers from the background due to non perfect alignment of the tracker bounding box with the actual target position.

- **On-line Evaluation of tracker output**. The output of the tracker must be evaluated on-line in absence of ground truth to decide whether or not to use it in model update. This is particularly important to avoid occluders appearance if the target undergoes occlusions.

In a recent work [93], a comparative analysis of the state of art adaptive tracker with respect to rare and continuous appearance changes, partial and total occlusions, initialization errors and wide angle views has been proposed. The adaptive trackers under analysis are: *BOOST TRACKER* [40], *SEMIBOOST TRACKER* [41], *BEYONDSEMIBOOST TRACKER* [101], *A-BHMC* (Adaptive Basin Hopping monte carlo)[58],*IVT*[90], *MILBOOST TRACKER*[6], *TLD*[55] and *STRUCK*[47].
From the proposed study, it has emerged that methods that let the model estimator to influence samples selection are more robust to the label noise introduced by imprecise object localization and consequently are less sensitive to the drifting problem.
Indeed four trackers exhibited good performance overall: the discriminative tracker *STRUCK* outperformed other approaches, followed by *TLD*, *IVT* and *MILBOOST*. These trackers deploy stable model representations and updates that are based on support vectors selection, ranking or subspace fitting and update.
Instead trackers that aim improving the sampling and labeling stage with priors, either fixed or adaptive, such as *BOOST TRACKER*, *SEMIBOOST TRACKER*, do not correctly update the prior and get meaningful labeled samples over time.
A promising direction of investigation has been suggested towards the hybrid methods. Combining the merits of both generative based on-line learning methods and discriminative based on-line learning

methods into a coherent framework has been acclared as the fundamental open problem to be solved in order to increase long term tracking stability. Such a combination define a classic open problem within th machine learning area [107].
Nowadays, *TLD* tracker represent one of best performing state of the art hybrid method, even if some critical behavior has been verified under condition of occlusion, fast appearance changes and resembling background.
Inspired by such analysis, in this thesis we extensively investigated the *TLD* approach with the aim of individuating the design choice that limited the performance of the approach and proposed two novel solutions that make the whole approach more robust.

Chapter 3

Tracking Learning Detection: open issues

Tracking Learning Detection TLD is an hybrid long term tracker based on a detector trained with examples found on the trajectory estimated by a short term tracker that itself is independent from the object detector. By decoupling object tracking and object detection this approach achieves high robustness and exhibits high performance respect with existing adaptive tracking-by-detection methods. In this chapter we briefly review the *Tracking Learning Detection* framework to contextualize in details, the contribution provided in this thesis.
In section 3.1, we give an overview of the approach, presenting the main components building the *TLD* architecture. In section 3.2, we present the adopted adaptive appearance model while in section 3.3, we describe the method adopted to solve the state estimation problem based on the optical flow; in section 3.4 the cascaded approach building the object detector is presented. In section 3.5 the data fusion strategy employed by the integrator component is presented, while in 3.6 the object detector bootstrapping approach is discussed. In section 3.7 we provide an analysis of *TLD* weakness, introducing open problems and challenges that should be solved in order to improve the capabilities of this high performing framework.

3.1 TLD: system overview

TLD is an hybrid system designed for long-term tracking of unknown objects in unconstrained environments. The approach is built on four interacting components namely the *tracker*, the *learner*, the *detector* and *integrator*.
In figure 3.1 a schema depicting the complete system is showed. Each

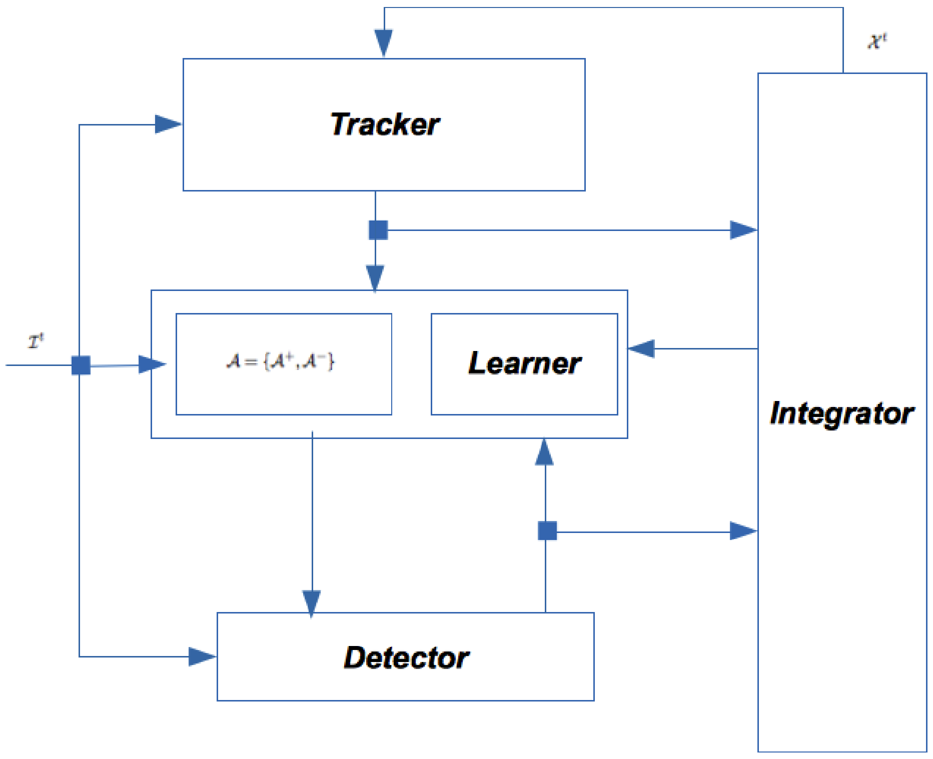


Figure 3.1: Tracking Learning and Detection architecture.

module is designed to perform a particular subtask involved with the long term tracking process.
In particular:

- The *tracker* is an exploratory and error-prone component. it is employed to solve the short term tracking problem by estimating object frame-to-frame correspondences. It is adaptive in order to handle appearance and illumination changes. Automatic detection of tracking failures is an important, but not required feature.
- The *detector* is a stabilizing component of the system designed to detect the appearances learned by the object model. It is build entirely on-line by efficient incremental update and analyze the entire image to find appearances similar to the target.
- The *learner* is designed to perform adaptive object appearance modeling. It constantly analyzes the output of the tracker and the detector, estimating errors performed by the detector and updating the object model to avoid these errors in the future.
- The *integrator* analyzes the hypothesis from the detector and the tracker and outputs the final hypothesis about the object state.

The whole system perform long-term visual tracking in two phases, namely the *start up stage* and the *run-time stage.* The *start up stage* requires the first frame of the sequence and the object to be tracked, in order to perform the following initialization procedures:

- *Initialization of the tracker* involves setting the initial state of the tracker by generating the set of points to be tracked in the next frame.
- *Initialization of the object model* involves the insertion of positive examples extracted from overlapping regions with the target initial state and examples of the background into the object model.
- *Initialization of the detector* involves training of the initial object detector to localize in the next frames the appearances represented in the object model.

After initialization, the *TLD* framework is ready to process the video stream frame-by-frame. At each time instance, the system process the incoming video frame and passes it to the tracker, the detector. The tracker estimates the object motion based on its previous state and outputs a single hypothesis. The detector provides a number of hypotheses about the location of the target object.
The integrator, analyzes the tracker and detector hypothesis in order to find the best hypothesis according to the appearance model and returns the final state that is then output of the system. Outputs of the tracker, the detector and the integrator are analyzed by the learning component, which estimates errors and updates the detector to avoid these errors in the future. When relevant changes in object appearances are detected the learning component proceeds on integration of such new information into appearance model. The process is repeated, according to the tracking loop.

3.2 Object Appearance Modeling

According to the description provided in section 2.1, object appearance modeling is performed specifying an *object state* and an *object appearance representation*.

Object state. Object state is represented by a rectangle enclosing its appearance. Such bounding box has a fixed aspect ratio (given by the initial bounding box) and is parameterized by its location and scale. Other state information such as in-plane rotation are not considered. Spatial similarity between object states is measured in terms of *overlap* , which is defined as a ratio between intersection and union of the two bounding boxes defining the states.

Object appearance representation. A single instance of the object appearance is represented by an image patch P. The patch is

extracted within the object bounding box and then is scaled to a normalized resolution (typically 15 × 15 pixels) regardless of the aspect ratio.
A measure based on normalized cross correlation (ncc) $S(P_i, P_j)$ is employed to evaluate similarity between two patches P_i, P_j:

$$ncc(P_1, P_2) = \frac{1}{n-1} \sum_{x=1}^{n} \frac{(P_1(x) - E(P_1))(P_2(x) - E(P_2))}{Var(P_1)Var(P_2)} \tag{3.1}$$

where $E(P)$ and $Var(P)$ denotes the mean and variance of gray intensity level computed over all the pixel $p \in P$, respectively.
Such measure is scaled in range [0 ,1], according to the following expression:

$$S(P_i, P_j) = 1 - \frac{ncc(P_1, P_2) + 1}{2} \tag{3.2}$$

The whole appearance model is specified by a dynamic data structure that maintains the object appearances and its surroundings observed so far. Basically it is defined by collection of positive and negative patches:

$$\mathcal{A} = \{\mathcal{A}_1^+, \dots \mathcal{A}_n^+ \ , \mathcal{A}_1^- \dots \mathcal{A}_m^-\}$$

where $\mathcal{A}^+$ and $\mathcal{A}^-$ denotes the object and background patches, reported on the left and on the right of figure 3.2 respectively.

Positive patches are ordered according to the insertion time into the collection $\mathcal{A}$ so that $\mathcal{A}_1^+$ is the first positive patch added to the collection while $\mathcal{A}_n^+$ is the last positive patch added to the set. At each iteration at least one positive sample is added to the collection of positive patches. Multiple negative patches detected around the positive one can be inserted into the model.
To evaluate how much an arbitrary patch P resembles the object appearances represented in the model $\mathcal{A}$, four similarity measures are derived from eq. 5.3:

1. **Similarity with the positive nearest neighbor**:

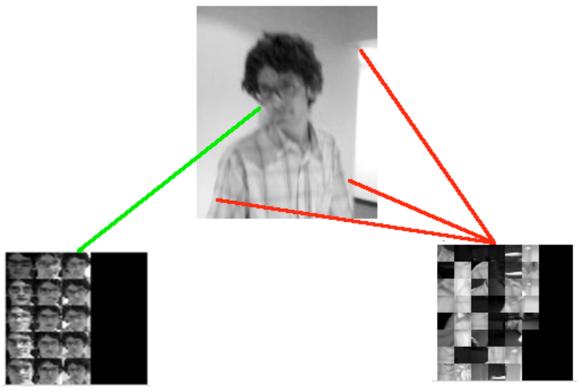


Figure 3.2: TLD appearance model, build on *David* from *MILBOOST Dataset*. On the left, $\mathcal{A}^{+}$, the collection of positive samples. On the right, $\mathcal{A}^{-}$ the collection of positive samples.

$$S^{+}(P,\mathcal{A}) = \min_{A_i \in \mathcal{A}^{+}} S(P,\mathcal{A}_i^{+}) \tag{3.3}$$

It defines the distance of the patch P with respect to the nearest positive sample stored in the appearance model.

2. **Similarity with the negative nearest neighbor**:

$$S^{-}(P,\mathcal{A}) = \min_{A_i \in \mathcal{A}^{-}} S(P,\mathcal{A}_i^{-}) \tag{3.4}$$

It defines the distance of the patch P with respect to the nearest negative sample stored in the appearance model.

3. **Relative similarity** S^{r} .

$$S^r(P,\mathcal{A}) = \frac{S^-(P,\mathcal{A}^-)}{S^+(P,\mathcal{A}^+) + S^-(P,\mathcal{A}^-)} \tag{3.5}$$

Such measure ranges from 0 to 1 and can be interpreted as the probability that the visual appearance encoded in the patch P belongs to the object model.

4. **Conservative similarity** S^c.

$$S^c(P,\mathcal{A}) = \frac{S^-(P,\mathcal{A}^-)}{S^+(P,\mathcal{A}^+) + S^-(P,\mathcal{A}^-)} \tag{3.6}$$

It encodes the same information defined in eq. 5.3, with the exception that it considers only the first 50% of positive example in the object model. In such way the visual similarity is more related to the labeled samples, available during the start up stage.

S^r is used to define the Nearest Neighbor classifier (NNC) employed to discriminate the target from the background. Figure 3.2 depitcs a clarifying example of how NNC explores the feature space $\mathcal{A}$. A new

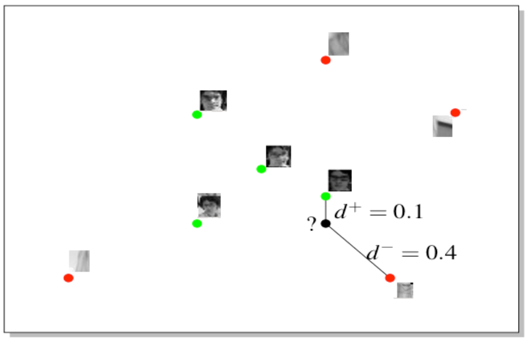


Figure 3.3: Feature Space $\mathcal{A}$ defined by *David* and *NN* classifier

patch $P \in \mathcal{P}$ is classified as positive if $S^r(P,\mathcal{A}) > \theta_{NN}$. Parameter θ_{NN} enables tuning the nearest neighbor classifier either towards

precision or recall.
The classification margin is defined as $S^r(P, \mathcal{A}) - \theta_{NN}$ and measures the confidence of the classification. The integration of new labeled patches into the object model is performed by adding only those samples miss-classified by NN classifier, as sketched in algorithm 1.

This strategy leads to a significant reduction of accepted patches

Algorithm 1 NN-Classifier

```
1: procedure NN-Classifier
2:     Input: $\mathcal{A} = \{\mathcal{A}^+,\ \mathcal{A}^-\},\ \mathcal{P} = \{\mathcal{P}^+,\ \mathcal{P}^-\}$
3:     for $P \in \mathcal{P}$ do
4:         $conf = S^r(P, \mathcal{A})$
5:         if $conf < \theta_{NN} \wedge P \in \mathcal{P}^+$ then
6:             $\mathcal{A}^+ = \{\mathcal{A}^+,\ P\}$
7:         end if
8:         if $conf > \theta^{NN} \wedge P \in \mathcal{P}^-$ then
9:             $\mathcal{A}^- = \{\mathcal{A}^-,\ P\}$
10:        end if
11:    end for
12: end procedure
```

[55] at the cost of coarser representation of the decision boundary.

3.3 Tracker: Median Flow

The tracker component is the short term generative tracker that self-learns the appearance model at tracking iteration.
It is based on *median-flow tracker* [54] extended with failure detection and is designed to estimates object motion between consecutive frames under the assumption that the motion is limited and the object is always visible.
Median-flow tracker represents an object by a rectangle and perform object localization by estimating displacements of a fixed set of pixels $\mathcal{K} = \{\mathcal{K}_i^t = (x, y, t)|\ i = 1 \ldots n^2\}$ lying on a regular $2D$ grid overlapping with object, regenerated at each tracking iteration inside the predicted target state.
To estimate point correspondences between consecutive frames, *median-*

flow tracker exploits *pyramidal Lucas-Kanade tracker (KLT)* [120] with 2 levels of the pyramid and patches of 10×10 pixels.
KLT is a differential approach for optical flow estimation that try to calculate the motion between two image frames which are taken at times t and $t + \Delta t$ at every pixel position.
Given a pixel location (x, y) at time t with intensity value $I(x, y, t)$ the objective is to estimate its displacment at time $t + \Delta t$, indicated by the variable Δx, Δy.
Assuming no change of pixel intesinty, the "brightness constancy constraint" gives the relation:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{3.7}$$

Assuming the movement to be small, the image constraint at $I(x, y, t)$ with Taylor series can be developed to get:

$$\begin{gathered} I(x + \Delta x, y + \Delta y, t + \Delta t) = \\ I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + h.o.t. \end{gathered} \tag{3.8}$$

From these equations it follows that:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{3.9}$$

which results in:

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \tag{3.10}$$

where V_x, V_y are the x and y components of the velocity or optical flow of $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x, y, t) in the corresponding directions.
I_x,I_y and I_t can be written for the derivatives in the following.

$$I_x V_x + I_y V_y = -I_t \tag{3.11}$$

This is equation in two unknowns, represents the *aperture problem* of the optical flow algorithms and needs additional constraint to be solved. In [120], the *spatial coherence* constraint provides a new set of equations o solve the problem. It assumes that the displacement of the image contents between two nearby instants (frames) is small and approximately constant within a neighborhood of the point p under consideration. As consequence, the local image flow (velocity) vector (V_x, V_y) satisfies:

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \tag{3.12}$$

where $q_1, q_2, \ldots, q_n$ are the pixels inside the window, and $I_x(q_i)$, $I_y(q_i)$, $I_t(q_i)$ are the partial derivatives of the image I with respect to position (x, y) and time t, evaluated at the point q_i and at the current time. These equations can be written in matrix form $Av = b$, where

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \quad , \quad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \quad , \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \tag{3.13}$$

This system is over-determined and a linear least squares solution is obtained by solving the 2×2 system $A^T Av = A^T b$, leading to:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \tag{3.14}$$

$$\begin{bmatrix} \sum_i^n I_x(q_i)^2 & \sum_i^n I_x(q_i)I_y(q_i) \\ \sum_i^n I_y(q_i)I_x(q_i) & \sum_i^n I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i^n I_x(q_i)I_t(q_i) \\ -\sum_i^n I_y(q_i)I_t(q_i) \end{bmatrix}$$

The matrix $S = A^T A$ is referred to as the *structure tensor* of the image at the point p.

By analyzing the *structure tensor* for each layer building a pyramidal representation of an image is possible to estimate big displacement

within an iterative approach. The key idea is to solve the optical flow equation 3.12 for each image building the pyramid, starting from the lowest resolution. For each layer, the iterative last square solution is computed providing as starting flow vector the solution estimated in the previous-level. Addition implementation details are given in [120]. The basic assumption employed by *KLT* method are often violated in real video streams.
For this reason, refinement stages based on motion analysis are employed to discard wrong correspondences.
The *median-flow tracker* (described in algorithm 2) exploits two type of error measures to reject points that were tracked unreliably, namely *forward-backward error* and *normalized cross correlation.*

Algorithm 2 Median Flow Tracker

```
1: procedure MEDIAN FLOW TRACKER
2:     Input: $B^t$  $I^t$  $I^{t+1}$
3:     Output: $B^{t+1}$
4:     Initialize $X = \{(x_i, y_i), i = 1 \dots n^2\} = GridPoints(B^t)$
5:     while $i < n^2$ do
6:         %KLT tracking
7:         $\mathcal{T}_i^f = KLT_{forward}(x_i)$
8:         $\mathcal{T}_i^b = KLT_{backward}(x_i)$
9:         %Error Measures
10:        $e_i^{fb} = ||\mathcal{T}_i^f - \mathcal{T}_i^b||$
11:        $e_i^{ncc} = ncc(P_1, P_2)$
12:    end while
13:    %Median Filtering
14:    $med_{FB} = median(e_i^{fb}, \dots, e_{n^2}^{fb})$
15:    $med_{NCC} = median(e_i^{ncc}, \dots, e_{n^2}^{ncc})$
16:    if $med_{FB} > \theta_{FB}$ then
17:        $B^t = \varnothing$
18:    else
19:        %Select Reliable Points
20:        $X_{reliable} = \{(\mathbf{x}^{t+1}, \mathbf{x}^{t+1})|\mathbf{x}^{t+1} \neq \varnothing, e_i^{fb} \leq med_{FB}, e_i^{ncc} \geq med_{NCC}\}$
21:        %Estimate the Affine Transform: scale S - translation T
22:        $[S\ \ T] = Transform(X_{reliable})$
23:        %Final State Estimation
24:        $B^{t+1} = SB^t + \mathrm{T}$
25:    end if
26: end procedure
```

The *forward-backward error* is based on the idea that the tracking of points must be reversible and is derived as follow.
Let $\mathcal{T}_f^t = \{\mathbf{x}^{t-k} \cdots \mathbf{x}^t\}$ be the trajectory of a target point $\mathbf{x}$ tracked up to time t and let $\mathbf{x}^{t+1}$ the corresponding point tracked by KLT at time $t+1$. The validity of $\mathbf{x}^{t+1}$ is established by tracking it backward for k previous frames and extracting the backward trajectory $\mathcal{T}_b^t = \{\hat{\mathbf{x}}^{t-k} \cdots \hat{\mathbf{x}}^t\}$ (lines 8-10).
The euclidean distance between the computed trajectories $||\mathcal{T}_f^t - \mathcal{T}_b^t||$ (line 13) defines a measure of reliability of the tracked point, allowing to discard those points whose trajectory is unstable over time.
Additionally a measure of visual similarity based on normalized cross correlation (ncc) evaluated on patches $\mathbf{P}$ centered on each trajectory location $\hat{\mathbf{x}}$ is employed to establish the visual consistency of tracked points (line 14). A filtering approach (lines 18-32) employs the median of all forward-backward errors med_{FB} and the median med_{NCC} of all similarity measures to reject those points exhibiting a forward-backward error larger than med_{FB} and a similarity measure lower than med_{NCC} (line 25). This strategy is able to reliably identify failures caused by fast motion or fast occlusion of the object of interest. Indeed, when such events occur the individual displacement become scattered around the image and the residual rapidly increases (a threshold of 10 pixels was experimentally found not critical). This heuristic is able to reliably identify most failures caused by fast motion or fast occlusion of the object of interest. If the failure is detected, the tracker does not return any bounding box. Otherwise, the remaining points are used in order to estimate the position of the new bounding box in the second frame by employing a transformation model based on changes in translation and scale (line 28). In particular, the pairwise distances between all points are calculated before and the relative increase is interpreted as the change in scale. The translation in x-direction is computed using the median of the horizontal translations of all points. The translation in y-direction is calculated in the same way.

This method works most reliably if grid points are located on corners but becomes very unstable when tracking points are generated over homogeneous regions. Furthermore, median flow tracker does not maintain an object model and is therefore unable to recover from failure. Object detector is the component designed to allow to re-initialize short term tracker when drifting.

3.4 Object Detector

While the recursive tracker depends on the location of the object in the previous frame, the object detection mechanism employs an exhaustive search based on a sliding-window approach [59]. In such way, re-detection capabilities and invariance to fast changes of target motion dynamic are enabled in the whole system.
Since several thousands of sub-windows should be tested independently to check if they contains the object of interest, a cascaded object detector based on three stages is employed to reject as many non relevant sub-windows with a minimal amount of computation. The complete schema of the this component is depicted in figure 3.4.
In the first stage all sub-windows that exhibit a variance lower than

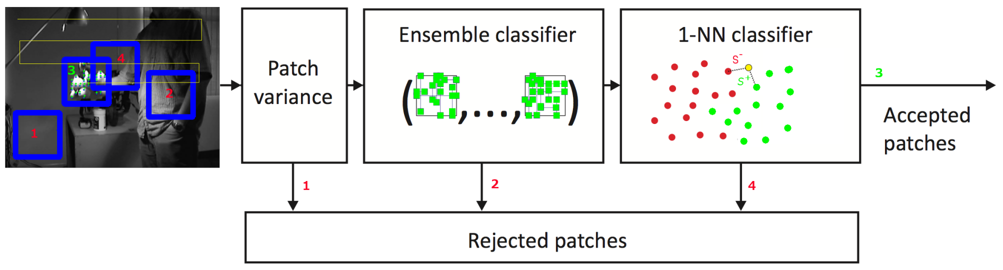


Figure 3.4: The Cascade Object Detector work-flow. 1) a candidate patch rejected by the variance filter. 2) a candidate patch rejected by the ensemble classifier. 3) a positive patch detected by the ensemble classifier. 4) the false positive patch detected by the ensamble classifier

a certain threshold Θ_{var} are rejected. Such a variance filter is able to rapidly reject uniform background regions (the box 1 in fig. 3.4) but unable to distinguish between different well-structured objects (boxes $2, 3, 4$ in fig. 3.4).

The second stage comprises an ensemble of base classifiers exploiting *ferns features* [61]. As pointed out in section 2.3.2, *ferns feature* is a patch texture descriptor based on a binary comparisons of the intensity values of d random couple of pixels $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ drawn from a uniform distribution once at startup and remained constant over time. Such binary test, invariant against constant brightness variations, defines a binary digit f_i as follow:

$$f_i = \begin{cases} 0 & \text{if } I(p_1) < I(p_2) \\ 1 & \text{otherwise} \end{cases} \tag{3.15}$$

By concatenation of the d responses, a binary number F, is generated and its decimal form defines the *ferns* response used to derive (as will explained in section 3.6) the posterior probability $P(y = 1|F)$ that a sub-window belongs to positive class ($y = 1$). The fern classifier maintains a distribution of posterior probabilities of 2^d entries. In [55] 13 comparisons are used, leading to a space of 8192 possible binary codes indexing the posterior probability. The approach adopted to estimate the posterior probability associated with the fern F is directly encoded into the learning component described in section 3.6. The ensemble classifier is constructed by averaging M different ferns as showed in fig. 3.5. To guarantee the independence of the base classifiers [17], the space of pixel locations within a normalized patch is first discretized and then all possible horizontal and vertical pixel comparisons are generated. Comparisons of zero length are not discarded and after random permutation, the remaining comparisons are split into the base classifiers.

As a result, every classifier is guaranteed to be based on a different set of random pixel, uniformly covering the entire patch. This is in contrast to other approaches [61], where pixel comparisons are generated independently one other. For each tested sub-window, if posterior probability P is smaller than a threshold ($\Theta_{ensemble} = 0.5$)

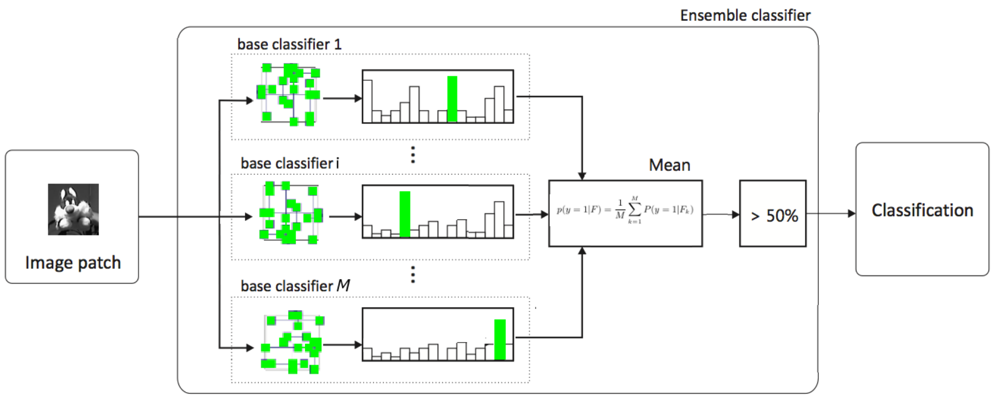


Figure 3.5: Ensemble of ferns classifier: average of M fearns feature

the patch is classified as negative and is rejected (boxes 2 in fig. 3.4). In [55] 10 base classifiers have demonstrated a good trade-off between detection accuracy and real-time performance capabilities. Clearly, if the speed is not an issue new base classifiers can be added increasing the performance of the ensemble. Within the third stage, the detection provided by ensemble classifier are evaluated by the nearest neighbor classifier building the appearance model.

3.5 Integrator

Median flow tracker and *object detector* run in parallel in order to realize the *sampling and labeling* stage and the *feature extraction and refinement* stage described in section 2.1.
They have identical priorities, and they can be thought fundamentally as two competitive estimators of the object state. While the detector localizes already known templates, the tracker localizes potentially new templates and thus can bring new data for the detector. The *integrator* is the component designed to combine the tracker and the detector hypotheses to predict object location in the current image. A naive implementation is sketched in algorithm 3.

If neither the tracker nor the detector output a bounding box,the object is declared as not visible. Otherwise the integrator outputs

Algorithm 3 INTEGRATOR Component

```
1: procedure INTEGRATOR
2:     Input: $B^t_{tracker}$ , $B^t_{detector}$
3:     Output: $B^t_{final}$
4:     if $S^r(I^t(B^t_{tracker}) > S^r(I^t(B^t_{detector})$ then
5:         $B^t_{final} = B^t_{tracker}$
6:     else
7:         $B^t_{final} = \sum_{B \cap B_{detector} > 0.8} B^t_{detector}$
8:     end if
9: end procedure
```

the maximally confident bounding box, measured using the Conservative similarity S^c.
To increase the localization accuracy, when multiple detection are available in proximity of the the maximally confident detection, the tracker bounding box is averaged with all detections that exhibits an overlap > 0.8. If the maximally confident detection is far from the tracker (overlap < 0.8), the tracker is re-initialized by regenerating the fixed grid of point inside the estimated state.

3.6 Object detector learning strategy

The objective of the learning component is to train the ensemble detector in the first frame and bootstrap its performance at run-time using the P/N learning strategy [55] .
The initial training of the detector is performed by synthesizing 200 positive examples from the initial object appearance, providing scaled, translated and in plane rotated versions of the initial bounding box. Negative patches are collected from the surrounding of the initial patch, without generating synthetic examples.
The detector update, at run-time is performed through the P/N learning, a semi-supervised learning approach that, exploiting so-called *structural constraints*, extracts training data from new unlabeled data and updates the classifier building the detector.
Such learning scheme employs two classes of constraint, namely the

P-Constraint and the N-constraint, in order to update on-line each fern base classifier F_k.
The P-Constraint assumes that all the patches highly overlapped with the final result must be classified as positive examples. According to such constraint, the learner perform a label switch for negative classified patterns that overlaps with final solution. The N-Constraint requires that all the patches that are not overlapped with the valid final result must be classified as negative examples. Consequently it involves a label switch for positive classified patterns that are far from the current solution. The number of label switching associated to each ferns F_k measures the feature reliability, i.e. it indicates how such descriptor is representative for the object. Such concept is directly expressed in to strategy to update the posterior probability that is defined by:

$$p(y=1|F_k) = \begin{cases} \frac{p_{F_k}}{p_{F_k}+n_{F_k}} & \text{if } p_{F_k}+n_{F_k}>0 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

where p_{F_k} is the number of times a P-constraint is applied to F_k while n_{F_k} n_{F_k} refers to the number of times a N-constraint are applied to F_k.
Clearly, the initialization stage, set up $p_{F_k} = 1$ and $p_{F_k} = 0$ for each fern F_k. With this rule, structural constraints provide a feedback about the performance of the classifier which is iteratively improved in a bootstrapping fashion. In practice, at each tracking iteration, the training set is augmented with miss-classified examples and classification functions are updated by hard samples that are more challenging for the current model. In [55] a theoretical analysis provides conditions under which the P-N learning guarantees improvement of the classifier.

3.7 TLD: open problems and proposed solution

An extensive set of experiments[55] on standard benchmark datasets (*MILBOOST DATASETS*, *TLD DATASETS*), has demonstrated *TLD* capabilities with respect to the closest competitors under various challenging conditions such as abrupt camera motion, motion blur, appearance changes.
However some open problems have been recently highlighted in [93]. In particular, a fundamental limitation of the method has been proved under condition of partial, full occlusion and similar objects or background. Such events represent the most challenging scenario for this approach since they severely stress the interaction between the generative tracker and the discriminative object detector building the system. Indeed, when the target is partially occluded the generative tracker systematically learns the occluding appearance as a change in object appearance.
In the worst case, when the occluding object resembles the target appearance, the *tracker* component promotes wrong hypothesis to the *integrator* component, causing in the worst case, the learning of negative examples. Consequently the object detector is wrongly updated and becomes a persistent source of errors. In figure 3.6 a clarifying example of this critical behavior is depicted. The *coke* is characterized by areas of uniform color resembling surrounding background and as it moves behind the leaf the tracker component drifts. Indeed, after reinitialization, stationary local points on the leaf (*blue dots*) are identified by median operator as the correct ones, leading the final solution to drift (*yellow box*). Even if among ensemble detections (*all other colored boxes*), the correct one is present, its confidence is still too low to activate error correction. The time required to correct the tracker depends on leaf similarity respect to target appearance and on the ability of the *detector* component to recover with an higher confidence the correct target hypothesis that enables error correction trigger.

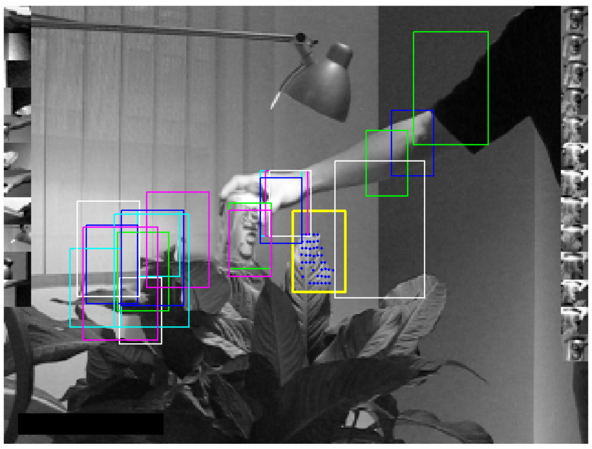


Figure 3.6: *TLD* failure on *Coke* sequence (*MILBOOST DATASET* [6]). On the right column *positive samples*, on the left *negative samples.* In yellow *TLD* final solution corresponding to *median flow tracker* solution, while other colored boxes are detections provided by *detector*

Avoiding such condition is the open problem of the *TLD* paradigm, and from a general point of view it face off an open question of the machine learning community on how integrating supervised learning with unsupervised learning methods.
The aim of the approaches proposed in this thesis is directed towards the solution of this specific problem.
In particular we integrate into the *TLD* approach two algorithm able to alleviate such critical behavior and improve the overall system performances.
The first approach,the *BTLD* tracker, has been designed to tackle the problem directly on its source (*the tracker component*). A deep investigation of *median flow* has reveled a systematic drifting behavior caused by *reinitialization strategy* adopted at each tracking iteration to regenerate the 2*D* grid defining the target state. In chapter 4, our contribution towards the solution of this critical behavior is presented. The second approach *BTLD+*, extend the *BTLD* tracker

into a discriminative learning framework, introducing a coupled layered visual representation of the object, that model the target at different levels of resolution. In such way, the *reinitialization strategy* is controlled by a classifier able to discriminate among target and background local features.

Chapter 4

BTLD: a Bayesian Approach to Tracking Learning Detection

As stated in [93], *TLD* paradigm is successful in several scenarios but exhibits a systematic drifting behavior in presence of occlusions and resembling background, due to the *reinitialization strategy* adopted to correct the median flow tracker at each tracking iteration.
Enforcing the robustness of such generative tracker has been pointed as an open problem for the *TLD* paradigm.
This challenge poses a fundamental question on how the unsupervised learning component of the system should select new data in order to find noise free unlabeled data necessary to update the supervised appearance model.
This chapter is organized as follows: section 4.1 analyzing the *TLD* weakness from a theoretical point of view, introduces the reason motivating the design of a novel generative tracker; in section 4.2, we present the novel generative tracker and its integration into the *TLD* framework; sections 4.4 and 4.5 present the object detector and the integrator component respectively; in section 4.6 experimental results are showed comparing the proposed approach with the original *TLD* and other state-of-the arts methods; finally, section 4.7 summarizes the main contributions and highlights open research challenges.

4.1 Motivation

The systematic drifting behavior of the *median flow tracker* is related to the design choice of tracking points lying over a fixed grid that is reinitialized at each tracking iteration on the estimated target location. Within reinitialization approach, authors assumes a complete visibility of the target and .
Such *reinitialization strategy* inevitably involves the selection of new points, that a priori may generate the failure of the *Lukas-Kande method*, since it is able to track points centered on high textured regions.
This condition represents a fundamental constraint to the above method, since it allows to verify the assumptions that lead to the resolution of the *aperture problem* of the optical flow. It can be seen from equation 3.13 that v can be calculated only if the *structure tensor* S is invertible. S is reliably invertible if it has two large eigenvalues (λ_1, λ_2), which is the case when the gradient of the image assume values in both directions. This is the reason that makes *corner points* "good feature to track" [96], since they are defined as high responses to the operator:

$$det(S) - \mathbf{k}trace(S)^2 \tag{4.1}$$

The *median* operator applied to *forward-backward error* and *normalized cross correlation* control such instability, filtering out unreliable tracked points. Such measures of spatio-temporal consistency and visual similarity allows the selection of reliable points in order to estimate the final box defining the current state.
Such box is parameterized by horizontal displacement, vertical displacement and scale change and all three parameters are estimated independently using the *median*.
When the target overlaps with other objects, the tracker does not detect any occlusion, since the most reliable points continue to maintain the same relative distance over time and is in accord to the median statistic. At each subsequent re-initialization, points belonging to

the target are replaced by the points of the occluding object leading the tracker to drift.
The detector is designed to correct such failure, but in some cases wrong samples are learned before the error correction take place, leading the whole system to an irreversible state. However, the time required to activate the error trigger is unpredictable. Furthermore, it strongly depends on how many target variability has been learned before the occlusion.
Providing a more stable tracker component and in particular a more flexible *reinitialization strategy*, is a crucial problem to improve the *TLD* performances.
The *reinitialization strategy* is a challenging problem in adaptive visual tracking. In [48] a set of simple features (e.g., optical flow features) is used to track individual parts of the object while distances among features are used to add or remove salient points during tracking. Since the set of features is *geometrically unconstrained*, the tracker is likely to get stuck on the background, losing the target.
In [118] *Harris corner* is used to detect stable regions for tracking and enforcing a single global affine transformation constraint to avoid drifting. However, authors assume that the shape of the object can be approximated with an ellipsoid and that the object does not deform, limiting the generality of the tracker.
In this work we approach the aforementioned problems by focusing short term tracking on high textured regions localized around *Harris* local maxima and by motion vector analysis, rejecting unreliable *KLT* points. A novel *reinitialization strategy*, directly encoded into our probabilistic framework, is proposed to reidentificate at each tracking iteration the set of reliable points.
The novel idea behind our *reinitialization strategy* is to add new regions of interest around high confident points tracked from the previous frame and filter out those regions that are not geometrically consistent with the best explanation of the target global appearance during the inference process.

Inspired by the outlier filtering scheme proposed in [24] for multiple target tracking, we designed an *markov chain monte carlo* (*MCMC*) particle filter that automatically rejects local features not consistent with the current estimate of the target location and scale. In this way stable regions are geometrically constrained to the estimated target area *without any assumption on its shape.*
We integrated our method into *TLD* approach, resulting in a new efficient and accurate long term tracker, that we named *Bayesian Tracking Learning Detection* (*BTLD*).
A schema depicting the complete *BTLD* architecture is showed in figure 4.1.

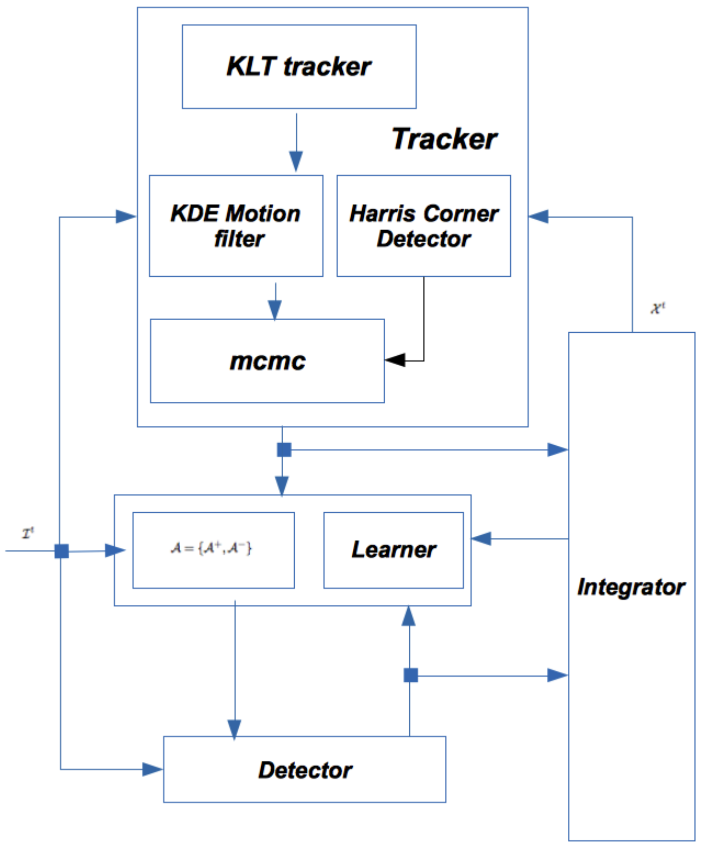


Figure 4.1: *BTLD* architecture.

4.2 Proposed Approach

With *BTLD*, we focus *KLT* only on salient regions defined around local maximum of *Harris* operator defined in 4.1. This strategy as previuosly discussed, reduces *KLT* serious failures to resembling background since it restricts the tracking on high textured regions. As consequence, at each tracking iteration a sparse set of points $\mathcal{K}^t$ is employed to track the target over time, as depicted in figure 4.2. From now on, Harris local maxima and regions of interest will re-

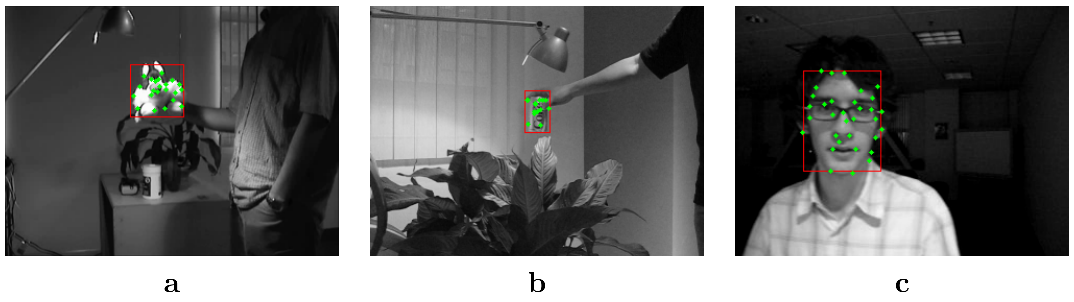


Figure 4.2: Tracking a sparse set of corners $\mathcal{K}^t$. *Sylv* (a), *Coke11* (b) ,*David* (c),from the *MILBOOST Dataset* [6]

ferred with the same meaning.
To remove erroneous correspondences generated by *KLT* failures, we apply a filtering scheme based on motion analysis. Assuming coherent motion of the tracked points, we remove those points whose motion does not agree with the motion distribution estimated by kernel density estimation, over the set $\mathcal{K}^{t+1}$ of tracked points.
This processes removes *KLT* serious failures but reduces drastically the number of local feature points. In order to guarantee a sufficient number of such salient points, new ones, detected near the remaining points are added. This is the most critical stage, since there is no prior knowledge about the "nature" of such new salient regions.
The main idea is to allow the selection of new unconstrained salient points followed by a refining step where not consistent elements are rejected.

In this way geometrical constraints can be encoded in our *MCMC* particle filter, where we introduce two competitive likelihood functions: one promotes the addiction of new salient points in $\mathcal{K}^{t+1}$, the other rejects local feature points that are not consistent with the appearance encoded into the target visual model.
At each tracking iteration t, the set of tracked points $\mathcal{K}^t$ is updated by two refining event, namely the growing and pruning.
Within the growing event new interest points $\mathcal{K}^t_{new}$ detected in the current image are proposed as new point of interest to be added to the set of reliable $\mathcal{K}^t$. On the other hand, the pruning event is designed to remove unreliable tracked points from $\mathcal{K}^t$. The complete procedure is reported in algorithm 4. At the t iteration, we track

Algorithm 4 *BTLD* Tracker

```
1: procedure BTLD Tracker
2:     Input: $\mathcal{K}^t$ , $B^t$ , $I^t$
3:     Output: $B^{t+1}$
4:
5:     Initialize $\mathcal{K}^0 = HarrisCornerDetector(B^0)$
6:
7:     while $t < N_{frames}$ do
8:         $\mathcal{K}^{t+1} = KLT_{tracker}(\mathcal{K}^t)$
9:
10:        %Build Motion Vectors
11:        $\mathcal{V} = |\mathcal{K}^{t+1} - \mathcal{K}^t)|$
12:
13:        %Build the set of reliable and unreliable points
14:        $[\mathcal{K}_r^{t+1} , \mathcal{K}_u^{t+1}] = KernelDensity(\mathcal{V})$
15:
16:        %Detect new corner
17:        $[\mathcal{K}_{new}^{t+1}] = HarrisCornerDetector(\theta_{corner})$
18:
19:        %Target State Estimation
20:        $[B^{t+1} , \mathcal{K}^{t+1}] = mcmc(\mathcal{K}_r^{t+1}\mathcal{K}_u^{t+1})$
21:    end while
22: end procedure
```

independently by *KLT* each point belonging to the sets $\mathcal{K}^t$ (line 8). We denote $\mathcal{K}^{t+1}$, the set of corresponding points, generated by the

KLT tracker.
In the next step, we compute $\mathcal{V}^t = \{|K_i^t - K_i^{t-1}|, i = 1 \dots |K_i^t|\}$, the set of motion vectors associated to tracked points (line 11).
We estimate the motion distribution $p(V|V^t, \Sigma_v)$ of such points by *gaussian kernel density estimation* (*KDE*) (line 14):

$$p(V|V^t) = \frac{1}{|\mathcal{V}^t|} \sum_{i=1}^{|\mathcal{V}^t|} K_v(V, V_i^t) \tag{4.2}$$

where K_v is gaussian kernel parametrized by the 2×2 covariance matrix Σ_v. Assuming a motion consistency among positive target points, we build the set of reliable points $\mathcal{K}_r^t$ and of unreliable points $\mathcal{K}_u^t$.
$\mathcal{K}_r^t$ contains the subset of tracked points that are moving in a coherent way respect to the peak V_{max} of $p(V|V^t)$ while $\mathcal{K}_u^t$ maintains rejected points. The measure of reliability for a motion vector V_i^t is obtained by:

$$R(V_i^t, V_{max}) = (V_i^t - V_{max})^T K_v (V_i^t - V_{max}) \tag{4.3}$$

In particular, $R(V_i^t, V_{max}) <= 1$, in relation to K_v properties, defines a circular or elliptical filter centered around V_{max}.
In addiction, new corners in proximity of the reliable points are detected and are maintained in $\mathcal{K}_{new}^t$ (line 17). The proximity is specified by the parameter d_{corner}, which measure the maximum distance allowed for new detected corner to be considered as:

- a possible change in object appearance that has not been yet observed.
- a background region that was not visible before, because occluded by the target.

The three sets $\mathcal{K}_r^t, \mathcal{K}_u^t, \mathcal{K}_{new}^t$ are then refined by our *mcmc* procedure that automatically search for the best state $X^t = [x^t\ , y^t\ , w^t\ , h^t]$ that jointly:

- contains the maximum number of reliable tracked points $\mathcal{K}_r^t$.
- encloses a visual content that maximizes the visual similarity to the object appearance model.

In such way, new detected points $\mathcal{K}_{new}^t$ and $\mathcal{K}_u^t$ unreliable tracked points are automatically evaluated and inserted in the final set $\mathcal{K}^t$ if they belongs to a state that exhibits high similarity to the global appearance model.
Figure 4.3 depicts the result produced by the most relevant stages of the proposed algorithm, while processing the sequence *David*. The

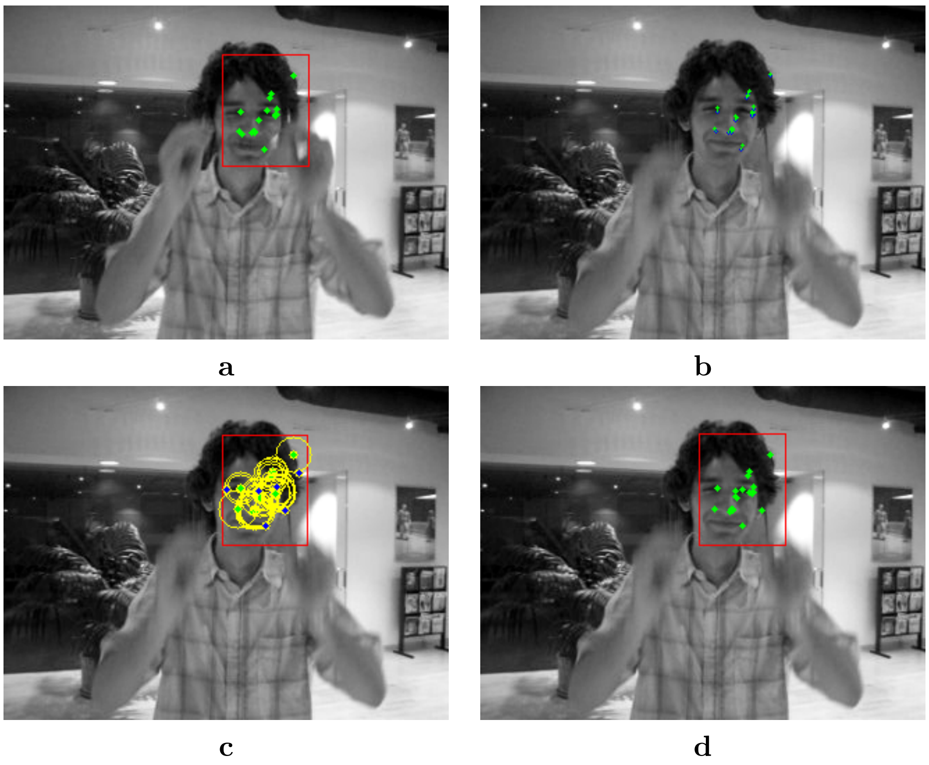


Figure 4.3: Intermediate results while tracking *David* from *MILBOOST Dataset*[6]. **a)** The set of tracked point from the frame 295, $\mathcal{K}^{295}$. **b)** The set $\mathcal{K}_r^{296}$ of reliable tracked point on the frame 296, after *motion filtering*. **b)** The set $\mathcal{K}_{new}^{296}$ of new detected point on the frame 296. In blue the new detected points, in green the reliable tracked points. **c)** The final set $\mathcal{K}^{296}$ after *mcmc filtering*.

four images, illustrate the intermediate results while tracking the

David from frame 295 to 296.
In particular, we can observe:

- the pruning approach realized by the *KDE motion filter* in order to detect and remove unreliable tracked points.
- the growing approach realized by the *MCMC state estimator* to add new corners (blu points in fig. 4.3-(c)) detected near the reliable points (green points in figure 4.3-(c)).

4.3 Bayesian Formulation

Following the Sequential Bayesian formulation, the posterior probability of target state X^t a time t is given by:

$$\underbrace{p(X^t|\mathcal{O}^t)}_{posterior} \approx \underbrace{p(\mathcal{O}^t|X^t)}_{a} \int \underbrace{p(X^t|X^{t-1})}_{b} \underbrace{p(X^{t-1}|\mathcal{O}^{t-1})}_{c} \, dX^{t-1} \tag{4.4}$$

where (a), (b) and (c) in Eq. 4.4 represent the *observation likelihood*, the *motion model* and the *posterior* from previous time, respectively. The hidden state $X^t = [\ x^t,\ y^t,\ w^t,\ h^t]$ is encoded by location (x^t, y^t) and size (w^t, h^t) information of the 2D box enclosing the target, resulting in a 4D state space $\mathcal{X}$.
$\mathcal{O}^t = [\mathcal{K}^t\ \mathcal{A}^t]$ represents the measurement space, where $\mathcal{K}^t = \{\mathbf{K}_i^t \in \mathcal{R}^2\}$ is the set of local points tracked from previous frame and $\mathcal{A}^t$ represent the adaptive global appearance model analyzed in 3.2 and updated on-line employing the NN classifier described in 7.
Assuming $\mathcal{K}^t$ and $\mathcal{A}^t$ independent, the *observation likelihood* $\mathcal{O}$ is factorized as:

$$p(\mathcal{A}^t, \mathcal{K}^t|X^t) = p(\mathcal{A}^t|X^t)p(\mathcal{K}^t|X^t) \tag{4.5}$$

Observation likelihood of $\mathcal{K}^t$ measures the fraction of local feature points lying inside the candidate target state X^t:

$$p(\mathcal{K}^t|X^t) = \frac{\mathbf{K_i}^t\ \in \mathbf{X}^t}{|\mathcal{K}^t|}. \tag{4.6}$$

Such distribution promotes candidate states containing the maximum number of tracked local features, assuming that they are free of errors. *KLT* failures, are automatically rejected by the global appearance likelihood modeled by *TLD*.
It assign low confidence to hypothesis containing local tracked points and not resembling target appearance, assuming an "outliers-rejection" role similar to ramdom sampling consensus (*RANSAC* [34]).
The visual likelihood, measuring the visual similarity of a candidate state respect to the target appearance model, is directly derived by *relative similarity* defined in equation 3.5:

$$p(\mathcal{A}^t|X^t) = S^r(P, \mathcal{A}) \tag{4.7}$$

We assume a linear *dynamic model* modeled by a gaussian distribution over $\mathcal{X}$, centered on previous target X^{t-1} location and scale:

$$p(X^t|X^{t-1}) = N(X|X^{t-1}, \Sigma_x) \tag{4.8}$$

Considering the complexity of the given probabilistic formulation, it is extremely challenging to design an analytical inference method for estimating the maximum a posterior (*MAP*) solution:

$$\hat{X}^t = \underset{X^t \in \mathcal{X}}{\operatorname{argmax}}\, p(X^t|\mathcal{O}^t) \tag{4.9}$$

This challenge is due to the presence of the high nonlinearity of observation likelihood functions. We propose to employ a sampling based sequential filtering technique based on the *MCMC* particle filter[66].
At each time step t, we approximate the posterior by a number N of samples:

$$p(X^{t-1}|\mathcal{O}^{t-1}) \approx \{X_s^{t-1}\}_{s=1}^N. \tag{4.10}$$

Propagating samples through the *motion model*, we generate particles for the *predictive distribution* and approximate the posterior distribution at time t by Monte Carlo integration:

$$p(X^t|\mathcal{A}^t,\mathcal{K}^t) \propto \tag{4.11}$$
$$p(\mathcal{A}^t|X^t)p(\mathcal{K}^t|X^t)\sum_{s=1}^{N} p(X^t|X_s^{t-1})p(X_s^{t-1}|\mathcal{A}^{t-1},\mathcal{K}^{t-1})$$

Approximation in eq. 4.11 is achieved by a Markov chain over the joint space of $\mathcal{X}$ that converges over the posterior distribution $p(X^t|\mathcal{A}^t,\mathcal{K}^t)$.
The whole *Metropolis-Hasting* procedure with component-wise updating scheme is sketched in algorithm 5.

Algorithm 5 MCMC Particle Filter

```
1: procedure MCMC Particle Filter
2:     Input: $\mathcal{K}^t$, $\mathcal{Y}^t$, $X_i^{t-1}$
3:     Output: $p(X^t|\mathcal{Y}^t,\mathcal{K}^t)$
4:
5:     Initialize $X_0^t = X^{t-1} = [x,\ y,\ w,\ h]^{t-1} = \{X_l,\ X_s\}^{t-1}$
6:
7:     while $i < N_{accept}$ do
8:         Select uniformly the candidate component $j \in [1\ \ 2]$
9:
10:        Propose $X_j^* \sim N(X_j^i|X_j^{i-1})$
11:
12:        Build the hypothesis $X^* = \{X_j\ ,\ X_{k\neq j}\}$
13:
14:        Evaluate the acceptance probability $\alpha = min(1, \frac{p(X_s^*|\mathcal{A}^t,\mathcal{K}^t)}{p(X_s^{i-1}|\mathcal{A}^t,\mathcal{K}^t)})$
15:
16:        Accept $X_s^* \rightarrow X_s^{i+1}$ if $\alpha < u \leftarrow$ uniform sample $\in [0\ 1]$
17:    end while
18: end procedure
```

For *MCMC* sampling to be successful, it is critical to have a good proposal distribution which can explore the hypothesis space efficiently.
Our *proposal distribution* generates separate random hypothesis for location ($\mathcal{X}_1$) and scale ($\mathcal{X}_2$) subspaces (line 8), according to normal deviates from previous accepted hypothesis (line 10).
The stopping criteria adopted to terminate the proposal generation

relies on the number of accepted samples N_{accept} (line 7). Once the sampling method has reached convergence, the maximum a posterior estimate for X^t is analyzed by the *TLD integrator* that establishes the final solution.

4.4 Object Detector

The object detector is built following the approach described in section 3.4. In addiction, a refinement step has been introduced to reduce the number of overlapping produce detections. All relevant detections are clustered by the agglomerative hierarchical clustering approach proposed in [82], exploiting a similarity measure based on overlap ratio between rectangles. According to such measure, the distance $d(B_1, B_2)$ between two detections B_1, B_2 is a value in [0 1] defined by $\frac{B_1 \cap B_2}{B_1 \cup B_2}$. A threshold value of 0.5 has been used to generate new clusters, in all of our experiments. All bounding boxes assigned to a cluster are finally averaged and compressed into a single final detection.

4.5 Integrator

The *tracker* and *object detector* run in parallel with identical objectives. In essence, they defines two experts that compete to object state estimation. While the detector localizes already known templates, the tracker localizes potentially new templates and thus can bring new data for detector. The *Integrator* is the component designed to fuse the tracker and the detector hypothesis to predict object location in the current image. The adopted data fusion procedure is sketched in algorithm 4.5.

The decision is based on the confidence of the tracker hypothesis $\mathbf{B}^t_{tracker}$, and on the detection with maximum confidence $\mathbf{B}^t_{detector}$. If the detector yields a confidence higher than the result from the tracker, then the response of the detector is assigned to the final re-

Algorithm 6 INTEGRATOR Component

```
 1: procedure INTEGRATOR
 2:     Input: $\mathbf{B}^t_{tracker}$, $\mathbf{B}^t_{detector}$
 3:     Output: $\mathbf{B}^t_{final}$, $V^t$
 4:     $\mathbf{B}^t_{final} = \emptyset$
 5:     $V^t = false$
 6:
 7:     if $S^r(B^t_{tracker}, \mathcal{A}^t) < S^r(B^t_{detector}, \mathcal{A}^t)$ then
 8:         $\mathbf{B}^t_{final} = \mathbf{B}^t_{detector}$
 9:         $V^t =$ true
10:     else
11:         $\mathbf{B}^t_{final} = \mathbf{B}^t_{tracker}$
12:         if $S^r(\mathbf{B}^t_{final}) > \theta^+_{NN}$ then
13:             $V^t =$ true
14:         else
15:             if $V^{t-1} \wedge S^r(\mathbf{B}^t_{final}, \mathcal{A}^t) > \theta^-_{NN}$ then
16:                 $V^t =$ true
17:             end if
18:         end if
19:     end if
20: end procedure
```

sult (line 8-10). This corresponds to a re-initialization of the tracker. If the tracker produced a valid result and is not re-initialized by the detector because it is less confident than the tracker, the result of the tracker is assigned to the final result (line 11). In all other cases the final result remains empty (line 4-5), indicating that the object is not visible in the current frame. We use a logical variable V^t to memorize the validity of the final result $\mathbf{B}^t_{final}$ over time. Only if the final result is valid the learning step is performed. The final result is valid under the following two circumstances, both of which assume that the tracker was not re-initialized by the detector. The final result is valid if the tracker produced a result with a confidence value being larger than θ^+_{NN} (line 13).

The final result is also valid if the previous result was valid and the tracker produced a result with a confidence larger than θ^-_{NN} (Line 16).

In all other cases, the final result is not valid. The first bounding box is always valid. The confidence interval $[\theta^-_{NN}\ \theta^+_{NN}]$ defines a range of confidence values to add new samples into the appearance model, as described in algorithm 7. In particular a negative sample P^- is

Algorithm 7 NN-Classifier with confidence interval $[\theta^-_{NN}\ \theta^+_{NN}]$

```
1: procedure NN-Classifier
2:     Input: $\mathcal{A} = \{\mathcal{A}^+,\ \mathcal{A}^-\},\ \mathcal{P} = \{\mathcal{P}^+,\ \mathcal{P}^-\}$
3:     for $P \in \mathcal{P}$ do
4:         $conf = S^r(P, \mathcal{A})$
5:         if $conf < \theta^+_{NN} \wedge P \in \mathcal{P}^+$ then
6:             $\mathcal{A}^+ = \{\mathcal{A}^+,\ P\}$
7:         end if
8:         if $conf > \theta^-_{NN} \wedge P \in \mathcal{P}^-$ then
9:             $\mathcal{A}^- = \{\mathcal{A}^-,\ P\}$
10:        end if
11:    end for
12: end procedure
```

added to $\mathcal{A}$ if $S^r(P^-, \mathcal{A}) > \theta^-_{NN}$, while a positive sample P^+ is added $\mathcal{A}$ if $S^r(P^+, \mathcal{A}) < \theta^+_{NN}$.
This strategy allows a significant reduction of accepted patches at the cost of coarser representation of the decision boundary.

4.6 Experimental Results

We evaluate, quantitatively, *BTLD* using challenging sequences from the *MILBoost dataset*[6]. Each experiment in this section adopts the evaluation protocol proposed in [55].
The performance are evaluated by *Recall* measure, indicating the average percentage of frames for which the overlap between the identified bounding box and the ground-truth bounding box is at least 50%.
The tracker is initialized in the first frame of a sequence and tracks the object of interest up to the end.
Authors in [93], identified in *Coke* and *Faceocc2* the most critical sequences for *TLD*.

The *Coke* sequence (fig. 4.4-**a**,**b**,**c**) proves sensitivity to occlusion and resembling background. The target is affected by several occlusions with the leaf at the beginning of the sequence (figure 4.4-**a**) leading the tracker component to drift (figure 4.4-**b**-**c**). Furthermore, rotated

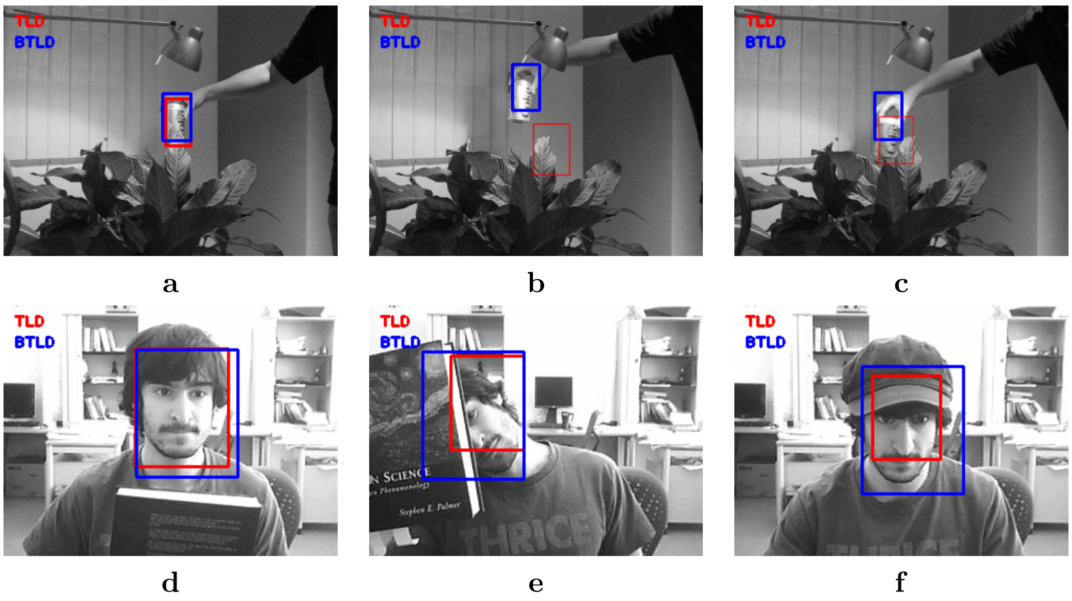


Figure 4.4: From top to bottom: *Coke*,*Faceocc2*. In Red *TLD* estimated object state, in blue *BTLD* estimated object state

version of *Coke* are not learned by the ensemble classifier that employs several frames to redetect that target and restarting the target as showed in figure 4.4-**c**,**d**). Our method, by tracking only stable points, does not lose the target (figures 4.4-**b**,**c**) outperforming the baseline method with a Recall 30% higher.

In sequence *Faceocc2* sensitivity to occlusions and permanent changes of appearance is analyzed, since a man is continuously occluding his face behind a book (fig. 4.4-**e**). Moreover during the sequence the man wears a hat (fig. 4.4-**c**), so that the adaptivity of the tracker to permanent changes of appearance can be evaluated. Reported frames (**d**,**e**,**f**) highlight how *BTLD* produces more accurate detection results since target state estimation is exploited by temporal consistency that controls variation in position and scale over time.

Quantitative results reported in table 4.1 confirm the improvement in accuracy achieved respect to the baseline method.

We evaluated our method also on sequences *Sylvester*,*Faceocc*,

Sequence	*frames*	**PROST** [94]	**OB** [40]	**FTRACK** [1]	**MIL** [6]	**ORF** [92]	**TLD** [55]	**BTLD**
1. *David*	1200	0.8	0.23	0.47	0.70	0.95	1.00	**1.00**
2. *FaceOcc*	820	**1.00**	0.35	**1.00**	0.96	0.70	0.96	**1.00**
3. *Sylvester*	1440	0.73	0.51	0.74	0.93	0.71	0.97	**1.00**
4. *Coke*	292	—	—	—	0.46	0.17	0.60	**0.91**
5. *Tiger1*	353	0.79	0.38	0.20	0.78	0.27	0.88	**0.92**
6. *Tiger2*	364	—	—	—	0.80	0.21	0.85	**0.94**
7. *Dollar*	326	—	—	—	**1.00**	—	0.86	0.93
8. *Girl*	945	0.8	0.24	0.7	47.0	—	0.93	**0.95**
9. *FaceOcc2*	812	0.82	0.75	0.48	0.96	0.82	0.96	**1.0**

Table 4.1: *Recall* measures. The best performance on each video is boldfaced.

Girl, *David*, *Tiger1*, *Tiger2* , *Dollar*. The selected sequences depicts scenarios where the baseline method produces overall good tracking results.
Dollar(fig. 4.5-**a**,**b**,**c**) is a simple but useful sequence to understand the robustness to distractors and the degree of adaptiveness of the algorithms in a very controlled and predictable situation.
The target (fig. 4.5-**a**) suddenly changes appearance(fig. 4.5-**b**). After a while a distractor equal to the original appearance of the target pops out close to the target (4.5-**c**) and then moves next to it. With such scenario the stability of the tracker component is essential to avoid drifting. Our approach outperforms the baseline method up to 7% points, highlighting the correctness of our intuition. However, the *MILBOOST* tracker exhibits higher performances respect to the proposed approach. Indeed, since the object is moving very slow, for few frames (%7) the estimated solution presents an high overlap with the distractor until the motion of the two objects becomes more discriminative. For such reason the *MILBOOST* approach, due to the high stability of its learning component is not affected by this

condition. In contrast, in condition where the appearance variability is high *MILBOOST* slows down its performances, since the it is not able to detect such variations.

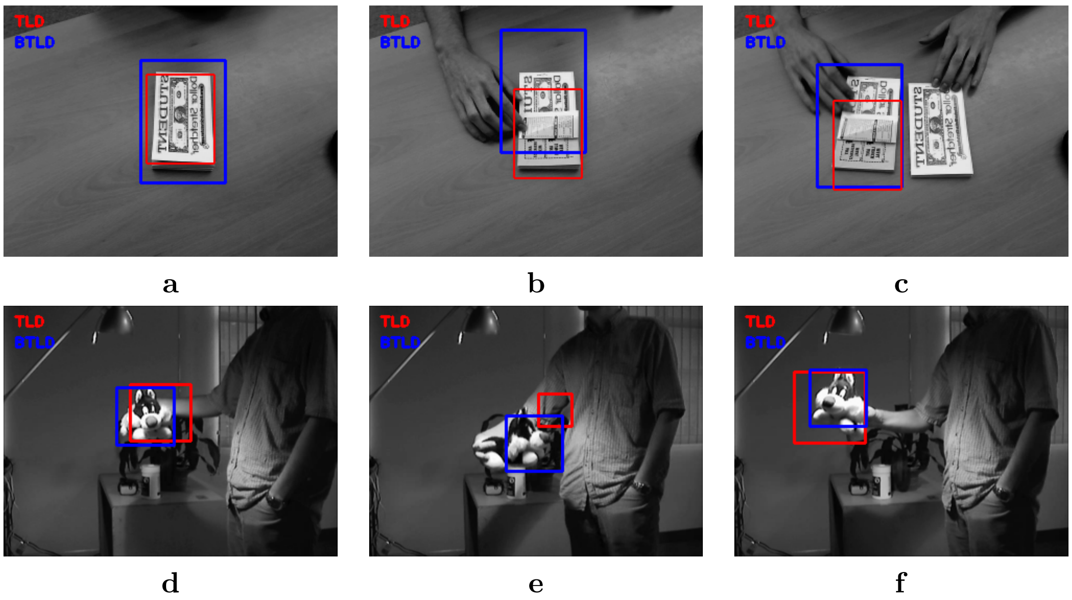


Figure 4.5: From top to bottom: *Dollar*, *Sylv.* In Red *TLD* estimated object state, in blue *BTLD* estimated object state

Sylvester(fig. 4.5-**d**,**e**,**f**) sequence, is a moderately difficult scene, testing target characterized by uniform regions of color that moves with out of plane rotation. In such scenario *TLD* presents several difficulties since median flow tracker is distracted by similar background appearances (fig. 4.5-**e**).
The detector is able to correct the tracker (fig. 4.5-**e**), but time required to re-detect the object reduces overall tracking performances, as shown in table 4.1. Our approach gains 3% on the *TLD* approach reaching the maximum recall value.

Faceocc2 sequence , is a moderately difficult scene, testing face tracking under occlusion. The target is continuously occluding with a book (fig. 4.6 -**b**,**c**), bat once again our approach result not sensitive to such event.

Tiger1 and *Tiger2* depict the same challenging scenario. The main

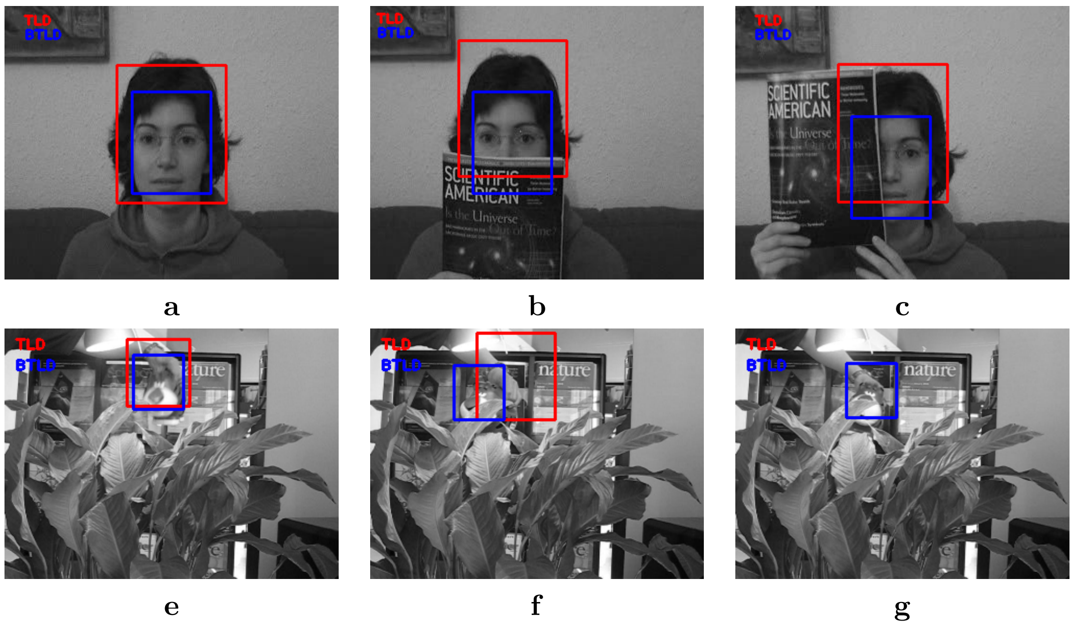


Figure 4.6: From top to bottom: *Faceocc*, *Tiger*. In Red *TLD* estimated object state, in blue *BTLD* estimated object state

difficulties arises from the low size of the target and the continuous occlusions (fig. 4.6 **-f-g**). This conjunction of event distract the *TLD* tracker that lost the target (fig. 4.6-**g**). Also in this scenario our approach is not distracted by occlusions reaching the best performances. *Girl* analyzes tracker capabilities under low quality images(fig. 4.7 **-a-b**). Furthermore, large occlusion and fast appearance changes are event that occurs in this scenario (fig. 4.7 **-a-b**).

David sequence (fig. 4.7 **-d-e-f**) stresses tracker capabilities under camera motion and view point changes. Our approach as well *TLD* tracker localize the target with maximum recall. In summary, results reported in table 4.1 underline the improvement achieved by integrating our component into the *TLD* approach. Furthermore, experiments underline how *BTLD* also affects appearance modeling avoiding the selection of wrong sample during the update of the appearance model.

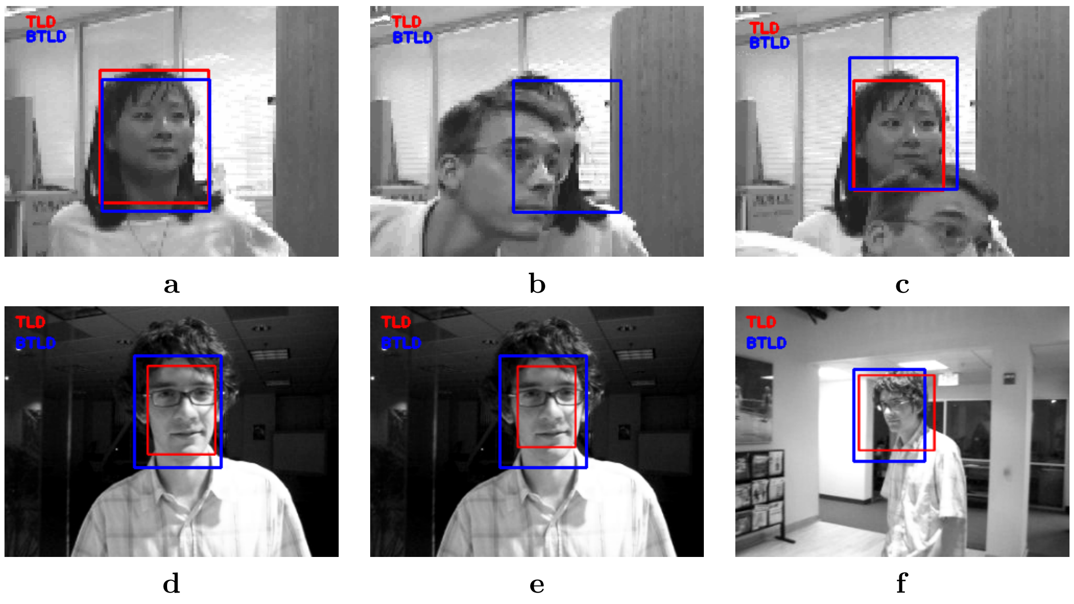


Figure 4.7: From top to bottom: *Girl*, *David*,. In Red *TLD* estimated object state, in blue *BTLD* estimated object state

4.6.1 Computational complexity

An analysis of the computation complexity of each independent component is presented. The computation complexity of the NN-classifier is $\mathcal{O}(m^2|\mathcal{A}|)$ where m^2 is the size of the patch and $|\mathcal{A}|$ is the total number of samples defining the appearance model.
The computation complexity of the tracker component depends on the number of generated samples N_{accept} during the chain evolution and on the size of $\mathcal{A}$, leading to $\mathcal{O}(n_s m^2|\mathcal{A}|)$.
As concern the object detector, the computation time for a sub-window that passes all filtering stages building the cascade is:

- $\mathcal{O}(4)$ for the variance filter adopting the integral image approach that has a pre-computation time of $\mathcal{O}(MN)$.
- $\mathcal{O}(2dK)$ for the ensemble classifier, where d is the number of pixel comparison building the fern feature space and K is the number of ferns building the ensemble.

- $\mathcal{O}(m^2|\mathcal{A}|)$ for the confidence evaluation.

Furthermore, the hierarchical clustering to reduces detections requires $\mathcal{O}(C^2)$ distances evaluations, where C is the number of hypothesis produced by the cascade detector. The final computation complexity is dominated by the tracker, leading to $\mathcal{O}(N_{accept}m^2|\mathcal{A}|)$. In our experiments, a fixed size for $\mathcal{A}$ of 40 samples (20 positives and 20 negatives) has lead to a significant reduction of the computation time without affecting the tracking accuracy. When the number of samples exceeds such values, the oldest samples are replaced, keeping constant the size of $\mathcal{A}$. With this short-term memory mechanism, the system could master the challenge of an unstructured environment as well as moving objects with a rate of 10 fps.

4.6.2 Parameter Selection

Since tuning is a very tedious and time consuming task, here we report the parameters used in our simulations, to obtain results reported in 4.1

- **Appearance Model sample size**. We use 15×15 to represent object and background visual appearance.
- **Nearest Neighboor classifier threshold**. In our experiments, we used $\theta^-_{NN} = 0.5$ and $\theta^+_{NN} = 0.65$ which compromises the accuracy of representation and the speed of growing of the object model. Exact setting of this parameter is not critical.
- **TLD Object Detector parameters**.
  - **Sliding Window**. In sliding-window-based approaches for object detection, sub-images of an input image are tested whether they contain the object of interest Given an image $n \times n$ the number of possible sub-windows grows as n^4. We restrict the search space to a subspace $\mathcal{W}$ by employing the following constraints. First, we assume that the object of interest retains its aspect ratio. Furthermore, we introduce

margins d_x and d_y between two adjacent sub-windows and set d_x and d_y to be 1 of the values of the original bounding box. In order to employ the search on 10 multiple scales, we use a scaling factor $s = 1.2^a, a \in \{-5 \dots 5\}$ for the original bounding box of the object of interest. We also consider sub-windows with a minimum area of 15×15 pixels only. The size of the set of all sub-windows $\mathcal{W}$ constrained in this manner is then:

$$|\mathcal{W}| = \sum_{s \in \{1.2^{-5} \dots 1.2^{5}\}} \frac{n - s(w + d_x)}{sd_x} \frac{m - s(h + d_y)}{sd_y} \tag{4.12}$$

In 4.12 w and h denote the size of the initial bounding box and n and m the width and height of the image. For an initial bounding box of size $w = 80$ and $h = 60$ the number of sub-windows in a VGA image is $146,190$.

  – **Variance threshold**. We adopted a $\Theta_{var} = 10\%$ of the variance exhibited by the object at first frame.
  – **Ferns Feature** . We adopted 13 binary comparison ($S = 13$) to build the fern feature.
  – **Ensamble Detector** . We adopted 10 base classifier ($M = 10$) to build the ensemble classifier and $\Theta_{ensamble} = 0.5$.

- **KLT window size**. We employ window 11×11 to solve the optical flow aperture problem and 2 levels for the image pyramid.

- **Motion Filter**. The 2×2 covariance matrix Σ_v specifying the gaussian kernel employed to estimate the motion distribution $p(V|V^t, \Sigma_v)$ of salient points is:

$$\Sigma_v = 2\mathbb{I}$$

- **Proposal density**. We split the proposal π density into two independent proposal function, namely $\pi_{location}$ and π_{size}. $\pi_{location}$ generates hypothesis for the location (x, y) defining the center of bounding box representing the target according to:

$$\pi_{location} = N(X|X^{t-1}, \Sigma_{location})$$

where:

$$\Sigma_{location} = 500\mathbb{I}$$

π_{size} generates hypothesis for the parameters (w, h) specifying the dimension of the bounding box representing the target according to:

$$\pi_{size} = N(X|X^{t-1}, \Sigma_{location})$$

where:

$$\Sigma_{location} = 10\mathbb{I}$$

- **Motion dynamic**. The motion dynamic $p(X^t|X^{t-1}) = N(X|X^{t-1}, \Sigma_X)$ is linear motion model:

$$\Sigma_X = \begin{bmatrix} 2000\mathbb{I}_{2\times2} & 0 \\ 0 & 250\mathbb{I}_{2\times2} \end{bmatrix},$$

- **Corner Maximum distance** Θ_{corner}. It defines the maximum distance in pixel, that new detected corner should have from the reliable points to be considered as possible changing appearance of the target. We use $\Theta_{corner} = 30$.

- **Markov Chain evolution**. We found experimentally that 1000 accepted samples is good trade off between accuracy in detection and real-time performances requirements.

4.7 Conclusions

In conclusion *BTLD*, defines a novel generative tracker that corrects a systematic drifting behavior revealed in the short term tracker component building the *TLD* approach. The novelty of such generative tracker relies on its capabilities of solving in joint fashion feature selection and re-sampling exploiting the global adaptive appearance model as outlier removal. A real-time implementation of the *MCMC* particle filter framework has been described in detail and an extensive set of experiments was performed in order to highlight the ability of our approach to increase robustness of *TLD* tracker.

4.8 Related Publications

- Giorgio Gemignani, Wongun Choi, Alessio Ferone, Alfredo Petrosino, Silvio Savarese. *A Bayesian Approach to Tracking Learning Detection.* ICIAP 2013. Lecture Notes in Computer Science Volume 8156, 2013, pp 803-812

Chapter 5

Application of BTLD in an Ambient Assistent Living scenario

In the last years, ceiling mounted camera, due to its limited invasiveness and cost-effective installation and management, has become an emerging solution to enable people localization capabilities in Ambient Assisted Living (*AAL*) systems, a specific user oriented application of the ambient intelligence (*AMI*) discipline. However humans detection and tracking, exploiting only visual informations is still a challenging problem due to the high variability exhibited by humans appearance observed from a top view [79, 95, 91, 98, 116].
In this chapter we present the *BTLD* approach extended with motion detection for tracking humans in ceiling mounted camera video stream. In 5.1 we introduce the reasons motivating the proposed extension; In 5.2 the complete system architecture is described analyzing in detail all the building blocks of the extended system; in 5.2.1, we introduce the motion detection component; in subsection 5.2.2 we describe the proposed human appearance model; in 5.2.3 and 5.2.4 we introduce the tracker and the detector components respectively. In 5.3 experimental results and computational complexity of the approach are presented, followed by a conclusion.

5.1 Motivation

Despite its high performances, *BTLD* is not suitable to control complex deformations of target appearance observed from a ceiling mounted static camera for several reasons. The most critical problems come from the state representation and appearance descriptor. Indeed, representing the target state as a rectangle and adopting a similarity measure based on *ncc* easily produce low confidence values for new hypothesis generated by both the tracker and the detector. Moreover, the high density of corners into the surrounding background leads the tracker to drift as depicted in figure 5.1. As the target

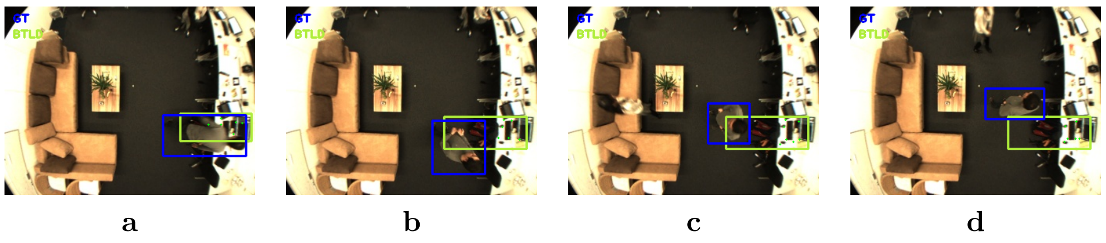


Figure 5.1: BTLD failure *video1* from *KSERA Dataset* [116]

moves (figure 5.1 b), erroneous feature points lying inside the target estimated state, are selected to grow the set of reliable points. After several frames the tracker drifts and the modeling component learns wrong samples (fig. 5.1 c-d).

To overcame this limitation, since the camera is static, we integrated a motion detection component based on robust background subtraction [72] providing a new source of information able to stabilize the performance of each component building the *BTLD* architecture. In essence, the motion detector classifies image regions in moving (the *foreground*) and static (the *background*) as showed in figure 5.2. In particular, the tracker component exploits foreground regions to handle the selection of new salient point of the target. At each tracking iteration, new points of interest detected in proximity of the target are added to the set $\mathcal{K}_{new}^{t+1}$ only if they belong to a foreground

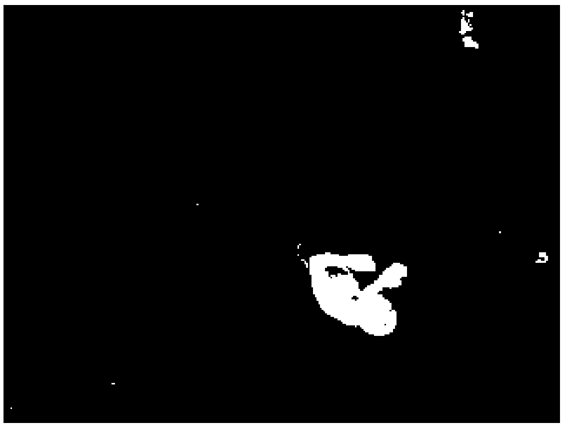


Figure 5.2: The background subtraction allows to detect moving elements (white regions) and static elements (black regions) in the current frame.

region. In other words, the motion detector defines a new filtering stage that rejects corner points belonging to static image regions and accept corner points belonging to moving regions. With this strategy, the growing-pruning approach building the *BTLD* approach is less sensitive to the wrong sample selection problem. The analysis of foreground image regions, has been incorporated also in to the cascade object detector as a new preliminary filtering stage . Furthermore, human appearance representation has been modeled by a color histogram representation, extracted from moving regions detected by robust background subtraction [72]. In such way a more discriminative representation of the target appearance is available and described limitations are avoided, stabilizing performances on real-time human detection and tracking.

5.2 Proposed Architecture

The complete architecture of our tracking system is now built on five interacting components: the *tracker*, the *object detector*, the *motion detector*, *learning component* and *integrator* as depicted in figure 5.3.

The *motion detector* maintaining a background appearance model

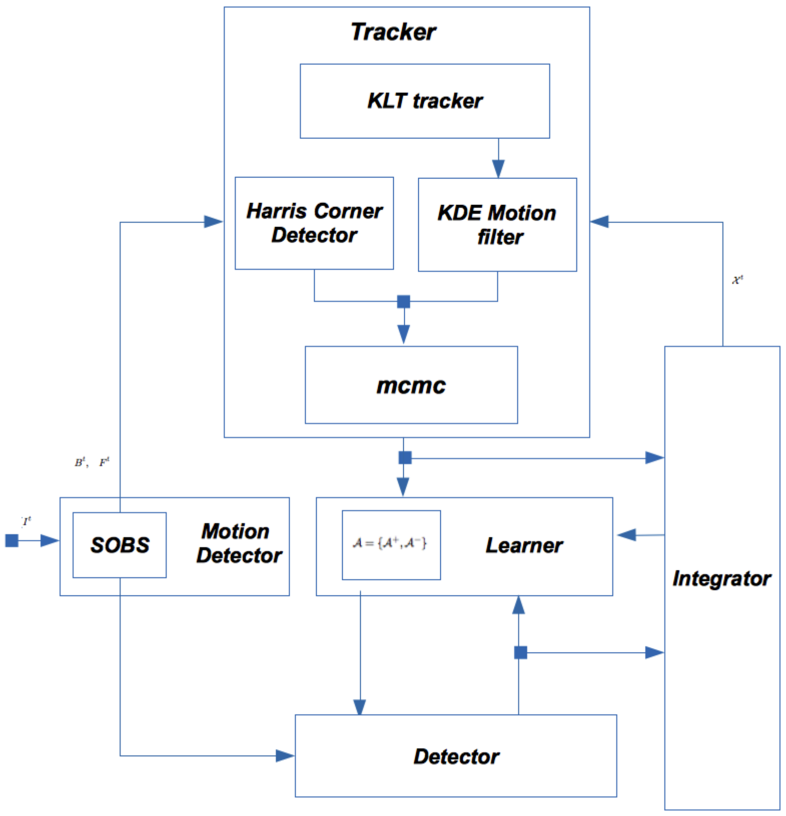


Figure 5.3: The proposed system architecture.

B^t over time, detects moving regions into the scene employing robust background subtraction. Both the *tracker* and *object detector* exploit motion classification to support target localization.
The *tracker* has been re-designed to solve the short term tracking problem including moving regions analysis to constraint the selection of new salient points. Indeed, only foreground regions are allowed to generate new candidate local feature points. The *cascade object detector* performs a full scanning analysis of the image in order to localize all appearances that are similar to the ones that have been observed and learned in the past. It is still built on three filtering stages to rejects as many non relevant sub-windows with a minimal amount of computation. We replaced the first filtering stage of

the cascade with a background filter that reject static regions and all moving regions having density of foreground pixels lower than a threshold θ_{fg}. The *integrator component* continues to perform the data fusion task as described in sec. 3.5. The *learning component* exploits a color histogram based representation, to build the Nearest Neighbor classifier able to discriminate among the target and the surrounding background. With the collaborative contribution of each component, the tracking performance are improved significantly.

5.2.1 Motion detector component

We approach motion detection by the Self-Organizing Background Subtraction (*SOBS*) algorithm [72]. It is based on the neural background model automatically generated by a self-organizing method, without prior knowledge about the involved patterns. Such adaptive model have demonstrated good capabilities in handling scenes containing moving backgrounds, gradual illumination variations and camouflage, can include into the background model shadows cast by moving objects, and achieves robust detection for different types of videos taken with stationary cameras.
In essence, the background model is built on a self-organizing neural network arranged as a 2-D flat grid of neurons. Each neuron computes a function of the weighted linear combination of incoming inputs, with weights resembling the neural network learning, and can be therefore represented by a weight vector obtained collecting the weights related to incoming links. An incoming pattern is mapped to the neuron whose set of weight vectors is most similar to the pattern, and weight vectors in a neighborhood of such node are updated; such learning of the neuronal map allows to adapt the background model to scene modifications. Specifically, each pixel $\mathbf{x}$ is modeled by a neuronal map consisting of $n \times n$ weight vectors $b_i^0(\mathbf{x}), i = 1, \ldots, n^2$ where each weight vector is a 3D vector initialized to the color components of the corresponding pixel of the first sequence frame I^0.
The complete set of weight vectors for all pixels of an image I^0 with

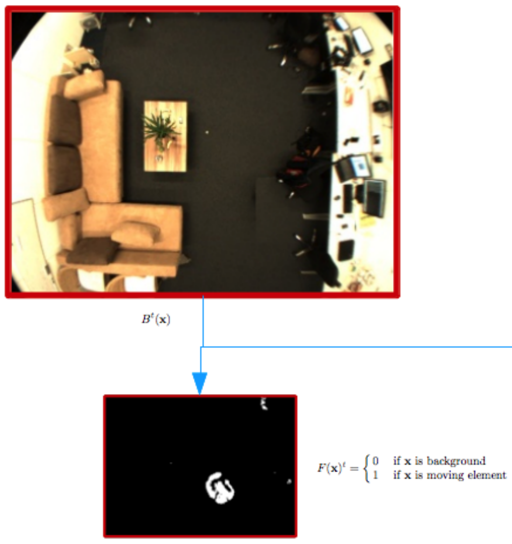

$B^t(\mathbf{x})$
$I^t(\mathbf{x})$
$F(\mathbf{x})^t = \begin{cases} 0 & \text{if } \mathbf{x} \text{ is background} \\ 1 & \text{if } \mathbf{x} \text{ is moving element} \end{cases}$

Figure 5.4: The motion detection approach based on the *SOBS* algorithm: the incoming image $I^t(x)$ is compared with the background model $B^t(x)$ in order to generate the foreground image $F^t(x)$

N rows and M columns is represented as a neuronal map B^0 with $n \times N$ rows and $n \times M$ columns, where adjacent blocks of $n \times n$ weight vectors correspond to adjacent pixels in image I^0. The value $n = 3$, suggested and justified in [72], is adopted for all experiments reported in 5.3. For each frame I^t, the color $I^t(\mathbf{x})$, at position $\mathbf{x}$ is compared to the weight vectors $b_1^{t-1}(\mathbf{x}), \ldots, b_{n^2}^{t-1}(\mathbf{x})$ related to it in the model B^{t-1}, to determine the weight vector $b_{BM}^{t-1}(\mathbf{x})$ that best matches it according to a metric $d(\cdot)$:

$$d(b_{BM}^{t-1}(\mathbf{x}), I^t(\mathbf{x})) = \min_{i=1,\ldots,n^2} d(b_i^{t-1}(\mathbf{x}), I^t(\mathbf{x})). \tag{5.1}$$

The best matching weight vector is used as the pixel's encoding approximation, and therefore $\mathbf{x}$ is detected as foreground if the distance in (5.1) exceeds a threshold ϵ; otherwise, it is classified as background as depicted in figure 5.4. Exploiting such classification a foreground mask F^t, is set according to:

$$F^t(\mathbf{x}) = \begin{cases} 0 & \text{if background} \\ 1 & \text{if moving element} \end{cases}$$

According to [72] we adopted the HSV color space and the Euclidean distance as distance metric. Learning is able to adapt the background model B^{t-1} to scene modifications and is achieved by updating the best matching weight vector $b_{BM}^{t-1}(\mathbf{x})$, supposed at position $\mathbf{z}$ of B^{t-1}, and all other weight vectors in its neighborhood $N_\mathbf{z}$ according to:

$$b^t(\mathbf{y}) = (1 - \alpha^t(\mathbf{y}, \mathbf{z}))b^{t-1}(\mathbf{y}) + \alpha^t(\mathbf{y}, \mathbf{z})I^t(\mathbf{x}), \quad \forall \mathbf{y} \in N_\mathbf{z} \tag{5.2}$$

Here $\alpha(\mathbf{y, z}) = \gamma_t \cdot G(\mathbf{y\text{-}z})$, where γ_t represents the learning rate, that depends from scene variability, while $G(\cdot) = G(\cdot; 0, \sigma^2)$ are the values of a Gaussian filter with zero mean and σ^2 variance in $N_\mathbf{z}$.

5.2.2 Appearance Modeling Component

Human appearance is modeled by a dynamic data structure that maintains the object appearance and its surroundings observed so far. To this aim, we define a collection of patch histograms $\mathcal{A}^t = \{\mathcal{A}^+, \mathcal{A}^-\}$, where $\mathcal{A}^+ = \{\mathcal{A}_1^+ \dots \mathcal{A}_n^+\}$ and $\mathcal{A}^- = \{\mathcal{A}_1^- \dots \mathcal{A}_m^-\}$ denotes the sets of object and background patches histogram respectively.
A measure $S(A_i, A_j)$ based on normalized cross correlation (ncc) is defined to compare patches histograms A_i, A_j, according to the following expression:

$$S(A_i, A_j) = \frac{1}{2}(ncc(A_i, A_j) + 1) \tag{5.3}$$

Such measure produces values in the range [0 , 1], assuming 0 when A_i is equal to A_j and 1 when they are not. In order to evaluate the degree of similarity of an arbitrary patch histogram H to the object appearances encoded in $\mathcal{A}^+$, we define three similarity measures:

1. **Similarity with the positive nearest neighbor** defines the distance of a candidate histogram A and its nearest positive

sample $\mathcal{A}_i^+ \in \mathcal{A}^+$:

$$S^+(H, \mathcal{A}^+) = \min_{A_i \in \mathcal{A}^+} S(H, \mathcal{A}_i^+) \tag{5.4}$$

2. **Similarity with the negative nearest neighbor** defines the distance of a candidate histogram A and its nearest negative sample $\mathcal{A}_i^- \in \mathcal{A}^-$:

$$S^-(H, \mathcal{A}^-) = \min_{A_i \in \mathcal{A}^-} S(H, \mathcal{A}_i^-) \tag{5.5}$$

3. **Relative similarity** S^r defines the confidence that the patch histogram A belongs to the object appearance model $\mathcal{A}^+$.

$$S^r(H, \mathcal{A}) = \frac{S^-(H, \mathcal{A}^-)}{S^+(H, \mathcal{A}^+) + S^-(H, \mathcal{A}^-)} \tag{5.6}$$

The relative similarity is employed to build a Nearest Neighbor (NN) classifier that discriminates target's samples from the background as depicted in figure 5.5. A candidate histogram H is classified as pos-

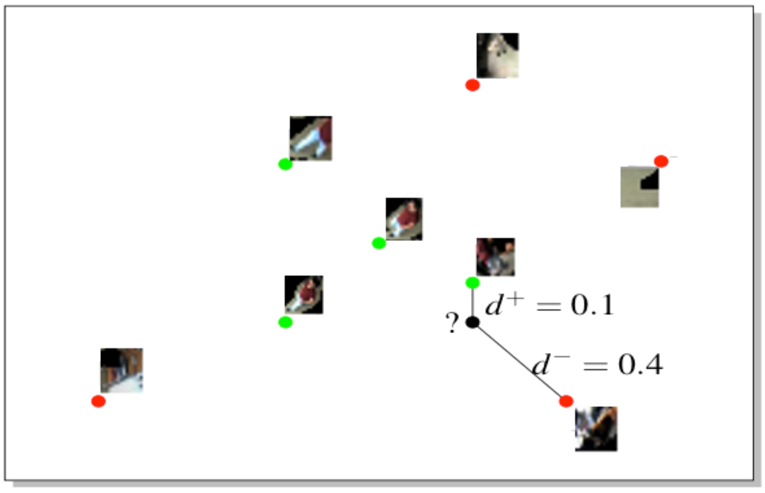


Figure 5.5: Histogram Feature Space and NN-classifier. In green target feature samples. In red, background feature samples.

itive if $S^r(H, \mathcal{A}) \leq \theta^{NN}$, where the parameters θ_{NN} enables tuning the nearest neighbor classifier either towards precision or recall. A new unlabeled patch histogram is added to $\mathcal{A}$ only if the NN-classifier prediction differs from the final label assigned by the tracker or detector results. This leads to a significant reduction of accepted patches at the cost of a coarser representation of the decision boundary. We introduce a confidence interval $[\theta^-_{NN}, \theta^+_{NN}]$ representing the range of confidence values to add new samples into the appearance model. In particular a negative sample H is added to $\mathcal{A}^-$ if $S^r(H, \mathcal{A}) > \theta^-_{NN}$, while a positive sample is added $\mathcal{A}^+$ if $S^r(H, \mathcal{A}) < \theta^+_{NN}$.
In our experiments, we used $\theta^+_{NN} = 0.5$ and $\theta^+_{NN} = 0.65$ which compromises the accuracy of representation and the speed of growing of the object model.

5.2.3 Tracker Component

Following the approach prosed in sec. 4.2 the tracker component solves the frame to frame correspondences by a growing pruning approach. The growing procedure is designed to guarantee a sufficient number of such salient points at each tracking iteration.
In essence, it tries to repopulate the set $\mathcal{K}^t$ with new corners detected in proximity of the reliable points and unreliable points that were removed by the motion filter. This is the most critical stage, since there is no prior knowledge about the "nature" of new detected salient regions. The spatial distance respect to the reliable points coming from the past and global appearance similarity are the only available information used to infer a decision on the state of such points.
By exploiting motion detection information, the selection of new salient points is constrained to moving regions identified by robust background subtraction(white regions in fig. 5.6-c).

In other words, motion detection allow to discard local feature points that lies into the target bounding box, but belong to the surrounding background.

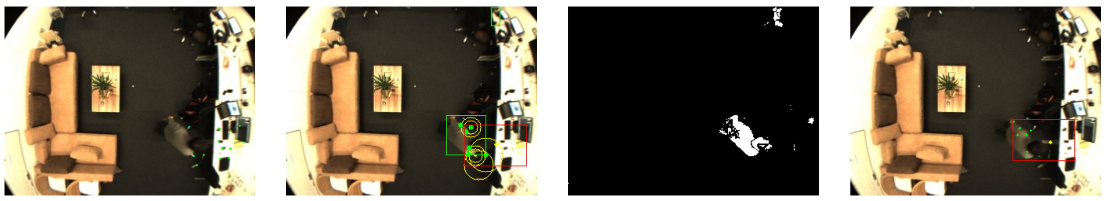


Figure 5.6: Intermediate results while tracking *Video1* from *KSERA Dataset* [116]. **a)** The set of tracked point from the frame 49, $\mathcal{K}^{50}$. **b)** The set $\mathcal{K}^{50}_{new}$ of new detected point on the frame 50. In yellow the new detected points, in green the reliable tracked points. **c)** The foreground mask produced by the motion detector component. **d)** The final set $\mathcal{K}^{50}$ after *mcmc filtering*.

The *BTLD* tracker extended with motion detection is sketched in algorithm 8.

Algorithm 8 Tracker component

```
procedure $BTLD$ Tracker
    Input: $\mathcal{K}^t$, $B^t$, $F^t$
    Output: $X^{t+1}$

    $Initialize\ \mathcal{K}^0 = HarrisCornerDetector(B^0)$

    while $t < N_{frames}$ do

        $\mathcal{K}^t = KLT_{tracker}(\mathcal{K}^{t-1})$

        $\mathcal{V} = |\mathcal{K}^t - \mathcal{K}^t)|$

        $[p(V|V^t)] = KernelDensity(\mathcal{V})$

        $[\mathcal{K}^t_r\ ,\ \mathcal{K}^t_u] = MotionFilter(V_{mode})$

        $[\mathcal{K}^t_{new}] = HarrisCornerDetector(F^t, \theta_{corner})$

        $[X^t,\ \mathcal{K}^t] = mcmc(\mathcal{K}^t_r,\ \mathcal{K}^t_u,\ F^t)$

    end while
end procedure
```

At the t tracking iteration, the points correspondences $\mathcal{K}^t$ and $\mathcal{K}^{t+1}$

are computed by *KLT* method (line 8). The set of motion vectors $\mathcal{V}^t = \{|K_i^t - K_i^{t-1}|, i = 1 \dots |K_i^t|\}$ associated with the tracked points are filtered according to the motion filter described in 4.2 and the the sets of reliable points $\mathcal{K}_r^t$ and unreliable points $\mathcal{K}_u^t$ are populated (lines $10 - 14$).
To capture appearance variability, new corners in proximity of the reliable points are detected on the foreground mask calculated by the background and are maintained in $\mathcal{K}_{new}^t$ (line 16).
A proximity threshold θ_{corner}, controls the maximum distance allowed to consider new detected corner as a possible change in object appearance that has not been yet observed or a background region that was not visible before, because occluded by the target.
The three sets are then refined by our *mcmc* procedure that automatically search for the best state $\mathcal{X}^t = [x\ ,\ y\ ,\ w\ ,\ h]$ that jointly:

- contains the maximum number of reliable tracked points $\mathcal{K}_r^t$.
- enclose a visual content that maximizes the visual similarity S^r to the object appearance model defined in 5.6.
- contains an high number of foreground pixels.

In such way, new detected points $\mathcal{K}_{new}^t$ and $\mathcal{K}_u^t$ unreliable tracked points are automatically re-evaluated and re-inserted in $\mathcal{K}^t$ if they belongs to a state that exhibits high similarity to the global appearance model.
Following the Sequential Bayesian formulation, the posterior probability of target state X^t a time t is given by:

$$\underbrace{p(X^t|\mathcal{O}^t)}_{posterior} \approx \underbrace{p(\mathcal{O}^t|X^t)}_{a} \int \underbrace{p(X^t|X^{t-1})}_{b}\underbrace{p(X^{t-1}|\mathcal{O}^{t-1})}_{c}\, dX^{t-1} \tag{5.7}$$

where (a), (b) and (c) in Eq. 5.7 represent the *observation likelihood*, the *motion model* and the *posterior* from previous time, respectively. The hidden state $X^t = [\ x^t,\ y^t,\ w^t,\ h^t]$ is encoded by location (x^t, y^t) and size (w^t, h^t) information of the 2D box enclosing the target, resulting in a 4D state space $\mathcal{X}$.

$\mathcal{O}^t = [\mathcal{K}^t\ , \mathcal{A}^t\ , \mathcal{F}^t]$ defines the observation space, where $\mathcal{K}^t = \{\mathbf{K}_i^t \in \mathcal{R}^2\}$ is the set of local points tracked from previous frame, $\mathcal{A}^t$ represent the adaptive global appearance model learned on-line by $BTLD$ and $\mathcal{F}^t$ is the foreground mask produced by the background subtraction at time t.
Assuming $\mathcal{K}^t$, $\mathcal{F}^t$ and $\mathcal{A}^t$ independent, the *observation likelihood* $\mathcal{O}$ is factorized as:

$$p(\mathcal{A}^t, \mathcal{K}_r^t, \mathcal{F}^t|X^t) = p(\mathcal{A}^t|X^t)p(\mathcal{K}_r^t|X^t)p(\mathcal{F}^t|X^t) \tag{5.8}$$

Observation likelihood of $\mathcal{K}_r^t$ measures the fraction of reliable local feature points lying inside the candidate target state X^t:

$$p(\mathcal{K}_r^t|X^t) = \frac{\mathbf{K_i}^t\ \in \mathbf{X}^t}{|\mathcal{K}_r^t|}. \tag{5.9}$$

Such distribution promotes candidate states containing the maximum number of tracked local features, assuming that they are free of errors. KLT failures, are automatically rejected by the global appearance likelihood modeled by $BTLD$. It assign low confidence to hypothesis containing local tracked points and not resembling target appearance, assuming an “outliers-rejection” role similar to $RANSAC$. The visual likelihood, measuring the visual similarity of a candidate state respect to the target’s appearance model, is directly given by *relative similarity* defined in equation 5.6, i.e. $p(\mathcal{A}^t|X^t) = S^r(H, \mathcal{A})$. Observation likelihood of $\mathcal{F}^t$ measures the density of foreground pixels $\mathbf{x} = (x, y)$ lying inside the box defined by candidate target state X^t:

$$p(\mathcal{F}^t|X^t) = \frac{1}{|BB(X^t)|} \sum_{\forall x \in BOX(X^t)} F^t(\mathbf{x}) \tag{5.10}$$

where $BB(X^t)$ is the bounding box defined by the hypothesis X^t. Such factor promotes candidate states containing an high number of foreground pixels increasing the localization accuracy.
We employ a linear *dynamic model* defined by a gaussian distribution over $\mathcal{X}$, centered on previous target X^{t-1} location and scale, leading

to $p(X^t|X^{t-1}) = N(X^{t-1}, \sigma I)$. Considering the complexity of the given probabilistic formulation, it is extremely challenging to design an analytical inference method for estimating the *MAP* solution:

$$\hat{X}^t = \underset{X^t \in \mathcal{X}}{\operatorname{argmax}}\, p(X^t|\mathcal{O}^t) \tag{5.11}$$

This challenge is due to the presence of the high nonlinearity of observation likelihood functions. We propose to employ a sampling based sequential filtering technique based on the *MCMC* particle filter. At each time step t, we approximate the posterior by a number N of samples:

$$p(X^{t-1}|\mathcal{O}^{t-1}) \approx \{X_s^{t-1}\}_{s=1}^N. \tag{5.12}$$

Propagating samples through the *motion model*, we generate particles for the *predictive distribution* $p(X^t|X^{t-1}, \mathcal{O}^{t-1})$ and approximate the posterior distribution at time t by Monte Carlo integration:

$$p(X^t|\mathcal{O}^t) \propto p(\mathcal{A}^t, \mathcal{K}_r^t, \mathcal{F}^t|X^t) \sum_{s=1}^{N} p(X^t|X_s^{t-1}) \tag{5.13}$$

Approximation in eq. 5.13 is achieved by a Markov chain over the joint space of $\mathcal{X}$ that converges over the posterior distribution $p(X^t|\mathcal{O}^t)$.
For *MCMC* sampling to be successful, it is critical to have a good proposal distribution which can explore the hypothesis space efficiently. Our *proposal distribution* generates separate random hypothesis for location ($\mathcal{X}_1$) and scale ($\mathcal{X}_2$) subspaces, according to normal deviates from previous accepted hypothesis.
The whole *Metropolis-Hasting* procedure with component-wise updating scheme is sketched in algorithm 9.

Algorithm 9 MCMC Particle Filter

```
procedure MCMC PARTICLE FILTER
    Input: $\mathcal{A}^t,\ \mathcal{K}^t,\ \mathcal{F}^t,\ X_i^{t-1}$
    Output: $p(X^t|\mathcal{X}^t, \mathcal{K}^t)$

    Initialize $X_0^t = X^{t-1} = [x, y, w, h]^{t-1} = \{X_l X_s\}^{t-1}$

    while $i < N_{accept}$ do
        Select uniformly the candidate component $j \in [12]$

        Propose $X_j^* \sim N(X_j^i|X_j^{i-1})$

        Build the hypothesis $X^* = \{X_j\ ,\ X_{k \neq j}\}$

        Evaluate the acceptance probability $\alpha = min(1, \frac{p(X_s^*|\mathcal{A}^t, \mathcal{K}^t, \mathcal{F}^t)}{p(X_s^{i-1}|\mathcal{A}^t, \mathcal{K}^t)})$

        Accept $X_s^* \rightarrow X_s^{i+1}$ if $\alpha < u \leftarrow$ uniform sample $\in [0\ 1]$
    end while
end procedure
```

Once the sampling method has reached convergence, the maximum a posterior estimate for X^t is analyzed by the *integrator* that establishes the final solution.

5.2.4 Detector component

While the recursive tracker performs target localization based on a smoth variation of estimated location and scale in the previous frame, the object detector employs an exhaustive search at multiple scales based on a sliding-window approach [59]. In such way, re-detection capabilities and recognition of changes in target's motion dynamic are enabled in the whole system. Since several thousands of sub-windows should be tested independently whether it contains the object of interest, a cascaded object detector based on four stages is employed to reject as many non relevant sub-windows with a minimal amount of computation. The complete schema of the this component is depicted in fig. 5.7. The motion detection result provides useful informations to reduce the search space, since it restrict to moving

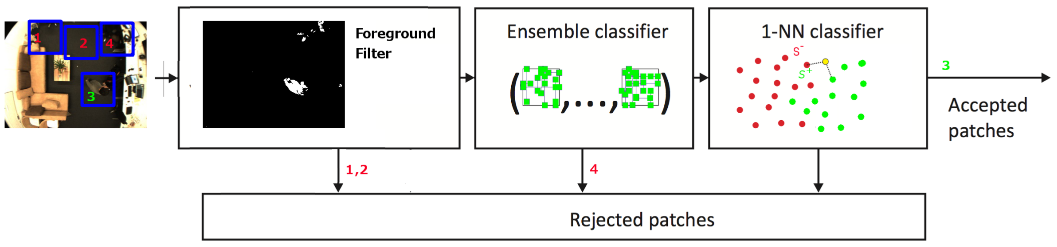


Figure 5.7: Integration of the motion detection into the cascade object detector. 1-2) candidate patches rejected by the foreground filter. 3) a positive patch detected by the ensemble classifier. 4) the true negative patch detected by the ensemble classifier

regions all the other stages building the cascade object detector as depicted in fig. 5.7.
In the second stage all the sub-windows that exhibit a density of foreground pixels lower than a certain threshold Θ_{fg} are rejected. Such a foreground filter is able to rapidly reject small foreground regions (the box 1 in fig. 5.7) but is unable to distinguish between different well-structured objects (boxes $2, 3, 4$ in fig. 5.7).
The second stage comprises an ensemble of base classifiers exploiting *ferns features* as described in sec. 3.4. Within the third stage, the detection provided by ensemble classifier are evaluated by the nearest neighbor classifier building the appearance model. All relevant detections are clustered as described in sec. 4.4.

5.3 Experiments and Results

We quantitatively evaluate our approach using challenging sequences from the *KSERA* [116] and *BOMNI* [27] dataset.
The *BOMNI* datasets collects sequences at resolution 640×480 registered in a laboratory and was created to evaluate single-target tracking as well multi-target tracking systems.
The *KSERA* dataset proposes video streams at resolution 320×240

collected by the ambient assistant living system and was created to analyze single-target tracker proposed in [116].
Both the datasets depicts common conditions of daily life where the tracker should be able to perform its task. In table 5.1 we summarize the experimental conditions that have been simulated in each dataset, in order to evaluate tracker capabilities in handling common situation of every day life.

Each experiment in this section adopts the evaluation protocol pro-

Table 5.1: Simulated conditions in the *KSERA* dataset

		Moving	*Sitting*	*Changing light*	*Night condition*	*Distracter person*	*Similar dress*
KSERA	*video1*	☑	☑	☐	☐	☑	☑
	video2	☑	☐	☑	☑	☐	☐
	video3	☑	☑	☐	☐	☑	☐
	video4	☑	☑	☐	☐	☑	☐
BOMNI	*top-1312975204642*	☑	☑	☐	☐	☐	☐
	top-1313674683690	☑	☑	☑	☐	☐	☐
	top-1313677458790	☑	☑	☑	☐	☐	☐
	top-1313678291554	☑	☑	☐	☐	☑	☑

posed in [116]:

1. The background model is initialized on the first frame of the sequence.
2. the tracker and the detector component are initialized as soon as the target has entered in the room.

The tracking process is performed till the end of the sequence. If the target leave the room, the detector component will be able to relocalize it. In videos *top-1313678291554* ground-truth are available for all individuals allowing to test each people individually. Results are reported in table 5.2 where *person1, person2* and *person3* refer to each individual test. The camera frames are sub-sampled to the resolution 320 × 240, to uniform the test conditions. The experimental results reported in table 5.2, have been evaluated according to the multiple object tracking accuracy metric (*MOTA*) [10]. Such

Table 5.2: *MOTA* accuracy. The best performance on each video is boldfaced.

Sequence	*frames*	[116]	**BTLD**
1. *video1*	1564	0.94	**1.00**
2. *video2*	160	0.96	**0.98**
4. *video3*	700	0.88	**0.91**
3. *video4*	1190	**0.91**	0.88
5. *top-1312975204642*	1000	–	**0.98**
6. *top-1313674683690*	793	–	**0.97**
7. *top-1313677458790*	906	–	**0.92**
8. *top-1313678291554 person1*	431	–	**0.97**
9. *top-1313678291554 person2*	431	–	**0.94**
10. *top-1313678291554 person3*	431	–	**0.90**

metric has been proposed to measure the performances of a multiple object tracking system but is also adopted to evaluate single target trackers.
Since only a single person is tracked in the system, based on our goal design,at each iteration we count the number of misses m_t, the false positives fp_t and the mismatches mme_t.
The *MOTA* is then calculated according to:

$$MOTA = 1 - \frac{\sum_t m_t + fp_t + mme_t}{\sum_t gt_t} \tag{5.14}$$

This measure accounts for all object configuration errors made by the tracker, false positives, misses, mismatches, over all frames. It provides a very intuitive measure of the tracker's performance at detecting objects and keeping their trajectories, independent of the precision with which the object locations are estimated. The threshold distance of a false positive was defined as 40 pixels; 4 videos made available by authors of [116], have been evaluated.
On average 90% of the images are tracked correctly. Each test condition is analyzed in detail in the following subsections.

5.3.1 Tracking a moving person

Tracking a moving person in the room (fig. 5.8 a-b-d-e), is the common task for all the analyzed sequences. After a person has entered

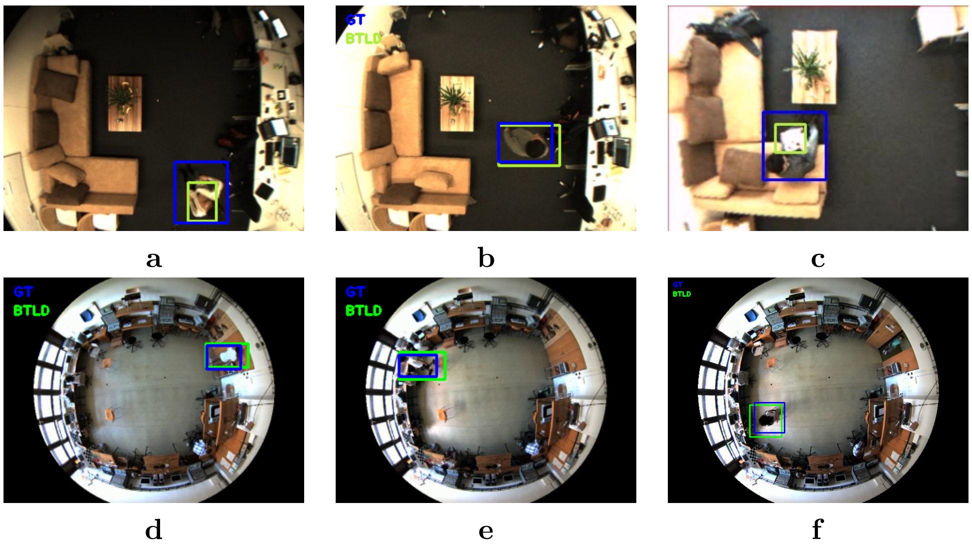


Figure 5.8: *Moving and sitting person in KSERA* and *BOMNI* dataset. In green the estimated target location, in blue the ground truth box. **(a)** *video1-KSERA* frame 78. **(b)** *video2-KSERA* frame 273. **(c)** *video4-KSERA* frame 734. **(d)** *top-1313677458790-BOMNI* frame 392. **(e)** *top-1313674683690-BOMNI* frame 273. **(f)** *top-1313678291554-BOMNI* frame 112.

in the scene, he can sit in a position for a long time, for example while watching television, reading a book or talking with other persons. In this case, is essential that the motion detector component classifies the stopped object as a static region without updating the background model. In the *KSERA* dataset, the person sits down on the sofa (fig. 5.8-c) while in the *BOMNI* dataset several chairs have been placed in the room to simulate such condition (fig. 5.8-f).
In this scenario color and motion information are feasible to easily track a person. In test *video4*, due to the slow velocity of the target and its overlap with a jacket having similar color, the accuracy of the predicted solution decreases. However, as soon as the motion becomes consistent the tracker increase its accuracy. In this particular

condition, the approach proposed in [116], exploiting multiple visual cues for target localization, achieves more accuracy (%3) at the cost of an higher computational cost.

5.3.2 Tracking in changing light conditions

Sensitivity to changing light condition is an essential feature for a tracking system based on motion detection. Light variation generated by shadows are easily detected by the motion detection component in all the tested sequences(*video4*, *top-1313674683690*, *top-1313677458790*). Obtained results indicates that the slight change of light under sufficient sunshine does not disturb the tracking system at all. Furthermore the *KSERA* dataset propose a challenging sequence (*video2*) where the lights have been switched off after the person is localized. Due to the dramatic change of the intensity, both motion detector and ensemble classifier lose the target person (fig. 5.9-b). However, as soon as the ambient becomes more lighted,

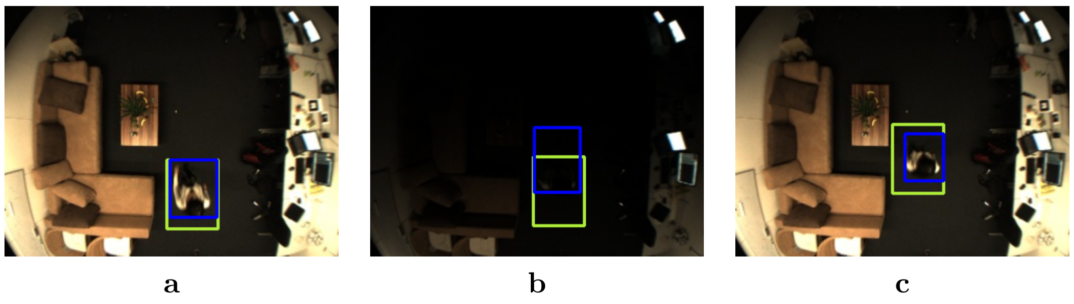


Figure 5.9: *Change light condition in night scenario(video2 -KSERA)*. In green the estimated target location, in blue the ground truth box. **(a)** frame 92. **(b)** frame 97.**(c)** frame 101.

the detector component re-localize the the target and the tracker is able to continue the tracking task leading to a (98%) mota accuracy. Compared to [116] we achieve more %2 points accuracy indicating less sensitivity to changes of light conditions.

5.3.3 Tracking interacting persons

Another important scenario proposed in both the datasets is the presence of multiple people interacting with the target. While in the *KSERA* dataset, three people are simultaneously in the scene with a limited amount of time interaction, in the *BOMNI* dataset four people are interacting in to the room for a long time, making more challenging the tracking task (fig. 5.10-b).

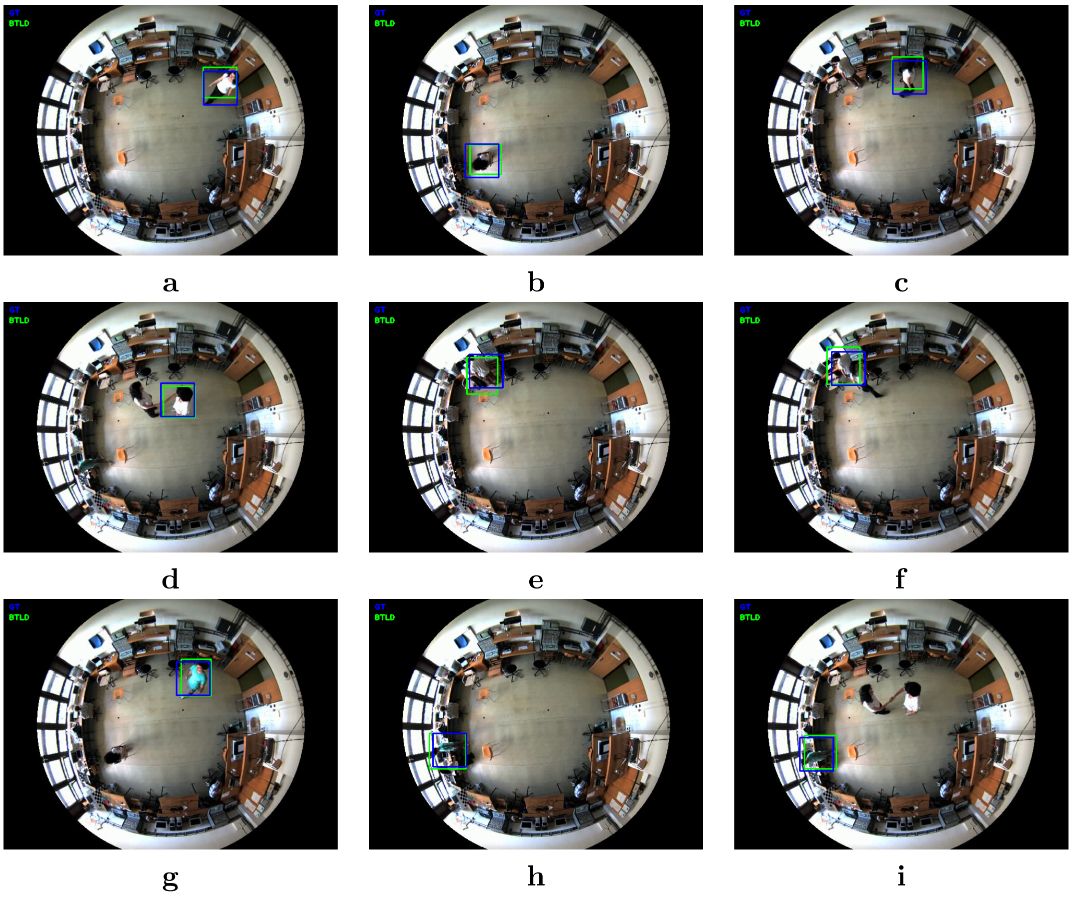


Figure 5.10: Sequence *top-1313678291554-BOMNI* dataset. Tracking *person1*:**(a)** frame 66, **(b)** frame 109, **(c)** frame 341. Tracking *person2*:**(d)** frame 238, **(e)** frame 299, **(f)** frame 333. Tracking *person3*:**(g)** frame 128, **(h)** frame 177, **(i)** frame 227

Reported results highlight how motion information combined with color features is sufficient to avoid drift when the target interacts with another person moving into the scene.

5.3.4 Computational complexity

The computational complexity of the proposed approach is given taking into account *BTLD* extended computational complexity and the *SOBS* computational complexity. The former complexity is the same as *BTLD* described in 4.6.1, since each component presents the same complexity of the analogous into the baseline approach. The *SOBS* is both in terms of space and time, is $\mathcal{O}(n^2MN)$ where n^2 is the number of weight vectors used to model each pixel and $M \times N$ is the image dimension. Such requirements are not a problem for our system since we used the *gpu* based implementation of the algorithm, we proposed in [37], that enables real-time processing up to 70 fps. The final computational complexity is dominated by the tracker, leading to $\mathcal{O}(n_s m^2 |\mathcal{A}|)$. In experiments, a fixed size for $\mathcal{A}$ of 40 samples (20 positives and 20 negatives) has lead to a significant reduction of the computation time without affecting the tracking accuracy. When the number of samples exceeds such values, the oldest samples are replaced, keeping constant the size of $\mathcal{A}$. With this short-term memory mechanism, the system could master the challenge of an unstructured environment as well as moving objects in a real ambient intelligent system with a rate of 7 fps.

5.4 Conclusion

We have presented an extension of the *BTLD* approach for indoor person detection and tracking in an *AMI* environment with a ceiling mounted camera. The system works in real-time and it is able to track a moving person in the scene learning an appearance model. Experimental results have been performed on indoor video-sequences with different illumination conditions and with different number of moving people. The system reaches a percentage of correct people detection ranging from 90% to 98% depending on test conditions.

5.5 Related Publications

- Giorgio Gemignani,Lucia Maddalena, Alfredo Petrosino. *Real-time stopped object detection by neural dual background modeling.* Proceedings of the 2010 conference on Parallel processing, pages 357-364.
- Giorgio Gemignani, Alessio Ferone, Alfredo Petrosino. *Tracking, Learning and Detection of Humans in ceiling mounted camera video streams.* Submitted to the Journal of Ambient Intelligence and Humanized Computing

Chapter 6

Tracking Learning Detection by parts

The critical point of the *BTLD* approach, is the *growing-pruning* approach employed to select new points of interest characterizing the changing appearance of the tracked object. This filtering procedure is completely unsupervised, since new detected points are directly validated during the inference process defining the state estimation of the generative tracker.
This approach, overcomes the *reinitialization strategy* error of the median flow tracker, but raises an interesting question on how detect and validate new local visual features corresponding to target's changing appearance. *BTLD* tackles this problem by exploiting global similarity measure and spatial distances respect to the points in track from the previous frame. With $BTLD^+$, we propose a novel tracker that controls within a supervised approach the discovery and selection of new unlabeled data during tracking.
Section 6.1 motivates the proposed approach; section 6.2 introduces an overview of the proposed framework; subsection 6.3 and 6.4 analyze the learning component and its integration into the *BTLD* framework, respectively. section 6.5 shows experimental results.

6.1 Motivation

A critical problem characterizing the *BTLD* approach relies on the selection of local feature points to guide the state estimation procedure encoded into the *MCMC* particle filter.
According to fast appearance changes, the number of reliable points reduces drastically over time. The growing-pruning approach described in section 4.2 has been designed to guarantee a sufficient number of such salient points at each tracking iteration. In essence, it tries to repopulate the set $\mathcal{K}^t$ with new corners detected in proximity of the reliable points and unreliable points that were removed by the motion filter.
This is the most critical stage, since there is no prior knowledge about the "nature" of new detected salient regions. Their spatial distance respect to the reliable points tracked from the past and the global appearance similarity are the only available information to infer a decision on the state of such points. The effectiveness of such assumption has been proved in short term tracking rigid objects, but reveals limitations while tracking deformable objects. Fur such type of objects, feature points could follow different motion patterns over time, leading to a consistent loss of points and the subsequent selection background feature points.
To overcome such limitation, inspired by feature-based tracking approaches [44, 42], we train on-line a classifier able to establish correspondences for local feature points that disappear when the target changes its pose.
The main idea here, is that the periodic structure of the data is exploited to automatically infer a labeling , which is then used to control the prediction of a tracker.

6.2 Proposed Approach

With $BTLD^{+}$, we extended the $BTLD$ local feature selection strategy into a supervised context. We exploit supervised learning to select new salient regions of interest describing target appearance during tracking. The complete $BTLD+$ architecture is depicted in figure 6.1. We introduced a feature pool that maintains salient point

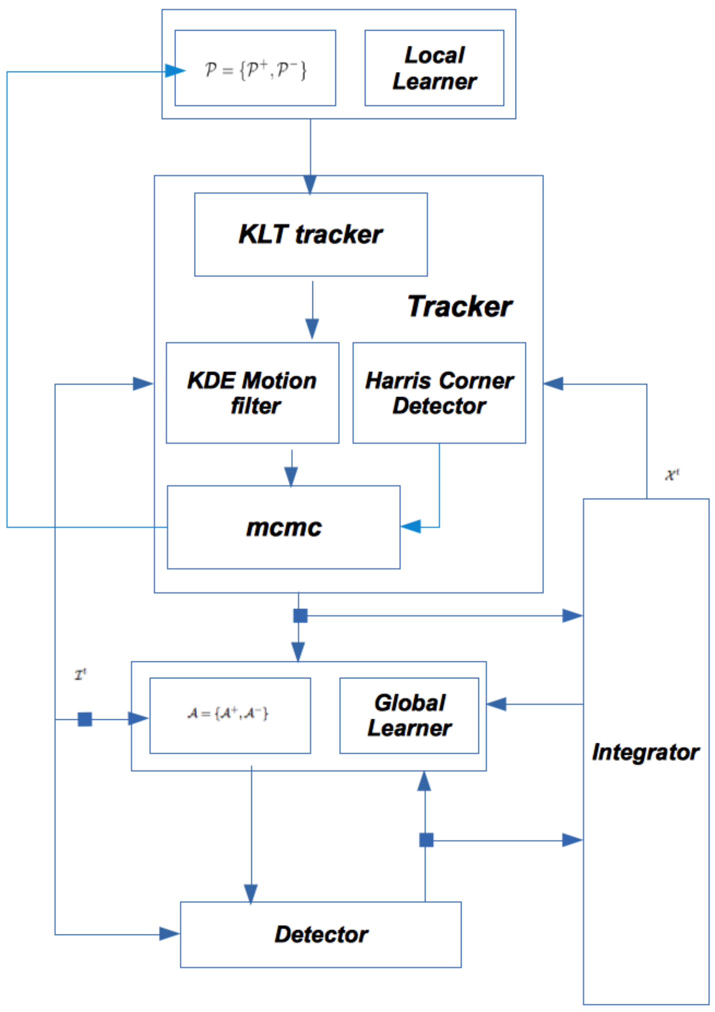


Figure 6.1: $BTLDP^{+}$ architecture.

belonging to the target and the surrounding background, corresponding to Harris operator local maxima.
In such way, we are able to build a nearest neighbor classifier, referred as the *Part classifier*, able to discriminate among such local feature points. This feature enables *memory capabilities* into tracker component building the $BTLD$ framework, since it becomes able in recognizing if a subregion of the target has been observed before.
Moreover, negative visual features bring new useful information that makes the target state estimation more robust to occlusion and false positive detections. Indeed, this information is explicitly integrated in our probabilistic framework as a geometrical constraint on the position of the target. In other words, we exploit background feature points to localize positions in the image where the tracked object can not be. Similar approaches have been proposed in [42, 20], where the concept of *supporters* has been introduced. Basically, *supporters* points define a subset of highly discriminative regions of the image which are useful to predict the target object position.
They at least temporarily move in a way which is statistically related to the motion of the target points (*green points* in fig. 6.2). A supporter can be very discriminative respect to an object feature. At the same time it can provide helpful information to predict the position of the target.
In [20] a single supporter, defined by a single "companion" region close to the target, improves object tracking. However, the companion has a $2D$ affine relationship to the target limiting the flexibility of the tracker in controlling non planar motion.
In [42] supporters are defined by local patches extracted around local maximum of *Harris* operator. No constraint on their motion and location are assumed. Such points are tracked independently over time. A model consisting of the supporters, continuously estimate, update, and refine the model which predicts the position of the target. With such flexibility, the tracker can re-detect the target under occlusion.

In contrast to the aforementioned approaches, we consider supporters as salient points having strong correlation to the object motion and scattered around the target estimated bounding box. Furthermore we assume that the class of such points point is not completely determined.
The key idea beyond our *supporters* definition is to consider them as possible new salient regions capturing a change of target appearance that has not been yet observed. In figure 6.2, we show a clarifying example of our definition of *supporters*.
The yellow points define regions that are located around the current

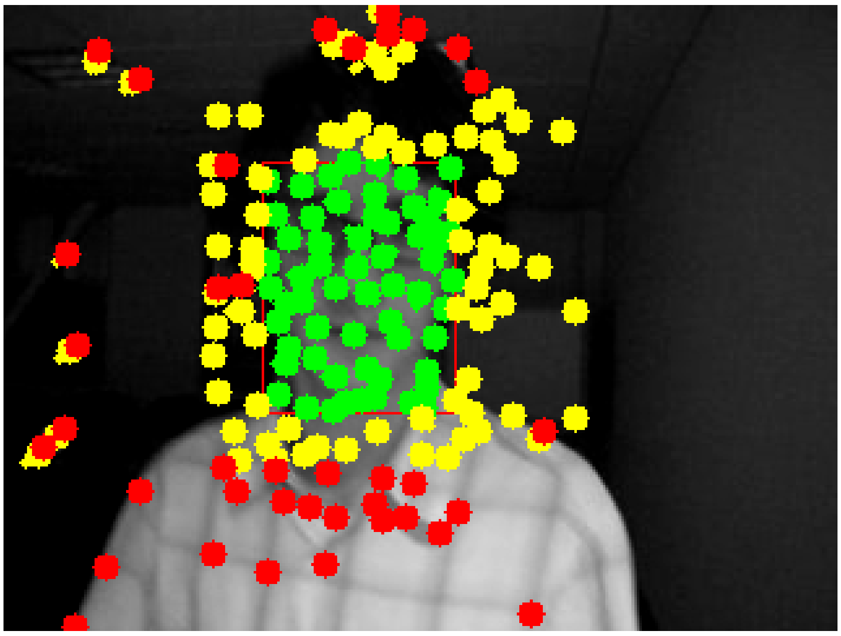


Figure 6.2: Object parts in *green*, *Supporters regions* in yellow *Bacground points* in red.

estimated target state.
Due to appearance changes, tracking failures and imprecise localization *supporters* can be regions of the target or region of the background. Another goal of our algorithm is to discover the nature of the *supporters* while tracking the object of interest.
At each tracking iteration, the set of *supporters* referred to as $\mathcal{K}_s$ is updated by two refining event, namely the growing and pruning.

Within the growing event new interest points $\mathcal{K}^t$ detected around the previous estimated target box are added to set $\mathcal{K}_s^t$ of supporters. The initial state of such samples is defined according to the *part classifier* score. Label switching are possible in order to counteract the problem of wrong sample selection, generated by an not accurate estimated solution.
In contrast the pruning event is designed to remove points from $\mathcal{K}_s^t$ and assign them to the set of positive samples $\mathcal{K}_p^t$ or negative samples $\mathcal{K}_n^t$ building the feature pool. The whole procedure is sketched in algorithm 10.

A the tracking iteration t, each point belonging to the sets $\mathcal{K}_p^t$,

Algorithm 10 $BTLD^+$ Tracker

```
1: procedure $BTLD^+$TRACKER
2:     Input: $\mathcal{K}_p^t,\ \mathcal{K}_s^t,\ \mathcal{K}_n^t,\ B^t,\ I^t,\ I^{t+1}$
3:     Output: $B^{t+1}$
4:
5:     Initialize $\mathcal{K}^0 = CornerDetector(B^0)$
6:     while $t < N_{frames}$ do
7:         %Track Points
8:         $\mathcal{K}_p^{t+1} = KLT(\mathcal{K}_p^t)$
9:         $\mathcal{K}_n^{t+1} = KLT(\mathcal{K}_n^t)$
10:        $\mathcal{K}_s^{t+1} = KLT(\mathcal{K}_s^t)$
11:        %Build Motion Vectors
12:        $\mathcal{V}_p = |\mathcal{K}_p^{t+1} - \mathcal{K}_p^t)|$
13:        $\mathcal{V}_n = |\mathcal{K}_n^{t+1} - \mathcal{K}_n^t)|$
14:        $\mathcal{V}_s = |\mathcal{K}_s^{t+1} - \mathcal{K}_s^t)|$
15:        % Kernel Density
16:        $[p(V|V_p)] = KernelDensity(\mathcal{V}_p)$
17:        %Select Reliable points
18:        $[\mathcal{K}_r^{t+1},\ \mathcal{K}_u^{t+1}] = MotionFilter(p(V|V_p), \mathcal{V}_p, \mathcal{V}_n, \mathcal{V}_s)$
19:        %Detect new regions of interest
20:        $[\mathcal{K}_r^{t+1},\ \mathcal{K}_u^{t+1}] = CornerDetector(B^{t+1})$
21:        %Target State Estimation
22:        $B^{t+1} = mcmc(\mathcal{K}_r^{t+1}\mathcal{K}_u^{t+1})$
23:    end while
24: end procedure
```

$\mathcal{K}_s^t$ and $\mathcal{K}_n^t$ is tracked independently by *KLT* (lines 9-11). We denote $\mathcal{K}_p^{t+1}, \mathcal{K}_s^{t+1}, \mathcal{K}_n^{t+1}$ the set of corresponding points, found by *KLT*

method.
A refinement stage based on motion analysis is employed to remove *KLT* failures. We estimate by gaussian kernel density estimation the motion distribution $p(V|V^t)$ of the positive tracked points, as described in 4.2 (lines 13-19).
Furthermore, assuming a motion consistency among target points, we build two sets $\mathcal{K}_r^{t+1}$, $\mathcal{K}_u^{t+1}$ containing reliable points and of unreliable points respectively (line 22).
A point is reliable if it is undergoing the same motion defined by the peak of the motion distribution $p(V|V^t)$; Furthermore, supporter points observing such motion are inserted in $\mathcal{K}_r^t$.
$\mathcal{K}_u^t$ will contain all the rejected *supporter* points and the negative points. New detected corners according to the classification results are added to $\mathcal{K}_r^{t+1}$ if they have strong confidence or to $\mathcal{K}_u^{t+1}$ (line 25).
This sets allows the introduction of new geometrical constraints, that we encode in our novel *MCMC* particle filter. Such constraints express the following concepts:

- Unreliable tracked points $\mathcal{K}_u^{t+1}$ cannot belong to the target object appearance.
- Reliable tracked points $\mathcal{K}_r^{t+1}$ should belong to the target object appearance as many as possible.

This geometrical constraints are directly encoded in a novel *MCMC* particle filter, where we introduce two competitive likelihood functions: one promotes the addiction of reliable tracked points in $\mathcal{K}_r^{t+1}$, the other discard image regions containing unreliable tracked points in $\mathcal{K}_u^{t+1}$.
As within *BTLD*, the visual likelihood evaluates the visual similarity that a proposed state exhibits respect to the target global appearance model.

6.3 Parts Appearance Model

Inspired by [57, 90, 55, 6] our learning model implement an incremental growing and pruning approach.
While tracking the object, we learn both new positive and negative parts of the target and the background are added to the training pool $\mathcal{P}$.
The whole appearance model is defined as a dynamic data structure that maintains the objectś parts appearances and its surroundings observed so far.
We define it as a collection of positive and negative patches:

$$\mathcal{P} = \{\mathcal{P}^+_{0,0}, \mathcal{P}^+_{i,0}, \dots \mathcal{P}^+_{n,t}\ , \mathcal{P}^-_{1,t} \dots \mathcal{P}^-_{m,t}\}$$

where:

- $\mathcal{P}^+$ denotes the object local patches extracted around *Harris local maxima*
- $\mathcal{P}^-$ denotes the background local patches.

Both positive and negative patches are ordered according to the insertion time into the collection so that $\mathcal{P}^+_{0,0}$ is the first positive part added to the collection while $\mathcal{P}^+_{n,t}$ is the last positive patch added to the set at time t.
The fundamental problem in learning is the choice of the new positive sample: we must learn a new sample only if the object appearance is undergoing changes.
In such way we avoid saturating the training pool with duplicated sample and add valuable information to the learning component.
We extended the *relative similarity* measure defined in equation3.5 to measure Part's similarity respect to $\mathcal{P}$ as follow:

1. **Similarity with the positive nearest neighbor Part**:

$$S^+(P, \mathcal{P}^+) = \min_{P_i \in \mathcal{P}^+} S(P, \mathcal{P}^+_i) \tag{6.1}$$

It defines the distance of the patch P respect to the nearest positive sample stored in the appearance model.

2. **Similarity with the negative nearest neighbor Part**:

$$S^-(P,\mathcal{P}^-) = \min_{P_i \in \mathcal{P}^-} S(P,\mathcal{P}_i^-) \tag{6.2}$$

   It defines the distance of the patch P respect to the nearest negative sample stored in the appearance model.

3. **Relative similarity** S^r .

$$S^r(P,\mathcal{P}) = \frac{S^-(P,\mathcal{P}^-)}{S^+(P,\mathcal{P}^+) + S^-(P,\mathcal{P}^-)} \tag{6.3}$$

   Such measure ranges from 0 to 1 and can be interpreted as the probability that the visual appearance encoded in the patch P belongs to the part based visual model.

The classifier building the part based object model is built, following the strategy described in 4.5. A negative part P^- is added to $\mathcal{P}$ if $S^r(P^-,\mathcal{P}) > \theta^-_{NNpart}$, while a positive part P^+ is added to $\mathcal{P}$ if $S^r(P^+,\mathcal{P}) < \theta^+_{NNpart}$.
Introducing a part based representation, the complete appearance models becomes a couple layered model that exploit both global appearance and local appearance appearance, as depicted in figure 6.3.

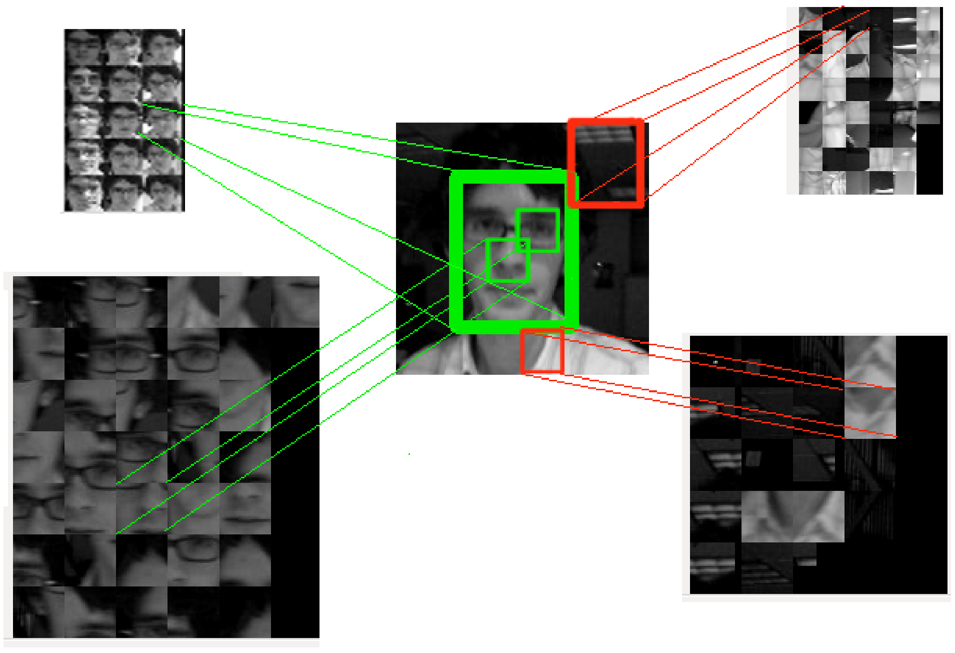


Figure 6.3: The coupled layered appearance model. In red negative samples encoded into the Global Appearance Model (big box) and the Part based Appearance Model. In green positive samples encoded into the Global Appearance Model (big box) and the Part based Appearance Model

6.4 Bayesian Formulation

Following the Sequential Bayesian formulation, the posterior probability of target state X^t a time t is given by

$$\underbrace{p(X^t|\mathcal{O}^t)}_{posterior} \approx \underbrace{p(\mathcal{O}^t|X^t)}_{a} \int \underbrace{p(X^t|X^{t-1})}_{b} \underbrace{p(X^{t-1}|\mathcal{O}^{t-1})}_{c} \, dX^{t-1} \tag{6.4}$$

where (a), (b) and (c) represent the *observation likelihood*, the *motion model* and the *posterior* from previous time, respectively.
The hidden state $X^t = [\ x^t,\ y^t,\ w^t,\ h^t]$ is encoded by location (x^t, y^t) and size (w^t, h^t) information of the 2D box enclosing the target, resulting in a 4D state space $\mathcal{X}$.

$\mathcal{O}^t = [\mathcal{K}_r^t \ , \mathcal{K}_p^t \ , \mathcal{A}^t]$ represents the measurement space, where:

- $\mathcal{A}^t$ is the adaptive global appearance model learned on-line by *BTLD.*

- $\mathcal{K}_r^t = \{\mathbf{K}_i^t \in \mathcal{R}^2 | R(V_i^t, V_{max}) \leq 1\}$ is the set of reliable local points.

- $\mathcal{K}_u^t = \{\mathbf{K}_i^t \in \mathcal{R}^2 | R(V_i^t, V_{max}) > 1\}$ is the set of unreliable local tracked points.

As previously described V_i^t represent the motion flow associated to a tracked point, while V_{max} defines the mode of the motion distribution estimated for $\mathcal{K}_p^t$.
Assuming $\mathcal{K}_r^t$, $\mathcal{K}_u^t$ and $\mathcal{A}^t$ independent, the *observation likelihood* $\mathcal{O}$ is factorized as:

$$p(\mathcal{A}^t, \mathcal{K}^t | X^t) = p(\mathcal{A}^t | X^t) p(\mathcal{K}_u^t | X^t) p(\mathcal{K}_r^t | X^t) \tag{6.5}$$

Reliable points likelihood of $\mathcal{K}_r^t$ measures the fraction of local feature points lying inside the candidate target state X^t:

$$p(\mathcal{K}_r^t | X^t) = \frac{\mathbf{K}_r^t \ \in \mathbf{X}^t}{|\mathcal{K}^t|} \tag{6.6}$$

Such distribution promotes candidate states containing the maximum number of reliable tracked local features, assuming that they are free of errors.
In practice, such component try to enlarge the box enclosing the target in order to find the best similar patch containing the maximum number of reliable points.
In contrast, unreliable points likelihood of $\mathcal{K}_u^t$ measures the fraction of unreliable tracked points lying inside the candidate target state X^t:

$$p(\mathcal{K}_u^t | X^t) = 1 - \frac{\mathbf{K}_u^t \ \in \mathbf{X}^t}{|\mathcal{K}^t|} \tag{6.7}$$

Such likelihood component is designed to give high confidence to candidate states containing the minimum number of unreliable tracked local features. In practice, this component try to reduce the box enclosing the target in order to find the best similar patch containing the minimum number of unreliable points.
The global visual likelihood $p(\mathcal{A}^t|X^t)$ as in *BTLD* evaluates the visual content of an hypothesis X^t. It is once again derived by *relative similarity* defined in equation 6.3.
We assume a linear *dynamic model* modeled by a Gaussian distribution over $\mathcal{X}$, centered on previous target X^{t-1} location and scale.
Considering the high nonlinearity of observation likelihood functions, it is extremely challenging to design an analytical inference method for estimating the *MAP* solution:

$$\hat{X}^t = \operatorname*{argmax}_{X^t \in \mathcal{X}} p(X^t|\mathcal{A}^t, \mathcal{K}^t) \tag{6.8}$$

We propose a sampling based sequential filtering technique based on the *MCMC* particle filter.
At each time step t, we approximate the posterior by a number N of samples:

$$p(X^{t-1}|\mathcal{O}^{t-1}) \approx \{X_s^{t-1}\}_{s=1}^{N}. \tag{6.9}$$

Propagating samples through the *motion model*, we generate particles for the *predictive distribution* and approximate the posterior distribution at time t by Monte Carlo integration:

$$p(X^t|\mathcal{A}^t, \mathcal{K}^t) \propto \tag{6.10}$$

$$p(\mathcal{A}^t|X^t)p(\mathcal{K}^t|X^t)\sum_{s=1}^{N} p(X^t|X_s^{t-1})p(X_s^{t-1}|\mathcal{A}^{t-1}, \mathcal{K}^{t-1})$$

Approximation in eq. 6.10 is achieved by a Markov chain over the joint space of $\mathcal{X}$ that converges over the posterior distribution $p(X^t|\mathcal{A}^t, \mathcal{K}^t)$.
The whole *Metropolis-Hasting* procedure is sketched in algorithm 11.

Algorithm 11 MCMC Particle Filter

```
1: procedure MCMC Particle Filter
2:     Input: $\mathcal{K}^t$, $\mathcal{A}^t$, $X_i^{t-1}$
3:     Output: $p(X^t|\mathcal{A}^t, \mathcal{K}^t)$
4:
5:     Initialize $X_0^t = X^{t-1} = \{xywh\}^{t-1} = \{X_l X_s\}^{t-1}$
6:
7:     while $i < N_{accept}$ do
8:         Select uniformly the candidate component $j \in [1 , 2]$
9:
10:        Propose $X_j^* \sim N(X_j^i|X_j^{i-1})$
11:
12:        Build the hypothesis $X^* = \{X_j , X_{k \neq j}\}$
13:
14:        Evaluate the acceptance probability $\alpha = min(1, \frac{p(X_s^*|\mathcal{A}^t,\mathcal{K}^t)}{p(X_s^{i-1}|\mathcal{A}^t,\mathcal{K}^t)})$
15:
16:        Accept $X_s^* \to X_s^{i+1}$ if $\alpha < u \leftarrow$ uniform sample $\in [0 , 1]$
17:    end while
18: end procedure
```

For *MCMC* sampling to be successful, it is critical to have a good proposal distribution which can explore the hypothesis space efficiently. Our *proposal distribution* generates separate random hypothesis for location ($\mathcal{X}_1$) and scale ($\mathcal{X}_2$) subspaces, according to normal deviates from previous accepted hypothesis.
Once the sampling method has reached convergence, the maximum a posterior estimate for X^t is analyzed by the *TLD integrator* that establishes the final solution and provides to perform the update of both the appearance models.

6.5 Experimental Results

We evaluate, quantitatively, $BTLD^+$ on *MILBoost dataset*, adopting the same evaluation protocol described in section 4.6. Results reported in table 6.1 underlines an improvement of the baseline method on each tested sequence.

The previous experiments show that $BTLD^+$ performs well on

Sequence	*frames*	**PROST** [94]	**OB** [40]	**FTRACK** [1]	**MIL** [6]	**ORF** [92]	**TLD** [55]	**BTLD**	*BTLD*$^+$
1. *David*	1200	0.8	0.23	0.47	0.70	0.95	1.00	**1.00**	**1.00**
2. *FaceOcc*	820	**1.00**	0.35	**1.00**	0.96	0.70	0.96	**1.00**	**1.00**
3. *Sylvester*	1440	0.73	0.51	0.74	0.93	0.71	0.97	**1.00**	**1.00**
4. *Coke*	292	—	—	—	0.46	0.17	0.60	0.91	**1.00**
5. *Tiger1*	353	0.79	0.38	0.20	0.78	0.27	0.88	0.92	**1.00**
6. *Tiger2*	364	—	—	—	0.80	0.21	0.85	0.94	**1.00**
7. *Dollar*	326	—	—	—	**1.00**	—	0.86	0.93	**0.96**
8. *Girl*	945	0.8	0.24	0.7	47.0	—	0.93	0.95	**1.00**
9. *FaceOcc2*	812	0.82	0.75	0.48	0.96	0.82	0.96	**1.0**	**1.00**

Table 6.1: *recall* measures. The best performance on each video is boldfaced.

benchmark sequences where the *recall* is in the range $90 - 100$. The *Dollar* sequence presents the same problem discussed in section 4.6, even if the time elapsed to filter out wrong feature points has reduced leading to an improvement of %3 points of accuracy.

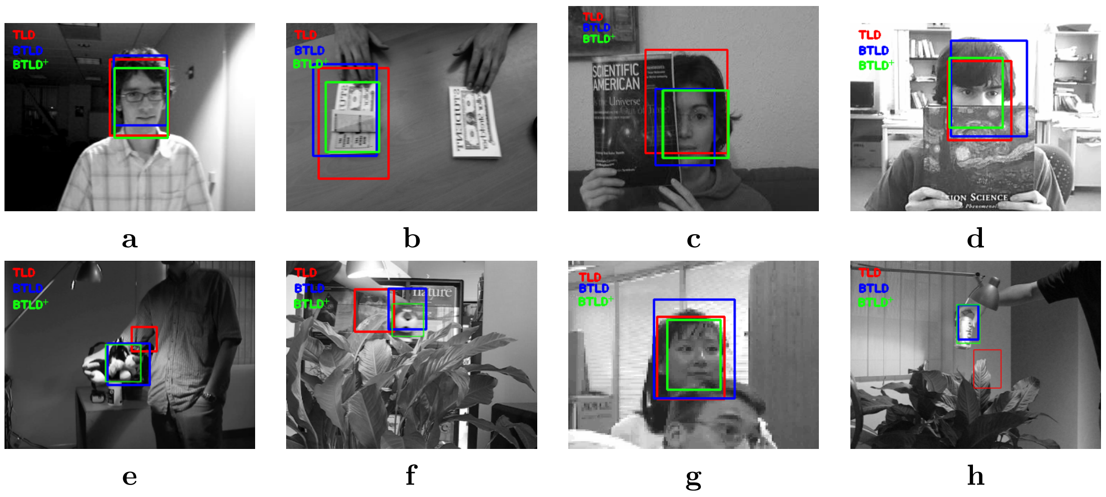


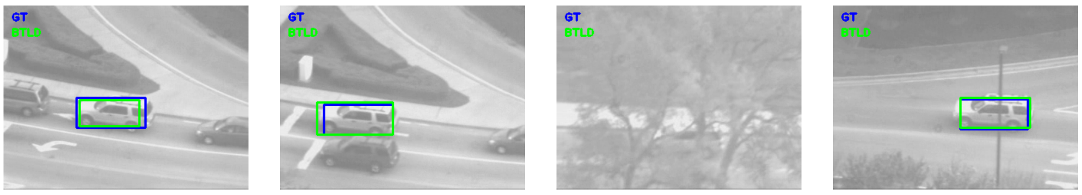
Figure 6.4: From top to bottom: *David, Dollar, Faceocc, Faceocc2, Sylv, Tiger, Girl, Coke11,* . In Red *TLD* estimated object state, in blue *BTLD* estimated object state,in green *BTLD*$^+$ estimated object state

We consider these sequences saturated. New challenging scenarios have been taken from the *TLD DATASET* [55]. We selected different sequences in order to analyze our tracker performances on long sequences, fast motion generating motion blur and out of plane rotations. In particular, we analyzed the sequences *Car*,*Panda*, *Carchase*, *Jumping* whose critical condition are summarized in table 6.2.

Sequence	*frames*	*Fast Motion*	*Full Occlusion*	*Partial Occlusion*	*Motion Blurring*	*Out of plane rotation*	*Illumination Changes*
Car	945	☐	☑	☑	☐	☐	☐
Panda	3000	☐	☑	☑	☐	☑	☑
Carchase	9928	☑	☑	☑	☑	☐	☑
Jumping	313	☑	☐	☑	☑	☐	☐

Table 6.2: Simulated conditions in the *TLD* dataset

We compared our approach to *TLD* ([55]),*OB* ([40]) *SB* [41], *BS* ([101]), *MIL* ([6]) and CoGD ([119]). The *Car* sequence (fig. 6.5 a-b-c-d), represent a moderately difficult sequence since the target is affected by several occlusion with the surrounding background. Our approach is tolerant to occlusion as well is able to re-detect the object as a soon as it becomes again visibile (fig. 6.5 d) in the scene. Results reported in table highlight low performances for the approaches *OB*, *SB* , *BS* , *MIL*. Since their architecture was not designed with a detection component, when the object disappears or is occluded, they are not able to re-localize it achieving low recall rate.



a **b** **c** **d**

Figure 6.5: Sequence *Car*(a)

The *Panda* sequence (fig. 6.6 a-b-c-d), represent a very challenging scenario since a fast deformable target is tracked. Furthermore, occlusions, camouflage with the sorrounding background (fig. 6.6 d) and object disappearance (fig. 6.6 c) are other events that affects the tracking task. Our approach correctly tracks the object outperforming the *TLD* approach as reported in results showed table 6.3. Also in this scenario, the target camouflage with the backgrounds leads methods without re-detection capability to low recall values.

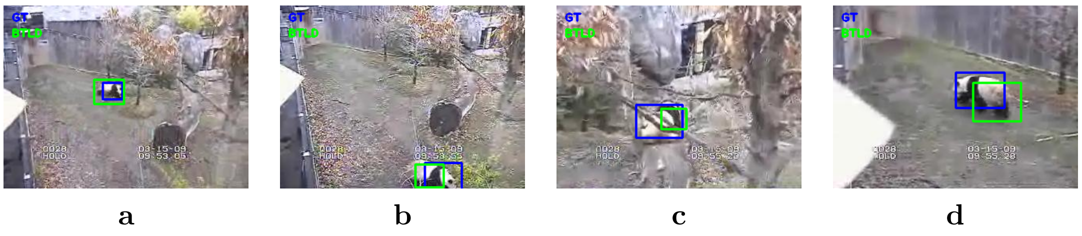


Figure 6.6: Sequence *Panda*(a)

The *Carchase* sequence (fig. 6.7 a-b-c-d), is the longest analyzed sequence. Fast motion and camera point of view changes (figure 6.7 d) as well object partial and full occlusion (fig. 6.7 c) makes the tracking process very difficult. Results reported in table 6.3 underlines the ability of our approach to perform the long term tracking task.

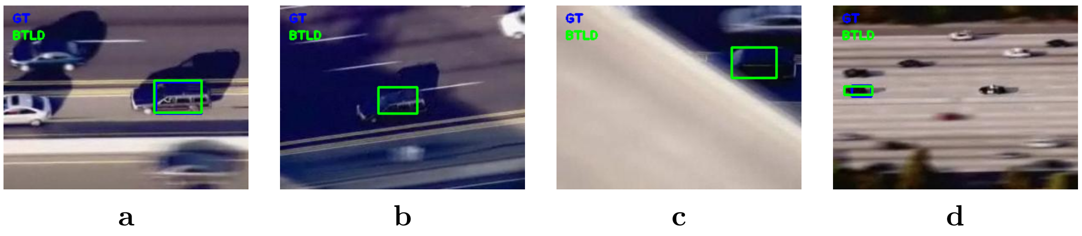
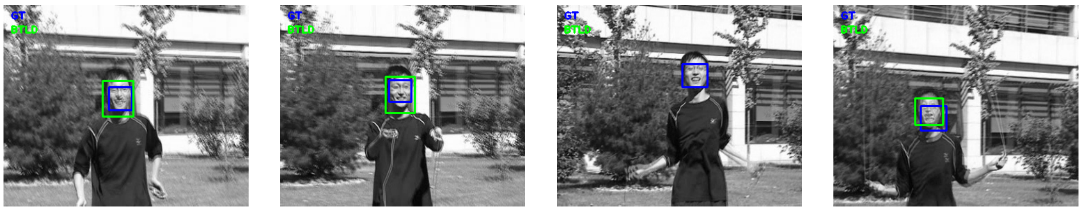


Figure 6.7: Sequence *Carchase*)

The *Jumping* sequence (fig. 6.8 a-b-c-d), tests the sensitivity to fast motion changes. The object appearance is affected by severe motion blurring (fig. 6.8 a-d). In such scenario, our approach is not accurate as the *TLD* method, since our detector implementation does not integrate the learning of synthetic blurred version of positive samples. This is an essential feature in order to control se-



a **b** **c** **d**

Figure 6.8: Sequence *Jumping*

vere motion blurring. Indeed, the *KLT approach* building the tracker component exibiths an high failure rate. Since the detector is not able to recognize blurred versions of the target, when the tracker loses the object (fig. 6.8 c) it is not restarted until motion blurring becomes less severe (fig. 6.8 d).

Sequence	OB [40]]	SB [41]	BS [101]	MIL [6]	CoGD [119]	TLD[55]	*BTLD*$^+$
1. Car	0.94 / 0.59 / 0.73	1.00 / 0.67 / 0.80	0.99 / 0.56 / 0.72	0.23 / 0.25 / 0.24	0.95 / 0.96 / 0.96	0.92 / 0.97 / 0.94	**0.99 / 0.98 / 0.98**
2. Panda	0.95 / 0.35 / 0.51	1.00 / 0.17 / 0.29	0.99 / 0.17 / 0.30	0.36 / 0.40 / 0.38	0.12 / 0.12 / 0.12	0.58 / 0.63 / 0.60	**0.9 / 0.8 / 0.84**
3. Carchase	0.79 / 0.03 / 0.06	0.80 / 0.04 / 0.09	0.52 / 0.12 / 0.19	0.62 / 0.04 / 0.07	0.95 / 0.04 / 0.08	0.86 / 0.70 / 0.77	**0.89 / 0.77 / 0.84**
4. Jumping	0.47 / 0.05 / 0.09	0.25 / 0.13 / 0.17	0.17 / 0.14 / 0.15	1.00 / 1.00 / 1.00	1.00 / 0.99 / 1.00	**1.00 / 1.00 / 1.00**	0.9 / 0.7 / 0.78

Table 6.3: Performance evaluation on *TLD* dataset measured by Precision/Recall/F-measure. Bold numbers indicate the best score. *BTLD*$^+$ outperformed in 3/4 sequences

6.5.1 Computational complexity

The computation complexity of the proposed approach is derived adding to the *BTLD* computational complexity, the computational complexity of *part besed* appearance modeling component. The NN-classifier building such component exibiths the same complexity of the global appearance model described in 4.6.1. It is $\mathcal{O}(m^2|\mathcal{P}|)$ where m^2 is the size of the patch and $|\mathcal{P}|$ is the total number of samples defining the *part besed* appearance model. The parts are evaluated only once at each tracking iteration, without affecting significantly the execution time of the overall approach. Indeed the system could master the challenge of an unstructured environment as well as moving objects with a rate of 9 fps.

6.6 Conclusions

In conclusion $BTLD^+$, defines a novel discriminative tracker that enforces the feature selection step building the $BTLD$ approach. The novelty of such discriminative tracker relies on the integration of memory capabilities into the $BTLD$ approach through the definition of part classifier able to recognize local feature points belonging to tracked object. With such capability, wrong sample selection problem is reduced drastically, increasing the accuracy of $MCMC$ particle filter framework proposed to solve the state estimation problem. A real-time implementation of the $MCMC$ particle filter framework has been described in detail and an extensive set of experiments was performed in order to highlight the ability of our approach to increase robustness of $BTLD$ tracker.

6.7 Related Publications

- Giorgio Gemignani, Alessio Ferone, Alfredo Petrosino. *A Bayesian Approach to Tracking Learning and Detecting by Parts* . Submitted to ECCV 2014.

Chapter 7

Conclusion

This chapter discusses the contributions proposed in this thesis and reviews recent developments.

7.1 Contribution

This dissertation has presented the research activity concerning adaptive visual tracking carried out during the Ph.D. course.
In particular, a critical analysis of the tracking learning detection approach has been proposed.
The main weaknesses of TLD approach have been pointed out and two contributions, namely $BTLD$ and $BTLD^+$ have been proposed.
Concerning $BTLD$, we proposed a novel tracking component that employs recursive Bayesian estimation framework to solve the state estimation problem, defining the essence of the long term tracking task. The proposed tracker has been shown to outperform the baseline solution, due to a novel strategy used to exploit the KLT feature tracker. Our experiments have demonstrated that our novel generative tracker is able to overcame the main limitations of the baseline

approach, while maintaining unchanged its strengths.
The novelty of our new component relies on the out-lier filtering scheme that exploit visual similarity to validate new point of interest detected on the target appearance. An application of the $BTLD$ approach into the Ambient Assisted Living context has been proposed, revealing how application dependent constraint (the static background) can be integrated into the proposed framework to improve the tracking accuracy.
The proposed solution has highlighted new interesting directions of research that led us towards the formulation of $BTLD^+$.
With such approach, we introduced into $BTLD$ architecture a discriminative part based appearance representation able of explicitly perform occlusion reasoning and enabling memory capabilities into the tracker component. Exploiting a part based representation, the tracker component is now able to . Moreover, the ability to discriminate target and background feature points, lead us to introduce geometrical constraint into the optimization problem defined into the $MCMC$ particle filter framework.
Preliminary experimental results, where our proposal was compared on challenging sequences against many state of the art trackers are encouraging and reveal an essential feature to be introduced into the detector component: the learning of motion blurred versions of the target appearance.

7.2 On going directions

An extension for our proposal would be introducing different visual representations, such as gradient and color.
Color information have demonstrated effective discriminative capabilities when it is possible to segment the target appearance.
Unfortunately, since camera motion is possible strategies to model a moving background and applying background subtraction should be studied, in order to detect foreground regions and extract only color

informations associated to the target.
A promising direction of research is in the use of the proposed part based model as a detector component. Such approach enforces re-detection capabilities under occlusion, but poses interesting questions on how controlling false positive and negative detections.
Preliminary results, obtained by using the part-based classifier enabling detection capabilities are depicted in figure 7.1. The detec-

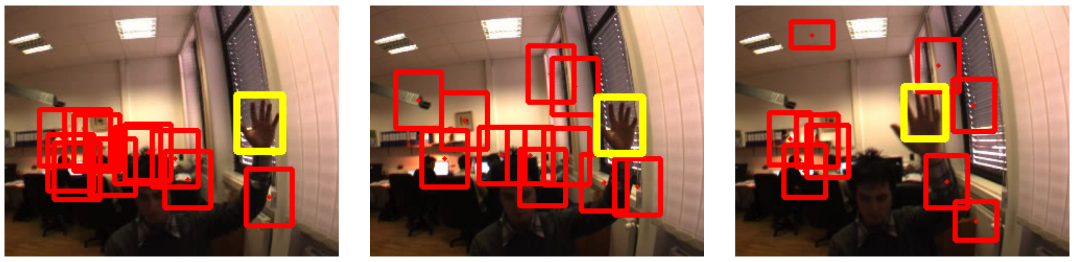


Figure 7.1: Sequence *Hand* from the *VOT 2013* Dataset. Preliminary results obtained exploiting the part based appearance model as a part detector. In red, negative detections generated by clustering negative detected parts. In yellow, the positive detection.

tions produced by the part based classifier are clustered according their spatial proximity and then integrated as constraints on the possible location where the object may be.
Moreover, exploiting spatial relationship among parts into the generative model is another cue that could improve tracking performances.

7.3 Related Publication

Here we report the list of the publications related to this thesis.

- Giorgio Gemignani, Wongun Choi, Alessio Ferone, Alfredo Petrosino, Silvio Savarese. *A Bayesian Approach to Tracking Learning Detection.* ICIAP 2013. Lecture Notes in Computer Science Volume 8156, 2013, pp 803-812.
- Giorgio Gemignani,Lucia Maddalena, Alfredo Petrosino. *Real-time stopped object detection by neural dual background modeling.* Proceedings of the 2010 conference on Parallel processing, pages 357-364.
- Giorgio Gemignani, Alessio Ferone, Alfredo Petrosino. *Tracking, Learning and Detection of Humans in ceiling mounted camera video streams.* Submitted to the Journal of Ambient Intelligence and Humanized Computing.
- Giorgio Gemignani, Alessio Ferone, Alfredo Petrosino. *A Bayesian Approach to Tracking Learning and Detecting by Parts* . Submitted to ECCV 2014.

Bibliography

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *In IEEE Conf. Computer Vision and Pattern Recognition (CVPR*, pages 798–805, 2006.

[2] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, 2008.

[3] N. Anjum and A. Cavallaro. Multi-feature object trajectory clustering for video analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1555–1564, November 2008.

[4] S. Avidan. Support vector tracking. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 184–191, 2001.

[5] S. Avidan. Ensemble tracking. In *In CVPR*, pages 494–501, 2005.

[6] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1619–1632, 2011.

[7] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vision*, 56(3):221–255, 2004.

[8] Y. Bar-Shalom. *Tracking and data association.* Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[9] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[10] K. Bernardin, E. Elbs, and R. Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment.

[11] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '98, pages 232–243, Washington, DC, USA, 1998. IEEE Computer Society.

[12] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[13] M. J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Comput. Vis. Image Underst.*, 63(1):75–104, 1996.

[14] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *International Journal of Computer Vision*, pages 329–342, 1998.

[15] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[16] Y. Boykov and D. Huttenlocher. Adaptive bayesian recognition in tracking rigid objects. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 697–704 vol.2, 2000.

[17] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[18] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[19] L. Cehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(4):941–953, 2013.

[20] M. Cerman and Vaclav. Sputnik tracker: Having a companion improves robustness of the tracker. In A.-B. Salberg, J. Y. Hardeberg, and R. Jenssen, editors, *SCIA*, volume 5575 of *Lecture Notes in Computer Science*, pages 291–300. Springer, 2009.

[21] T.-J. Cham and J. M. Rehg. A multiple hypothesis approach to figure tracking. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 2:239–244, 1999.

[22] W.-Y. Chang, C.-S. Chen, and Y.-P. Hung. Tracking by parts: A bayesian approach with component collaboration. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 39(2):375–388, 2009.

[23] T.-J. Chin and D. Suter. Incremental kernel principal component analysis. *IEEE Transactions on Image Processing*, 16(6):1662–1674, 2007.

[24] W. Choi and S. Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. In *Proceedings of the 11th European conference on Computer vision: Part IV*, pages 553–567, 2010.

[25] D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

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

[27] B. Demiroz, I. Ari, O. Eroglu, A. Salah, and L. Akarun. Feature-based tracking on a multi-omnidirectional camera dataset. pages 1–5, 2012.

[28] N. D. H. Dowson and R. Bowden. Simultaneous modeling and tracking (smat) of feature sets. In *CVPR (2)*, pages 99–105, 2005.

[29] D. G. E. Kochavi and H. A. M. for rapid MRI needle tracking. *Magnetic Resonance in Medicine*, 51(5):1083–1087, 2004.

[30] G. G. Electrics. Visiowave - intelligent video platform. `http://www.axxonsoft.com/partners/partner_locator/technological_partners/45565/`.

[31] E. Erdem, S. Dubuisson, and I. Bloch. Visual tracking by fusing multiple cues with context-sensitive reliabilities. *Pattern Recognition*, 45(5):1948–1959, 2012.

[32] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[33] V. Ferrari, T. Tuytelaars, and L. J. V. Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 67(2):159–188, 2006.

[34] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[35] M. C. R. W. for Xbox 360. Kinect. `http://www.microsoft.com/en-us/kinectforwindows`.

[36] D. M. Gavrila. Pedestrian detection from a moving vehicle. pages 37–49, 2000.

[37] G. Gemignani, L. Maddalena, and A. Petrosino. Real-time stopped object detection by neural dual background modeling. In *Euro-Par Workshops'10*, pages 357–364, 2010.

[38] J. Gemmell, K. Toyama, C. L. Zitnick, T. Kang, and S. Seitz. Gaze awareness for video-conferencing: A software approach. *IEEE MultiMedia*, 7(4):26–35, Oct.

[39] M. Godec, C. Leistner, A. Saffari, and H. Bischof. On-line random naive bayes for tracking. In *ICPR*, pages 3545–3548, 2010.

[40] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proc. BMVC*, pages 6.1–6.10, 2006.

[41] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking, 2008.

[42] H. Grabner, J. Matas, L. J. V. Gool, and P. C. Cattin. Tracking the invisible: Learning where the object might be. In *CVPR*, pages 1285–1292. IEEE, 2010.

[43] H. Grabner, P. M. Roth, and H. Bischof. Eigenboosting: Combining discriminative and generative information. In *CVPR*, 2007.

[44] M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking. In *CVPR*. IEEE Computer Society, 2007.

[45] S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, 1987.

[46] B. Han and L. Davis. On-line density-based appearance modeling for object tracking. In *International Conference on Computer Vision (ICCV)*, 2005.

[47] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270. IEEE, 2011.

[48] J. Hoey. Tracking using flocks of features, with application to assisted handwashing. In *British Machine Vision Conference BMVC*, 2006.

[49] B. K. P. Horn and B. G. Schunck. Determining optical flow. *ARTIFICAL INTELLIGENCE*, 17:185–203, 1981.

[50] IBM. Ibm smart surveillance system (s3). `http://researcher.watson.ibm.com/researcher/view_project.php?id=1394`.

[51] INTELLIVID. Intellivid retail video investigatortm. `http://tienyi.com/iv/pr_2007_01_29.html`.

[52] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[53] A. D. Jepson, D. J. Fleet, T. F. El-maraghi, I. C. Society, I. C. Society, and I. C. Society. Robust online appearance models for visual tracking. pages 415–422, 2001.

[54] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, pages 2756–2759, 2010.

[55] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

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

[57] W. Kloihofer and M. Kampel. Interest point based tracking. In *ICPR*, pages 3549–3552. IEEE, 2010.

[58] J. Kwon and K. M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *CVPR*, pages 1208–1215, 2009.

[59] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR*, pages 1–8, 2008.

[60] C. Leistner, A. Saffari, and H. Bischof. Miforests: Multiple-instance learning with randomized trees. In *ECCV (6)*, volume 6316, pages 29–42, 2010.

[61] V. Lepetit. Keypoint recognition using randomized trees. *IEEE Trans. Pattern Anal. Mach. Intell*, 2006.

[62] Y. Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518, 2004.

[63] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li. Learning multi-scale block local binary patterns for face recognition. In *ICB*, volume 4642, pages 828–837, 2007.

[64] H. Lim, V. I. Morariu, O. I. Camps, and M. Sznaier. Dynamic appearance modeling for human tracking. In *Computer Vision and Pattern Recognition (CVPR)*, pages 751–757, 2006.

[65] Z. Lin, L. S. Davis, D. Doermann, and D. Dementhon. Dementhon d.: Hierarchical part-template matching for human detection and segmentation. In *In: ICCV (2007*, 2007.

[66] J. S. Liu. *Monte Carlo strategies in scientific computing*. 2001.

[67] R. Liu, J. Cheng, and H. Lu. A robust boosting tracker with minimum error bound in a co-training framework. In *ICCV*, pages 1459–1466, 2009.

[68] X. Liu and T. Yu. Gradient feature selection for online boosting. In *ICCV*, pages 1–8. IEEE, 2007.

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

[70] L. Lu and G. D. Hager. A nonparametric treatment for location/segmentation based visual tracking. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*, 2007.

[71] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. pages 674–679, 1981.

[72] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Transactions on Image Processing*, 17(7):1168–1177, 2008.

[73] E. Maggio and A. Cavallaro. *Video tracking.* 2009.

[74] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

[75] B. S. Manjunath and W.-Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(8):837–842, 1996.

[76] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* 1982.

[77] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE PAMI*, 26:810–815, 2003.

[78] G. Máttyus, C. Benedek, and T. Szirányi. Multi target tracking on aerial videos. In *ISPRS Workshop on Modeling of Optical Airborne and Space Borne Sensors*, Istanbul, Turkey, 2010.

[79] A. Mecocci, F. Bartolini, and V. Cappellini. Image sequence analysis for counting in real time people getting in and out of a bus. *Signal Processing*, 35(2):105–116, Jan. 1994.

[80] K. Mori, D. Deguchi, K. Akiyama, T. Kitasaka, C. R. M. Jr., Y. Suenaga, H. Takabatake, M. Mori, and H. Natori. Hybrid bronchoscope tracking using a magnetic tracking sensor and image registration. In *MICCAI (2)*, volume 3750, pages 543–550, 2005.

[81] Y. Mu, S. Yan, Y. Liu, T. S. Huang, and B. Zhou. Discriminative local binary patterns for human detection in personal album. In *CVPR*. IEEE Computer Society, 2008.

[82] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *Computer Journal*, 26(4):354–359, 1983.

[83] S. Notomi, Lysyansky. Measurement of ventricular torsion by two-dimensional ultrasound speckle tracking imaging. *Journal of the American College of Cardiology*, 45(12):2034–2041, 2005.

[84] N. C. Oza and S. Russell. Online bagging and boosting. In *In Artificial Intelligence and Statistics 2001*, pages 105–112, 2001.

[85] I. Patras and M. Pantic. Particle filtering with factorized likelihoods for tracking facial features. In *FGR*, pages 97–104, 2004.

[86] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[87] C. Rasmussen and G. D. Hager. Probabilistic data association methods for tracking complex visual objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(6):560–576, 2001.

[88] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: particle filters for tracking applications.* 2004.

[89] R. Rosales and S. Sclaroff. 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *In CVPR*, pages 117–123, 1999.

[90] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking, 2008.

[91] C. Sacchi, G. Gera, L. Marcenaro, and C. S. Regazzoni. Advanced image-processing tools for counting people in tourist site-monitoring applications. *Signal Processing*, 81(5):1017 – 1040, 2001.

[92] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests.

[93] S. Salti, A. Cavallaro, and L. di Stefano. Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE TIP*, 21(10):4334–4348, 2012.

[94] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *CVPR*, pages 723–730, 2010.

[95] A. Schofield, P. Mehta, and T. Stonham. A system for counting people in video images using neural networks to identify the background scene. *Pattern Recognition*, 29(8):1421 – 1428, 1996.

[96] J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[97] D. Skocaj and A. Leonardis. Weighted and robust incremental method for subspace learning. In *ICCV*, pages 1494–1501. IEEE Computer Society, 2003.

[98] L. Snidaro, C. Micheloni, and C. Chiavedale. Video security for ambient intelligence. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(1):133–144, 2005.

[99] X. Song, J. Cui, H. Zha, and H. Zhao. Vision-based multiple interacting targets tracking via on-line supervised learning. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, pages 642–655, 2008.

[100] P. sports graphics system. Bbc. `http://www.bbc.co.uk/rd/projects/piero`.

[101] S. Stalder, H. Grabner, and L. V. Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *OLCV 09: 3rd On-line learning for Computer Vision Workshop*, 2009.

[102] O. T. and T. Pietikainen, M. ; Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, 2002.

[103] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19(6):1635–1650, 2010.

[104] Z. Tang, Brennan and Tao. Co-tracking using semi-supervised support vector machines. *IEEE Trans. Pattern Analysis Machine Intelligence*, pages 1–8, 2007.

[105] A. Teichman and S. Thrun. Tracking-based semi-supervised learning. *Int. J. Rob. Res.*, 31(7):804–818, 2012.

[106] M. Tian, W. Zhang, and F. Liu. On-line ensemble svm for robust object tracking. In Y. Yagi, S. B. Kang, I.-S. Kweon, and H. Zha, editors, *ACCV (1)*, volume 4843 of *Lecture Notes in Computer Science*, pages 355–364. Springer, 2007.

[107] I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *CVPR (2)*, pages 258–265, 2005.

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

[109] E. A. Wan and R. V. D. Merwe. The unscented kalman filter for nonlinear estimation. pages 153–158, 2000.

[110] H. Wang, D. Suter, K. Schindler, and C. Shen. Adaptive object tracking based on an effective appearance filter. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(9):1661–1667, 2007.

[111] T. Wang, I. Y. H. Gu, and P. Shi. Object tracking using incremental 2d-pca learning and ml estimation. In *ICASSP (1)*, pages 933–936, 2007.

[112] G. Welch and G. Bishop. An introduction to the kalman filter, 1995.

[113] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic huber-l1 optical flow. In *BMVC*, 2009.

[114] J. Xiao, S. Baker, I. Matthews, and T. Kanade. Real-time combined 2d+3d active appearance models. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 535–542, 2004.

[115] J. Xie, S. Khan, and M. Shah. Automatic tracking of escherichia coli bacteria. In *MICCAI (1)*, volume 5241, pages 824–832, 2008.

[116] W. Yan, C. Weber, and S. Wermter. A hybrid probabilistic neural model for person tracking based on a ceiling-mounted camera. *J. Ambient Intell. Smart Environ.*, 3(3):237–252, Aug. 2011.

[117] M. Yang, F. Lv, W. Xu, and Y. Gong. Detection driven adaptive multi-cue integration for multiple human tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1554–1561, 2009.

[118] Z. Yin and R. T. Collins. On-the-fly object modeling while tracking. In *2007 (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*, 2007.

[119] Q. Yu, T. B. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. In *in Proc. Eur. Conf. Comput. Vision*, pages 678–691.

[120] J. yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.

[121] B. Zeisl, C. Leistner, A. Saffari, and H. Bischof. On-line Semi-supervised Multiple-Instance Boosting. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[122] K. Zhang, L. Z. 0006, and M.-H. Yang. Real-time compressive tracking. In *ECCV (3)*, volume 7574 of *Lecture Notes in Computer Science*, pages 864–877. Springer, 2012.

[123] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang. Local gabor binary pattern histogram sequence (lgbphs): A novel non-statistical model for face representation and recognition. In *ICCV*, pages 786–791, 2005.

[124] Q. Zhu, Q. Zhu, S. Avidan, S. Avidan, M. chen Yeh, M. chen Yeh, K. ting Cheng, and K. ting Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *In CVPR06*, pages 1491–1498, 2006.

[125] K. Zimmermann, J. Matas, and T. Svoboda. Tracking by an optimal sequence of linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):677–692, 2009.

[126] M. Zuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(3):448–461, 2010.