ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEIS – DIPARTIMENTO DI INFORMATICA,
ELETTRONICA E SISTEMISTICA

ON-LINE ADAPTIVE VISUAL TRACKING

Samuele Salti

TUTOR
Professor
Tullio Salmon Cinotti

COORDINATOR
Professor
Paola Mello

CO-TUTOR
Professor
Luigi Di Stefano

PHD. THESIS

January, 2008 – December, 2010

PHD PROGRAM IN ELECTRONIC, COMPUTER SCIENCE AND TELECOMMUNICATIONS ENGINEERING

CYCLE XXIII – ING-INF/05

Contents

Introduction

Visual tracking is the problem of estimating some variables related to a target given a video sequence depicting the target. In its simplest form, it consists of estimating the position of the target while it wanders in the scene, *i.e.* its trajectory in the image plane. Depending on the final application and the tracker complexity, additional target variables can be estimated, such as scale, orientation, joint angles between its parts, velocity, etc. These variables form the target *state*, *i.e.* the set of hidden variables that the tracker tries to recover from noisy observations of it, *i.e.* the video frames.

Visual tracking is a fundamental feature for the automation of many tasks, such as visual surveillance, robot or vehicle autonomous navigation, automatic video indexing in multimedia databases, etc. . . It is also a basic enabling factor for making machines able to interpret human motion and deliver a whole new branch of services and applications, such as natural human-computer interfaces, smart homes, offices or urban environments and computer-aided diagnosis or rehabilitation.

Visual tracking is difficult because of the classical nuisances that computer vision faces, such as scene illumination changes, loss of information due to perspective projection, sensor noise, etc... , and because of peculiar difficulties, such as complex motion patterns of the target, non-rigid or appearance-changing targets, partial and full target occlusions.

Despite many years of research, long term tracking in real world scenarios for generic targets is still unaccomplished. The main contribution of this thesis is the definition of effective algorithms that can bring visual tracking closer to a solution by letting the tracker adapt to mutating working conditions. In particular, we propose to adapt two crucial components of visual trackers: the transition model and the appearance model. The adaptation is performed on-line, *i.e.* frame-by-frame while the tracker runs. To better contextualize our contributions, we first introduce the standard formulation of the tracking problem and the tools typically used to solve it.

As noted in [17], two major components can be distinguished in a typical visual tracker: *Filtering and Data Association* is mostly a top-down process dealing with the dynamics of the tracked object and evaluation of different hypotheses; *Target Representation and Localization* is mostly a bottom-up process which has to cope with changes in the appearance of the target and has to provide an effective description of it in presence of similar objects (distractors). The way the two components are combined and weighted is application dependent and plays a decisive role in the robustness and efficiency of the tracker. Nevertheless, for a general tracker both components are key to tracker success.

As far as the Filtering and Data Association component is concerned, to deal with all the nuisances and to take into account the uncertainty onto the final estimation they introduce, one widespread approach is to formulate tracking as a probabilistic inference problem on the space of all possible states. The probabilistic formulation and the requirement for the updating of state estimation on receipt of new measurements naturally lead to the Bayesian approach. It provides a rigorous general framework for dynamic state estimation.

In the Bayesian approach the output is the *posterior* probability density function (PDF) of the state, based on all available information, *i.e.* the sequence of previous states and received measurements. Since the posterior PDF encompasses all available statistical information, an optimal estimation of the state with respect to any criterion may be obtained from it.

In this thesis we deal only with causal trackers, *i.e.* we do not take into account visual trackers using future frames and states to estimate the state at a given time. In a causal tracker an estimate of the state

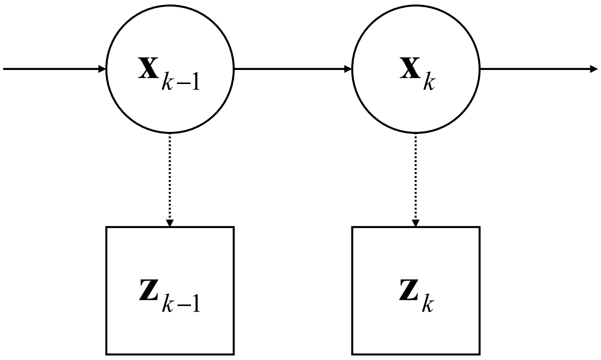


Figure 1: The first order Markov chain structure assumed for the target state.

is computed every time a measurement is received, *i.e.* a new frame is available in the frame buffer, using only past states and measures. A recursive filter is the natural solution in this case. Hence, Recursive Bayesian Estimation (RBE) [3, 79] is the standard tool to tackle the state estimation in causal visual trackers.

RBE is solved, at least from a theoretical point of view, under the standard assumption that the system can be modeled as a first order Markov model (Fig. 1), *i.e.*

- the state at time k, $\mathbf{x}_k \in \mathbb{R}^N$, depends only on the previous state $\mathbf{x}_{k-1}$;
- the measure at time k, $\mathbf{z}_k \in \mathbb{R}^M$, depends only on $\mathbf{x}_k$.

In the case of visual tracking, the measure $\mathbf{z}_k$ typically coincides with the current frame I_k, hence the two terms and symbols will be used interchangeably.

From the first order Markovian assumption it follows that the system is completely specified by:

- a law of evolution of the state,

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \nu_k) \tag{1}$$

where v_k is an i.i.d. process noise sequence and f_k is a possibly non-linear function relating the state at time k with the previous one;

- a measurement process,

$$\mathbf{z}_k = h_k(\mathbf{x}_k, \eta_k) \tag{2}$$

  where η_k is an i.i.d. measurement noise sequence and h_k is a possibly non-linear function relating the measurement at time k with the current state;

- an initial state $\mathbf{x}_0$.

Process noise takes into account any modeling errors or unforeseen disturbances in the state evolution model

In a Bayesian probabilistic approach, given the noise affecting the low of evolution of the state and the measurement process, the entities comprising the system are defined by PDFs, *i.e.*

- the transition model,

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \tag{3}$$

  defined by (1) and the statistics of v_k;

- the observation likelihood,

$$p(\mathbf{z}_k \mid \mathbf{x}_k) \tag{4}$$

  defined by (2) and the statistics of η_k;

- the initial target PDF $p(\mathbf{x}_0)$.

These PDFs are generally assumed to be known a priori and never updated.

Given this characterization of the target, a general but conceptual solution can be obtained in two steps: prediction and update. In the prediction stage, the Chapman-Kolmogorov equation is used to propagate the belief on the state at time $k-1$ to time k

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{z}_{1:k-1})\, d\mathbf{x}_{k-1}. \tag{5}$$

where $\mathbf{z}_{1:k-1}$ is the set of all measurements up to frame $k-1$, $\{\mathbf{z}_1, \ldots, \mathbf{z}_{k-1}\}$ This usually corresponds to a spreading of the belief on the state, due to the increasing distance in time from the last measurement. In the update stage, the PDF is sharpened again by using the current measure $\mathbf{z}_k$ and the Bayes rule

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) \propto p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}). \tag{6}$$

This conceptual solution is analytically solvable only in few cases. A notable one is when the law of evolution of the state and the measurement equations are linear and the noises are Gaussian. In this situation, the optimal solution is provided by the Kalman filter [42]. The RBE framework for this case becomes:

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \nu_k) \;\Rightarrow\; \mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \nu_k, \;\; \mathbb{E}\left[\nu_k \nu_k^T\right] = \mathbf{Q}_k \tag{7}$$

$$\mathbf{z}_k = h_k(\mathbf{x}_k, \eta_k) \;\Rightarrow\; \mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \eta_k, \;\; \mathbb{E}\left[\eta_k \eta_k^T\right] = \mathbf{R}_k. \tag{8}$$

and the mean and covariance matrix of the Gaussian posterior can be optimally estimated using the Kalman filter equations:

- prediction,

$$\mathbf{x}_k^- = \mathbf{F}_k \mathbf{x}_{k-1} \tag{9}$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{10}$$

  where $\mathbf{x}_{k-1}$ and $\mathbf{P}_{k-1}$ are the previous estimates of, respectively, the mean vector and the covariance matrix and $\mathbf{x}_k^-$ and $\mathbf{P}_k^-$ are, following the typical Kalman notation, the estimates of, respectively, the mean vector and the covariance matrix for the current frame *before* a new measure is available;

- update,

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- * \mathbf{H}_k^T + \mathbf{R}_k \tag{11}$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k \mathbf{S}_k^- 1 \tag{12}$$

$$\mathbf{x}_k = \mathbf{x}_k^- - \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}_k \mathbf{z}_k^- \right) \tag{13}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \mathbf{P}_k^- \tag{14}$$

  where $\mathbf{x}_k$ and $\mathbf{P}_k$ are the optimal estimates of, respectively, the mean vector and the covariance matrix.

When the assumptions made by the Kalman filter do not hold, a sub-optimal solution to the RBE problem can be obtained with particle filters [79]. Particle filters performs sequential Monte Carlo estimation. Given the posterior, $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$ we want to obtain an estimate of the state from it:

$$\hat{\mathbf{x}}_k = \int_{\mathbb{R}^n} f(\mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) \, d\mathbf{x}_k \, . \tag{15}$$

The Monte Carlo solution is a numerical evaluation of the integral, that requires to sample L samples $\mathbf{x}_k^i$ from the posterior and then compute the estimate as the sample mean

$$\hat{\mathbf{x}}_k = \frac{1}{L} \sum_{i=1}^{L} f(\mathbf{x}_k^i) \, . \tag{16}$$

Unfortunately, it is impossible to sample from the posterior in the general, non Gaussian / non linear case, since it has a non standard form and it is usually known only up to a proportionality constant. However, if it is possible to generate samples from a density $q(\mathbf{x}_k)$ that is similar to the posterior (*i.e.*, it is not 0 when the posterior is not 0), then we can still use the Monte Carlo method to approximate the integral in (15) by drawing sample from $q(\mathbf{x}_k)$ and weighting them accordingly,

$$\hat{\mathbf{x}}_k = \frac{1}{L} \sum_{i=1}^{L} f(\mathbf{x}_k^i) w(\mathbf{x}_k^i) \quad \text{with} \quad w(\mathbf{x}_k^i) = \frac{p(\mathbf{x}_k^i \mid \mathbf{z}_{1:k})}{q(\mathbf{x}_k^i)} \, . \tag{17}$$

This technique is known as importance sampling and the PDF q 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, the *particles*. The posterior PDF can then be approximated as

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) = \sum_{i=1}^{L} w(\mathbf{x}_k^i)\delta(\mathbf{x}_k - \mathbf{x}_k^i) \tag{18}$$

where samples are obtained at each time step from the proposal density $q(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{z}_k)$, weights are updated at each time step as [79]

$$w(\mathbf{x}_k^i) \propto \frac{p(\mathbf{x}_k \mid \mathbf{z}_{1:k})}{q(\mathbf{x}_k \mid \mathbf{x}_{1:k-1}, \mathbf{z}_{1:k})} \propto w(\mathbf{x}_{k-1}^i)\frac{p(\mathbf{z}_k \mid \mathbf{x}_k^i)p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{19}$$

and then normalized to sum to one. It can be shown that as $N \to \infty$ the approximation in (18) converges to the true posterior density.

The main problem with sequential importance sampling is represented by particle degeneracy. In particular, the variance of the particles weights can only increase with sequential importance sampling. In practice, this means that after a certain number of recursive steps, all but one particle will have negligible weights. To counteract this effect resampling algorithms are introduced, leading to so called sequential importance resampling algorithms. Resampling eliminates samples with low weights and multiplies samples with high importance weights. 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 resampling with replacement L times from the cumulative sum of normalized weights of the particles [79].

Within the RBE framework, our major contribution, described in Chapter 1, is an algorithm to effectively and efficiently estimate the transition model $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ on-line from the tracker output in the Gaussian and linear case. This reduces the number of parameters to be set by the user, in particular the process noise covariance which are typically hard to estimate but play a significant role for the filter performance. Our algorithm also allows for obtaining a time-variant estimation of the transition model, and therefore results in a more adaptive filter.

As far as Target Representation and Localization is concerned, two main ingredients constitute it, namely the choice of the features space and the target appearance model.

The regions of the current frame I_k analyzed by the recursive Bayesian filter are generally projected in some feature space. For instance, in a standard approach for tracking with particle filters [78], the samples of the state generated by the importance density are then represented as color histograms [17].

The feature representation usually is:

- more compact than the corresponding region of I_k;
- invariant to some (geometric or photometric) variations.

A variety of features has been used to describe the target, *e.g.* motion vectors, change detection, object classification, low-level features such as pixel colors or gradients, or mid-level features such as edges and interest points (see [104] for a survey). A main discriminant characteristic among features is their spatial extent:

- **Part-wise features.** Features are extracted from small patches or even single pixels (*e.g.* 5×5 HoGs [20]). 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 (*e.g.* color histograms [17]). This kind of features can typically tolerate target deformations and rigid transformations. Correct handling of occlusions represents the most serious limitation of these representations.

The link between the Filtering and the Representation stage of a tracker is represented by the observation likelihood $p(\mathbf{z}_k \mid \mathbf{x}_k^i)$ defined in (4). To evaluate it, in oder to update a particle weight, the appearance model of the target, that we indicate with $\mathbf{A}$, is compared with the features extracted from the state candidate $\mathbf{x}_k^i$. The target model lives, of course, in the same feature space used to describe the current candidates. The target model is usually learned once, either offline from training data or online from the first frame(s), and then used throughout the sequence.

The use of a fixed model for locating the target makes it difficult to cope with illumination changes and deformable targets. Hence, recently, the idea of appearance model update has been proposed by several researchers to aim at successful long-term tracking despite these difficulties. By letting the model evolve across frames to include and adapt to the potential geometric and photometric changes of the target, these methods are inherently able to cope with target deformations and lightening variations. On the other hand, they expose the tracker to the risk of *drift*, *i.e.* the inclusion of background appearance in the appearance model that can eventually lead to loss of track.

In chapter 2, we analyze the recent advances in target model update and present our proposal, which is based on the deployment of the Recursive Bayesian Estimation framework to tackle target model update, too. This allows for exploiting the robustness of this framework also in the crucial step of target model update and introduces a probabilistic treatment as an interesting solution for this open problem.

Chapter 3 deals with adaptive tracking with a static camera. Our contribution in this context concerns both Target Representation and Filtering. As for the former, we introduce a novel, efficient change detection algorithm, robust to sudden illumination changes, based on the joint histogram of background and foreground intensities and on Bayesian inference. As for the latter, we propose a sound way to obtain an adaptive observation likelihood from the output of the change detection and a method to obtain a proper prior for the change detection from the prediction step of the recursive Bayesian filter employed as tracker. The two flows of information realize a full adaptive Bayesian loop encompassing tracking and change detection.

Finally, in Chapter 4 we present our work on the detection of categories in 3D data. In a real automatic deployment a visual tracker is usually initialized with the output of a detector for the category of interest (*e.g.* humans, cars, faces). While detection in images has reached a high level of maturity [20, 50, 100], data coming from 3D sensors have not been fully exploited yet. Moreover, we have recently seen an increasing interest on the automatic analysis of such data due to the release of cheap modern sensors such as the *Kinect* device by Microsoft, that lets foresee an ubiquitous presence of 3D data for human computer interaction. In our work we adapt the well-known Implicit Shape Models [50], proposed for images, to the detection of categories in 3D data. This extension is based on our novel descriptor for 3D data, dubbed SHOT, that obtains state of the art performance in various experiments of shape matching, also presented in the chapter. Finally, the extension of SHOT for the description of textured 3D data like those provided by the Kinect sensor is described and compared with another texture-aware descriptor [106].

All the tracking results for the first three chapters are available as videos at the author website [1].

[1] `www.vision.deis.unibo.it/ssalti`

Chapter 1

Adaptive Transition Model

Recursive Bayesian Estimation (RBE) is a widespread solution for visual tracking as well as for applications in other domains where a hidden state is estimated recursively from noisy measurements. Although theoretically sound and unquestionably powerful, from a practical point of view RBE suffers from the assumption of complete a priori knowledge of the transition model, that is typically unknown. The use of a wrong a priori transition model may lead to large estimation errors or even to divergence. We propose to prevent these problems, in case of fully observable systems, learning the transition model on-line via Support Vector Regression [86]. An application of this general framework is proposed in the context of linear/Gaussian systems, where we dub it Support Vector Kalman (SVK), and shown to be superior to a standard, non adaptive solution.

1.1 Motivation

The difficulty of identifying a proper transition model for a specific application typically leads to empirical and suboptimal tuning of the estimator parameters. The most widespread solutions to specify a transition model for tracking are to empirically select it among a restricted set of standard ones (such as constant position, *i.e.* Brownian motion,

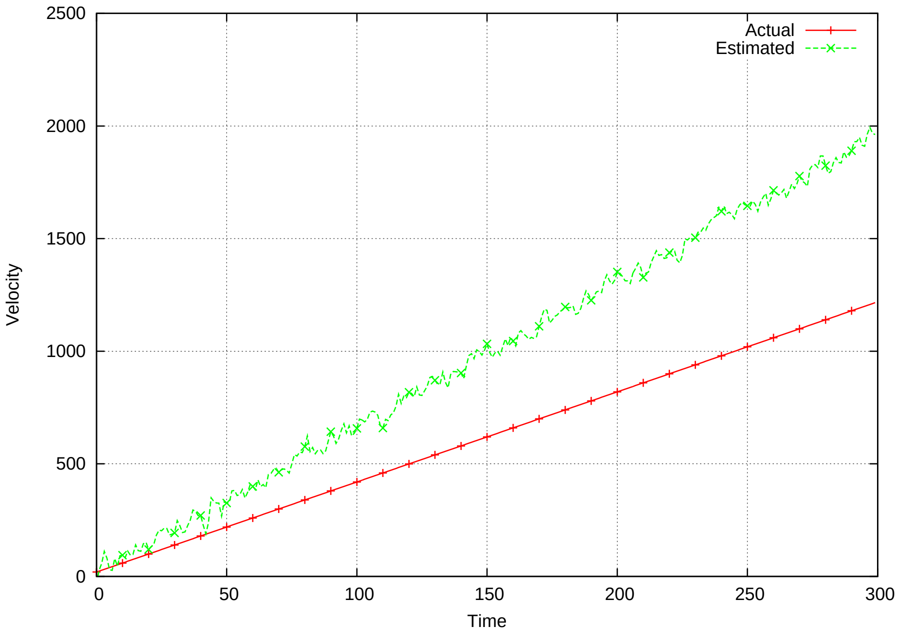


Figure 1.1: The effect of the use of a wrong transition model: the Kalman estimation diverges from the true velocity.

[1, 4, 16] or constant velocity [17, 32, 34, 38]) or learn it off-line from representative training data [78]. Besides the availability of these training sequences, which depends on the particular application, the major shortcoming of these solutions is that they do not allow to change the transition model trough time, although this can be beneficial and neither the conceptual solution nor the solving algorithms require it to be fixed.

Approximate tuning of a recursive Bayesian filter may seriously degrade its performances, that could be optimal (*e.g.*, when the assumptions of a Kalman filter are met) or sub-optimal (*e.g.*, in all the other cases where a particle filter is used) in case of correct system identification. In Fig. 1.1 we present a simple experiment to highlight the strong, detrimental impact of a wrong transition model on an otherwise optimal and correctly tuned recursive Bayesian filter. In this simulation a point is moving along a line with constant acceleration and we try to estimate its position and velocity by a Kalman filter from measurements corrupted with Gaussian noise, whose constant covariance matrix is known and used as the measurement noise covariance matrix of the filter, $\mathbf{R}_k$. Hence, we are using the optimal estimator for the experimental setup. The only parameter that is wrongly tuned is the transition model, in particular we are using a constant velocity matrix $\mathbf{F}_k$ instead of a constant acceleration one. The process covariance matrix, $\mathbf{Q}_k$, was set very high, in order to compensate for the wrong transition matrix. Despite this, the estimation and the true value of the velocity diverge. In other words, the estimation of an otherwise optimal estimator like the Kalman filter can be arbitrarily wrong when an incorrect transition model is assumed. This is the main motivation behind our work.

1.2 Previous work

Closely related to our work are the efforts devoted to the derivation of adaptive Kalman filters, that have been studied since the introduction of this filtering technique. In fact, our proposal can be seen as a new approach to build an adaptive Kalman filter. The main idea behind adaptive filtering schemes is that the basic source of uncertainty is due to the unknown noise covariances, and the proposed solution is to estimate them on-line from observed data. One of the most comprehensive contributions is given by Mehra [58]. He reviews proposed approaches and classify them according to four categories:

1. Bayesian Estimation (BE)
2. Maximum Likelihood Estimation (MLE)
3. Correlation Methods (CM)
4. Covariance-Matching Techniques (CMT).

Methods in the first category imply integration over a large dimensional space and can be solved only with special assumptions on the PDF of the noise parameters. MLE requires the solution of a non-linear equation that, in turns, is solvable only under the assumptions that the system is time invariant and completely observable and the filter has reached a steady state. Under these assumptions, however, only a time invariant estimation of the parameters of the noise PDF can be obtained. Correlation Methods, too, are applicable only to time invariant and completely observable systems. Finally, Covariance-Matching Techniques can estimate either process or measurement noise parameters and turn out to provide good and time-varying approximations for the measurement noise when the process noise is known.

In the work of Oussalah and De Schutter [70], an improved correlation method is proposed, but the requirement on the stationarity of the system is not dropped. In the context of visual tracking, Weng et al. [101] present the application of an adaptive Kalman filter. The process and measurement errors are modified in every frame taking into account the degree of occlusion of the target: greater occlusion corresponds to greater value of measurement noise and vice versa. The two noises always sum up to one. In the extreme case of total occlusion, measurement noise is set to infinity and process noise to 0. Zhang et al. [109] use the term Adaptive to refer to an adaptive forgetting factor, that is used to trade off the contribution to the covariance estimate for the current time step of the covariance estimate for the previous time step and the process noise. This is done in order to improve the responsiveness of the filter in case of abrupt state changes.

Compared to all these proposals, our method makes less assumptions on the system, the only one being its complete observability. This allows it to be more generally applicable and, in particular, to fit better the usual working conditions of visual trackers. Moreover, unlike BE, MLE and CM techniques our proposal provides a time-varying noise statistics estimation. This is extremely important to allow the filter to dynamically weight the prediction on the state and the noisy measurement it has to fuse at each frame, *e.g.* to cope with occlusions when the measurement can be totally wrong and the prediction on the state is the only reliable source of information to keep on tracking the target. Unlike the work of Weng et al. [101], our proposal is not specifically conceived for visual tracking and, hence, generally applicable. Finally, it is worth pointing out that, unlike all reviewed approaches, our proposal is adaptive in a broader sense, for it identifies on-line not only the process noise covariance matrix but also the transition matrix.

1.3 On-line adaptation of the transition model

We propose to overcome the difficulties and the shortcomings due to the empirical tuning of the transition model by adapting it *on-line* . If the state is completely observable, as it is the case in most practical applications, *i.e.* the h_k function just adds measurement noise on the state, the transition model is directly related to the dynamics exhibited by the measurements. Hence, it is possible to exploit their temporal evolution in order to *learn* the function f_k, and, implicitly, the PDF $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. That is, we can avoid to define $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, and instead use in its place a learned PDF $\tilde{p}_{\mathbf{z}_{1:k-1}}(\mathbf{x}_k|\mathbf{x}_{k-1})$, derived from a learned $\tilde{f}_{\mathbf{z}_{1:k-1}}$. Here, $\tilde{p}_{\mathbf{z}_{1:k-1}}$ formally indicates that the PDF is learned using as training data the relationships between all the consecutive measures from 1 to $k-1$.

Furthermore, we propose to learn the motion model using Support Vector Machine [99] in ϵ-regression mode (SVR) [86]. SVMs are well known and effective tools in pattern recognition based on the statistical learning theory developed by Vapnik and Chervonenkis. Their widespread use is due to their solid theoretical bases which guarantee their ability to generalize from training data minimizing the over-fitting problem. Their use as regressors is probably less popular but even in this field they provide excellent performances [86]. In the case of linear and Gaussian systems, there is another important reason to use SVR in combination with Kalman filters (the optimal RBE filter in such a case). The noise model assumed by an SVR is Gaussian, with mean and covariance being random variables whose distributions depend on two of its parameters, C and ϵ, as discussed in the very interesting work of Pontil et al. [76]. The mean, in particular, is uniformly distributed between $-\epsilon$ and ϵ. Therefore, the SVR noise model is a superset of that assumed by the

Kalman filter, *i.e.* a zero-mean Gaussian. In other words, the SVR is a theoretically sound regressor to apply in all the situations where the Kalman is the optimal filter.

1.3.1 SVMs in ϵ-regression mode

To introduce SVMs as regressors, and in particular in ϵ-regression mode, let us have a quick look at the regression of a linear model given a series of data $(\mathbf{x}_i, y_i)$. In ϵ-regression mode the SVR tries to estimate a function of $\mathbf{x}$ that is far from training data y_i at most ϵ and is at the same time as flat as possible. The requirement of flatness comes from the theory of complexity developed by Vapnik [99] and ensures that we will get a solution with minimal complexity (hence, with better generalization abilities). In the linear case, the model we fit on the data is

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{1.1}$$

and the solution with minimal complexity is given by the one and only solution of the following convex optimization problem

$$\begin{gathered} min\tfrac{1}{2}||\mathbf{w}||^2 + C\textstyle\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b & \leq \quad \epsilon + \xi_i \\ y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b & \geq \quad -\epsilon - \xi_i^* \end{cases} \end{gathered} \tag{1.2}$$

The constant C is an algorithm parameter and weights the deviations from the model greater than ϵ. The problem is then usually solved using its dual form, that is easier to solve and to extend to estimate also non-linear functions ([99]).

1.3.2 SVRs for transition model estimation

In the context of RBE, given the first order Markovian assumption, one is left with two options to regress f_k:

1. to learn it from measures, that is to provide to the SVR as training data at time k the tuples

$$\langle \hat{\mathbf{x}}_1, \mathbf{z}_2 \rangle, \ldots, \langle \hat{\mathbf{x}}_{k-2}, \mathbf{z}_{k-1} \rangle \tag{1.3}$$

   where $\hat{\mathbf{x}}_k$ is the state vector estimate obtained from the recursive Bayesian filter at time k;

2. to learn if from states, that is to provide to the SVR as training data at time k the tuples

$$\langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \rangle, \ldots, \langle \hat{\mathbf{x}}_{k-2}, \hat{\mathbf{x}}_{k-1} \rangle. \tag{1.4}$$

Generally speaking, to learn the transition model from the relation between consecutive filtered states may cause the filter to repeatedly confirm itself, *i.e.* to regress the transition model that the filter itself is imposing on the training data. While this effect may guarantee a certain level of smoothness of the output, if this loop degenerates the filter trusts too much the learned model and diverges from the real state of the system by ignoring subsequent measures. On the other hand, learning form measures avoids this risk and results in a more responsive filter; yet, for the same reasons, it produces a filter more sensitive to noise, whose effects on the output of the filter or on the quality of the learned transition model cannot easily be mitigated. Therefore, we advocate the use of the learning from states strategy and will introduce a specific mechanism to avoid the degeneracy of the learning loop.

Since the SVR can only regress functions $f : \mathbb{R}^n \to \mathbb{R}$, if the state vector has dimension n, n SVRs are used, and each one is fed with tuples of the form $\langle \hat{\mathbf{x}}_{k-2}, \hat{\mathbf{x}}^i_{k-1} \rangle$, where the superscript i indicates the i-th component of a vector.

Another important design choice is the nature and length of the temporal window used to select states (or measures) for training. It does not make sense to use all the state transitions since the beginning of observations to learn the transition model for the current time slot, or, at least, it does not make sense during regression to equally weight their contributions. A solution that may be used to address this problem is dynamic SVR for time series regression, introduced by Cao and Gu [11]. While we believe that this may be beneficial, and can be an interesting investigation to carry on in the future, so far we have relied on a simpler solution, namely a sliding window of fixed length, to prevent too old samples from polluting the current estimate.

Finally, the influence of the time variable must be considered during regression. To understand this, consider the circular motion on the unit circle depicted in the leftmost chart of Fig.1.2. Assuming for clarity of the graphical explanation the state vector to be composed only by the x position of the point, some of the samples from which the SVR has to regress the transition model of this point are depicted in the second chart. As can be seen, without taking into account the evolution of the state through time, even with a perfect regression (represented by the dotted line in the second chart), it is impossible to have a correct prediction of the state at time t, given the state at time $t-1$: for example, at time $t = 4$ and $t = 6$ the previous state, x_{t-1}, is equal for the two positions, but the output of the regression should be different, namely $x_4 = -1$ and $x_6 = 0$. This situation can be disambiguated adding time as an input variable to the function to be regressed, as shown by the last chart.

To summarize, n SVRs are used, where n is the dimension of the state vector $\mathbf{x}_k$. The i-th SVR is trained at frame k by using the following training set

$$\{\langle k-2-W, \hat{\mathbf{x}}_{k-1-W}, \hat{\mathbf{x}}^{i}_{k-2-W}\rangle, ..., \langle k-1, \hat{\mathbf{x}}_{k-2}, \hat{\mathbf{x}}^{i}_{k-1}\rangle\} \tag{1.5}$$

where W is the length of the sliding window. We always use $W = 10$ in our experiments.

In the following section we address in detail the linear-Gaussian case, when the Kalman filter is the optimal solution, and show how our framework can be instantiated to successfully and advantageously adapt the transition matrix and the associated noise covariance matrix on-line.

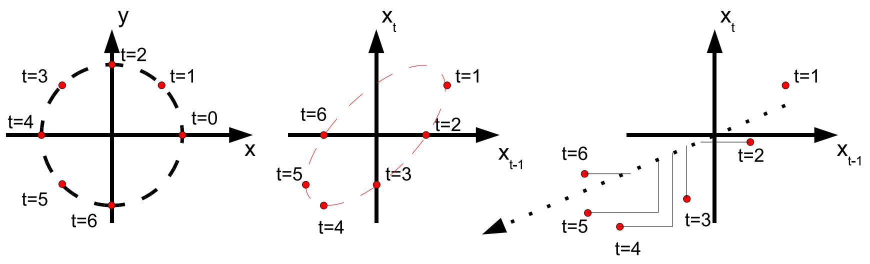


Figure 1.2: An example showing the importance of the inclusion of the temporal variable among those used for regression.

1.4 Support Vector Kalman

In the case of linear process and measurement functions, of Gaussian zero-mean noise and of Gaussian PDF for the initial state, all the subsequent PDFs of the state are (multivariate) Gaussians as well. Therefore, they are completely specified by their mean vector, that is usually considered also the estimation of the state, and their covariance matrix. The Kalman filter is the optimal estimator for this case.

Since between the hypotheses of the Kalman filter there is the linearity of f_k, two consequences immediately arise:

1. we must use a linear kernel, *i.e.* the SVR formulation introduced in 1.3.1;

2. we must modify it in order to regress a linear function.

In fact, the standard function learned by an SVR is (1.1), *i.e.* an affine mapping. As discussed by Poggio et al. [75], a linear mapping can be learned without harming the general theory underneath SVM algorithms, since the linear kernel is a positive definite kernel. Moreover, a solving algorithm for the linear mapping was also proposed in the paper of Platt [74] that introduced the standard and widespread solution for the affine case, *i.e.* the Sequential Minimal Optimization (SMO) algorithm.

Using this flavor of SVRs, it is possible, given the training data in the considered temporal window, to obtain an estimate of F_k. Each vector of weights $\mathbf{w}_k^i$ regressed by the i-th SVR at time k can be almost directly used as the i-th row of the estimated transition matrix $\hat{F}_k$. The last but not least issue to be solved in order to deploy the SVR weights as rows of the Kalman transition matrix is the problem of normalization.

Typical implementations of SVMs require the input and output to be normalized within the range $[0,1]$ or $[-1,+1]$. While this normalization is a neutral preprocessing as far as the SVR output is concerned, it has subtle consequences when the weight vectors of the SVR are used within our proposal. To illustrate this, let us consider a simple example where a mapping from a scalar x to y is regressed, and the variables are normalized to the range $[-1,+1]$. Then

$$\tilde{x} = \frac{2x - x_{max} - x_{min}}{x_{max} - x_{min}}, \quad \tilde{y} = \frac{2y - y_{max} - y_{min}}{y_{max} - y_{min}}, \tag{1.6}$$

where the superscript $\tilde{}$ denotes the normalized variables and x_{max}, x_{min} are the maximum and minimum value of the variable within the considered temporal window. Hence, the function of x that gives the unnormalized y is

$$\begin{aligned} \tilde{y} = w\tilde{x} \;\Rightarrow\; y = ax + b,\; a &= \frac{2(y_{max} - y_{min})w}{x_{max} - x_{min}} \\ b = y_{max} + y_{min} &- \frac{(y_{max} - y_{min})(x_{max} + x_{min})w}{x_{max} - x_{min}} \end{aligned} \tag{1.7}$$

i.e., again an affine mapping. Therefore, using the unnormalized coefficient a as an entry of the transition matrix $\hat{F}_k$ results in poor prediction, since the constant term is not taken into account. In order to obtain a linear mapping, that fits directly into the transition matrix of a Kalman filter, a two steps normalization must be carried out. Given a sequence of training data, a first normalization is applied,

$$\bar{x} = x - \frac{x_{max} + x_{min}}{2}, \quad \bar{y} = y - \frac{y_{max} + y_{min}}{2}. \tag{1.8}$$

These are the data on which the Kalman filter has to work. In other words, at every time step, the output of the previous time step must be renormalized if its value changes the minimum or maximum within the temporal window. This is equivalent to a translation of the origin of the state space and does not affect the Kalman filter itself. No normalization is required for the covariance matrix. After this normalization, the data can be scaled in the range $[-1, +1]$, as required by the SVR, according to

$$\tilde{x} = \frac{2}{\overline{x}_{max} - \overline{x}_{min}} \overline{x}, \quad \tilde{y} = \frac{2}{\overline{y}_{max} - \overline{y}_{min}} \overline{y} \tag{1.9}$$

where the subscripts have the same meaning as in (1.6). Using this two steps normalization, the unnormalized function of the Kalman data is

$$\tilde{y} = w\tilde{x} \;\; \Rightarrow \;\; \hat{y} = \frac{(\overline{y}_{max} - \overline{y}_{min})}{(\overline{x}_{max} - \overline{x}_{min})} w\overline{x}, \tag{1.10}$$

i.e. the required linear mapping.

1.4.1 Adaptive process noise model

As discussed in Sec. 1.2, the classical definition of an adaptive Kalman filter is more concerned with dynamic adjustment of $\mathbf{Q}_k$ than with the adaptation of the transition model [70, 109]. Our proposal makes it easy to learn on-line the value of $\mathbf{F}_k$, but provides also an effective and efficient way to dynamically adjust the value of the process noise. The value of $\mathbf{Q}_k$, in fact, is crucial for the performances of the Kalman filter. In particular, the ratio between the uncertainties on the transition model and on the measurements tunes the filter to be either more responsive but more sensitive to noise or smoother but with a greater latency in reacting to sharp changes in the dynamics of the observed system.

Within our framework, a probabilistic interpretation of the output of the SVR allows to dynamically quantify the degree of belief on the regressed transition model, and, consequently, the value of $\mathbf{Q}_k$. Some works have already addressed the probabilistic interpretation of the output of a SVR [13, 28, 51]. All of them estimate error bars on the prediction, *i.e.* the variance of the prediction. Therefore they are all suitable for estimating the Gaussian covariance matrix of the regression output. We choose to use [51] since it is the simplest method and turned out also the most effective in the comparison proposed in [51].

Given a training set, this method performs k-fold cross validation on it and considers the histogram of the residuals, *i.e.* the difference between the known function value at $\mathbf{x}_i$ and the value of the function regressed using only the training data not in the $\mathbf{x}_i$ fold. Then it fits a Gaussian or a Laplace PDF to the histogram, using a robust statistical test to select between the two PDFs. In our implementation, in accordance with the hypothesis of the Kalman filter, we avoid the test and always fit a Gaussian, *i.e.* we estimate the covariance as the mean squared residual. We also keep $\mathbf{Q}_k$ diagonal for simplicity. Hence, every SVR provides only the value of the diagonal entry of its row of $\mathbf{Q}_k$. As discussed before, however, learning from states is prone to degeneration of the learning loop into a filter unaffected by measurements. To avoid this, we prevent the covariance of every SVR to fall down a predetermined percentage of the corresponding entry of $\mathbf{R}$ (10% in our implementation). This has experimentally proved to be effective enough to avoid the coalescence of the filter while at the same time preserving its ability to dynamically adapt the values of $\mathbf{Q}$.

Finally, this method of estimation of the process noise covariance matrix allows for an intuitive interpretation of the C parameter of the SVRs. Since C weights the deviations from the regressed function greater than ϵ, it is directly related with the smoothness of the Support Vector Kalman output. In fact, if C is high, errors will be highly penalized, and the regressed function will tend to overfit the data, leading to greater residuals during the cross validation and to a bigger uncertainty on the transition model. This will result in a more noisy but more responsive output of the Kalman estimation. If, instead, C is low, the SVR output will be smoother and the residuals during the cross validation will be smaller. The resulting tighter covariances will guide the Kalman filter towards smoother estimates of the state.

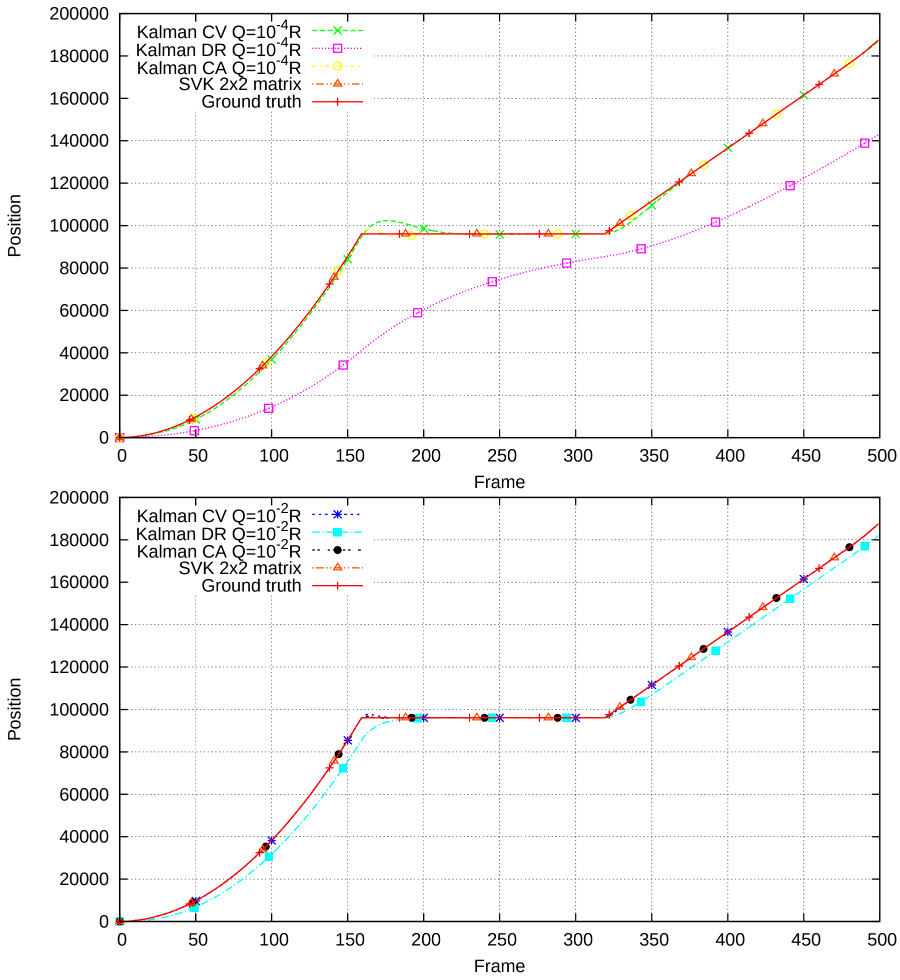


Figure 1.3: Charts showing the evolution of the filters against ground truth data in case of linear motion: the top one compares SVK to Kalman filters tuned for smoothness, the bottom one to Kalman filters tuned for responsiveness.

1.5 Experimental results

We provide first two simulations concerning a simple 1D estimation problem (*i.e.*, a point moving along a line). In the first experiment, the motion is kept within the assumptions required by the Kalman filter, in particular there is a linear relationship between consecutive states. In the second one, a case of non-linear motion is considered. Finally, we provide experimental results concerning tracking of the 3D position and orientation of a moving camera for real-time video augmentation and of tracking of various targets in the image plane.

1.5.1 Simulation of linear motion

In both simulations, comparisons have been carried out versus three Kalman filters adopting different motion models: drift (Kalman DR), constant velocity (Kalman CV) and constant acceleration (Kalman CA). Their model matrices are as follows:

$$\mathbf{F}_{DR} = [1]\,,\ \mathbf{F}_{CV} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix},\ \mathbf{F}_{CA} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.11}$$

Two different tunings were considered for each Kalman filter: a more responsive one, when $\mathbf{Q}$ has been set equal to $10^{-2}\mathbf{R}$; and a smoother one, with $\mathbf{Q} = 10^{-4}\mathbf{R}$. As far as SVK is concerned, it was fed with noisy measures of the position and the velocity of the point, therefore regressing a 2×2 model matrix. The only rough tuning regards C, which is set equal to 2^{-10} in this simulation and to 2 in the non-linear case: intuitively, an easier sequence allows for using a smoother filter.

During the linear motion sequence, motion is switched every 160 samples between a constant acceleration, a constant position and a constant velocity law. Therefore, each Kalman filter has a time frame wherein the real motion of the point is exactly that described by its transition matrix. Results on the whole sequence are reported in Fig.1.3 and Tab.1.1. As for simulation parameters, $\mathbf{R}$ has been kept constant in time and equal to $100 * \mathbf{I}$, with $\mathbf{I}$ denoting the identity matrix; constant acceleration was 30.0 m/s^2, constant velocity was 1000 m/s and Δt was 0.5. Gaussian noise with covariance matrix $\mathbf{R}$ was added to the data to produce noisy measurements for the filters.

As shown by the first column of Tab.1.1, our proposal achieves the best Root Mean Squared Error (RMSE) on the whole sequence. This

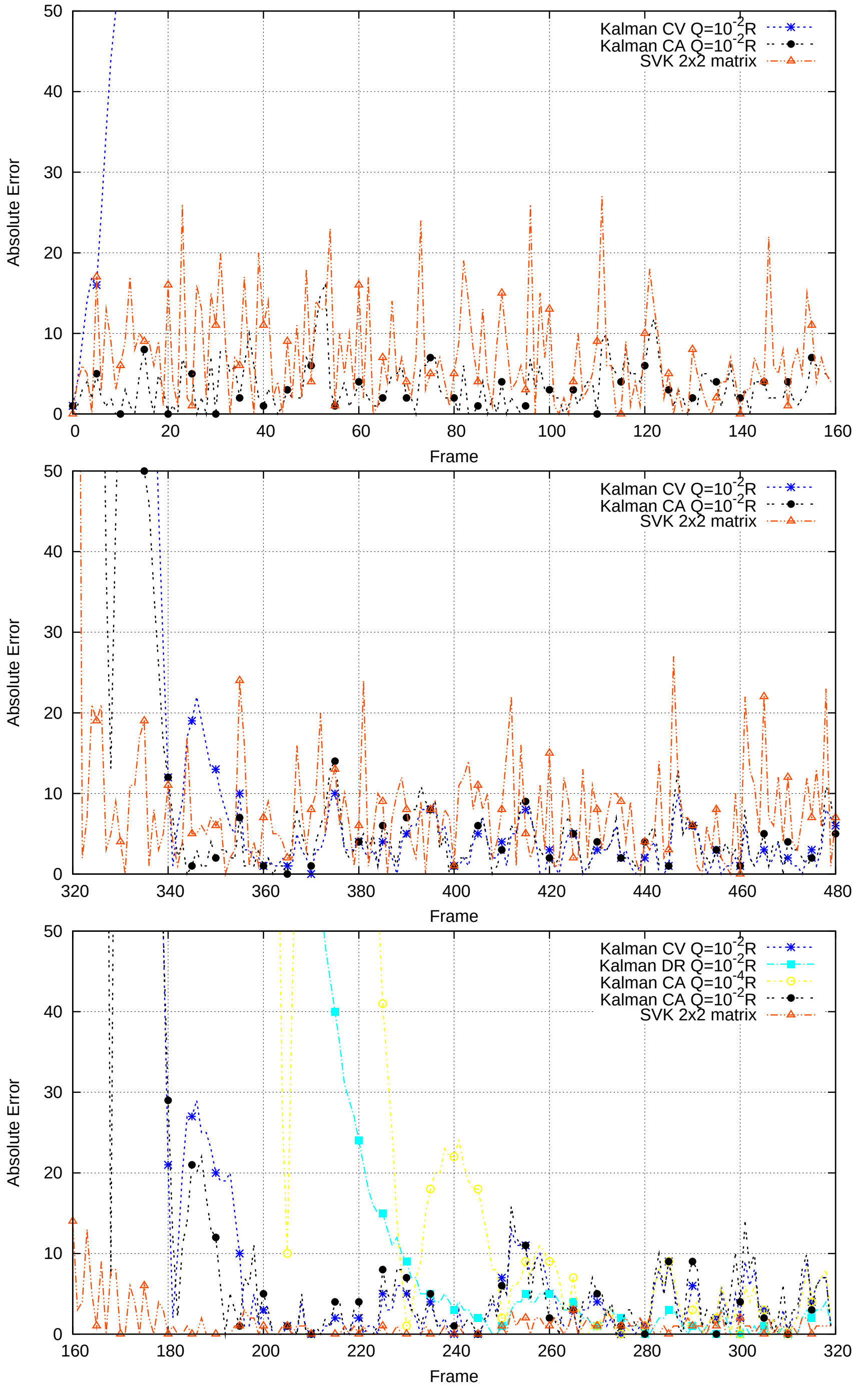


Figure 1.4: The charts report absolute errors for, respectively, the constant acceleration, the constant velocity and the constant position intervals.

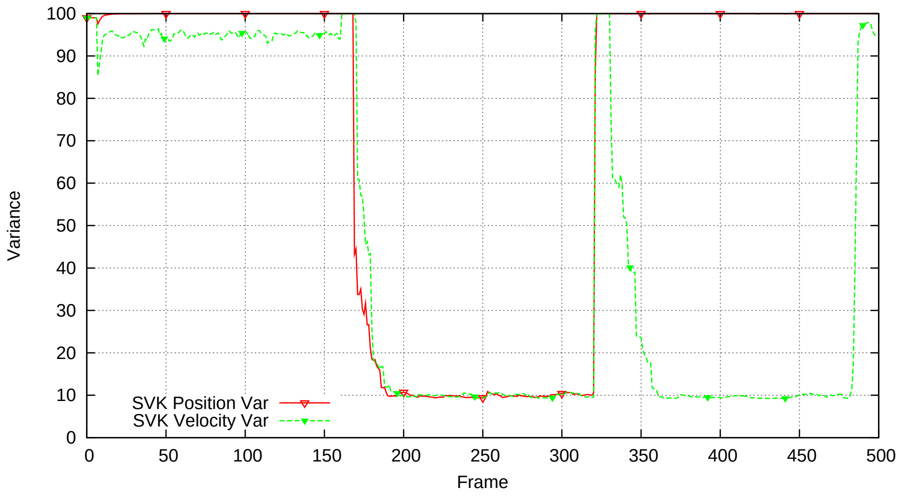


Figure 1.5: The chart shows the covariances on state variables provided by SVK throughout the whole sequence.

shows the benefits that on-line adaptation of the transition model can produce on the state estimate. This is also shown by the two charts in Fig.1.3. At the scale of the charts, the estimation of our filter is indistinguishable from the real state of the system, whereas the delay of Kalman DR and the overshots/undershots of Kalman CA and Kalman CV in presence of sharp changes of motion are clearly visible.

Going into more details, we separately analyze each of the three different parts of motion (Fig. 1.4). Here, we discuss not only the performance on the whole interval associated with each motion law, but, also, those achieved in the final part of each interval (*i.e.*, the last 80 samples). In fact, final samples allow to evaluate the accuracy of the steady state of the estimators, filtering out the impact of the delays due to the filter degree of responsiveness.

During the constant acceleration interval, Kalman CA performs best, both with the responsive and the smooth tuning. This is reasonable, since theoretically it is the optimal filter for this specific part of motion. Our filter, however, performs slightly worse than Kalman CA, but definitely better than Kalman CV and Kalman DR (2-nd column of tab.1.1). This is also demonstrated by the first chart of Fig. 1.3, which, for better visu-

Filter	Whole	CA	CV	Drift	CA*	CV*	Drift*
SVK 2x2 Model	22.41	9.79	38.02	35.41	8.91	9.63	1.67
Kalman CA $Q = 10^{-2}R$	76.62	4.83	51.3	125.87	4.59	4.55	6.06
Kalman CA $Q = 10^{-4}R$	357.45	4.26	242.19	581.52	3.72	4.04	7.87
Kalman CV $Q = 10^{-2}R$	227.38	100.12	155.13	355.71	104.84	3.74	5.31
Kalman CV $Q = 10^{-4}R$	1680.37	1213.78	1160.73	2439.37	1416.30	49.82	109.30
Kalman DR $Q = 10^{-2}R$	4498.51	6015.22	4536.67	1793.30	8056.45	4757.75	2.77
Kalman DR $Q = 10^{-4}R$	29698.38	25771.38	31583.97	29279.53	35763.45	37809.42	16743.08

Table 1.1: Comparison of RMSE on linear motion: first column reports the RMSEs on the whole sequence; then, partial RMSEs on each piece of motion are given as well as RMSEs concerning only the final part of each interval (marked with *), when the filter may have reached the steady state.

alization, displays only absolute errors less than 50. Only our filter stays in the visualized range, apart from the optimal one. When considering only the steady state part (5-th column of tab.1.1) the analysis does not change, partly because this interval is the very first one and, hence, there are no delays to recover, and partly because the Kalman CV and DR do not have the proper transition matrix for this part and, thus, cannot recover from errors.

During the constant velocity part, SVK has the best overall RMSE (3-rd column of tab.1.1). This is due to the delay accumulated by Kalman CV, theoretically the optimal filter, during the previous intervals. Therefore, we can highlight one of the major advantages brought in by SKV: in case of sharp changes of the motion law, dynamical update of parameters renders SVK even more accurate than the optimal filter due to its higher responsiveness. This is confirmed by Fig. 1.5, showing the position and the velocity variances estimated by SVK. It can be seen that, immediately after the change of motion from constant position to constant velocity at sample 320, both variances significantly increase, somehow ”detecting” such a change, thanks to the adaptive process noise modeling embodied into our filter. The resulting lower confidence in the predictions automatically turns the filter from smoothness to responsiveness, preventing the overshots/undershots exhibited by standard Kalman filters. After few samples the covariance on the velocity decreases again, proving that SVK has confidently learned the new model. Considering only the steady state (6-th column of tab.1.1) Kalman CV is, as expected, the best one. Unlike the CA interval, however, only the responsive tuning performs well since the smoother Kalman CV has accumulated too much delay to recover. This difference is due to the intrinsically higher smoothness of the CV model with respect to the CA one. Kalman CA, with both tunings, is the second best and this is also predictable since a constant velocity motion may be seen as a special case of a constant acceleration one. Again, SVK is by far closer to the optimal filters than to those adopting a wrong motion model and, visualizing only errors less than 50, it is the only one visible in the corresponding chart of Fig. 1.4, apart from the optimal ones.

Finally, due to the delay accumulated by the other filters, SVK turns out the best estimator also in the constant position interval (4-th column of Tab.1.1). As far as the steady state is concerned, all the filters exhibit a good RMSE apart from the very smooth ones, namely CV and DR tuned towards smoothness, since they do not recover from delays even after 80 samples. Unlike the other motion intervals, SVK keeps on being the best, even when the steady state only is considered. A reason for this is provided again by the chart of covariances (Fig. 1.5). During the constant position part, the SVR is able to regress a very good transition matrix and both the uncertainties are kept really low compared to the values in **R**. Therefore, the filter is highly smooth, as can be seen in the chart of absolute errors, and this keeps the RMSE low also in the last part.

Our proposal is robust to higher measurement noise, too. We report in Tab.1.2 the RMSEs for the same simulation, but with $\mathbf{R} = 1000\mathbf{I}$. Even in this case SVK turns out to be the overall best thanks to its adaptive behavior. Considerations similar to previous ones apply to the three different parts of motion.

To summarize, simulations with linear motion laws show that the proposed SVR-based approach to on-line adaptation of the transition model is an effective solution for the tracking problem when the assump-

Filter	Whole	Drift	CV	CA	Drift*	CV*	CA*
SVK 2x2 Model R=1000	43.36	36.36	67.93	31.35	5.23	30.56	28.29
Kalman CA $Q = 10^{-2}R$	79.65	130.17	52.94	15.36	19.17	14.3	14.52
Kalman CA $Q = 10^{-4}R$	357.69	581.70	242.46	13.33	17.28	10.94	11.75
Kalman CV $Q = 10^{-2}R$	228.08	356.26	156.61	100.97	16.81	11.71	106.77
Kalman CV $Q = 10^{-4}R$	1681.04	2439.48	1162.36	1214.90	106.66	49.56	1418.82
Kalman DR $Q = 10^{-2}R$	4500.00	1793.01	4539.23	6016.82	8.78	4761.46	8059.09
Kalman DR $Q = 10^{-4}R$	29699.11	29279.76	31584.70	25772.48	16742.06	37810.78	35764.94

Table 1.2: Comparison of RMSE between different filters in case of higher measurement noise.

R = 100	Whole	R=1000	Whole
SVK 2x2 Model	20.61	SVK 2x2 Model	47.98
Kalman CA resp.	61.92	Kalman CA resp.	62.32
Kalman CA smooth	308.32	Kalman CA smooth	308.66
Kalman CV resp.	72.69	Kalman CV resp.	72.95
Kalman CV smooth	248.30	Kalman CV smooth	248.46
Kalman DR resp.	143.63	Kalman DR resp.	144.87
Kalman DR smooth	434.83	Kalman DR smooth	435.20

Table 1.3: Comparison of RMSE on non-linear motion.

tion of stationary transition matrix cannot hold due to the tracked system undergoing significant changes in its motion traits.

1.5.2 Simulation of non-linear motion

Given its ability to dynamically adapt the transition matrix, we expect SVK to be superior to a standard Kalman filter also in the case of non-linear motion. In such a case, in fact, a time-varying linear function can approximate better than a fixed linear function the real non-linear motion. Hence, to assess its merits we have run simulations with a motion compound of two different sinusoidal parts linked by a constant position interval. The motion law of the two sinusoidal parts is as follows:

$$x_1(t) = 300t + 300\sin(2\pi t) + 300\cos(2\pi t), \tag{1.12}$$

$$x_2(t) = 300t - 300\sin(2\pi t) - 300\cos(2\pi t). \tag{1.13}$$

Aggregate results are shown in Fig. 1.6, Fig. 1.7 and Tab.1.3 for the same levels of measurement noise as in 1.5.1. Our filter proves again to be the overall best.

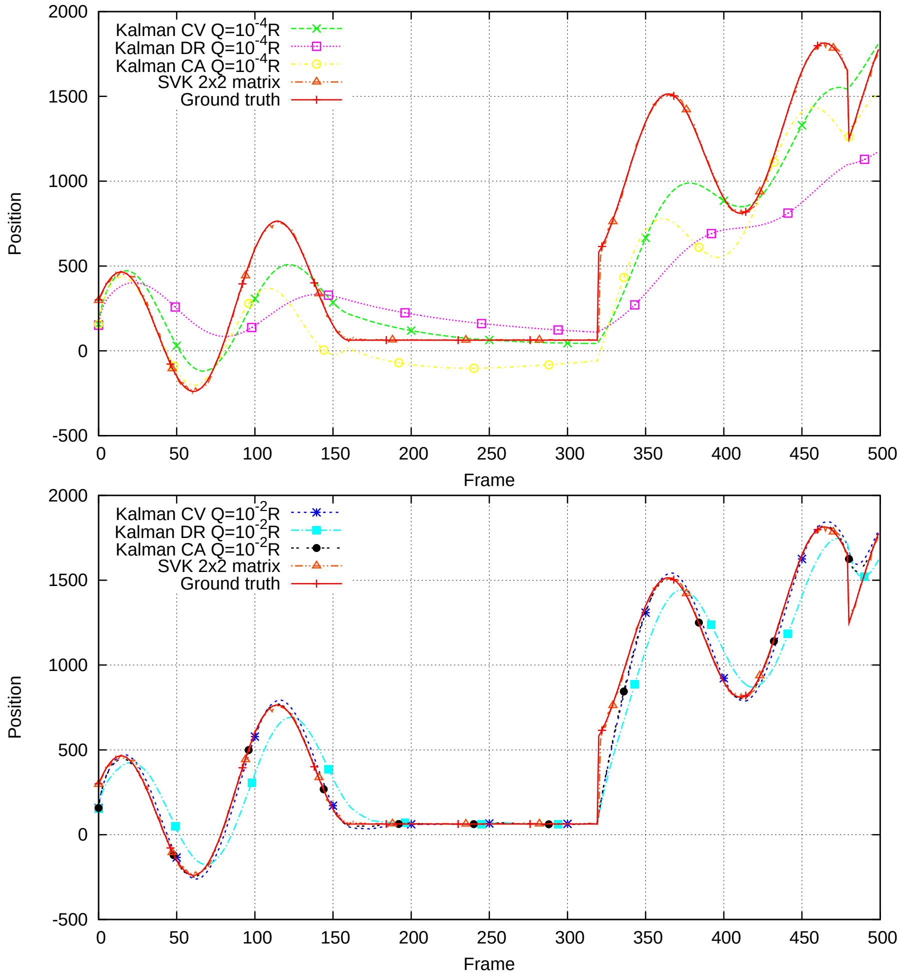


Figure 1.6: Simulation dealing with non-linear motion with $\mathbf{R} = 100\mathbf{I}$. Chart on top compares SVK to Kalman filters tuned for smoothness, the bottom one to Kalman filters tuned for responsiveness. At this scale, the estimation of our filter is almost indistinguishable from the ground truth.

1.5.3 3D camera tracking

In this experiment, we track the 3D position of a moving camera in order to augment the video content, taking as measurement the output of a standard pose estimation algorithm [81] fed with point correspondences established matching invariant local features, in particular SURF features [6]. Some snapshots are reported in Fig. 1.8. The snapshots show side-by-side the augmentation resulting from the use of Kalman CA and

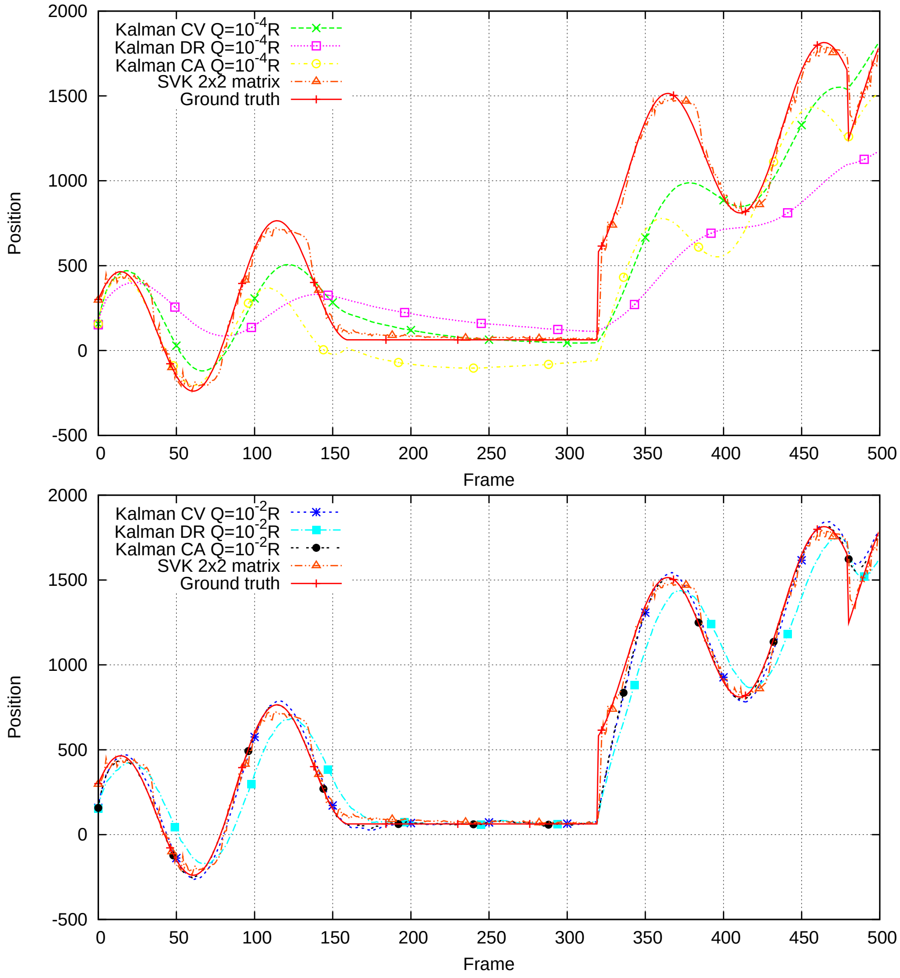


Figure 1.7: Simulation dealing with non-linear motion with $\mathbf{R} = 1000\mathbf{I}$. Chart on top compares SVK to Kalman filters tuned for smoothness, the bottom one to Kalman filters tuned for responsiveness.

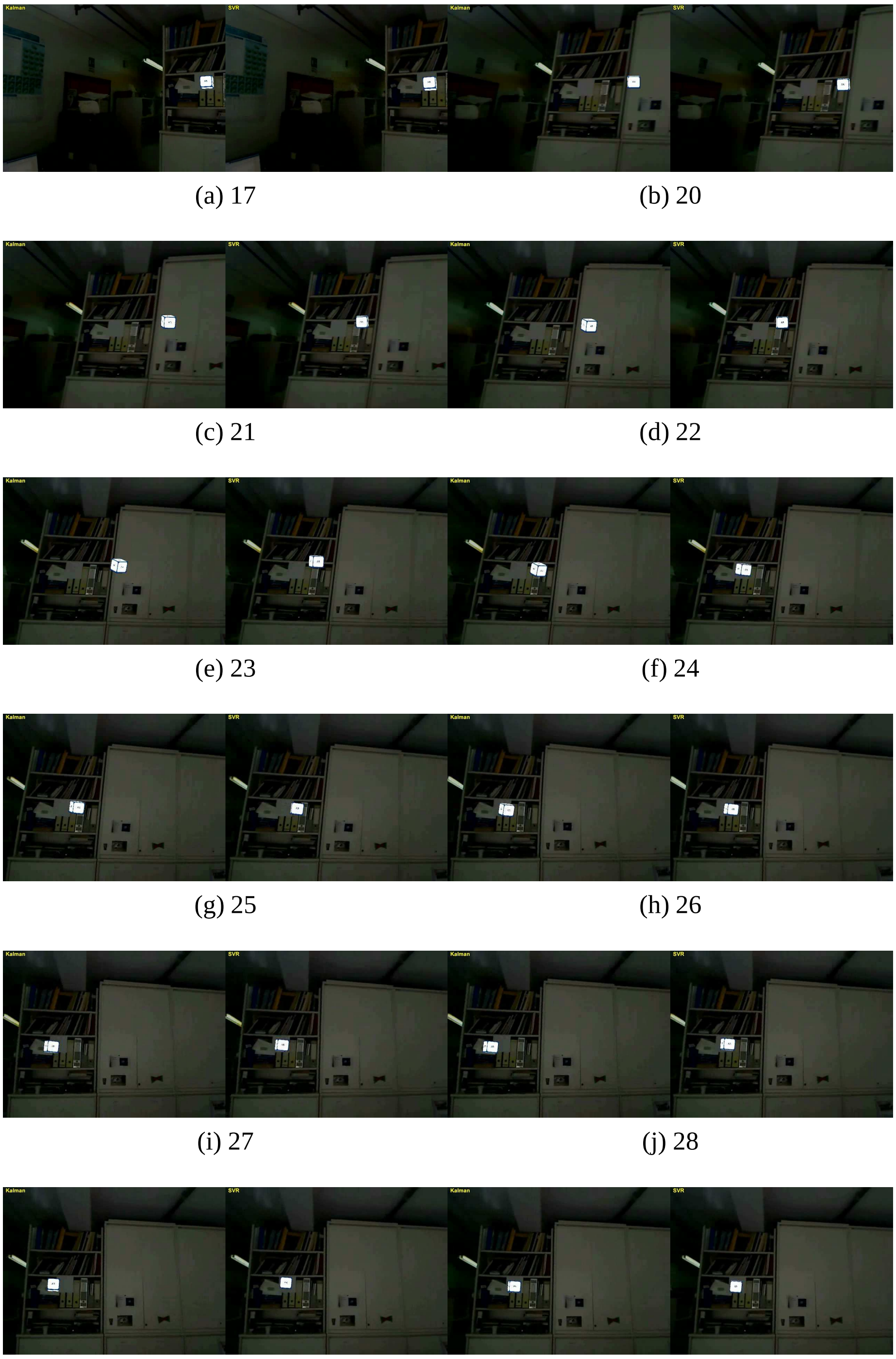
our SVK. Both filters have been tuned to be as responsive as in 1.5.2 and measurement noise covariances has been adjusted to match the range of the input data. The sequence shows a fast change of motion of the camera, the purpose of filters being to keep the virtual object spatially aligned with the reference position, denoted for easier interpretation of results by a white sheet of paper. We can see that both filters exhibit a delay following the sharp motion change at frame 19, but SVK is subject to a smaller translation error (*e.g.* frame 23), recovers much faster (SVK is again on the target by frame 27, Kalman CA only by frame 40) and, unlike Kalman CA, without any overshot (which Kalman CA exhibits from frames 27 to 40).

1.5.4 Mean-shift tracking through occlusions

In the last experiment, we compare our SVK to standard, non adaptive solutions for estimating an object trajectory in the image plane based on the mean-shift tracker introduced by Comaniciu et al. [17]. We compare the original mean-shift (MS) tracker and the non-adaptive Kalman filter (Kalman-MS tracker) to the SVK. Both KalmanMS and SVK use the MS tracker as the measurements source . The MS tracker and the Kalman-MS tracker have been proposed in the original work by Comaniciu et al. [17].

The MS tracker implicitly assumes a constant position motion model by letting the tracker start its search for the best position in each new exactly where the object was found in the previous frame. The Kalman-MS tracker in our experiment uses a constant velocity motion model.

Some snapshots of the test sequence are depicted in Fig. 1.9. The mean-shift technique is generally speaking not robust to total occlusions, like that shown in the third snapshot (Frame # 067), because the MS tracker can be attracted by the background structure (*e.g.* the road in our experiment) if this is more similar to the target than the occluder. For this reason the MS Tracker is unwilling to follow the object while it passes below the advertisement panel and stays in the last position where it could locate the target (frame # 067 of Fig. 1.9). The Kalman-MS tracker follows the previous dynamic of the target, thanks to the smoothness brought in by the Kalman filter transition model (frame # 067 of Fig. 1.9). Nevertheless, since the way it weights the contribution of the measure and the prediction on the state is fixed, it is finally caught back by the measures (the MS tracker) continuously claiming the presence of the target in the old location, before the occluder. Only the SVK is able to correctly guess the trajectory of the target while the lat-



(a) 17

(b) 20

(c) 21

(d) 22

(e) 23

(f) 24

(g) 25

(h) 26

(i) 27

(j) 28

(k) 34

(l) 40

Figure 1.8: Some of the most significant frames from the experiment on 3D camera tracking.

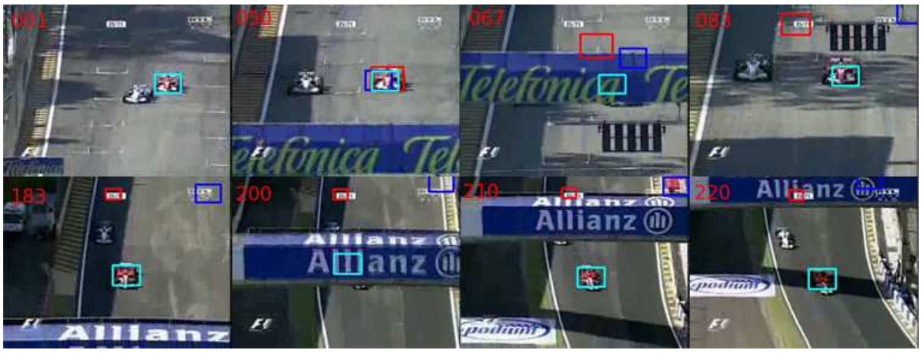


Figure 1.9: Some of the most significant frames from the experiment on object tracking in the image plane. In cyan the SVK tracker, red the MS tracker, blue the KalmanMS tracker.

ter is occluded (frame # 067 of Fig. 1.9) and continues to track it after the occlusion (frame # 083 and subsequent frames of Fig. 1.9). This is due to the ability of the SVK to dynamically adjust the process noise covariance matrix, increasing its confidence on the motion of the object (*i.e.* to decrease the variance) while the object keeps moving with an approximatively constant motion law on the image plane (first part of the sequence, first two snapshots, from frame # 001 to frame # 050 of Fig. 1.9). Thanks to the high confidence gained on the motion model, the filter is able to reject the wrong measurements coming from the MS tracker during the occlusion. This happens again during the second occlusions at frame # 200 of Fig. 1.9.

Chapter 2

Adaptive Appearance Model

Every visual tracker uses an internal representation of the appearance of the target, that it compares with the current frame I_k in order to locate the target. We refer to this internal representation as *appearance model* or *target model*, $\mathbf{A}$, and we denote the instance used by the tracker at time k as $\mathbf{A}_k$. This model is usually learned once, either offline from training data or online from the first frame(s).

In the works on tracking up to the last decade this model was usually fixed throughout the sequence [15, 17, 32, 37, 38, 78]. The main efforts of these works were devoted to develop robust ways to use the fixed model for locating the target in the current frame, despite all the nuisances that realistic video sequences may contain, such as clutter and distractors, illumination changes and deformable targets.

More recently, the idea of appearance model update has been proposed by several researchers to aim at successful long-term tracking despite these difficulties. By letting the model evolve across frames to include and adapt to the potential geometric and photometric changes of the target, these methods are inherently able to cope with target deformations and lightening variations. On the other hand, they expose the tracker to the risk of *drift, i.e.* the inclusion of background appearance in the appearance model that can eventually lead to loss of track.

In our work on adaptive appearance modeling we define the general structure of an adaptive modeling tracker and identify and discuss the main alternatives that have been proposed for each main building block of such systems. Recently, adaptive modeling trackers have been extended also to the multi-target case [8, 88, 103]. Our review, however, focuses on the single target case, that has reached a higher level of maturity. The multiple target trackers are covered by this review only as far as the part of their proposal covering single target tracking is concerned.

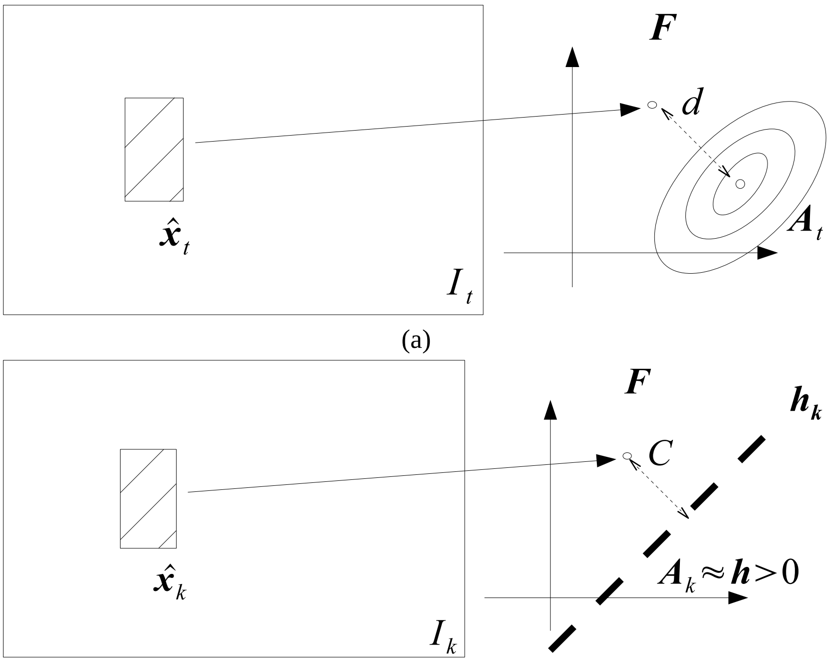
Then, we formulate our proposal for target model adaptation, based on the idea that tracking and target model update are similar in spirit and in practice: they both try to estimate the state of a system from noisy measures, under the assumption that the system state exhibits temporal consistency in consecutive frames. The state for target model update is the target appearance instead of the cinematic characteristics of the target, but the conceptual problems are highly similar. Therefore, we cast the problem of model update as a recursive Bayesian problem, and try to utilize the same tools, in particular the particle filter, to accomplish it. The work presented in this chapter has been carried out while the author was visiting Prof. Andrea Cavallaro's group within the Multimedia and Vision Group of the Queen Mary University of London.

2.1 Additional definitions

We presented the classical framework for visual tracking in the Introduction. Here, we add two notions that are used in the context of target model update, namely the confidence map and the division in generative and discriminative trackers.

2.1.1 Confidence map

Typically the tracker evaluates several state candidates $\hat{\mathbf{x}}_k^i$ to select the current state $\mathbf{x}_k$. The candidates are sampled according to a variety of strategies, but they typically belong to a neighborhood of the previous state. This enforces temporal smoothness, upon which tracking is based.



(a)

(b)

Figure 2.1: Generative versus discriminative trackers. A state candidate $\tilde{\mathbf{x}}_k$ from the current frame I_k is projected in the feature space $\mathcal{F}$ and its likelihood of being the target is computed. The likelihood is a function of a distance or similarity measure between the current model $\mathbf{A}_k$ and the candidate features in a generative tracker, a function of the confidence value of a classifier h_k in a discriminative tracker.

The evaluation results in the assignment of a score C_k^i to each candidate, (e.g. the weight of the corresponding particle in a particle filter [78], the feature similarity in a Mean-Shift tracker [17], the confidence of a classifier in a tracking-by-detection approach [4], ...). We refer to the set of pairs $\left\langle \hat{\mathbf{x}}_k^i, C_k^i \right\rangle$ as the confidence map $\mathbf{C}_k$.

2.1.2 Generative vs. Discriminative Trackers

An important classification of visual trackers, as far as target model update is concerned, is the division between generative and discriminative trackers (Fig. 2.1).

Generative Trackers

The tracker [107] [46] [57] [80] [40] [49] is guided by a generative observation likelihood, i.e. “the state estimation boils down to the problem of finding the state which has the most similar object appearance to the model in a maximum-likelihood or maximum-a-posterior formulation” [93]. Generative models of the foreground try to represent the object appearance without considering its discriminative power with respect to the background or other targets appearance. In these methods the observation likelihood is based on a similarity function defined on the feature space $\mathbf{F}$, that compares the current model $\mathbf{A}_k$ with the current candidate state $\tilde{\mathbf{x}}_k$ features providing a similarity score or likelihood of the candidate state (Fig. 2.1a). A model is explicitly given and the similarity to it assigns a likelihood value to every point of the feature space, *i.e.* to every possible state candidate.

Discriminative Trackers

The tracker [16] [4] [5] [29] [30] [89] [93] is guided by a discriminative observation likelihood, i.e. a classifier trained to learn “a decision boundary that can best separate the object and the background” [93] . Classifiers able to produce a confidence value for the predicted label can be used in this framework. In these proposals the appearance model $\mathbf{A}_k$ is not explicitly given, it is implicitly defined by the subset of the set of all possible appearances $\mathbf{F}$ that is positively labeled by the classifier (Fig. 2.1b). In these methods the observation likelihood is the confidence value of the classifier on the classification as foreground of the current candidate state $\tilde{\mathbf{x}}_k$, and it is 0 if the candidate is classified as background.

Hybrid Trackers

Some methods have proposed hybrid solutions such as: switching between discriminative and generative observation models according to the targets proximity in a multi-target scenario [88]; using co-training [7] between a long-term generative observation model and a short-term

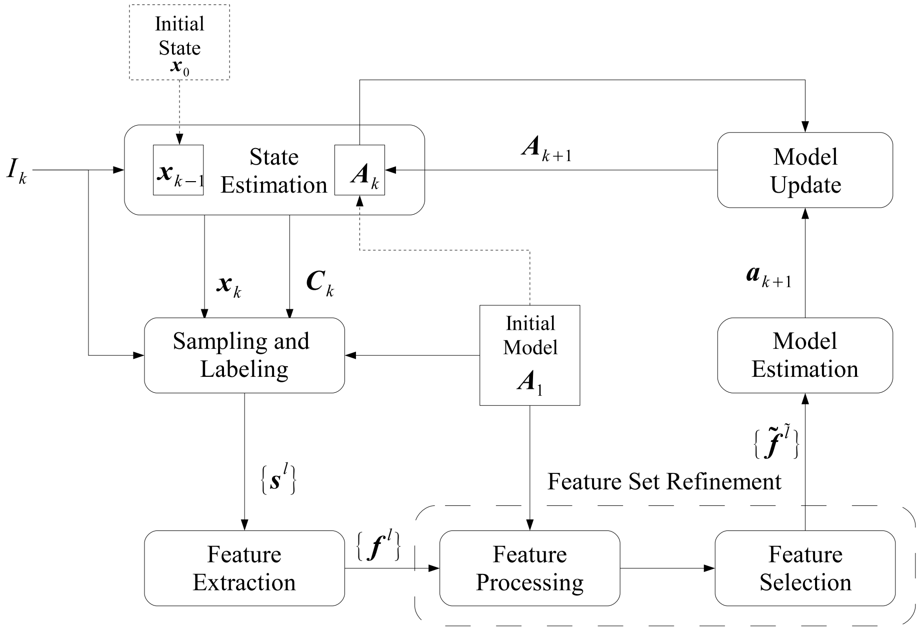


Figure 2.2: The general structure of the target model update flow in an adaptive tracker, $k \geq 1$.

discriminative one [105]; using several generative models but discriminatively learn in each frame the weights to combine them in order to maximize the distance with the neighboring regions [103]; store and update two generative non parametric models of foreground and background appearances and use them to train in every frame a discriminative tracker [55].

2.2 Elements of Adaptive Modeling in Visual Tracking

The general structure of an adaptive model tracker is sketched in Fig. 2.2.

1. Given the output of the tracker $\mathbf{x}_k$ and the confidence map $\mathbf{C}_k$ on the evaluated candidates, a set of samples $\mathbf{s}_i$ of the new target appearance are extracted from the current frame. If the tracker is a

	Sampling and Labeling	Feature Processing	Model Estimation	Model Update
Template Update [57]	Current State	Pivot Blended in	Direct Use of Features	Last model
IVT [80]	Current State	None	Direct Use of Features	Subspace
AdaptiveManifold [49]	Current State	None	Direct Use of Features	Manifold
$\mathcal{WSL}$ [40]	Current State	None	Direct Use of Features	Blending
Unified Bayesian [107]	Current State	Pivot Blended in	Direct Use of Features	Last model
Visual Tracking Decomposition [46]	Current State	Pivot Added	Direct Use of Features	Sliding Window
Ensemble Tracking [4]	Current State	Label Switch	New Classifier Training	Sliding Window
Non-Parametric Tracker [55]	Adaptive Classifier	Redundant and Outliers filtering	Direct Use of Features	Ranking
SVMs Co-Tracking - Tracker 1 [93] SVMs Co-Tracking - Tracker 2	Co-Training Co-Training	None None	Classifier Update Classifier Update	Sliding Window Sliding Window
Co-Training - Generative [105] Co-Training - Discriminative	Co-Training	None None	Direct Use of Features Classifier Update	Manifold Sliding Window
Adaptive Weights [103]	Current State	Pivot Blended in	Direct Use of Features	Blending
Discriminative Features Selection [16]	Current State	Pivot Blended in	Direct Use of Features	Last model
OnlineBoost [29]	Current State	None	New Classifier Training	Ranking
SemiBoost [30]	Fixed Classifier.	None	New Classifier Training	Ranking
BeyondSemiBoost [89]	Fixed and Adaptive Classifier.	None	New Classifier Training	Ranking
MILTracker [5]	Current-State-Centered	None	New Classifier Training	Ranking

Table 2.1: Reviewed Methods.

discriminative tracker, a set of samples are extracted also from the background. Samples are hard or soft labeled as target or background samples yielding a labeled sample set $\{\mathbf{s}_i^l, l \in [0,1]\}$.

2. Sample extracted from the current frame are projected into the feature space used for tracking, generating a set of labeled features $\{\mathbf{f}_i^l, l \in [0,1]\}$.

3. Feature can be filtered and/or selected.

   (a) Filtering: the set of features may be pruned to remove outliers or augmented with reliable features from trusted target appearances. Labels may be switched or modified, too.

   (b) Selection: if multiple cues are used as features (such as color, edges, shape, motion vectors, etc...) feature selection may be performed to select the most effective features for the current frame.

   These steps aim at providing a more representative and effective feature set $\{\tilde{\mathbf{f}}_i^l, \tilde{l} \in [0,1]\}$.

4. Given the selected labeled features the model $\mathbf{a}_{k+1}$ of the target in the current frame is estimated.

5. The model for the current frame $\mathbf{a}_{k+1}$ is merged with the previous

overall model $\mathbf{A}_k$, yielding the model $\mathbf{A}_{k+1}$ used in the next frame for state estimation.

This section describes the alternatives to implement each of these main building blocks.

To limit the chances of drift, an adaptive model tracker has to try to solve the following sub-problems:

- **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.

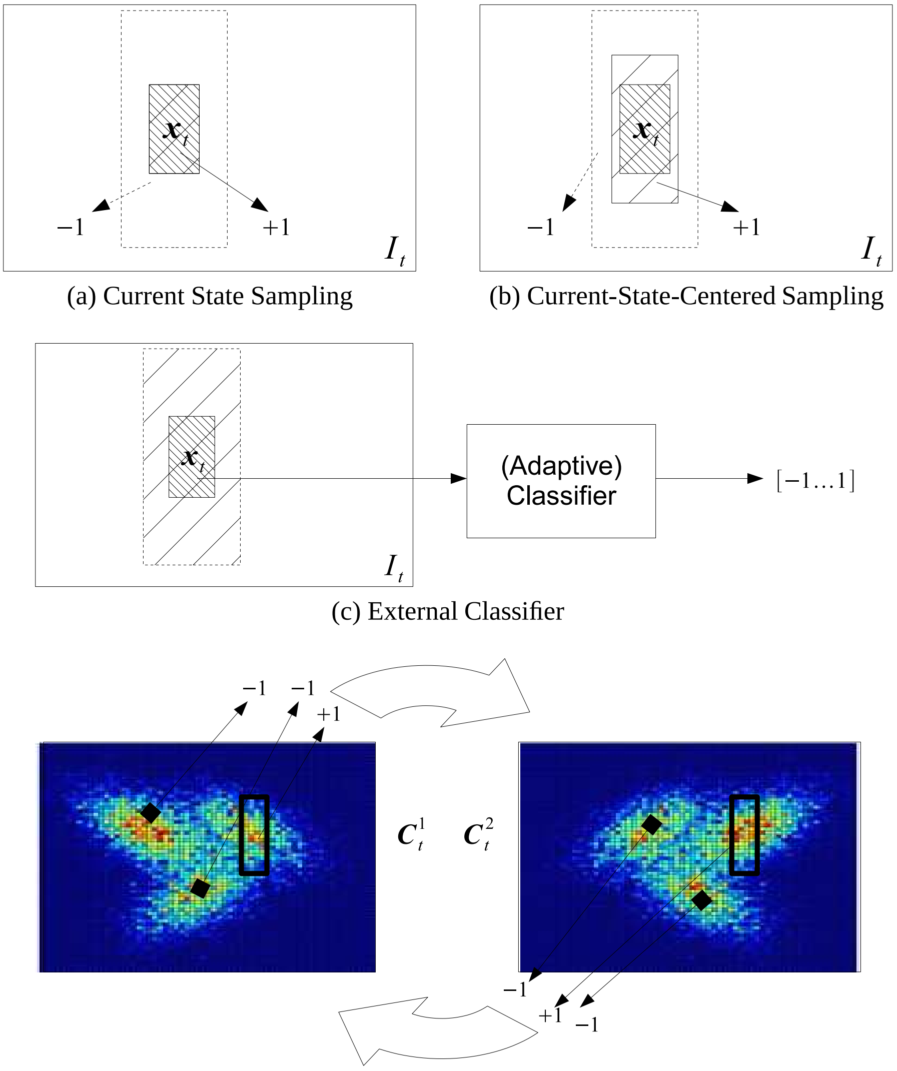
- **Stability/Plasticity Dilemma [31].** The simultaneous requirement for rapid learning and stable memory. This is a common problem of all on-line adaptive systems.

Each of the above mentioned building blocks deals with one or more of these sub-problems.

2.2.1 Sampling and Labeling

Given the output state $\mathbf{x}_k$ of the tracker in the current frame I_k and the confidence map $\mathbf{C}_k$, this step selects the regions of the current frame that are then used to update the model and, in a discriminative tracker, assign them either to the target or the background class.

The different proposals are presented according to the degree of reliability they assign to the tracker.



(a) Current State Sampling

(b) Current-State-Centered Sampling

(c) External Classifier

(d) Co-Training

Figure 2.3: Sampling and labeling strategies. In (a), (b) and (c) the thicker hatch represents the current state estimate, the wider hatch the sampling region for foreground labeled samples and the wider dotted rectangle defines the region for background samples. Note that in (c) the last two regions coincide. In (d), the images represent the confidence maps of two trackers: blue low likelihood, red high likelihood.

- **Current State (Fig. 2.3a).** The region defined by $\mathbf{x}_k$ is the only one used to update the target model. In case of discriminative trackers, samples from a region surrounding the current state are used as background appearance sample. This method assumes that the tracker is always correct and leaves to the subsequent stages the task of attenuating the effects of misaligned current states.

- **Current-State-Centered Sampling. (Fig. 2.3b).** Introduced in MILBoost [5]. Samples are extracted in the region defined by $\mathbf{x}_k$ plus its neighborhood. Samples extracted in the proximity of $\mathbf{x}_k$ are grouped in bags of samples and at least one sample of each bag is assumed to be a target sample whereas samples from the outer sampling region are used as samples for the background. It is up to the subsequent stages of the algorithm to disambiguate the uncertainty left in the target samples, for example by using Multiple Instance Learning as done in [5]. This method assumes that the tracker can by slightly off the target, but is always close to it.

- **Co-Training Sampling.** Introduced in Co-Tracking [93]. Two subtrackers that use independent features make up the tracker. The output $\mathbf{x}_k$ is given by the combination of their output, but the sampling and labeling for model update of each tracker is carried on independently, within the framework of co-training [7]. Each subtracker provides the training samples for the other. Target samples come from the global maxima of the other subtracker confidence maps whereas background samples are taken from the local maxima not overlapping with the global maximum. In this way, each subtracker is trained to be able to discriminate the cases that are difficult for the other tracker. This method assumes that in a given frame at least one of the two features alone is able to correctly track the target.

- **External Classifier (Fig. 2.3c.** Samples are extracted in the region defined by $\mathbf{x}_k$ plus its neighborhood but are not labeled according to their position with respect to $\mathbf{x}_k$. Instead, labeling is performed by means of an external classifier. Samples are soft labeled as samples of the target or the background according to the confidence of the classifier. Although this option makes sense for both generative and discriminative trackers, it has been used only by discriminative or hybrid approaches.

Generally speaking, the use of a classifier to guide the tracker updates is an interesting solution to break the self learning loop. Nevertheless it leads to a chicken-and-egg problem: if an external algorithm, like this classifier, can reliably tell if a patch selected from the output of the tracker belongs to the object of interest in spite of all the changes in appearance the target underwent, such a powerful algorithm could be successfully used as the observation model for the tracker and there would be no need to update the target model. Of course this is not the case: if the detector has to cope with all the possible changes it has to be updated as well, and this introduces the problem of drift for it, too.

By considering how the proposed solutions cope with the issue of classifier adaptability, this category can be further specified as follows:

– **Fixed Classifier.** Introduced in [30]. The classifier in this case may be an object detector or a similarity function with a fixed pivotal appearance model. It is created off-line or in the first frame and never updated. These methods assume that the classifier is able to cope with all the variations the target will undergoes in a sequence or, alternatively, that there will be no more variations of the target appearance than those that the classifier is invariant to. Therefore, this choice limits the degree of adaptability of the tracker. On the other hand, it does not make any assumption on the correctness of the current state, besides the proximity with the target.

- **Adaptive Classifier.** Introduced in [55]. The classifier is a similarity function with respect to the previous model. This method does not assume any reliability of the current state but it requires the absence of sudden changes in the target or background appearance evolution. Moreover, the degree of adaptability, *i.e.* the maximum variation in appearance between consecutive frames, is dictated by hard thresholds that may be difficult to set. Finally, by using the previous model to label the current samples, this method is prone to the drift introduced by self learning, although, unlike the other proposals, this loop is based on models rather than on states.
- **Fixed and Adaptive Classifiers.** Introduced in [89]. Two classifiers are used. One is fixed and its trained on the first frame. Another one is adaptive, and it is the one used to label the samples. This method tries to obtain the benefits of not assuming any correctness of the current state, introduced by using a classifier for samples labeling, without limiting the adaptability of the tracker, by letting the classifier adapt to target or background changes. This rises the problem of drift for the adaptive classifier. The proposed solution is to update the classifier only when the tracker and the fixed classifier are in agreement. Although this may limit the chances of drift for the adaptive classifier, it results in similar limits on the degree of adaptability introduced by the fixed classfier solution.

2.2.2 Feature Extraction

Features are extracted for each sample s_i^l of I_k, producing a set of labeled feature vectors $\{f^l\}$.

With reference to Tab. 2.2, we categorize features used by the adaptive modeling trackers according to the spatial extension of the features extracted from each sample. This has a direct impact on the ability of the tracker to correctly adapt in presence of partial occlusions:

- **Part-wise features.** Feature vectors are extracted from small patches or even single pixels . This makes it possible to reason explicitly about occlusions and to avoid to use features from the occluding object to update the target model. It also helps to deal with the approximation inherent to the modeling of the target as a rectangular object, since every feature can be classified either as foreground or background, even those laying inside the target bounding box.

- **Target-wise features.** Feature vectors represent the whole target appearance (*e.g.* color histograms [17]). As noted in the Introduction, this kind of features can typically tolerate target deformations and rigid transformations such as rotations and scaling even without model update. On the other hand, being a global representation of the target, it is difficult to correctly update it in presence of partial occlusions.

2.2.3 Feature Set Refinement

Given the features $\{f^l\}$ extracted and labeled from the current frame, this step processes the features and the labels in order to obtain a modified set $\{\tilde{f}^{\tilde{l}}\}$ that is more effective for model update. To this purpose, two main strategies have been followed, that can be deployed alternatively or sequentially: feature processing and feature selection.

Feature Processing

As fas as feature processing is concerned a tracker can perform:

- **Sample checking.** The idea behind the following filtering steps is that it is possible to decide a priori which samples are not suitable to perform model update given the current model. In particular some adaptive trackers perform:

			Parts						Target													
				Hist		Haar Filters			Template			Histogram					Contour	Detector		
			RBG values (pixel)	Color	HoGs [20]	SURF [6]	Steerable Filters [26]	Haar-like wavelets	Intensity	Separate HSI	Edges	Colour	HoGs [20]	LBP [69]	Bag of Features	Colour Filter Bank [16]	Ellipse	Head (ConvNN)	Human (HoGs [20])	Human (ISM [50])
Single	Template Update	[57]							x											
	IVT	[80]							x											
	AdaptiveManifold	[49]							x											
	$\mathcal{WSL}$	[40]					x													
Mixture	Ensable Tracking	[4]	x		x															
	Non-Parametric Tracker	[55]	x		x															
	Detector Confidence	[8]										x		x					x	x
	SVMs Co-Tracking - Tracker 1	[93]										x								
	SVMs Co-Tracking - Tracker 2												x							
	Co-Training - Generative	[105]							x											
	Co-Training - Discriminative												x							
	Unified Bayesian	[107]										x	x							
	Adaptive Weights**	[103]										x			x		x	x		
Selection	Discriminative Features Selection	[16]														x				
	OnlineBoost	[29]						x												
	SemiBoost	[30]						x												
	BeyondSemiBoost	[89]		x	x			x												
	MILTracker	[5]						x												
	Visual Tracking Decomposition*	[46]								x	x									

Table 2.2: Features. The single asterisk indicates use of multiple trackers, hence not all the features listed might be used in the same tracker. The double asterisk indicates the use of the Adaptive Multiple Features Blending strategy for the feature set composition (see Sec. 2.2.4).

- **Redundant Sample Removal.** Introduced in [55]. Feature vectors that are too similar to the current model are discarded as redundant.
- **Outliers filtering.** As far as outliers are concerned, two different strategies have been deployed:
  - **Outliers Removal.** Introduced in [55]. Feature vectors that are too different from the current model are discarded as outliers.
  - **Positive Label Switch.** Introduced in [4]. In case the confidence on a target-labeled feature vector is not high enough, the label is switched to background. This is done mainly to counteract the approximation inherent in the use of a rectangular box as target shape.

- **Pivot.** The initial appearance is used as a *pivot*, under the assumptions that the bounding box in the first frame was correct and that the target and the background appearance remains similar to the initial one in the feature space. In the proposals adopting this strategy, first frame data receive a special treatment: it is reasonable because usually first frame detection is assumed to be reliable, for example in a tag-and-track application for visual surveillance, where a human operator provides the first bounding box. For a full automatic deployment of tracking the first bounding box cannot be assumed to be particularly more accurate than the next ones. Another important issue with the use of the pivot for samples refinement is that it may not allow to adapt to sudden appearance changes nor to gradual changes in appearance that in the long run lead to great changes in target appearance compared to the first frame. This, depending on the application, may be a limitation that prevent the adoption of this filtering step. If general automatic visual tracking is the aim of an algorithm, then this filtering step should not be used, although it can greatly improve performances in more specific contexts. Use of features from the pivot to refine the current sample set has been proposed in two flavors:

– **Pivot added.** Features from samples of the pivot are added to the feature set with the proper label. With this strategy, subsequent stages of the algorithm can decide to ignore the added features and exploit only the features from the current frame for the update.

– **Pivot blended in.** Feature vectors are blended with the pivot features. With this choice the influence of the pivot cannot be discarded afterwards. On the other hand, the model update is guaranteed to keep the model in a neighborhood of the initial appearance, hence this solution trades off adaptability for robustness.

2.2.4 Feature selection

This is a key component of a generation of recently proposed family of discriminative tracking algorithms [5, 29, 30, 89] that perform model update by continuously updating the set of used features, selecting them according to their discriminative ability in distinguishing the target from the background. Beside these methods, that heavily base their efficacy on feature selection, feature selection is a fundamental step for all adaptive and even non-adaptive trackers, since different cues, such as edge patterns, color histograms or appearance patterns, may have a different ability to track a target in different parts of the sequence. Nevertheless, no standard way has emerged to tackle this fundamental problem. One of the main difficulties in performing on-line selection is given by the fact that different cues may have different score dynamics and ranges, which makes it hard to compare their effectiveness directly by comparing their scores. They can be compared by evaluating *a posteriori* their effects on the tracker accuracy, for example selecting the features to use at frame k by ranking them according to their effectiveness in locating the target in the previous frame $k-1$, under the assumption that the position estimated by the tracker at frame $k-1$ is correct. According to their treatment of this stage, trackers can be categorized in three classes (see also the vertical left-most column of Table 2.2):

- **Single Feature.** Only one kind of feature is used, *e.g.* one color histogram. No selection is carried out.

- **Mixture of (Independent) Features.** A fixed set of features is used. The composition of the set is never updated. Usually a certain degree of independence between the features is required (or assumed) for their simultaneous use to be effective. This is for example the case of trackers working in the co-training framework, that implicitly perform feature selection by weighting the contribution to the final estimation of classifiers using independent features.

- **On-line Feature Selection.** A fixed set of features is used. The composition of the subset used in each frame is updated according to the features effectiveness in the previous frame(s) [16].
    - **Online Boosting** Feature Selection is performed by applying on-line boosting [72] to weak classifiers that act as feature selectors [29].
- **Adaptive Multiple Features Weighting.** A fixed set of features is used. The weights of the features in the likelihood composition are updated in every frame based on the features effectiveness in the previous frame(s).

2.2.5 Model Estimation

Given the filtered feature set and the labels, a new partial model $\mathbf{a}_{k+1}$ that describe the target appearance in the current frame is built. This has no particular influence on the adaptation abilities of the tracker nor on its risk to drift. The main alternatives are:

- **Non parametric use of features.** The model estimated for the current frame is the non parametric ensemble of the feature extracted from the target or the background.
- **New Classifier Training.** The current samples are used to train a classifier that best separates the target and the background in the current frame.
- **Old Classifier(s) Update.** The current samples are used to update a previously trained classifier.

2.2.6 Model Update

Given the new model for the current frame $\mathbf{a}_{k+1}$, it has to be merged with the overall model used so far, $\mathbf{A}_k$, to obtain $\mathbf{A}_{k+1}$. This step directly addresses the Stability/Plasticity Dilemma presented above. Solutions are presented in order of Plasticity, *i.e.* starting from the most adaptive ones:

- **Last model only.** The result of the last frame is used as the model for the next frame.

- **Sliding Window.** A fixed amount of samples/classifiers is kept after every frame is processed. The newest is added and the oldest is discarded.

- **Ranking.** Up to a maximum fixed amount of samples/classifiers, the most effective ones are kept after every frame is processed, the new one is always added. This raises the problem of assessing their effectiveness, similar to the problem of evaluating features selection on-line. And again, the most widespread solution is to evaluate the models efficacy on the previous frame(s).

- **Blending.** Sample or classifier parameters estimated from the current frame are blended with their previous values. This in principle is more stable than the previous alternatives, since all the history up to the current frame has an influence on the new model. On the other hand, it is more prone to drift, since the inclusion of wrong samples for the target model cannot be fixed afterwards, only the inclusion of correct samples will eventually render the influence of the outlier negligible.

- **Subspace/Manifold.** A subspace or a set of subspaces (an approximation for a manifold in the feature space) is updated with the new sample from the current frame. It potentially retains the history of all the target appearances with a fixed amount of memory, hence it is the most stable solution. On the other hand, it is difficult to accommodate for sudden target appearance changes with such a model. Sometimes a forgetting factor is used to diminish through time the effect of the oldest samples on the subspace/manifold shape.

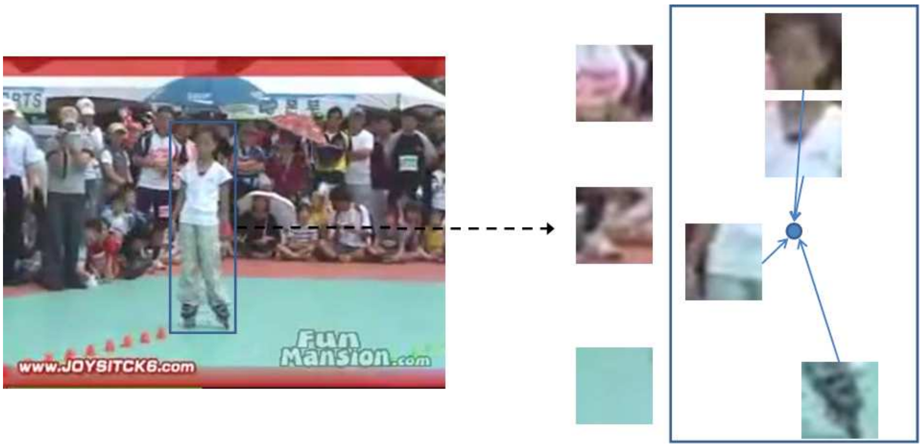


Figure 2.4: The patch based appearance model in our proposal.

2.3 Adaptive modeling with Particle Filtering

At the basis of our proposal lays the intuition that we can substitute some of the fundamental stages of the target model update algorithm described so far with equivalent steps performed by a particle filter estimating the target appearance.

Hence, in our proposal two RBE trackers are used. One tracks the target state, the other the target model. Since inference on high dimensionality spaces is hard and inefficient, we actually use an approximation of the particle filter when tracking appearance. Hence, although our formulation is deeply inspired by this filter and can easily be interpreted and implemented following its usual patterns, the appearance tracker is not strictly speaking a Bayesian filter. In particular, it is our definition of the observation likelihood that is not conformant, as detailed in the next sections.

The appearance model in our proposal is a part-based, Generalized Hough Transform-like model ([50], [1], [45]). It has been inspired also by the bag of patches non-parametric model of [55]. It offers several advantages over a global representation: it captures a coarse geometric structure of the target instead of global properties only; it naturally allows for dealing with partial occlusions; it can be used to obtained a segmentation of the target [50]. We model both the foreground and the background, in the spirit of recent discriminative trackers. Hence, our model is compound by a model for each class

$$\mathbf{A}_k = \{\mathbf{A}_k^{\mathcal{F}}, \mathbf{A}_k^{\mathcal{B}}\} \tag{2.1}$$

where the models are a set of graylevel square patches T of fixed side r with their geometric displacements v with respect to the object center (Fig. 2.4)

$$\mathbf{A}_k^{\mathcal{F}(\mathcal{B})} = \{(\mathbf{s}_k^i\}_{i=1}^M = \{(T_k^i, v_k^i)\}_{i=1}^M \qquad T_k^i \in [0, 255]^{r^2}, v_k^i \in \mathbb{R}^2 \tag{2.2}$$

The particle filter tracking the state of the target has the bounding box center coordinates as state variable and the current frame as measure. The tracker of the appearance, instead, has a patch and its displacement as state variable and the pair formed by the current frame and the current state estimation as measure. In fact, it is the output of the tracker estimating the bounding box that provides a new measure of the target appearance for the model update and, symmetrically, the tracker estimating the appearance provides a new model to update the state in the next frame. In other words, let

$$\mathbf{z}_k = I_k \tag{2.3}$$

$$\mathbf{y}_k = (\mathbf{x}_k, I_k) \tag{2.4}$$

denote the measure for the state tracker and the appearance, respectively. Then, the particle filter estimating the state computes the standard recursion:

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) \propto p(I_k \mid \mathbf{x}_k) \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{z}_{1:k-1})\, d\mathbf{x}_{k-1} \tag{2.5}$$

and then the particle filter estimating the appearance solves:

$$p(\mathbf{s}_{k+1} \mid \mathbf{y}_{1:k}) \propto p(\mathbf{y}_k \mid \mathbf{s}_{k+1}) \int p(\mathbf{s}_{k+1} \mid \mathbf{s}_k) p(\mathbf{s}_k \mid \mathbf{y}_{1:k-1})\, d\mathbf{y}_{k-1} \tag{2.6}$$

Given this formalization of model update as appearance tracking, in our proposal we replace (compare Fig. 2.5 with Fig. 2.2):

- the standard sampling and labeling step with the propagation of the appearance particles to the next frame, *i.e.* by sampling from the proposal on appearance $q(\mathbf{s}_{k+1} \mid \mathbf{s}_k, \mathbf{y}_{k+1})$.

- the sample refinement, in particular the sample processing, with the update step of the appearance particle filter, which dynamically weights samples according to the likelihood on appearance $p(\mathbf{y}_k|\mathbf{s}_{k+1})$ (in principle the update step can carry on also the on-line feature selection but is not done in our proposal yet);

- the model estimation for the current frame with the resampling step of the appearance tracker, which probabilistically discards down-weighted samples from the previous step and effectively produces the model that best explain the current frame, given the observations up to the current frame.

In the following we define the basic components of the particle filters we use to estimate the state and the appearance.

Appearance Proposal Density

$$q(\mathbf{s}_{k+1} \mid \mathbf{s}_k, \mathbf{y}_k) = q(\mathbf{s}_{k+1} \mid T_k, \mathbf{s}_k, I_k, \mathbf{x}_k) \tag{2.7}$$

To sample from it, we sample a new displacement with Gaussian Brownian motion relative to the displacement of this patch in the previous frame, v_k, and then extract a patch from I_k centered in the position given by the new displacement applied to $\mathbf{x}_k$. This gives a new particle to approximate the new posterior PDF on appearance.

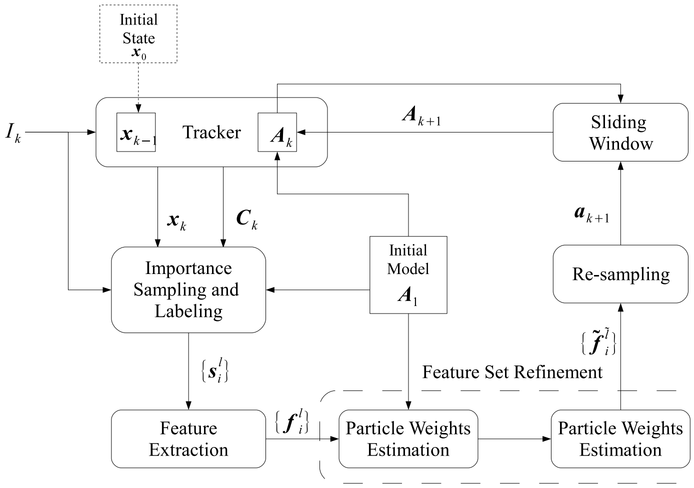


Figure 2.5: The structure of the target model update flow in our adaptive tracker, $k \geq 1$.

$$\hat{\mathbf{s}}_{k+1} = (\hat{T}_{k+1}, \hat{v}_{k+1}) \sim q(\mathbf{s}_{k+1} \mid \mathbf{s}_k, I_k, \mathbf{x}_k) \Leftrightarrow \tag{2.8}$$
$$\hat{v}_{k+1} \sim \mathcal{N}(\mu = v_k, \Sigma = \Sigma_v), \hat{T}_{k+1} = I_k\big|_{\mathbf{x}_k, v_{k+1}}$$

where, to indicate the extraction from a frame I_k of a patch defined by a displacement v with respect to a bounding box $\mathbf{x}$ with $I_t\big|_{\mathbf{x},v}$.

Our proposal density is a full definition of a proposal for particle filtering since it depends on both the previous state $\mathbf{s}_k$ and the current measure $\mathbf{y}_k$, whereas the classical proposal used in a particle filter discards the dependency on the current measure. In particular, we exploit the current measure to sample the new appearance of the patch, since to generate it according to a generative model of illumination changes and object deformations requires these models, which are difficult to obtain for a general purpose tracker, and it also requires to explore a high dimensionality space (*i.e.*, given the side of the patches r, the dimensionality of the space is r^2 and we use $r \sim 20$), which in turn requires a huge number of particles to obtain an acceptable approximation of the posterior. By letting the current measure guide the exploration of the state space we avoid these problems and obtain an efficient algorithm. Finally, the proposal density in our method accounts also for deformable objects by letting a patch move inside the object.

Appearance Observation Likelihood

$$p(\mathbf{y}_k \mid \mathbf{s}_{k+1}) = p(I_t, \mathbf{x}_k \mid T_{k+1}, v_{k+1}) \tag{2.9}$$

The likelihood of the measure under the hypothesis that the patch $\mathbf{s}_{k+1}$ belongs to the appearance model is where our proposal differs with respect to a standard particle filter. In particular, having exploited the current measure to guide the state space exploration and to sample the new patch appearance for $\mathbf{s}_{k+1}$, we cannot define the likelihood in terms of it, since $\mathbf{s}_{k+1}$ depends on $\mathbf{y}_k$. Therefore, we define the likelihood of $\mathbf{s}_{k+1}$ in terms of the particles of the distribution of the other class, *i.e.* we use the particles of the background class to assess the likelihood of the foreground particles and vice versa. Note that this way to evaluate $p(\mathbf{y}_k \mid \mathbf{s}_{k+1})$ implicitly takes still into account the measure $\mathbf{y}_k$, since the patches from both classes come from $\mathbf{y}_k$ through the proposal density.

We base our likelihood on the Zero-mean Normalized Cross Correlation (ZNCC). When applied to graylevel patches, this measure computes the similarity of the patches and is invariant to affine changes of the illumination. Therefore, the likelihood in our algorithm accounts for the robustness towards photometric changes of the target. The ZNCC of two vectors $\mathbf{a}, \mathbf{b}$ is defines as

$$ZNCC(\mathbf{a}, \mathbf{b}) = \frac{(\mathbf{a} - \mu(\mathbf{a})\mathbf{1})(\mathbf{b} - \mu(\mathbf{b})\mathbf{1})}{|\mathbf{a} - \mu(\mathbf{a})\mathbf{1}|\,|\mathbf{b} - \mu(\mathbf{b})\mathbf{1}|} \tag{2.10}$$

where $\mathbf{1}$ is the vector of 1s of the same dimension of $\mathbf{a}$ and $\mathbf{b}$, $\mu(\mathbf{x})$ is the mean of the components of the vector $\mathbf{x}$ and $|\mathbf{x}|$ its norm. Let

$$\bar{j} = \arg\max_{j=1,\dots,M} ZNCC\left(T_{k+1}, \bar{T}^{j}_{k+1}\right) \tag{2.11}$$

where $\bar{T}^{j}_{k+1}$ stand for the j-th particle of the other class with respect to the class of T_{k+1}. Then we compute the likelihood as

$$p(I_k, \mathbf{x}_k \mid T_{k+1}, v_{k+1}) \propto \exp(\frac{1 - ZNCC(T_{k+1}, \bar{T}^{\bar{j}}_{k+1})}{2}) \, . \tag{2.12}$$

Our definition of the likelihood is discriminative: the weight of each particle of the appearance model is higher the more discriminative with respect to the other class the particle is. This means that the resampling stage will be able to discard the particles not useful to track the target when estimating the model for the current frame. In other words, the weights computation performed with our likelihood realizes the Feature Processing stage of the scheme for model update presented before. If besides graylevel patches other features are used, their weighting and the subsequent resampling would effectively perform also probabilistic feature selection. The main difficulty to successfully carry out feature selection in this way is represented, as discuss in the previous section, by the different scales and dynamic responses of the similarity functions used to compare the features (*e.g.* the Bhattacharyya distance for histograms versus the ZNCC for patches), that makes it difficult to obtain comparable likelihood values.

State Proposal Density We employ a standard Gaussian proposal with a fixed, diagonal covariance matrix Σ_x.

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, I_k) = \mathcal{N}(x_k, \mu = \mathbf{x}_{k-1}, \Sigma = \Sigma_x) \tag{2.13}$$

State Observation Likelihood

$$p(I_k \mid \mathbf{x}_k) \tag{2.14}$$

Given the model estimated on the previous frame $\mathbf{A}_k = \{\mathbf{A}_k^{\mathcal{F}}, \mathbf{A}_k^{\mathcal{B}}\}$ let

$$\bar{j}_i = \arg\max_{j=1,\ldots,M} ZNCC\left(I_k\big|_{\mathbf{x}_k, v_k^i}, T_{k+1}^j\right) \qquad \forall\ \mathbf{s}_k^i \in \mathbf{A}_k^{\mathcal{F}} \tag{2.15}$$

i.e. for each foreground particle the index points to the patch in the background model that is the most similar to the current frame in the location given by the foreground particle displacement. In other words, it indicates the particle of the background that best explains the foreground appearance, given that the target is really at $\mathbf{x}_k$. Then, we compute the state likelihood as

$$p(I_k \mid \mathbf{x}_k) \propto \exp\left(\frac{1}{M}\sum_{i=1}^{M} \max(0, ZNCC(T_k^i, I_k\big|_{\mathbf{x}_k, v_k^i}) - ZNCC(T_k^{\bar{j}_i}, I_k\big|_{\mathbf{x}_k, v_k^i})\right) \quad (2.16)$$

i.e. as the mean likelihood obtained by the candidate $\mathbf{x}_k$ over all the particles of the foreground model, where the likelihood of a candidate with respect to a particle of the foreground is given by the similarity with the foreground patch and the dissimilarity from the best background patch of the patch at the location identified by the foreground particle displacement. This definition of the likelihood naturally deals with partial occlusions. To overcome also total occlusions we have to increase the stability of our algorithm by using one of the strategies introduced in Sec. 2.2.6. We deployed the sliding window strategy since it is the simplest, most efficient one and the overall probabilistic inference structure of our proposal already provides robustness against outliers, such as those included in the target model during occlusions. To include the sliding window strategy in our proposal the appearance tracker particles are no more patches with displacements, but sliding windows of patches and displacements. The proposal density is identical, whereas both likelihood values are computed as the average over the sliding window of the likelihoods for a single patch, presented above.

2.4 Experimental Results

2.4.1 Methodology

Trackers are initialized with the first bounding box available in the ground truth. Probabilistic trackers have been run 10 times and the mean of these runs is used for comparison with other trackers but the error bars for these trackers are plotted in the charts as well. Comparable or even better mean scores are not enough to assess that a probabilistic tracker is to be preferred: if the variance is higher the tracker is less reliable and, hence, less useful in a real deployment.

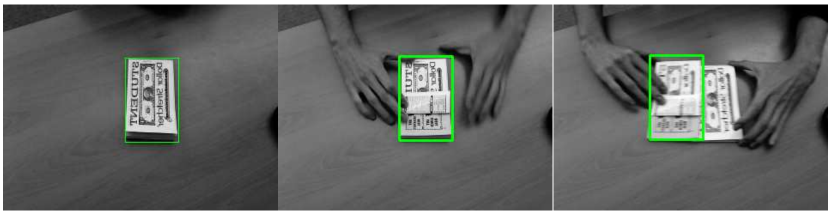
Two charts are used for each sequence. One reports the dice overlap with the ground truth in each frame of the sequence. *i.e.* the mean value of the ratio between 2 times the area of the intersection of the ground truth bounding box with the estimated bounding box and the sum of their areas:

$$d_k = \frac{2\left|\mathbf{x}_k \cap \mathbf{x}_k^{GT}\right|}{\left|\mathbf{x}_k\right| + \left|\mathbf{x}_k^{GT}\right|}\ . \tag{2.17}$$

This performance index varies in $[0, 1]$, the higher the better. Such index is also highly sensitive to small misalignment of the bounding boxes, hence values above 0.7 usually correspond to satisfactory tracking.

The second chart shows correct track ratio versus the mean overlap on correct frames, where we define correct frames those where the overlap is greater than a threshold and the correct track ratio is given by the ratio between the correct frames and the total frame of the sequence. An optimal tracker is represented by a line at the very top of the chart. This chart tries to cope with the fact that for different applications different correct track ratios (more commonly expressed as lost track ratio) may be required. By considering the chart at a defined x coordinate, it is possible to understand which trackers are able to provide such level of lost track ratio, if their line intersects such vertical axis, and with which accuracy, represented by their mean overlap.

We compare our proposal against several adaptive trackers selected for their relevance in the recent literature as well as for the availability of the implementations at the authors' website: Boost Tracker [29], Semi-Boost Tracker [30], BeyondSemiBoost Tracker [89], A-BHMC (Adaptive-BasinHopingMC) [45], IVT (Incremental Visual Tracker) [80].



(a) (b) (c)

Figure 2.6: From left to right: Initialization frame for the *Dollar* sequence; sudden change of appearance (frame 90); a distractor pops out (frame 130).The green rectangle represents the ground truth bounding box.

To evaluate the importance of model adaptation in the considered sequences as well as to rank the overall performance of adaptive solutions, results from three standard non adaptive solutions are also added, namely Frag-Track [1], a color-based particle filter [78] and Mean-shift [17].

All the sequences are part of the dataset provided by the authors of MILBoost [5].

2.4.2 Dollar sequence

This is a simple sequence, but it allows for some interesting considerations. There is no clutter. The target (Fig. 2.6a) suddenly changes appearance (Fig. 2.6b). After a while a distractor equal to the original appearance of the target pops out close to the target (Fig. 2.6c) and then moves next to it. It is useful to understand the robustness to distractors and the degree of adaptiveness of the algorithms in a very controlled and predictable situation.

SemiBoost uses a fixed external classifier. This allows for very good performances up to the sudden change. After that, the target is believed to have exited the scene by this tracker because nothing matches well with such prior model. When the distractor appears, this tracker believes the object is back in the scene, and follows it.

BeyondSemiBoost uses an adaptive slowly evolving prior in combination with a fixed one from the first frame. This allows the tracker to overcome the sudden appearance change. Nevertheless, when the distractor appears, the fixed prior misleads the tracker.

The behavior of Boost is slightly unexpected. Since it is not binded to the initial appearance by a prior, it should have been able to avoid the distractor, as well as the sudden change. It does indeed overcome the change in appearance but it many runs it jumps on the distractor as soon as it appears, much like BeyondSemiBoost. This explains the higher variance compared to the other trackers. The behaviour of A-BHMC is interesting. Since it is designed to cope with appearance changes steaming from geometric changes, it allows its patches to move independently from each other, similarly to our proposal, but not to vary much in appearance, since patches are matched across frames using a tracker assuming brightness constancy. This results in a greater instability than the other trackers. This also leads to two outcomes that limit its performace in this sequence: the lower part of the target is excluded from the model when it changes and some patches are attracted by the distractor when it appears close to the target. Therefore, the ouput of the tracker stretches between the target and the distractor. Our proposal, which updates also the particle appearance, does not suffer from these problems.

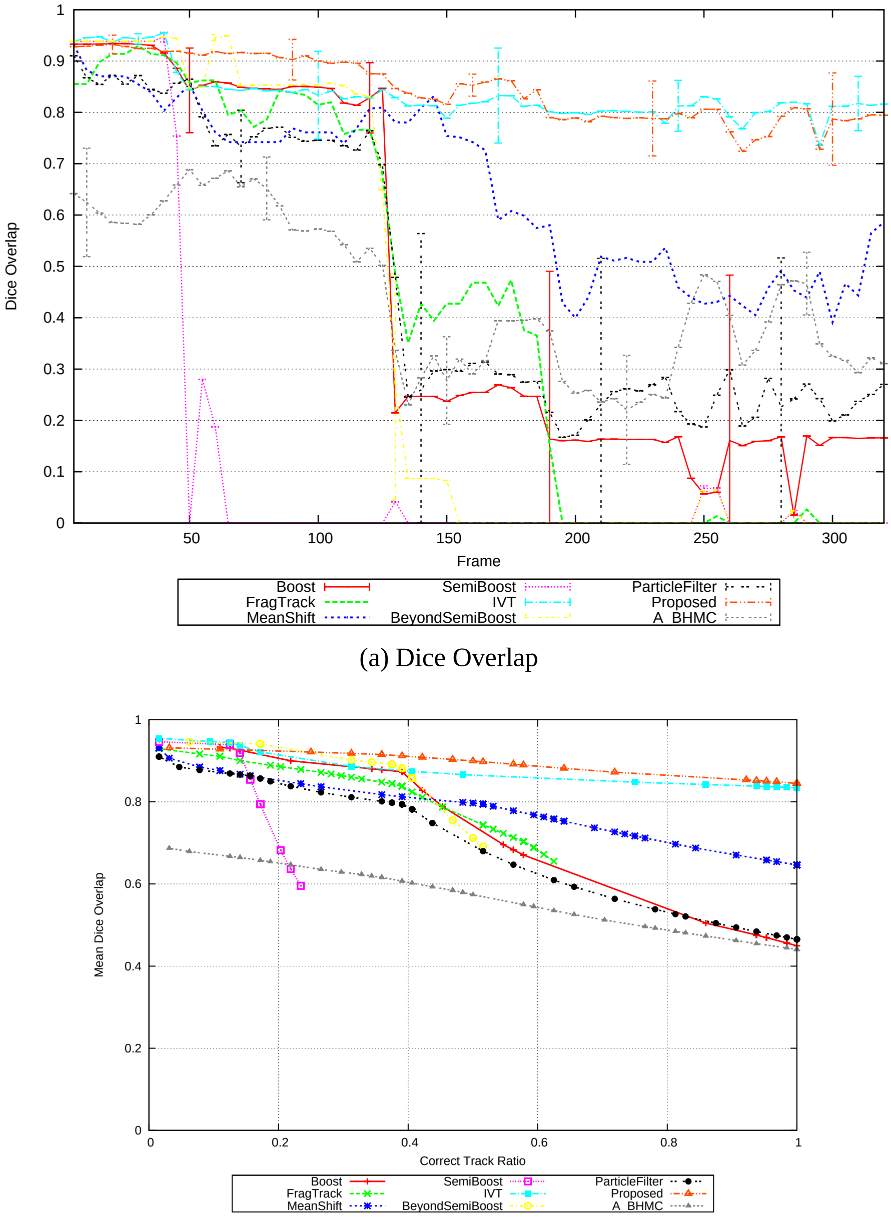
As for non adaptive solutions, the use of global statistics allows Mean Shift to overcome the nuisances of this scene, because the new appearance of the target is similar to the previous one as fas as the color histogram is concerned and the use of temporal consistency prevents it to jump completely onto the distractor. Nevertheless its performance after the appearance of the distractor is not satisfactory. FragTrack, using spatially localized histograms, is instead affected by the change and drifts to the distractor. The Particle Filter exhibits a large variance in its results, given by the fact that in the trials of the algorithms it was sometimes affected by the distractor and sometimes not: this indicates that the ability of the particle filter to avoid the distractor in this sequence is just a random event due to the random approximation of the posterior produced by the filter.

The best performer are IVT and our proposal. IVT deploy a particle filtering for state tracking as our tracker. Its target model is instead composed by global features, in particular the target graylevel template. A subspace of templates is constructed on-line and the distance from it constitutes the base for the definition of the observation likelihood. This is a very stable solution and has problems in adapting to sudden changes of appearance. Moreover the graylevel template has problems in dealing with deformable targets. None of these critics condition is met in this sequence, where from the object sudden change to the appearance of the distractor more then 40 frames elapses while the object is still and the majority of the target does not deform. Therefore, the tracker obtains a performance equivalent to ours both in terms of mean overlap and of variance. Both trackers are able to learn the new appearance of the target and do not confound it with the distractor in all the runs.

2.4.3 Faceocc2 sequence

This is a moderately difficult scene, targeting face tracking (Fig. 2.8). The main nuisances in these scenes are frequent and rather large occlusions. Beside, a permanent target appearance change happens about the middle of the sequence, followed by a last occlusion. Hence, the main ability a tracker has to show in this sequence is a high discriminative power between occlusions, *i.e.* spurious changes of the target appearance, and permanent changes of the target.

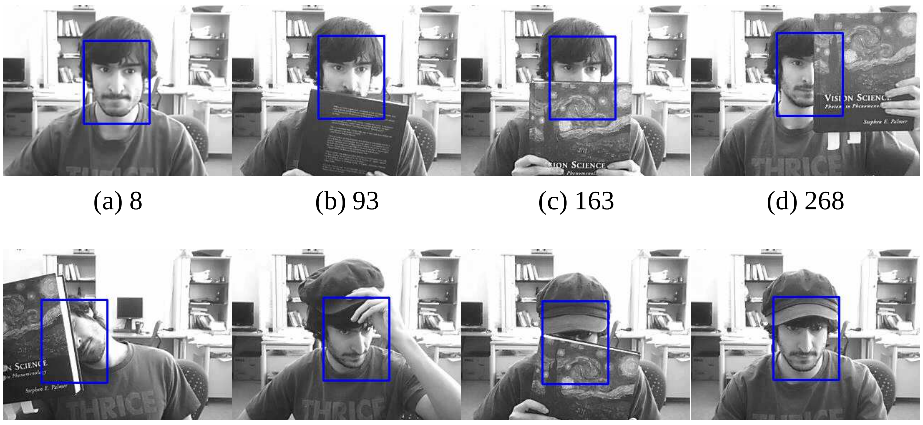
Results are reported in Fig. 2.9. Our proposal turns out the best again, as shown by the correct track ratio chart. Thanks to its formulation, our filter is able to discriminate between partial occlusions and changes of the target. In fact, when the book starts to occlude the face, its appearance has been already captured by the particle of our appearance model that are modeling the background. Hence, when performing weights update and resampling, the patches extracted on the book to perform target model update will receive a low score and will be likely discarded,



(a) Dice Overlap

(b) Lost Track Ratio

Figure 2.7: *Dollar* sequence



(a) 8 (b) 93 (c) 163 (d) 268

(e) 498 (f) 573 (g) 718 (h) 808

Figure 2.8: From left to right, top to bottom: Initialization frame for the *Faceocc2* sequence; first mild occlusion (frame 93); a larger occlusions (frame 163); third occlusions (frame 268); target rotation and large occlusions (frame 498); target appearance change (frame 573); large occlusion (frame 718); final appearance of the target (frame 808). The blue rectangle represents the ground truth bounding box.

therefore not corrupting the target model. On the other hand, the hat is fully included in the target bounding box, and therefore its patches are inserted in the target model.

IVT, deploying global features, suffers more than our proposal both the large occlusion around frame 500 and the target appearance deformation around frame 350 (head turning). Mean-shift deploying global features ,as well, and being not adaptive cannot cope with the challenges of this sequence. FragTrack, although non adaptive, too, is based on part-wise features. Since the target appearance does not change up to frame 550, the non adaptiveness of the tracker is compensated by the ability to correctly match the target in presence of occlusions, and the tracker is the second best in the correct track ratio chart. Nevertheless, the tracker suffers the target deformation around frame 350 and the appearance change after the last occlusion. This indicate the need to allow for target deformation when deploying part-wise features and the need to update the part-based representation to obtain better overlaps in this sequence.

Trackers deploying external classifiers for the sample and labeling stage (SemiBoost, BeyondSemiBoost) show good performances up to the large target deformation of frame 300. Again, the use of strong priors on the target appearance, assumed by using a detector to label new samples for appearance model update, limits their adaptability. On the other hand, a continuously adapting tracker like Boost suffer the same nuisances, and in particular occlusions, because of its lack of stability.

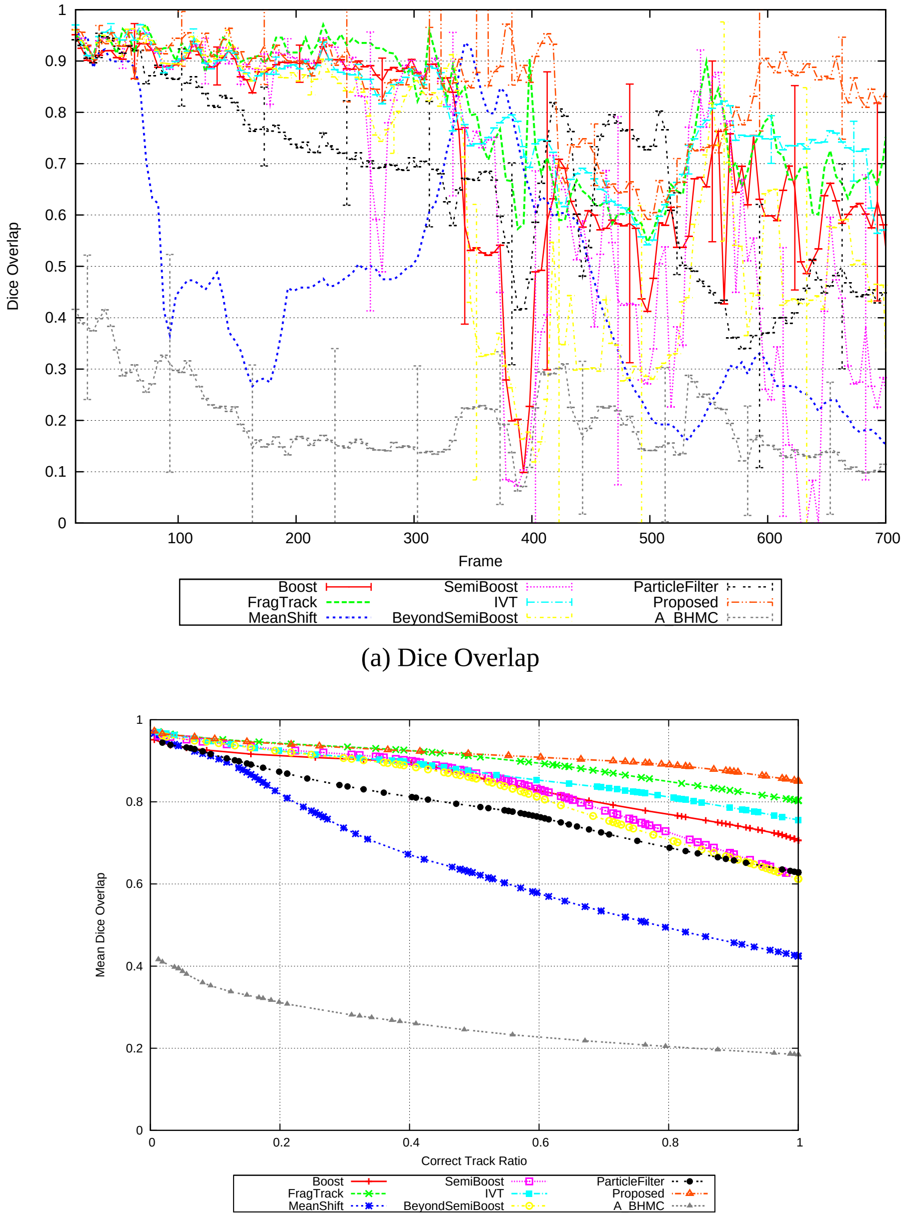
2.4.4 Coke sequence

A can of Coke is tracked in front of a uniform background. The can is moved behind a plant, causing partial and total occlusions. The can is also rotated, causing appearance changes. Finally, an artificial light stands very close to the target causing reflections and illumination changes. The target is also small and relatively untextured. Overall, a challenging sequence from many points of view.

Results are reported in Fig. 2.11. Basically, all trackers fail. The not adaptive solutions loose the target immediately since the can starts to rotate from the first frame. Appearance changes handling is of course fundamental in this sequence. The use of priors in SemiBoost and BeyondSemiBoost does not allow them to cope with a sequence with so many sudden changes of appearance. Also the prior cannot be really informative since the object is relatively untextured, very small and similar to the background. The use of salient regions by A-BHMC makes it loose the target as soon as an untextured side of the can is shown to the camera.

Even IVT looses the target in the first frame because it does not have the time to create an effective subspace representation for the can appearance in the first frames, where the can keeps on changing its appearance. Moreover, subspaces and manifolds do not seem the appropriate tools to cope with this sequence.

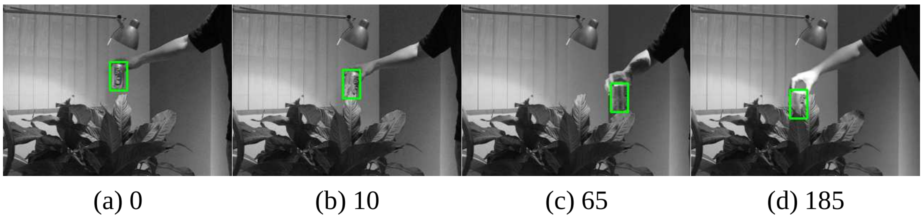
The only partially successful solutions are those that allows for con-



(a) Dice Overlap

(b) Correct Track Ratio

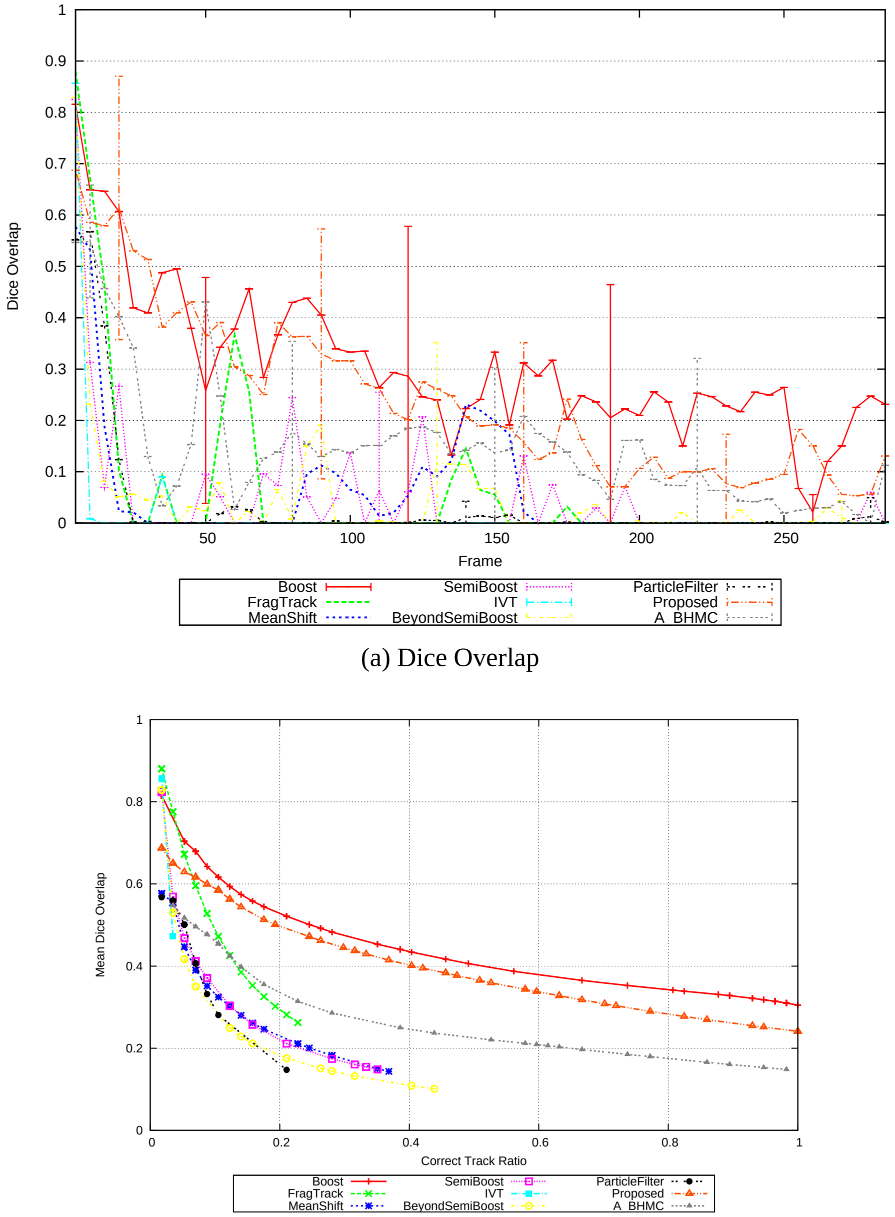
Figure 2.9: *Faceocc2* sequence



(a) 0 (b) 10 (c) 65 (d) 185

Figure 2.10: From left to right: Initialization frame for the *Coke* sequence; after ten frames the appearance of the can is already changed and the target undergoes a partial occlusion; then the can wanders around undergoing changes in appearance and illumination as in frame 65 and occlusions as in frame 185. The green rectangle represents the ground truth bounding box.

tinuous update, without priors, and with a part based model, namely Boost and our filter. We mainly impute the failure of our filter in this sequence to the lack of texture of the back of the object that is not correctly handled by our observation likelihood based on the ZNCC. We believe that with a proper mechanism to perform on-line feature selection and the inclusion of edge features our performance will likely improve.



(a) Dice Overlap

(b) Lost Track Ratio

Figure 2.11: *Coke* sequence

Chapter 3

Synergistic Change Detection and Tracking

In this chapter we investigate adaptive visual tracking with static cameras. The usual approach [15, 32, 34, 38, 88, 90, 104] in such a case is to ground tracking on *change detection*: a process that labels every pixel as changed (*i.e.* a target pixel) or unchanged (*i.e.* a background pixel) with respect to a static background. Although in these proposals change detection is key for tracking, little attention has been paid to sound modeling of the interaction between the change detector and the tracker. This negatively affects the quality of the information flowing between the two computational modules, as well as the soundness of the proposals. Moreover, the interaction can be highly influenced by heuristically tuned parameters, such as change detection thresholds, that limit the deployment of these solutions in real world applications.

Our work aims at sound modeling of the analysis of the output of the change detection that produces a new measure for the tracker. We also wish to have a limited number of parameters and that they can be easily interpreted and tuned. As we have seen, Recursive Bayesian Estimation (RBE) casts visual tracking as a Bayesian inference problem in state space given noisy observation of the hidden state. Bayesian reasoning has been recently used also to solve the problem of change detection in image sequences [47].

We introduce a novel Bayesian change detection approach aimed at efficiency and robustness to common sources of disturbance such as illumination changes, camera gain and exposure variations, noise. At each new frame, a binary Bayesian classifier is trained and then used to discriminate between pixels sensing a scene change and pixels sensing a spurious intensity variation due to disturbs. After efficient *non-parametric* estimation of likelihood distributions for both classes, the posterior probability of sensing a scene change at each pixel is obtained.

Given this Bayesian change detector and a generic recursive Bayesian filter as tracker, we develop a principled framework whereby both algorithms can virtuously influence each other according to a Bayesian loop. In particular:

- the output of the change detection is used to provide a fully specified observation likelihood to the RBE tracker;

- the RBE tracker provides a feedback to the Bayesian change detector by defining an informative prior for it;

- both PDFs are modeled and realized as marginalizations of the joint PDF on tracker state and change detector output.

The derivation of a measure for the tracker from the change detection output is a fundamental part of a every tracker based on change detection. The idea of letting the tracker provide a feedback to change detection is inspired by the emergence of cognitive feedback in Computer Vision [96]. The idea of cognitive feedback is to let not only low-level vision modules feed high-level ones, but also the latter influence the former. This creates a closure loop, reminiscent of effects found in psychophysics. This concept has not been deployed for the problem of visual tracking yet. Nevertheless, it fits surprisingly well in the case of Bayesian change detection, where priors can well model the *stimuli* coming from the tracker.

By exploiting the synergy between the two flows of information our system creates a full and synergistic Bayesian loop between the tracker and the change detection, whose benefits are presented in the Experimental Results section (Sec. 3.6), where the Kalman Filter is used as RBE tracker and the algorithm introduced in Sec. 3.4 as change detection. However, our proposal is general and in principle can be used with any RBE tracker and Bayesian change detection, such as e.g., respectively, particle filters and [47].

3.1 Related Works

Classical works on blob tracking based on change detection are $\mathcal{W}4$ [32] and the system developed at the Video Surveillance and Monitoring (VSAM) group of CMU [15]. In these systems the output of the change detector is thresholded and a connected component analysis is carried out to identify moving regions (blobs). A first or second order dynamical model of every tracked object is used to predict its position in the current frame from the previous ones. Positions are then refined by matching the predictions to the output of the change detection. In VSAM [15] any blob whose centroid falls within a neighborhood of the target predicted position is considered for matching. Matching is performed as correlation of an appearance template of the target to the changed pixels, and the position corresponding to the best correlation is selected as the new position for the object. In $\mathcal{W}4$ [32] the new position is that corresponding to the maximum of the binary edge correlation between the current and previous silhouette edge profiles. However, the interaction between tracking and change detection is limited, tracking is not formalized in the context of RBE, change detection depends on hard thresholds, no probabilistic reasoning is carried out to derive a new measure from the change detection output or to update the object position, (*i.e.* a bunch of heuristics are used to solve the case of not connected blobs for the same object).

[90] and [34] are examples of blob trackers based on change detection where the RBE framework is used in the form of the Kalman filter. Yet, the use of this powerful framework is impoverished by the absence of a truly probabilistic treatment of the change detection output. In practice, covariance matrices defining measurement and process uncertainties are constant, and the filter evolves toward its steady-state regardless of the quality of the measures obtained from change detection. A posteriori covariance matrices are sometimes deterministically increased by the algorithms, but this is mainly a shortcut to implement track management: if there is no match for the track in the current frame uncertainties are increased and if a posteriori uncertainties on state gets too high, the track is discarded.

[38] is one of the most famous attempt to integrate RBE in the form of a particle filter with a statistical treatment of background (and foreground) models. It proposes a multi-blob likelihood function that, given the frame and the background model, allows the system to reason probabilistically on the number of people present in the scene as well as on their positions. The main limitations are the use of a calibrated camera with reference to the ground plane and the use of a foreground model learned off-line. While the former can be reasonable, although cumbersome, the use of foreground models is always troublesome in practice, given the high intra-class variability of target appearances. Moreover, no cognitive feedback is provided from the Particle Filter to influence the change detection.

Sorts of cognitive feedbacks from tracking to change detection have been used so far only to deal with background maintenance and adaptive background modeling issues. For example, [95] proposes a method based on approximate inference on a dynamic Bayesian Network that simultaneously solves tracking and background model updating for every frame. Nevertheless, as discussed by the authors, this proposal do not take advantage of models of foreground motion as our algorithm does, although this would allow for better estimation of both the background model and the background/foreground labels, because it will also severely complicate inference. Another example of background maintenance is [33], where positive and negative feedbacks from high-level modules (a stereo-based people detector and tracker, a detector of rapid changes in global illumination, camera gain, and camera position) are used to update the parameters of the Gaussian distributions in the Gaussian Mixture Model used as background. These feedbacks come in the form of pixel-wise positive or negative real number maps that are generated as sum of the contributions of the high-level modules and are thresholded in order to decide if a pixel should be used to update the background. Contributions from the high-level modules are heuristically determined.

3.2 Models and assumptions

We first present assumptions and notations used to model RBE and Bayesian change detection separately, then we introduce the common framework that allows us to define probabilistically the bidirectional interaction between the two modules, *i.e.* the observation likelihood for the tracker defined on the change map and the prior for the change detection that implements the Cognitive Feedback.

3.2.1 RBE model

We assume a rectangular model for the tracked object, as done in many proposals such as *i.e.* [17]. Hence, the state of the RBE tracker, $\mathbf{x}_k$, comprises at least four variables

$$\mathbf{x}_k = \left\{ i_k^b, j_k^b, w_k, h_k, \ldots \right\} \tag{3.1}$$

where (i_k^b, j_k^b) are the coordinates of the barycenter of the rectangle and w_k and h_k its dimensions. These variables define the position and size at frame k of the tracked object. Of course, the state internally used by the tracker can beneficially include other cinematic variables (velocity,acceleration,...). Yet, change detection can only provide a measure and benefit from a prior on the position and size of the object. Hence, other variables are not used in the reminder of the presentation of the algorithm, though they can be used internally by the RBE filter, and are indeed used in our implementation (Sec. 3.6).

We can also represent the bonding box by defining new variables i_L, j_T, i_R, j_B as

$$\mathbf{A} = \begin{bmatrix} 1 & -\frac{1}{2} \\ 1 & \frac{1}{2} \end{bmatrix}, \quad \begin{bmatrix} i_L \\ i_R \end{bmatrix} = \mathbf{A} \begin{bmatrix} i^b_k \\ w_k \end{bmatrix}, \quad \begin{bmatrix} j_T \\ j_B \end{bmatrix} = \mathbf{A} \begin{bmatrix} j^b_k \\ h_k \end{bmatrix}. \tag{3.2}$$

We assume the variables i_L, j_T, i_R, j_B to be independent, since this is reasonable in our context and also simplifies the derivation of the information flows of our loop. This implies that the variables i^b_k, j^b_k, w_k, h_k defining the alternative representation are not independent, but this is not a problem since RBE can handle dependent variables (*e.g.* the Kalman filter does not require diagonal covariance matrices).

3.2.2 Bayesian change detection model

In Bayesian change detection each pixel of the image is modeled as a categorical Bernoulli-distributed random variable, c_{ij}, with the two possible realizations $c_{ij} = \mathcal{C}$ and $c_{ij} = \mathcal{U}$ indicating the event of pixel (i, j) being changed or unchanged, respectively.

In the following we refer to the matrix $\mathbf{c} = \left[c_{ij}\right]$ of all these random variables as the *change mask* and to the matrix $\mathbf{p} = \left[p(c_{ij} = \mathcal{C})\right]$ of probabilities defining the Bernoulli distribution of these variables as *change map*. The change mask and the change map assume values, respectively, in the $(w \times h)$-dimensional spaces $\Theta = \{\mathcal{C}, \mathcal{U}\}^{w \times h}$ and $\Omega = [0, 1]^{w \times h}$, with w and h denoting image width and height, respectively. The output of a Bayesian change detector is the posterior change map given the current frame f_k and background model b_k, *i.e.* the value of the Bernoulli distribution parameter for every pixel in the image given the frame and the

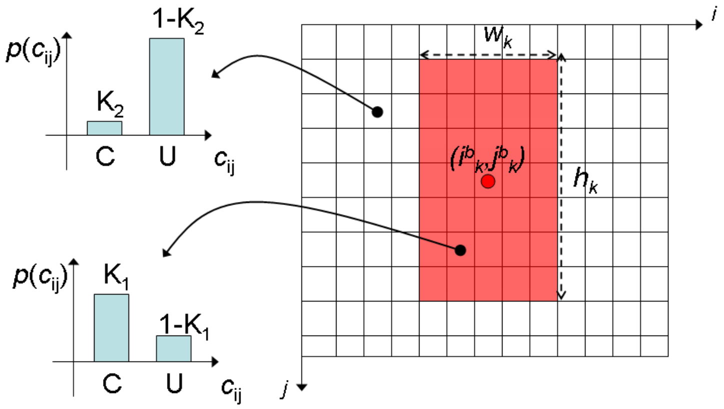


Figure 3.1: Model for the change map given a bounding box.

background:

$$p(c_{ij} = C \mid f_k, b_k) = \frac{p(f_k, b_k \mid c_{ij} = C)p(c_{ij} = C)}{p(f_k, b_k)} \tag{3.3}$$

Clearly, either a *non-informative* prior is used, such as a uniform prior, or this information has to be provided by an external module. We assume that the categorical random variables c_{ij} comprising the posterior change mask are independent, *i.e.* they are conditionally independent given f_k, b_k.

3.2.3 Bayesian loop models

All the information that can flow from the RBE filter to the Bayesian change detection and vice versa is in principle represented in every frame by the joint probability density function $p(\mathbf{x}_k, \mathbf{c})$ of the state vector and the change mask. Both information flows can be formalized and realized as its marginalization:

$$p(c_{ij}) = \iiiint_{\mathbb{R}^4} \sum_{\mathbf{c}^{ij}\in\Theta^{ij}} p\left(\mathbf{x}_k, c_{ij}, \mathbf{c}^{ij}\right) d\mathbf{x}_k \tag{3.4}$$

$$p(\mathbf{x}_k) = \sum_{\mathbf{c}\in\Theta} p\left(\mathbf{x}_k, \mathbf{c}\right) \tag{3.5}$$

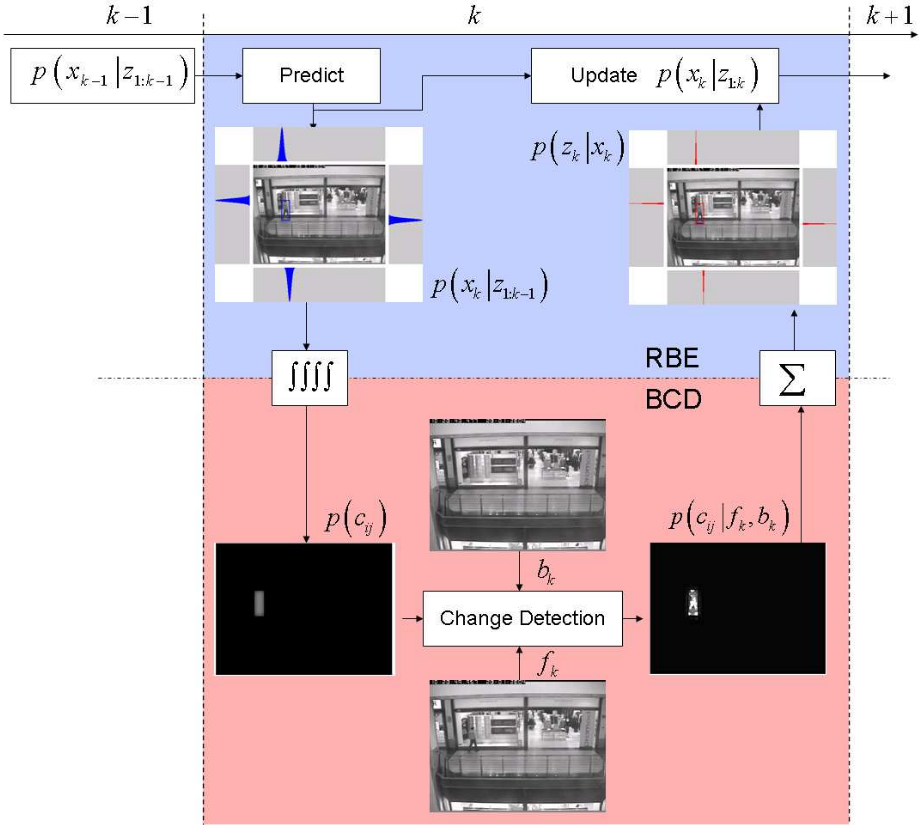

$k-1$
k
$k+1$
$p\left(x_{k-1} \mid z_{1:k-1}\right)$
Predict
Update $p\left(x_k \mid z_{1:k}\right)$
$p\left(z_k \mid x_k\right)$
$p\left(x_k \mid z_{1:k-1}\right)$
$\int\int\int\int$
RBE
BCD
$\sum$
$p\left(c_{ij}\right)$
$p\left(c_{ij} \mid f_k, b_k\right)$
b_k
Change Detection
f_k

Figure 3.2: Overall system description. In every frame the RBE tracker provides a prediction $p\left(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}\right)$ from the previous state that is used by our framework to generate a set of priors $p\left(c_{ij}\right)$, each one of them assessing the probability that a particular pixels is changed. This informative prior is used by a Bayesian change detection algorithm together with the current frame f_k and a model of the background b_k to produce a change map $p\left(c_{ij} \mid f_k, b_k\right)$. The change map is not thresholded but a probabilistic analysis is carried out in order to provide a new measure for the tracker $p\left(\mathbf{z}_k \mid \mathbf{x}_k\right)$, that is merged with the prediction in the update state of RBE. The blue and red histogram around respectively the prediction and the measure represent the variance associated with the four variables defining a bounding box, which are assumed to follow a Gaussian distribution in the specific example. Generally speaking, they are placed there to remind to the reader that completely specified probabilities are flowing from and into the RBE tracker thanks to our proposal.

where $\mathbf{c}^{ij}$ denotes the change mask without the (i, j)-*th* element, taking values inside the space $\Theta^{ij} = \{\mathcal{C}, \mathcal{U}\}^{w \times h-1}$.

The PDF computed with (3.4) defines an informative prior for the Bayesian change detection algorithm, and the estimation of the state obtained with (3.5) can then be used as the PDF of a new measure by the RBE tracker, *i.e.* as $p(\mathbf{z}_k \mid \mathbf{x}_k)$. We detail in Sec. 3.3 and Sec. 3.5 the solutions for (3.4) and (3.5). With reference to Fig. 3.2, it is worth notice that in our framework only fully defined probabilities flow among the modules, not just expectations or deterministic measures.

As we shall see in next sections, to use the above equations we need a statistical model that links the two random vectors $\mathbf{x}_k$ and $\mathbf{c}$. In agreement with our rectangular model of the tracked object, as shown in Fig. 3.1 we assume

$$p\left(c_{ij} = \mathcal{C} \mid \mathbf{x}_k\right) = \begin{cases} K_1 & \text{if } (i, j) \in R\left(\mathbf{x}_k\right) \\ K_2 & \text{otherwise} \end{cases} \tag{3.6}$$

where $R(\mathbf{x}_k)$ is the rectangular region delimited by the bounding box defined by the state $\mathbf{x}_k$ and $0 \le K_2 \le K_1 \le 1$ are two constant parameters specifying the probability that a pixel is changed inside and outside the bounding box, respectively. Moreover, we assume that the random variables c_{ij} are conditionally independent given a bounding box, *i.e.*

$$p\left(\mathbf{c} \mid \mathbf{x}_k\right) = \prod_{ij} p\left(c_{ij} \mid \mathbf{x}_k\right) \tag{3.7}$$

3.3 Cognitive Feedback

Given the assumptions in Sec. 3.2, we can obtain an exact solution for (3.4), *i.e.*, given the PDF of the state vector $p(\mathbf{x}_k)$, we can compute a prior $p(c_{ij})$ for each pixel of the frame that can then be used as prior in the Bayesian change detection algorithm. Starting from (3.4), we can rewrite it as

$$
\begin{aligned}
p\left(c_{ij}\right) &= \iiiint\limits_{\mathbb{R}^4} \sum_{\mathbf{c}^{ij}\in\Theta^{ij}} p\left(\mathbf{x}_k, c_{ij}, \mathbf{c}^{ij}\right)\, d\mathbf{x}_k \\
&= \iiiint\limits_{\mathbb{R}^4} p\left(\mathbf{x}_k, c_{ij}\right)\, d\mathbf{x}_k \\
&= \iiiint\limits_{\mathbb{R}^4} p\left(c_{ij} \mid \mathbf{x}_k\right) p\left(\mathbf{x}_k\right)\, d\mathbf{x}_k
\end{aligned} \tag{3.8}
$$

In the final marginalization we can recognize our model of the change map given a bounding box defined in (3.6) and the PDF of the state. Therefore, this equation provides a way to let the current estimation of the state computed by the RBE module influence the prior for the Bayesian change detection algorithm, thereby realizing the Cognitive Feedback. In particular, as discussed above, we will use the prediction computed for the current frame using the motion model, *i.e.* $p(\mathbf{x}_k \mid \mathbf{Z}_{1:k-1})$.

To solve (3.8) we have to span the space $\mathbb{R}^4$ of all possible bounding boxes $\mathbf{x}_k$. We partition $\mathbb{R}^4$ into the two complementary sub-spaces B_{ij} and $\overline{B}_{ij} = \mathbb{R}^4 \setminus B_{ij}$ of bounding boxes that contain or not the considered pixel (i, j), respectively. Given the assumed model (3.6), we obtain

$$
\begin{aligned}
p(c_{ij} = C) &= \iiiint\limits_{\mathbb{R}^4} p(c_{ij} \mid \mathbf{x}_k) p(\mathbf{x}_k)\, d\mathbf{x}_k \\
&= K_1 \iiiint\limits_{B_{ij}} p(\mathbf{x}_k)\, d\mathbf{x}_k + K_2 \iiiint\limits_{\overline{B}_{ij}} p(\mathbf{x}_k)\, d\mathbf{x}_k \\
&= K_1 \iiiint\limits_{\mathbf{x}_k\in B_{ij}} p(\mathbf{x}_k)\, d\mathbf{x}_k + K_2 \iiiint\limits_{\mathbf{x}_k\in\mathbb{R}^4} p(\mathbf{x}_k)\, d\mathbf{x}_k \\
&\quad - K_2 \iiiint\limits_{\mathbf{x}_k\in B_{ij}} p(\mathbf{x}_k)\, d\mathbf{x}_k \\
&= K_2 + (K_1 - K_2) I_{ij}\, , \; I_{ij} = \iiiint\limits_{B_{ij}} p\left(\mathbf{x}_k\right)\, d\mathbf{x}_k\, .
\end{aligned} \tag{3.9}
$$

Since I_{ij} varies in $[0, 1]$, it follows that $p(c_{ij} = C)$ varies in $[K_2, K_1]$: if no bounding box with non-zero probability contains the pixel, we expect a probability that the pixel is changed equal to K_2; if all the bounding boxes contain the pixel the probability is K_1; it is a weighted average otherwise.

By using the alternative representation for the bounding box defined in (3.2) and recalling that we assume i_L, j_T, i_R, j_B to be independent, the integral becomes

$$\begin{aligned} I_{ij} &= \iiiint\limits_{\left.\begin{array}{c} i_L \leq i \leq i_R \\ j_T \leq j \leq j_B \end{array}\right\} B_{ij}} p\left(i_L\right) p\left(i_R\right) p\left(j_T\right) p\left(j_B\right)\, di_L di_R dj_T dj_B \\ &= \int_{-\infty}^{i} p\left(i_L\right)\, di_L \int_{i}^{+\infty} p\left(i_R\right)\, di_R \int_{-\infty}^{j} p\left(j_T\right)\, dj_T \int_{j}^{+\infty} p\left(j_B\right)\, dj_B \\ &= F_{i_L}\left(i\right)\left(1 - F_{i_R}\left(i\right)\right) F_{j_T}\left(j\right)\left(1 - F_{j_B}\left(j\right)\right) \end{aligned} \tag{3.10}$$

where F_x stands for the CDF of the random variable x.

This reasoning holds for any distribution $p(\mathbf{x}_k)$ we might have on the state vector. If, for instance, we use a particle filter as RBE tracker, we can compute an approximation of the CDF from the approximation of the PDF provided by the weighted particles, after having propagated them according to the motion model and having marginalized them accordingly. In the case of the Kalman Filter all the PDFs are Gaussians, hence we can define all the factors of the product in (3.10) in terms of the standard Gaussian CDF, $\Phi(\cdot)$

$$I_{ij} = \Phi\left(\frac{i - \mu_{i_L}}{\sigma_{i_L}}\right)\Phi\left(\frac{\mu_{i_R} - i}{\sigma_{i_R}}\right)\Phi\left(\frac{j - \mu_{j_T}}{\sigma_{j_T}}\right)\Phi\left(\frac{\mu_{j_B} - j}{\sigma_{j_B}}\right) \tag{3.11}$$

where μ_x and σ_x stand for the mean and the standard deviation of the random variable x. The factors of the product in (3.11) can be computed efficiently with only 4 searches in a pre-computed Look-Up Table of the standard $\Phi(\cdot)$ values.

3.4 Bayesian change detection

The main difficulty with change detection consists in discerning changes of the monitored scene in presence of spurious intensity variations yielded by nuisances such as noise, gradual or sudden illumination changes, dynamic adjustments of camera parameters (e.g. auto-exposure, auto-gain). Many different algorithms for dealing with these issues have been proposed (see [24] for a recent survey).

A first class of popular algorithms based on statistical per-pixel background models, such as e.g. Mixture of Gaussians [90] or kernel-based non-parametric models [23], are effective in case of noise and gradual illumination changes (e.g. due to the time of the day). Unfortunately, though, they cannot deal with those disturbs causing sudden intensity changes (e.g. a light switch), yielding in such cases lots of false positives.

A second class of algorithms relies on a priori modeling the possible spurious intensity changes over small image patches yielded by disturbs. Following this idea, a pixel from the current frame is classified as *changed* if the intensity transformation between its local neighborhood and the corresponding neighborhood in the background can not be explained by the chosen a priori model. As a result, gradual as well as sudden photometric distortions do not yield false positives provided that they are explained by the model. Thus, the main issue concerns the choice of the a priori model: generally speaking, the more restrictive such a model, the higher is the ability to detect changes (sensitivity) but the lower is robustness to disturbs (specificity). Some proposals assume disturbs to yield linear intensity transformations [53, 68]. Nevertheless, as discussed in [102], many non-linearities may arise in the image formation process, so that a less constrained model is often required to achieve adequate robustness. Hence, other algorithms adopt order-preserving models, i.e. assume monotonic non-decreasing intensity transformations [48, 64, 102]

We propose a change detection approach that, instead of assum-

$x_{i,1}$	$x_{i,2}$	$x_{i,3}$
$x_{i,4}$	x_i	$x_{i,5}$
$x_{i,6}$	$x_{i,7}$	$x_{i,8}$

$y_{i,1}$	$y_{i,2}$	$y_{i,3}$
$y_{i,4}$	y_i	$y_{i,5}$
$y_{i,6}$	$y_{i,7}$	$y_{i,8}$

Figure 3.3: Notations adopted for the background (on the left) and the current frame (on the right) neighborhood intensities.

ing a-priori the model of intensity changes caused by disturbs, learns it on-line together with the model of intensity changes yielded by foreground objects. In particular, at each new frame a binary Bayesian classifier is trained and then used to discriminate between pixels sensing a scene change due to foreground objects and pixels sensing a spurious intensity variation due to disturbs. On-line learning of the models holds the potential for deploying on a frame-by-frame basis models as restrictive as needed to discriminate between the two classes, so that the algorithm can exhibit a high sensitivity without a significant loss of specificity. Moreover, the fully Bayesian formulation for the change detection problem allows for seamlessly incorporating in a sound way a prior probability to strengthen the change detection output. In our framework this prior is provided by the tracker via the cognitive feedback defined above.

3.4.1 On-line learning for change detection

By taking pixels in lexicographical order, let us denote the background and the current frame intensities, respectively, as

$$B = (x_1, \ldots, x_N) \quad \text{and} \quad F = (y_1, \ldots, y_N) \tag{3.12}$$

where $x_i, y_i \in [0, 255] \subset \mathbb{N},\ i = 1, \ldots, N$ and N is the total number of pixels in the images. The goal of change detection is to compute the binary change mask

$$M = (c_1, \ldots, c_N) \tag{3.13}$$

i.e. to classify each pixel i into one of the two classes:

$c_i \ = \ \mathcal{C}:$ the pixel is sensing a scene change;

$c_i \ = \mathcal{U}:$ the pixel is not sensing a scene change.

The idea at the basis of our proposal consists in training at each new frame a binary Bayesian classifier using as feature vector the background-frame pair of intensities (x, y) observed at a pixel, and then computing the change map by letting each pixel take the a-posteriori value of the probability of being changed:

$$p(c{=}\mathcal{C} \mid x, y) = \frac{p(c{=}\mathcal{C})p(x, y \mid c{=}\mathcal{C})}{p(x, y)}. \tag{3.14}$$

The prior $p(c{=}\mathcal{C})$ is obtained via the Bayesian loop from the tracker. In order to train the classifier we have to estimate the likelihood $p(x, y \mid c{=}\mathcal{C})$ and the evidence $p(x, y)$. We can avoid to estimate the evidence by the usual manipulation of (3.14) as

$$\begin{aligned} p\,(c = \mathcal{C} \mid x, y) &= \frac{p\,(c = \mathcal{C})\,p\,(x, y \mid c = \mathcal{C})}{p\,(x, y)} \\ &= \frac{p\,(c = \mathcal{C})\,p\,(x, y \mid c = \mathcal{C})}{p\,(c = \mathcal{C})\,p\,(x, y \mid c = \mathcal{C}) + p\,(c = \mathcal{U})\,p\,(x, y \mid c = \mathcal{U})} \\ &= \frac{1}{1 + \dfrac{p\,(c = \mathcal{U})\,p\,(x, y \mid c = \mathcal{U})}{p\,(c = \mathcal{C})\,p\,(x, y \mid c = \mathcal{C})}}\,. \end{aligned} \tag{3.15}$$

To estimate $p(x, y \mid c{=}\mathcal{C})$ and $p(x, y \mid c{=}\mathcal{U})$, we carry out a preliminary classification of pixels by means of a very simple and efficient neighborhood-based change detection algorithm. For a generic pixel i, let the intensities of a surrounding 3×3 neighborhood be denoted as in Fig. 3.3, let the intensity differences between the j-th and the central pixel of the neighborhood in the background and in the current frame be, respectively,

$$d_{i,j}^{(x)} = x_{i,j} - x_i \quad \text{and} \quad d_{i,j}^{(y)} = y_{i,j} - y_i \tag{3.16}$$

and let the pixel in the neighborhood yielding the maximum absolute value of the background intensity difference be

$$\bar{j}_i = \arg\max_{j=1,\dots,8} \left| d^{(x)}_{i,j} \right| \tag{3.17}$$

A preliminary change mask $\tilde{M} = (\tilde{c}_1, \dots, \tilde{c}_N)$ is computed by classifying each pixel as changed if the sign of the intensity differences $d^x_{i,\bar{j}_i}$ and $d^y_{i,\bar{j}_i}$ is the same, unchanged otherwise:

$$d^{(x)}_{i,\bar{j}_i} \cdot d^{(y)}_{i,\bar{j}_i} \underset{\tilde{c}_i = u}{\overset{\tilde{c}_i = c}{\gtrless}} 0 \tag{3.18}$$

This algorithm is a simplified version of that proposed in [102] and exhibits $O(N)$ complexity. In fact, since the background model is not updated, the computation of $\bar{j}_i$ for each pixel by (3.17) can be performed off-line after background initialization. Furthermore, the algorithm is threshold-free.

The preliminary change mask is thus used to label each pixel to create a training set out of the current frame. The two likelihood distributions $p(x, y \mid c{=}\mathcal{C})$ and $p(x, y \mid c{=}\mathcal{U})$ are estimated on this training set as follows:

$$p(x,y \mid c{=}\mathcal{C}){=}\frac{h_{\mathcal{C}}(x,y)}{N_{\mathcal{C}}} \tag{3.19}$$

$$p(x,y \mid c{=}\mathcal{U}){=}\frac{h_{\mathcal{U}}(x,y)}{N_{\mathcal{U}}} \tag{3.20}$$

where $N_{\mathcal{C}}$ is the number of pixels labeled as changed, $h_{\mathcal{C}}(x, y)$ and $h_{\mathcal{U}}(x, y)$ are the 2-D joint histograms of background versus frame intensity computed by considering, respectively, the pixels labeled as changed and those labeled as unchanged. Before being used in (3.15), both the histograms $h_{\mathcal{C}}(x, y)$ and $h_{\mathcal{U}}(x, y)$ are smoothed by averaging over a moving window of fixed size. The smoothing allow for correcting errors introduced by wrong labeled training data in the preliminary rough labeling as well as for introducing a small amount of spatial consistency among labels, under the hypothesis that pixels close to each other in the image space show similar intensity values both in the foreground and in the background.

3.5 Reasoning probabilistically on change maps

Given the change map $\mathbf{p} = \left[p(c_{ij} = C) \right]$ obtained by the Bayesian change detection algorithm, we aim at computing the probability density function $p(\mathbf{x}_k)$ of the current state of the RBE filter, to use it as the observation likelihood $p(\mathbf{z}_k \mid \mathbf{x}_k)$. To this purpose, from the marginalization in (3.5) we obtain:

$$\begin{aligned} p(\mathbf{x}_k) &= \sum_{\mathbf{c} \in \Theta} p(\mathbf{x}_k, \mathbf{c}) \\ &= \sum_{\mathbf{c} \in \Theta} p(\mathbf{x}_k \mid \mathbf{c}) p(\mathbf{c}) \\ &= \sum_{\mathbf{c} \in \Theta} p(\mathbf{x}_k \mid \mathbf{c}) \prod_{ij} p(c_{ij}) \end{aligned} \tag{3.21}$$

where the last equality follows from the assumption of independence among the categorical random variables c_{ij} comprising the posterior change map computed by the Bayesian change detection.

To use (3.21), we need an expression for the conditional probability $p(\mathbf{x}_k \mid \mathbf{c})$ of the state given a change mask, based on the assumed model (3.6), (3.7) for the conditional probability $p(\mathbf{c} \mid \mathbf{x}_k)$ of the change mask given a state. Informally speaking, we need to find the inverse of the model (3.6), (3.7).

By Bayes rule, eq. (3.7) and independence of the variables c_{ij}:

$$p(\mathbf{x}_k \mid \mathbf{c}) = p^*(\mathbf{x}_k) \frac{p(\mathbf{c} \mid \mathbf{x}_k)}{p^*(\mathbf{c})} = p^*(\mathbf{x}_k) \prod_{i,j} \frac{p(c_{ij} \mid \mathbf{x}_k)}{p^*(c_{ij})} \,. \tag{3.22}$$

We have used the notation $p^*(\mathbf{x}_k)$ and $p^*(c_{ij})$ in (3.22) since here these probabilities must be interpreted differently than in (3.21): in (3.21)

$p(\mathbf{x}_k)$ and $p(c_{ij})$ represent, respectively, the measurement and the change map of the current frame, whilst in (3.22) both must be interpreted as priors that form part of our model for $p(\mathbf{x}_k \mid \mathbf{c})$, which is independent of the current frame. Furthermore, using as prior on the state $p^*(\mathbf{x}_k)$ the prediction of the RBE filter, as done in the Cognitive Feedback section, would have created a strong coupling between the output of the sensor and the previous state of the filter, that does not fit the RBE framework, where measures depend only on the current state, and could easily lead the loop to diverge. Hence, we assume a uniform non-informative prior $p^*(\mathbf{x}_k) = \frac{1}{\alpha}$ for the state.

The analysis conducted for the Cognitive Feedback is useful to expand each $p^*(c_{ij})$ in (3.22). Since we are assuming a uniform prior on an infinite domain for the state variables, *i.e.* a symmetric PDF with respect to $x = 0$, it turns out that its CDF is constant and equals to $\frac{1}{2}$:

$$CDF(x) = \frac{1}{\alpha}x + \frac{1}{2} \xrightarrow{\alpha \to +\infty} \frac{1}{2} \tag{3.23}$$

Hence, every $p^*(c_{ij})$ in (3.22) can be expressed using (3.9) and (3.10) as:

$$p^*(c_{ij} = \mathcal{C}) = K_2 + (K_1 - K_2)\left(\frac{1}{2}\right)^4 = K_C. \tag{3.24}$$

By plugging (3.22) in (3.21) and defining $K_U = p^*(c_{ij} = \mathcal{U}) = 1 - K_C$:

$$\alpha p(\mathbf{x}_k) = \prod_{i,j}\left(\frac{p(\mathcal{C} \mid \mathbf{x}_k)p(\mathcal{C})}{K_C} + \frac{p(\mathcal{U} \mid \mathbf{x}_k)p(\mathcal{U})}{K_U}\right) \tag{3.25}$$

where, for simplicity of notation, we use $\mathcal{C}$ and $\mathcal{U}$ for $c_{ij} = \mathcal{C}$ and $c_{ij} = \mathcal{U}$, respectively. Since we know that $p(\mathcal{U}) = 1 - p(\mathcal{C})$ and $p(\mathcal{U} \mid \mathbf{x}_k) = 1 - p(\mathcal{C} \mid \mathbf{x}_k)$, we obtain:

$$\frac{p\left(\mathbf{x}_k\right)}{\beta} = \prod_{i,j}\left(p\left(\mathcal{C}\right)\left(p(\mathcal{C} \mid \mathbf{x}_k) - K_C\right) + K_C\left(1 - p\left(\mathcal{C} \mid \mathbf{x}_k\right)\right)\right) \tag{3.26}$$

with $\beta = 1/\alpha(K_C(1 - K_C))^{w \times h}$. By substituting the model (3.6) for $p(\mathcal{C} \mid$

$\mathbf{x}_k$) and taking the logarithm of both sides to improve the numeric stability, after some manipulations we get:

$$\gamma + \ln p(\mathbf{x}_k) = h(\mathbf{x}_k, \mathbf{p}) = \sum_{(i,j)\in R(\mathbf{x}_k)} \ln \frac{p(C)K_3 + K_4}{p(C)K_5 + K_6} \tag{3.27}$$

where $\gamma = -\ln\beta - \sum \ln\left(p(C)K_5 + K_6\right)$ and $h(\cdot)$ is a known function of the state vector value $\mathbf{x}_k$ for which we want to calculate the probability density, of the change map $\mathbf{p}$ provided by the Bayesian change detection algorithm, and of the constants

$$\begin{aligned} K_3 &= K_1 - K_C K_4 = K_C\,(1 - K_1) \\ K_5 &= K_2 - K_C K_6 = K_C\,(1 - K_2) \end{aligned} \tag{3.28}$$

Hence, by letting $\mathbf{x}_k$ vary over the space of all possible bounding boxes, (3.27) allows us to compute, up to the additive constant γ, a non-parametric estimation $h(\cdot)$ of the log-PDF of the current state vector of the RBE tracker. This holds independently of the PDF of the state.

In the case of the Kalman Filter, the PDF of the state vector (i^b, j^b, w, h) is Gaussian. In such a case, the variables (i_L, j_T, i_R, j_B) are a linear combination of Gaussian Random Variables. Moreover, we are assuming that variables (i_L, j_T, i_R, j_B) are independent. Therefore, the variables (i_L, j_T, i_R, j_B) are jointly Gaussian and the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ of the state variables are fully defined by the four means μ_L, μ_R, μ_T, μ_B and the four variances σ_L^2, σ_R^2, σ_T^2, σ_B^2 of (i_L, j_T, i_R, j_B).

To estimate these eight parameters, let us substitute the expression of the Gaussian PDF for $p(\mathbf{x}_k)$ in the left-hand side of (3.27), thus obtaining:

$$\delta - \ln(\sigma_L\sigma_R\sigma_T\sigma_B) - \frac{(i_L-\mu_L)^2}{2\sigma_L^2} - \frac{(i_R-\mu_R)^2}{2\sigma_R^2} - \frac{(j_T-\mu_T)^2}{2\sigma_T^2} - \frac{(j_B-\mu_B)^2}{2\sigma_B^2} = h(\mathbf{x}_k, \mathbf{p}) \tag{3.29}$$

where $\delta = \gamma - 2\ln(2\pi)$. The eight parameters of the PDF and the additive constant δ might be estimated by imposing (3.29) for a number $N > 9$ of different bounding boxes and then solving numerically the obtained over-determined system of N non-linear equations in 9 unknowns.

To avoid such a challenging problem, we propose an approximate procedure. First of all, an estimate $\widehat{\boldsymbol{\mu}}$ of the mean of the state vector $\boldsymbol{\mu} = (\mu_L, \mu_R, \mu_T, \mu_B)$ can be obtained by observing that, due to increasing monotonicity of logarithm, the mode of the computed log-PDF coincides with the mode of the PDF, and that, due to the Gaussianity assumption, the mode of the PDF coincides with its mean. Hence, we obtain an estimate $\widehat{\boldsymbol{\mu}}$ of $\boldsymbol{\mu}$ by searching for the bounding box maximizing $h(\cdot)$.

$$\widehat{\boldsymbol{\mu}} = \arg\max_{\mathbf{x}} h(\mathbf{x}, \mathbf{p}) \tag{3.30}$$

Then, we impose that (3.29) is satisfied at the estimated mean point $\widehat{\boldsymbol{\mu}}$ and that all the variances are equal, *i.e.* $\sigma_L^2 = \sigma_R^2 = \sigma_T^2 = \sigma_B^2 = \sigma^2$, thus obtaining a functional relationship between the two remaining parameters δ and σ^2:

$$\delta \;=\; 2\ln\sigma^2 + \; h(\widehat{\boldsymbol{\mu}}, \mathbf{p}) \tag{3.31}$$

By substituting in (3.29) the above expression for δ and the estimated $\widehat{\boldsymbol{\mu}}$ for $\boldsymbol{\mu}$, we can compute an estimate $\widehat{\sigma}^2(\mathbf{x})$ of the variance σ^2 by imposing (3.29) for whatever bounding box $\mathbf{x} \neq \widehat{\boldsymbol{\mu}}$. In particular, we obtain:

$$\widehat{\sigma}^2(\mathbf{x}) \;=\; \frac{1}{2}\,\frac{\left\|\widehat{\boldsymbol{\mu}} - \mathbf{x}\right\|_2^2}{h(\widehat{\boldsymbol{\mu}}, \mathbf{p}) - h(\mathbf{x}, \mathbf{p})} \tag{3.32}$$

To achieve a more robust estimate, we average $\widehat{\sigma}^2(\mathbf{x})$ over a neighborhood of the estimated mean bounding box $\widehat{\boldsymbol{\mu}}$. Finally, to obtain the means and covariance of the measurements for the Kalman Filter, we exploit the property of linear combinations of Gaussian variables:

$$\boldsymbol{\mu} = \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{-1} \end{bmatrix} \widehat{\boldsymbol{\mu}} \qquad \boldsymbol{\Sigma} = \widehat{\sigma}^2 \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{-1} \end{bmatrix}^T \tag{3.33}$$

3.6 Experimental Results

We have tested the proposed Bayesian loop on publicly available datasets with ground truth data, *i.e.* some videos from the CAVIAR[1] and ISSIA Soccer datasets [22]. The former comprises videos from typical video-surveillance scenarios, whereas the latter deals with a football match.

We have used a Kalman Filter with constant velocity motion model as RBE tracker and the algorithm introduced in Sec. 3.4 as Bayesian change detection. The detection to initialize the tracker was done manually from the ground truth (although change detection holds the potential to solve the detection problem in the same conceptual framework, an advantage over tracking systems based on other approaches such as e.g. color histograms). We have selected videos with a single person or where the tracked person was well separated from the others[2].

In particular, the complete system has been used to track people wondering in a shopping mall using three sequences from the CAVIAR dataset (referred to as CAVIAR1, CAVIAR2, CAVIAR3, respectively) and two players during a match in the sixth sequence of the ISSIA dataset (ISSIA_GK and ISSIA_P). Tracking results for these videos are available at the companion website.

As for the CAVIAR dataset, the main difficulties are changes in appearance of the target due to light changes inside and outside the shop, shadows, camouflage, small size of the target and, for sequence 2, dramatic changes in target size onto the image plane (he walks inside the shop until barely disappears). The ISSIA Soccer dataset is less challenging as far as color, lightening and size variations are concerned, and the players cast practically no shadow. Yet, it provides longer sequences and more dynamic targets. We used our system to track the goalkeeper and a player: the goalkeeper allows to test our system on a sequence 2500

[1] Data coming from the EC Funded CAVIAR project/IST 2001 37540, found at URL: `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`

[2] How to combine our system with proper data association algorithms and to take into account in the probabilistic analysis of the change map the multiple target scenario is an interesting subject for future work.

frames long; the player shows rapid motion changes and unpredictable poses (he even falls to the ground kicking the ball in the middle of the sequence).

Our system does not require to set a threshold to classify the output of the change detection, only the model for $p(c_{ij} = C \mid \mathbf{x}_k)$ must be set. To account for the differences between the reasoning of the cognitive feedback and the analysis of the change map, two different models must be defined, *i.e.* two different pairs of values for K_1 and K_2 must be tuned. We refer to them as K_1^{CF},K_2^{CF} and K_1^{PA}, K_2^{PA} respectively. We coarsely tuned these parameters on a sequence of the CAVIAR dataset not used for testing. The best values turned out to be

$$K_1^{CF} = 0.5, \quad K_2^{CF} = 0.0, \quad K_1^{PA} = 0.5, \quad K_2^{PA} = 0.2\ . \tag{3.34}$$

We expect these values to be generally applicable: we use them with success also on the ISSIA videos. They basically state:

- that the model for both analyses must allow for unchanged pixels into the bounding box ($K_1^{CF} = K_1^{PA} = 0.5$), due to the approximation inherent to the rectangular model in presence of non rectangular and deformable targets;

- that a good prior for the change detection dictates the absence of unchanged pixels outside he bounding box ($K_2^{CF} = 0.0$);

- that, even with a such a strong prior, we must allow for a small number of errors of the Bayesian change detection out of the bounding box and left them out of the estimation we provide when analyzing the change map ($K_2^{PA} = 0.2$).

These considerations hold regardless of the sequence at hand, the illumination condition and the characteristic of the target. Hence, we see our system as a step toward easily deployable solutions for visual tracking.

We also coarsely tuned the values for the Kalman filter state covariance matrix using the same sequence. We use a constant velocity motion model, thereby adding the velocity of the target along the *i* and *j* axes to the state vector. The best values turned out to be:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.35}$$

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.36}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.37}$$

with the state vector $\mathbf{x}_k$ given by

$$\mathbf{x}_k = \begin{bmatrix} i_k^b & \frac{di_k^b}{dk} & j_k^b & \frac{dj_k^b}{dk} & w_k & h_k \end{bmatrix}^T . \tag{3.38}$$

To quantitatively evaluate the performance we use the mean dice overlap d_k over a sequence, introduced in the previous chapter (Sec. 2.4.1 :

$$d_k = \frac{2\left|\mathbf{x}_k \cap \mathbf{x}_k^{GT}\right|}{|\mathbf{x}_k| + \left|\mathbf{x}_k^{GT}\right|} . \tag{3.39}$$

Quantitative evaluation is reported in Table 3.1. Our system, whose results are reported in the first column, successfully tracks all the tar-

Table 3.1: Performance scores. $^{(*)}$ indicates loss of target.

Seq.	Full Loop	Constant R	Kalm+MS	FragTrack
CAVIAR 1	0.74	0.64	$0.29^{(*)}$	0.55
CAVIAR 2	0.66	0.66	$0.01^{(*)}$	$0.01^{(*)}$
CAVIAR 3	0.70	0.64	$0.012^{(*)}$	$0.01^{(*)}$
ISSIA_GK	0.70	0.65	0.74	$0.02^{(*)}$
ISSIA_P	0.61	0.56	0.64	$0.02^{(*)}$

gets. The main source of misalignment between the bounding box and the ground truth in the CAVIAR dataset are shadows (first column of Fig. 3.5 and 3.6): because of the position of the artificial lights, cast shadows on the floor fit with our rectangular model and the analysis of the change map tends to include them, elongating the bounding box (*e.g.* the frames # 368 707 and 1046 of sequence CAVIAR 2, depicted in Fig. 3.5). Although many proposals for shadow removal exist [77] and could be used in a real deployment of our system, we present results without such post processing step to better characterize our proposal and show its robustness to disturbance factors.

On the ISSIA videos, too, our tracker was able to successfully track both targets throughout the whole sequence, as shown in Fig. 3.7. The main limitation of our algorithm in this case is due to the assumed rectangular model: in many frames, the players are running or performing extreme movements and their limbs cover a wider area than when a person is *e.g.* walking. Hence, the actual changed area inside the ground truth bounding box differs from a rectangular shape and the measures of our system are always too conservative in size with respect to the ground truth (*e.g.* frames # 656 and 768 of the player sequence in Fig. 3.7). Nevertheless, it is remarkable that our tracker is able to adapt to extreme situations, such as the player falling on the ground (second frame in the same sequence). It is also important that it succeeded in tracking the goalkeeper, although this sequence is easier than that of the player, because this is a long sequence, and it shows that the proposed loop does not incur in positive feedbacks and divergence.

To highlight the importance of the full Bayesian loop, we have performed the same experiments without considering the full PDF estimated during the change map analysis, but just the mean and a constant measurement covariance matrix **R** equal to

$$\mathbf{R} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}. \tag{3.40}$$

Results for this configuration are reported in the second column of Tab. 3.1: our proposal performs consistently better throughout all the sequences (only for one sequence, results are identical). Going into more details, the superior performance is given by the ability of our full loop to be closer to the ground truth bounding box even when the rectangular shape assumption is violated (*e.g.* compare frames # 720 in the CAVIAR1 experiment reported in Fig. 3.4 and # 487 in CAVIAR3 experiment reported in Fig. 3.6, where the feet and the head lay outside of the bounding box estimated by the partial loop). This is in turn due to the dynamic estimation of the measurement covariance matrix: in all the frames where the rectangular model is not adequate, the probabilistic analysis of the change map is able to detect such mismatch by obtaining a higher uncertainty on its bounding box estimation (that for such frames tends to concentrate on the target trunk) and this allows the Kalman filter to trust less the measure and, hence, to be more accurate. The same observation explains the difference in performance in the ISSIA dataset.

We also compare the performance of our tracker against two standard solution for visual tracking: Mean Shift tracker used in conjunction with a Kalman Filter (KalmanMS) [17] and FragTrack [1]. They are based, respectively, on the color histogram of the whole target (*i.e.* this tracker ignores spatial distribution of the colors on the target) and on the graylevel histogram of each cell of two grids superimposed on the target.

Results for these trackers are reported in the third and fourth column of Tab. 3.1, respectively.

The first sequence we consider from the CAVIAR dataset is the easiest one in our tests. There are no scale changes, no motion low changes (the person walks with practically constant velocity from right to left), and moderate changes in appearance, due to the not uniform light intensity in the corridor of the mall. Nevertheless this sequence turns out to be too difficult for the KalmanMS tracker and tough to handle for FragTrack. This is due to two factors: the moderate changes in appearance of the target and the hypothesis of a rectangular target, assumed also by these trackers. These two factors cause the KalmanMS tracker to provide poor tracking in the beginning of the sequence, not being able to adapt to the deformations of the target (*i.e.* to include in the bounding box the wide open legs in frame # 736 of Fig. 3.4) since the trunk alone fits better with the initial model; and then, to drift to the background and loose the target, since, due to the appearance change of the target, the best matching parts of the initial histogram are those of the background, that were included in the initial model, even if it was initialized from the ground truth, due to the approximate rectangular model. FragTrack performs definitely better, although it is less precise on the estimation of the bounding box than our system, *e.g.* it cuts the feet and the head of the target in the third and fourth frame of the sequence reported in Fig. 3.4. Similarly to KalmanMS, though, it can not handle appearance changes: at the end of the sequence it looses the target (last two frames in Fig. 3.4) by considering the background more similar to the initial appearance of the target.

The other two CAVIAR sequences are too difficult for a tracker based on color or graylevel histograms. Both the KalmanMS tracker and the FragTrack loose the target at the beginning of the sequence. The most likely cause for this is that they are also very sensitive to the initialization condition: in contrast with the previous sequence, where in the first frame it was possible to reasonably approximate the target with a rectangular bounding box, this is not possible in the first frames of these two sequences (compare the first row of Fig. 3.4 with those of Fig. 3.5 and 3.6). Because of this, a lot of background is included in the initial model, and this makes the tracker stick to the initial position and loose the target. Such sensitivity is less important for bigger targets. Therefore, we can conclude that our solution, which is unaffected by this initialization problem, is more suitable than the considered alternatives for visual surveillance scenarios, where targets are usually small and untextured.

On the ISSIA sequences, KalmanMS obtains slightly better performances than our proposal. Of course, color is an important cue to successfully track the players in such scenes. This is strengthen by the fact that, for the particular colors in these scenes, the compression to gray levels is particularly lossy: for example, yellow parts of the tracked players get really similar to the green background. This is confirmed by the poor performances of FragTrack, which uses graylevel images like our system. Despite this, the difference in performance between our solution and KalmanMS is encouraging, given the gap in the quality of the analyzed cues. We expect a sensible gain in performance by deploying color-based Bayesian change detection. This represents an interesting future direction of research to continue and extend this work.

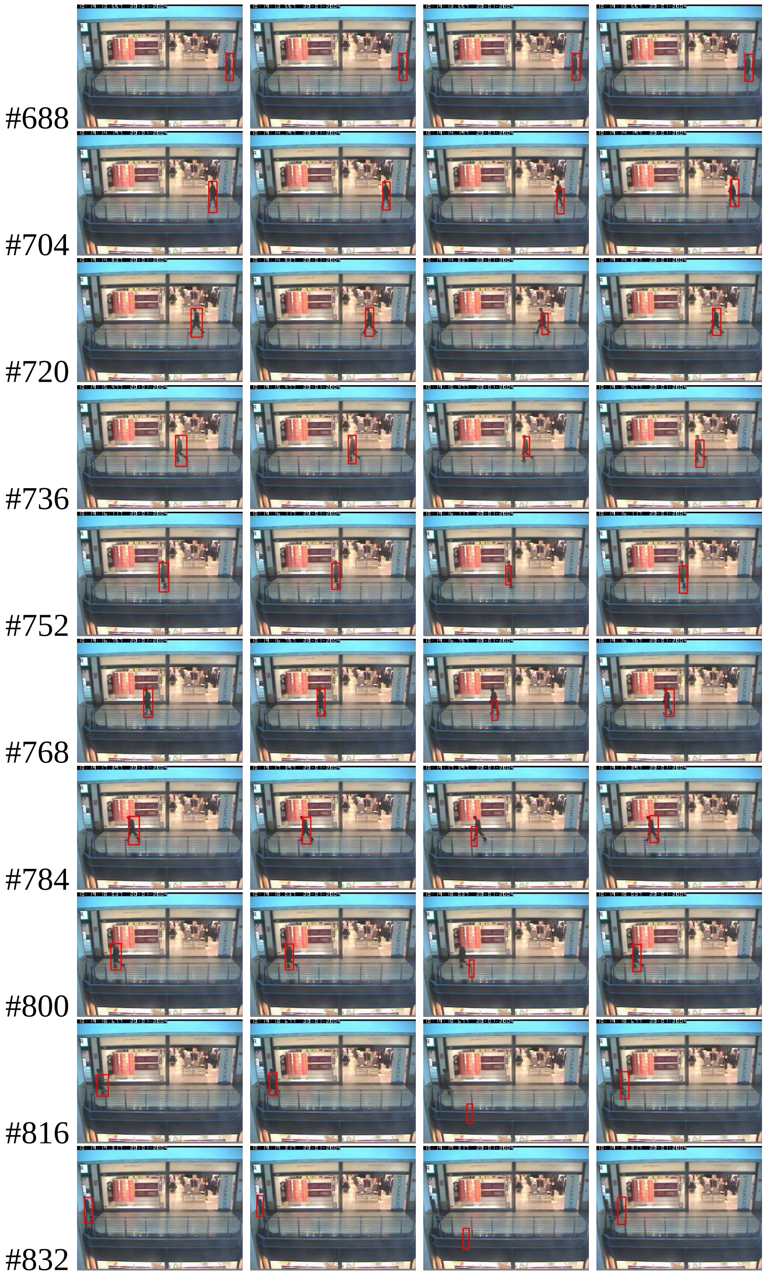


Figure 3.4: Samples equally spaced along the time axis from the CAVIAR1 experiment (sequence ”OneStopEnter2front” from the CAVIAR dataset). From left to right column: our method (full loop; our method with constant measurement covariance matrix(constant R); KalmanMS; FragTrack.

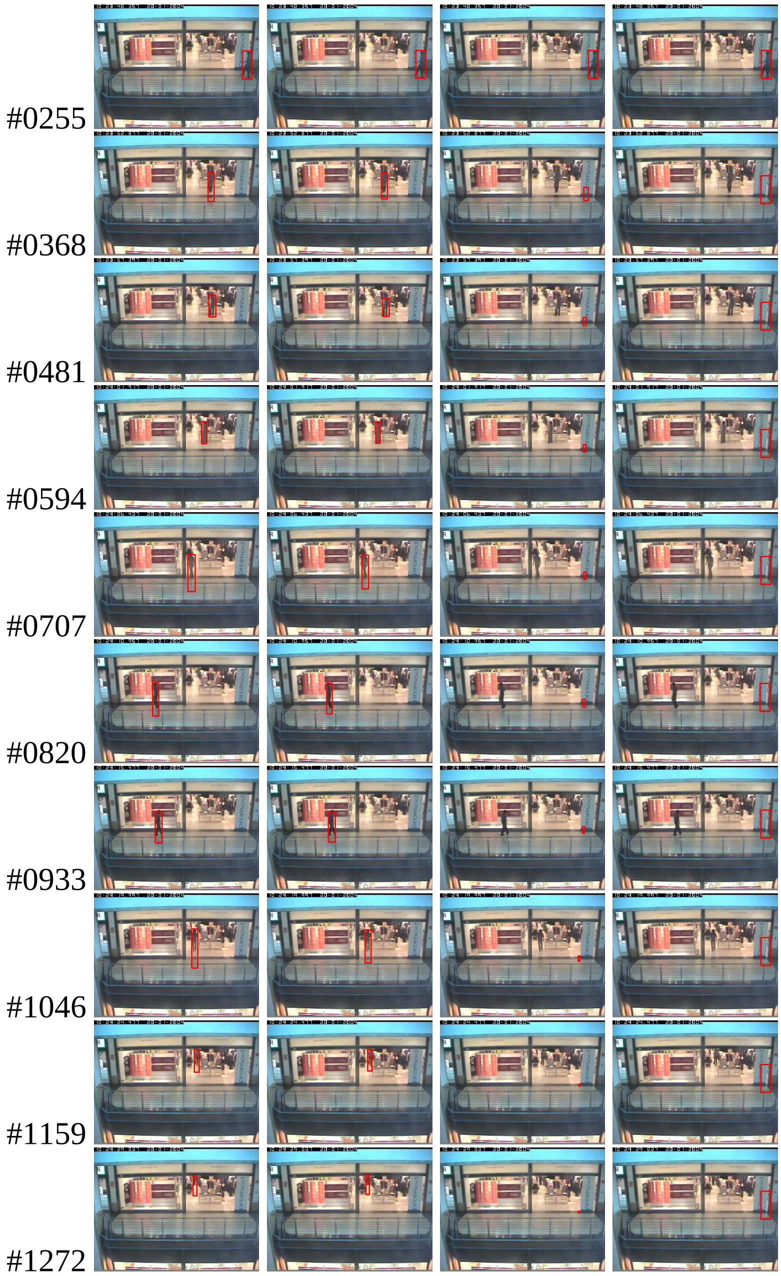


Figure 3.5: Samples equally spaced along the time axis from the CAVIAR2 experiment (sequence ”OneStopMoveEnter2front” from the CAVIAR dataset). From left to right column: our method (full loop); our method with constant measurement covariance matrix (constant R); KalmanMS; FragTrack.

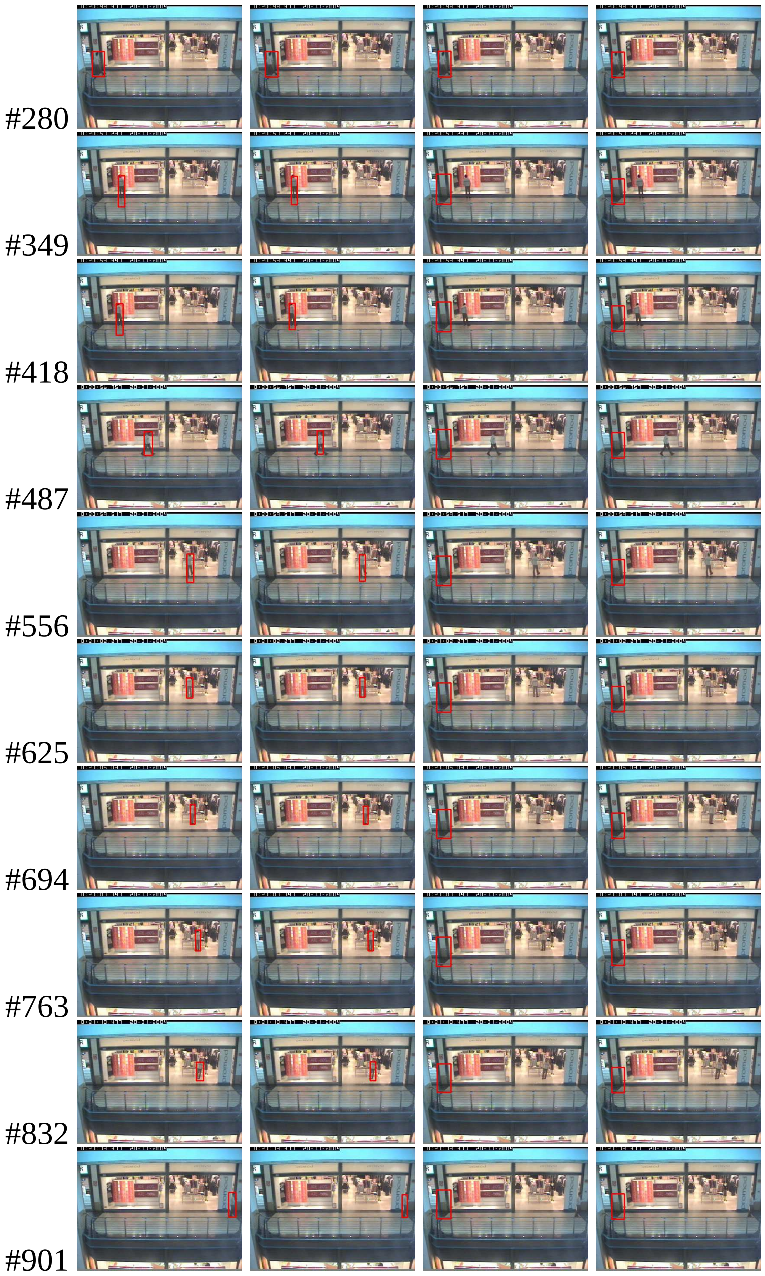


Figure 3.6: Samples equally spaced along the time axis from the CAVIAR3 experiment (sequence ”OneStopMoveNoEnter1front” from the CAVIAR dataset). From left to right column: our method (full loop); our method with constant measurement covariance matrix (constant R); KalmanMS; FragTrack.

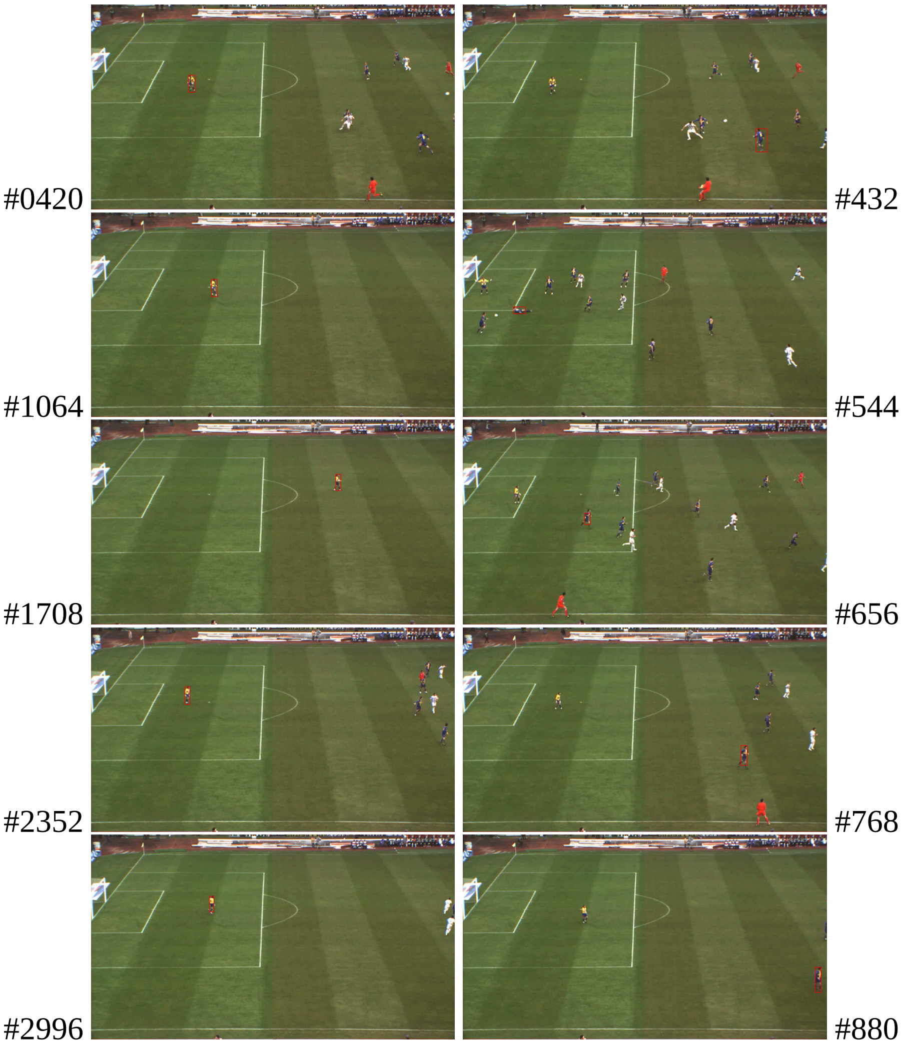


Figure 3.7: Exemplar frames equally spaced along the time axis from the ISSIA Soccer dataset: left column, the goalkeeper tracking experiment (ISSIA_GK); right column, the player tracking experiment (ISSIA_P).

Chapter 4

3D Surface Matching and Object Categorization

Automatic recognition of shapes in 3D data, also referred to as *shape matching,* is attracting a growing interest in the research community, with applications found in areas such as shape retrieval, shape registration, object recognition, manipulation and grasping, robot localization and navigation. An important enabling factor for the development of this technology is represented by the increasing availability of cheaper and more effective 3D sensors. Many of these sensors are able to acquire not only the 3D shape of the scene, but also its texture: this is the case, e.g. of stereo sensors, structure-from-motion systems, certain laser scanners as well as the recently proposed *Kinect* device by Microsoft.

Surface matching can be tackled by either a global or a local approach. According to the former, a surface is described entirely by means of global features, whereas the latter relies on local keypoints and regional feature descriptions to determine point-to-point correspondences between surfaces. Borrowing a denomination typical of the face recognition community [110] we refer here to these two approaches as, respectively, *holistic* and *feature-based*. While the holistic approach is popular in the context of 3D *object retrieval* [39, 71, 87], feature-based methods are inherently more effective for 3D *object recognition* in pres-

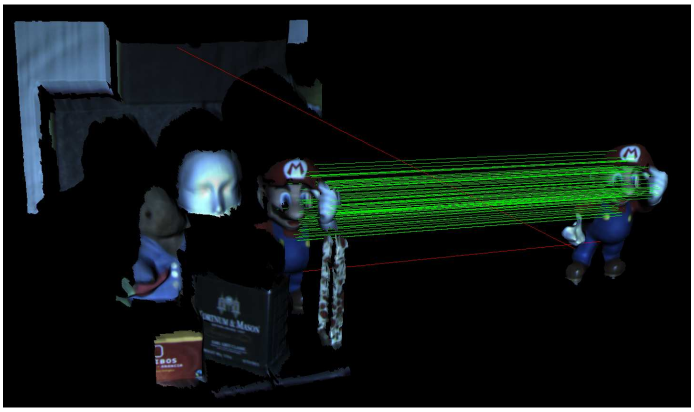


Figure 4.1: Example of matching local descriptors in a 3D object recognition scenario. Green lines identify correct matches, whereas red ones represent wrong correspondences.

ence of cluttered backgrounds and occlusions.

Feature-based methods rely on 3D keypoints that are extracted from a 3D surface. This task is accomplished by 3D detectors, whose aim is to determine points which are distinctive, to allow for effective description and matching, and repeatable with respect to point-of-view variations and noise [12, 60, 111]. Sometimes, a characteristic scale is also associated to each keypoint, so as to provide a local neighborhood to the following description stage [2, 60, 66, 98, 106]. Then, a description of the local neighborhood of each keypoint is computed by means of a 3D descriptor [12, 14, 27, 41, 60, 66, 106, 111] in order to obtain a compact local representation of the input data invariant up to a predefined level of transformation (rotation, scaling, affine warp, . . .). Descriptors are finally matched across different views to attain point-to-point correspondences (*e.g.* as in Fig. 4.1). This approach has become the standard paradigm in case of 2D data [6, 10, 43, 54, 56, 61, 62] for tackling classical computer vision problems such as object recognition, automatic registration, image indexing, etc...

Object categorization is among the most stimulating, yet challenging, computer vision tasks. It consists of automatically assigning a category to a particular object given its representation (an image, a point cloud, ..) and a predefined taxonomy. This is different from object recognition, which consists of recognizing a particular instance of a particular class (*i.e.* an object recognition algorithm is trained to recognize a specific car whereas an object category recognition algorithm is trained to recognize all cars as members of the same class) and more challenging.

We develop a novel object category recognition algorithm by solving the surface matching problem based on local features. The main contributions are as follows:

- a novel comprehensive proposal for surface representation, dubbed *SHOT*, which encompasses a new unique and repeatable local reference frame as well as a new 3D descriptor;
- the modification of this proposal to exploit texture, provided by the output of modern 3D sensors;
- the extension of the Implicit Shape Model [50] approach to the categorization of 3D data described by means of the SHOT method.

4.1 SHOT descriptor

This section deals with our proposal for local 3D description. First, we categorize existing methods into two classes: *Signatures* and *Histograms*. Then, by discussion and experiments alike, we point out the key issues of uniqueness and repeatability of the local reference frame. Based on these observations, we formulate a novel comprehensive proposal for surface representation, which encompasses a new unique and repeatable local reference frame as well as a new 3D descriptor. The latter lays at the intersection between Signatures and Histograms, so as to possibly achieve a better balance between descriptiveness and robustness. Experiments on publicly available datasets as well as on range scans obtained with *Spacetime Stereo* provide a thorough validation of our proposal, which is shown to outperform clearly three well-known state of the art methods.

4.1.1 Analysis of Previous Work

In Table 4.1 we propose a categorization of the main proposals in the field. As shown in the second column, we divide proposals for 3D descriptors into two main categories, namely *Signature* and *Histogram*. The first category, that includes earliest works on the subject, describes the 3D surface neighborhood of a given point (hereinafter *support*) by defining an invariant local Reference Frame (RF) and encoding, according to the local coordinates, one or more geometric measurements computed individually on each point of a subset of the support. On the other hand, Histogram-based methods describe the support by accumulating local geometrical or topological measurements (e.g. point counts, mesh triangle areas) into histograms according to a specific quantized domain (e.g. point coordinates, curvatures) which requires the definition of either a Reference Axis (RA) or a local RF. In broad terms, signatures are potentially highly descriptive thanks to the use of spatially well localized information, whereas histograms trade-off descriptive power for robustness by compressing geometric structure into bins.

As far as Signature-based methods are concerned, one of the first proposals is *Structural Indexing* [91], which builds up a representation based on either a *3D curve* or a *Splash* depending on the characteristics of the 3D support. The former encodes the angles between consecutive segments of the polygonal approximation of edges (corresponding to depth or orientation discontinuities) on the surface. The latter encodes as a 3D curve the local distribution of surface orientations along a geodesic circle centered on the point. In *Point Signatures* [14] the signature is given by the signed height of the 3D curve obtained by intersecting a sphere centered in the point with the surface. *3D Point Fingerprint* [92] encodes the normal angle variations and the contour radius variations along different geodesic circles projected on the tangent plane. Recently, *Exponential Mapping* [66] proposed a descriptor that encodes the components of the normals within the support by deploying a 2D parametrization of the local surface.

Table 4.1: Taxonomy of 3D descriptors.

Method	Category	Local RF	
		Unique	Unambig.
StInd [91]	Signature	No	Yes
PS [14]	Signature	No	Yes
3DPF [92]	Signature	No	Yes
EM [66]	Signature	Yes	No
SI [41]	Histogram	RA	
LSP [12]	Histogram	RA	
3DSC [27]	Histogram	No	Yes
ISS [111]	Histogram	Yes	No
Tensor [59]	Histogram	No	Yes
MeshHoG [106]	Both	Yes	Yes
SHOT	**Both**	**Yes**	**Yes**

As for Histogram-based methods, those relying on the definition of just a RA are typically based on the feature point normal. For example, *Spin Images* [41], arguably the most popular method for 3D mesh description, computes 2D histograms of points falling within a cylindrical volume by means of a plane that ”spins” around the normal. Within the same subclass, *Local Surface Patches* [12] computes histograms of normals and *shape indexes* [44] of the points belonging to the support. As for methods relying on the definition of a full local RF, *3D Shape Context* [27] modifies the basic idea of Spin Images by accumulating 3D histograms of points within a sphere centered at the feature point. *Intrinsic Shape Signatures* [111] proposed an improvement of [27] based on a different partitioning of the 3D local volume as well as on a different definition of the local RF. Finally, Mian et al. [59] accumulate 3D histograms (*Tensors*) of mesh triangle areas within a cubic support.

Two observations steam from the taxonomy proposed in Tab. 4.1. First, all proposals rely on the definition of a local RF or, at least, a repeatable RA. However, we believe that the importance of the choice of the local reference for a 3D descriptor is underrated in literature, with efforts mainly focused on the development of discriminative descriptors.

As a consequence, approaches for the choice of the local reference are ambiguous, or not unique, or too sensitive to noise and also lack specific experimental validation. Instead, as we will show in the remainder of the chapter, the repeatability of the local RF (or, analogously, of the RA) is mandatory to achieve effective local surface description.

Therefore, one of the contributions of our work is a specific study upon local RFs. We carry out an analysis of repeatability and robustness on proposed local RFs, and provide experiments that demonstrate the strong impact of the choice of the RF on the performance of a 3D descriptor (Sec. 4.1.2). Given the impact of such a choice, we introduce a robust local RF that, unlike all other proposals, is unique and unambiguous(Sec. 4.1.3).

Secondly, based on the nature of existing approaches highlighted by the proposed categorization, it is our belief that an effective and robust solution to the problem of 3D shape description can be found as a proper combination of *Signatures* and *Histograms*. Hence, we propose a novel 3D descriptor aware of the proposed categorization (Sec. 4.1.4). Its design, inspired by the analysis of the successful choices performed in the related field of 2D descriptors [54], has been explicitly conceived to achieve computational efficiency, descriptive power and robustness. Recently, MeshHoG [106] another approach for 3D data description that can be seen as an attempt to combine the benefits of Signatures and Histograms, was proposed. We will show in the experimental results that our proposal consistenly outperforms it.

4.1.2 On the traits and importance of the local RF

The definition of a local RF, invariant to translations and rotations and robust to noise and clutter, has been the preferred option to endow a 3D descriptor with invariance to the same sources of variations, similarly to the way rotation and/or scale invariance is injected into 2D descriptors. On the other hand, the definition of such an invariant frame is challenging. Furthermore, although almost every new proposal for local shape description is equipped with its own local RF, experimental validation has always been focused on the results obtained by the joint used of an RF and a descriptor, whilst the impact of the selected local RF on the descriptor performance has not been investigated in literature.

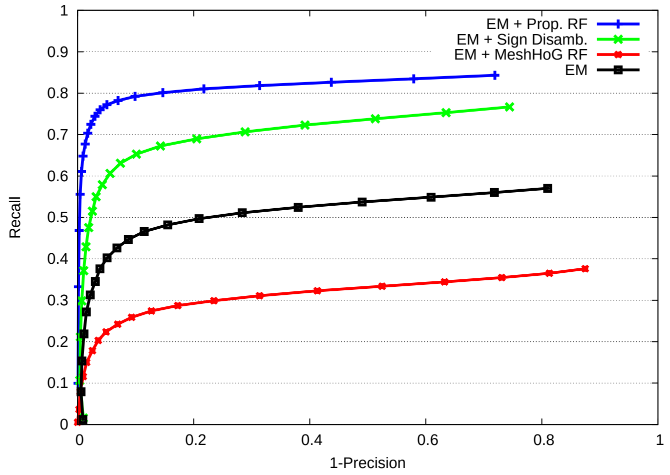


Figure 4.2: Impact of the local RF on a descriptor performance. The optimal point is located at the top left side of the chart.

In Table 4.1 we have reported for each proposal the properties of uniqueness and unambiguity of their local RF. As highlighted in the third column, the majority of proposals are based on RFs that are not *unique* [91] [14] [92] [27] [59], i.e. to obtain an invariant description they require multiple descriptors to be computed at each feature point. This is usually handled by describing a ”model” point using multiple descriptors, each based on a different local RFs, and a ”scene” point with just one of them. This approach causes additional ambiguity to the correspondence problem since it shifts the intrinsic non-uniqueness of the local RF to the matching stage, thus increasing potential mismatches, computational requirements and sometimes also memory footprint. Another disadvantage brought in by the use of multiple local RFs is that the proposed matching stage is so tailored on the descriptor that it prevents the use of off-the-shelf efficient solutions for matching and indexing, that in principle could be advantageously performed orthogonally with respect to the description. This may result in a severe loss of computational efficiency.

In addition to multiple RFs, another limit of current proposals consists in the intrinsic ambiguity of the sign of the local RF axes. For example, in [66] and [111], normals and principal curvature directions are used. The main problem with this choice is that principal directions are not vectors, i.e. their sign is not defined. From a practical point of view, principal directions are computed using Singular Value Decomposition (SVD) or Eigenvalue Decomposition (EVD) of the covariance matrix of the point coordinates within the support[1]. Of course, the output of the algorithm is a vector with a sign. Nevertheless, this sign is simply a numerical accident and, thus, is not repeatable on different (e.g. rotated) instances of the same mesh, even though the same SVD/EVD algorithm is used, as clearly discussed in [9]. Therefore, such an approach to the definition of the local RF is inherently ambiguous and thus not repeatable. [111] resorts to multiple RFs to overcome this limitation, while [66] does not deal with it explicitly.

To highlight the impact of the local RF on a descriptor performance, we show in Fig. 4.2 the performance of the EM descriptor [66] with different local RFs. Results are reported as *Recall vs 1-Precision* curves (see Sec. 4.1.5 for a discussion about this choice and for the settings used in all our experiments). The ambiguous RF used in [66] leads to unsatisfactory performances (black curve). Using exactly the same settings and exactly the same descriptor, we can boost performances simply by deploying the Sign Disambiguation technique recently proposed in [9] (green curve). Furthermore, using the more robust and more repeatable local RF that we propose in next section we can obtain another significant improvement (e.g. at recall 0.7 precision raises from 0.308 to 0.994) without changing the descriptive power of the descriptor (blue curve). It is also worth pointing out here that our local RF does not match perfectly the EM descriptor, for none of its axes provides an approximation of the local normal that is instead assumed by the theory underneath the

[1] From personal communication with the authors of [66] and as reported in [111].

EM descriptor. Nevertheless, performances with our local RF are better than those obtained with the original proposal, showing the overwhelming importance of a robust, repeatable local RF. The importance of a robust RF is confirmed by the use of the EM descriptors with the only other unique and unambiguous local RF, part of the MeshHoG algorithm [106]. Such local RF is based on curvatures, which are highly sensitive to noise. This results in a poorly repeatable RF, which negatively influence the descriptor performances (red line).

4.1.3 Disambiguated EVD for a repeatable RF

As shown by Table 4.1, none of current local RF proposals but that of MeshHoG is at the same time unique and unambiguous. The local RF defined by the MeshHoG descriptor is highly sensitive to noise, as shown in the previous section. Hence, there is a lack of a robust, unique and unambiguous RF. To fill this gap we have designed and extensively tested a variety of novel unique and unambiguous local RFs. We present here the method that turned out to be the most robust in our thorough experimental evaluation. It builds on a well known technique presented in [35] and [63], where the problem of normal estimation in presence of noise is specifically addressed. A Total Least Squares (TLS) estimation of the normal direction is obtained in [35] and [63] by EVD of the covariance matrix **M** of the $k-$nearest neighbors p_i of the point, defined by

$$\mathbf{M} = \frac{1}{k}\sum_{i=0}^{k}(\mathbf{p}_i - \hat{\mathbf{p}})(\mathbf{p}_i - \hat{\mathbf{p}})^T, \ \hat{\mathbf{p}} = \frac{1}{k}\sum_{i=0}^{k}\mathbf{p}_i \ . \tag{4.1}$$

In particular, the TLS estimation of the normal direction is given by the eigenvector corresponding to the smallest eigenvalue of M. Finally, they perform the sign disambiguation of the normals *globally* by means of sign consistency, i.e. propagating the sign from a seed chosen heuristically.

While this has proven to be a robust and effective technique for surface reconstruction of a single object, it cannot work for local surface description since in the latter case signs must be repeatable across any possible object pose as well as in scenes with multiple objects, so that a *local* rather than global sign disambiguation method is mandatory. Moreover, Hoppe's sign disambiguation concerns the normal only, hence it leaves ambiguous the signs of the remaining two axes.

In our proposal, we start by modifying (4.1) so as to assign distant points smaller weights, in order to increase repeatability in presence of clutter. Then, to improve robustness, all points laying within the spherical support (of radius R) which are used to compute the descriptor are used also to calculate $\mathbf{M}$. For the sake of efficiency, we also neglect the centroid computation, replacing it with the feature point $\mathbf{p}$. Therefore, we compute M as a weighted linear combination,

$$\mathbf{M} = \frac{1}{\sum_{i:d_i \le R} (R-d_i)} \sum_{i:d_i \le R} (R - d_i)(\mathbf{p}_i - \mathbf{p})(\mathbf{p}_i - \mathbf{p})^T \tag{4.2}$$

where $d_i = ||\mathbf{p}_i - \mathbf{p}||_2$. Our experimental evaluation indicates that the eigenvectors of $\mathbf{M}$ define repeatable, orthogonal directions in presence of noise and clutter. It is worth pointing out that, compared to [35] and [63], in our proposal the third eigenvector no longer represents the TLS estimation of the normal direction and sometimes it notably differs from it. However, this does not affect performance, since in the case of local surface description what matters is a highly repeatable and robust triplet of orthogonal directions, and not its geometrical or topological meaning.

Hence, eigenvectors of (4.2) represent a good starting point, but they need to be disambiguated to yield a repeatable local RF. The problem of sign disambiguation for EVD and SVD has been recently addressed in [9]. Their proposal basically reorients the sign of each singular or eigenvector so that its sign is coherent with the majority of the vectors it is representing. We determine the sign on the local x and z axes according to this principle. In the following we refer to the three eigenvectors in decreasing eigenvalue order as the $\mathbf{x}^+$, $\mathbf{y}^+$ and $\mathbf{z}^+$ axis, respectively. With $\mathbf{x}^-$, $\mathbf{y}^-$ and $\mathbf{z}^-$, we denote instead the opposite vectors. Hence, the final disambiguated x axis is defined as

$$S_x^+ \doteq \{i : d_i \leq R \ \wedge \ (\mathbf{p}_i - \mathbf{p}) \cdot \mathbf{x}^+ \geq 0\} \tag{4.3}$$

$$S_x^- \doteq \{i : d_i \leq R \ \wedge \ (\mathbf{p}_i - \mathbf{p}) \cdot \mathbf{x}^- > 0\} \tag{4.4}$$

$$\mathbf{x} = \begin{cases} \mathbf{x}^+, & |S_x^+| \geq |S_x^-| \\ \mathbf{x}^-, & \text{otherwise} \end{cases} \tag{4.5}$$

The same procedure is used to disambiguate the z axis. Finally, the y axis is obtained as $\mathbf{z} \times \mathbf{x}$.

We compare the repeatability of our proposal against three representative RFs: that of MeshHoG, that of PS and that of EM, respectively a not-robust solution, a not-unique solution and an ambiguous one. To prevent the shortcomings of not uniqueness and ambiguity from invalidating the comparison we consider only the global maximum of the height [14] for PS and we add the sign disambiguation of [9] to EM (EM+SD), thereby obtaining two unique and unambiguous RFs. We also consider the original EM approach to show the effectiveness of sign disambiguation. Using again the settings detailed in Sec. 4.1.5, in Fig. 4.3 we plot, for 5 increasing noise levels, the mean cosine between corresponding axes of the local RFs computed on two instances of the same mesh, i.e. the original one and a rotated and noisy instance. On one hand, ambiguity is clearly the most serious nuisance, as the low performances of the original EM proposal demonstrate. On the other hand, the use of a higher number of points to compute the local RF (i.e. the whole surface contained in the spherical support, as done by EM, instead of the 3D curve resulting by the intersection of the spherical support with the surface, as done by PS) yields better robustness, as shown by the relative drop of EM with respect to PS when noise increases. Nevertheless, the steepest drop of performance is provided by MeshHoG, which confirms the need to ground local RF computation on more robust features than second order differential entities like curvatures.The disambiguation introduced in EM+SD dramatically enhances repeatability. However, both EM and EM+SD subordinate computation of the directions on the tangent plane to the normal estimation (i.e. , the repeatable directions they compute are then projected onto the tangent plane to create an orthogonal basis). This choice sums noise on the normal to the noise inevitably affecting the other directions, thereby leading to increased sensitivity of the estimation of the axes on the tangent plane and finally to poor repeatability. Our proposal, instead, estimates all axes simultaneously and turns out to be the most effective, thanks to the combination of its noise and clutter-aware definition, the effectiveness of the proposed disambiguation and the inherent uniqueness deriving from its theoretical formulation.

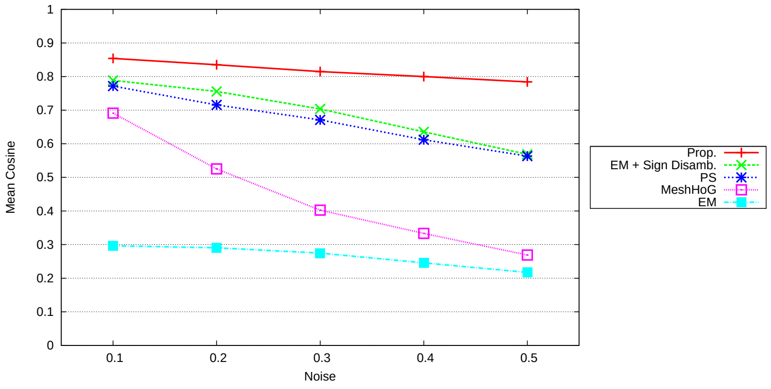


Figure 4.3: Comparison between local RFs.

4.1.4 Description by Signatures of Histograms

In Sec. 4.1.1 we have classified 3D descriptors as based on either histograms or signatures. We have designed our proposal following this intuition and aiming at a local representation that is efficient, descriptive, robust to noise and clutter as well as to point density variation. The point density issue is specific to the 3D scenario, where the same 3D volume of the real world may be represented with different amounts of vertexes in its mesh approximation, e.g. due to the use of different 3D sensors (stereo, Time-of-Flight cameras, LIDARs, etc...) or different acquisition distances.

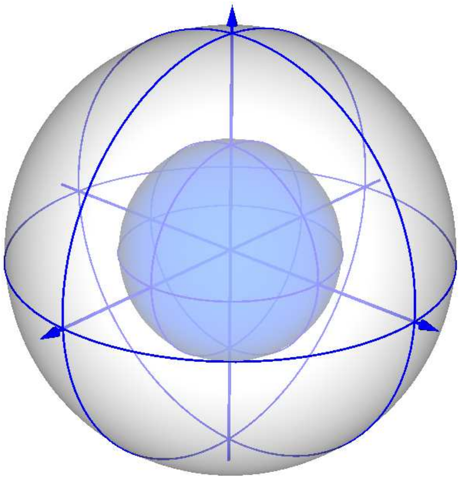


Figure 4.4: Signature structure for SHOT.

Besides our taxonomy, another source of inspiration has been the related field of 2D feature descriptors, which has reached a remarkable maturity during the last years. By analyzing SIFT [54], arguably the most successful and widespread proposal among 2D descriptors, we have singled out what we believe are among the major reasons behind its effectiveness. First of all, the use of histograms is spread throughout the algorithm, from the definition of the local orientation to the descriptor itself, this accounting for its robustness. The low descriptive power of a global histogram computed on the whole patch is balanced by the introduction of coarse geometric information: the descriptor is, in fact, a concatenation of histograms, each computed on a precise location in a regular grid superimposed on the patch. The use of this coarse geometric information creates what we identify as a signature-like structure.

Moreover, the elements of these local histograms are based on first order derivatives describing the signal of interest, i.e. intensity gradients. Although it has been argued that building a descriptor based on differential entities may result in poor robustness to noise [14], they hold high descriptive power, as the effectiveness of SIFT clearly demonstrates. Therefore, we believe they can provide a more effective solution for a descriptor than point coordinates [41] [27]. Yet, to achieve robustness to noise, differential entities have to be filtered, and not deployed directly, e.g. as done in [66].

Finally, an important part of the SIFT algorithm deals with the definition of a local invariant 2D reference frame (*i.e.* the characteristic orientation). The author states that in case of ambiguity in determining the local RF, a great benefit to the stability of matches is provided by the use of multiple orientations. This highlights the importance of a unique, unambiguous local RF for the effectiveness of a descriptor.

Based on these considerations, we propose a 3D descriptor that encodes histograms of basic first-order differential entities (i.e. the normals of the points within the support), which are more representative of the local structure of the surface compared to plain 3D coordinates. The use of histograms brings in the filtering effect required to achieve robustness to noise. Having defined an unique and robust 3D local RF (see Sec. 4.1.3), it is possible to enhance the discriminative power of the descriptor by introducing geometric information concerning the location of the points within the support, thereby mimicking a signature. This is done by first computing a set of local histograms over the 3D volumes defined by a 3D grid superimposed on the support and then grouping together all local histograms to form the actual descriptor. Hence, our descriptor lays at the intersection between Histograms and Signatures: we dub it Signature of Histograms of OrienTations (SHOT).

For each of the local histograms, we accumulate point counts into bins according to a function of the angle, θ_i, between the normal at each point within the corresponding part of the grid, $\mathbf{n}_{v_i}$, and the normal at the feature point, $\mathbf{n}_u$. This function is $cos\theta_i$, the reason being twofold: it can be computed fast, since $cos\theta_i = \mathbf{n}_u \cdot \mathbf{n}_{v_i}$; an equally spaced binning on $cos\theta_i$ is equivalent to a spatially varying binning on θ_i, whereby a coarser binning is created for directions close to the reference normal direction and a finer one for orthogonal directions. In this way, small differences in orthogonal directions to the normal, i.e. presumably the most informative ones, cause a point to be accumulated in different bins leading to different histograms. Moreover, in presence of quasi-planar regions (i.e. not very descriptive ones) this choice limits histogram differences due to noise by concentrating counts in a fewer number of bins.

As for the structure of the signature, we use an isotropic spherical grid that encompasses partitions along the radial, azimuth and elevation axes, as sketched in Fig. 4.4. Since each volume of the grid encodes a very descriptive entity represented by the local histogram, we can use a coarse partitioning of the spatial grid and hence a small cardinality of the descriptor. In particular, our experimentations indicate that 32 is a proper number of spatial bins, resulting from 8 azimuth divisions, 2 elevation divisions and 2 radial divisions (though, for clarity, only 4 azimuth divisions are shown in Fig. 4.4). Combined with the fact that the tuning we present in sec. 4.1.5 indicates a proper number of bins for the internal histograms to be around 10, we obtain a total descriptor length of 320, a good improvement over the 1980 proposed for 3DSC [27] or the 595 for ISS [111], that allows for faster indexing and matching.

Since our descriptor is based upon local histograms, it is important to avoid boundary effects, as pointed out e.g. in [41] [54]. Furthermore, due to the spatial subdivision of the support, boundary effects might arise also in presence of perturbations of the local RF. Therefore, for each point being accumulated into a specific local histogram bin, we perform quadrilinear interpolation with its neighbors, i.e. the neighboring bins in the local histogram and the bins having the same index in the local histograms corresponding to the neighboring volumes of the grid. In particular, each count is multiplied by a weight of $1 - d$ for each dimension. As for the local histogram, d is the distance of the current entry from the central value of the bin. As for elevation and azimuth, d is the angular distance of the entry from the central value of the volume. Along the radial dimension, d is the Euclidean distance of the entry from the central value of the volume. Along each dimension, d is measured in units of the histogram or grid spacing, i.e. it is normalized by the distance between two neighbor bins or volumes.

To achieve robustness to variations of the point density, we normalize the whole descriptor to sum up to 1. This is preferable to the solution proposed in [27], i.e. normalizing each bin with the inverse of the point density and bin volume. In fact, while [27] implicitly assumes that the sampling density may vary independently in every bin, and thus discards as not informative the differences in point density among bins, we

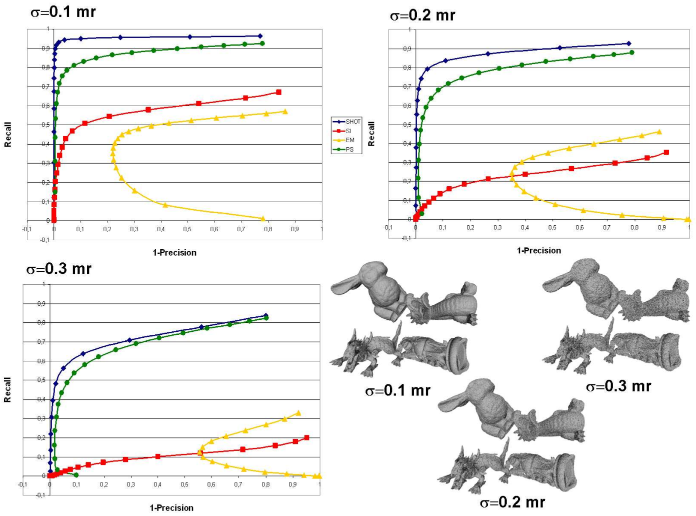


Figure 4.5: Exp. 1: Precision-Recall curves on Stanford dataset and a scene at the 3 noise levels.

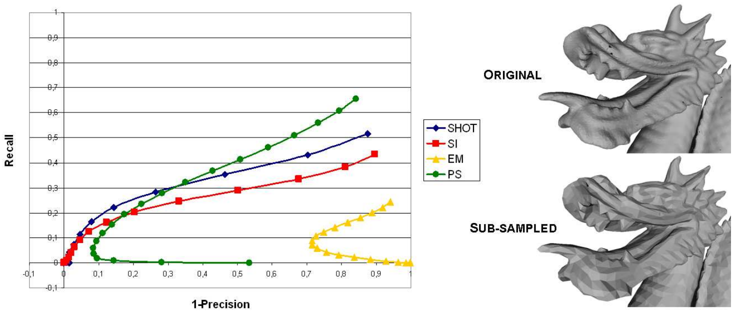


Figure 4.6: Exp. 2: Precision-Recall curves on subsampled dataset and a detail from one scene.

assume global (or at least regional) variations of the density and keep the local differences as a source of discriminative information.

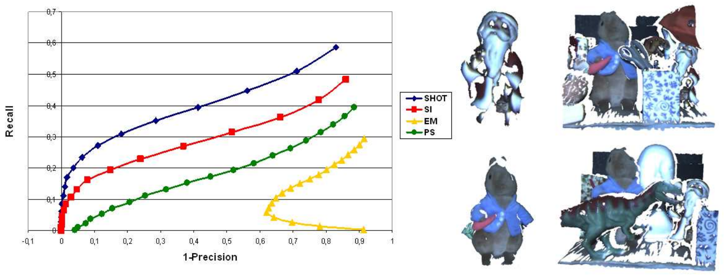
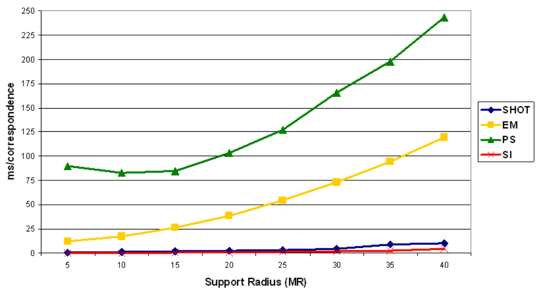
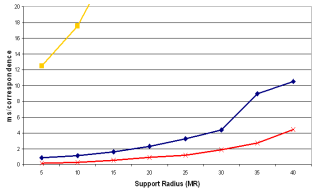


Figure 4.7: Exp. 3: Results on Spacetime Stereo dataset and two models (middle) and scenes (right).





	Time (s)	Radius (mr)	Length
SHOT	4.8	15	320
SI	5.6	30	100
EM	52.6	10	2700
PS	248.8	10	90

Figure 4.8: Charts: ms/correspondence vs. support radius (in the smaller chart the time axis is zoomed in for better comparison between SI and SHOT). Table: measured execution times (in Experiment 1) and tuned parameter values. Radius values are reported in mesh resolution units. As for SI, the support radius is the product of the bin size by the number of bins in each side of the spin image.

4.1.5 Experimental results

Surface Matching

In this section we provide experimental validation of our proposals, i.e. the unique local RF together with the SHOT descriptor. To this purpose, we carry out a quantitative comparison against three state-of-the-art approaches in a typical surface matching scenario, where correspondences have to be established between a set of features extracted from a scene and those extracted from a number of models. The considered approaches are: *Spin Images* (SI), as representative of Histogram-based methods due to its vast popularity in the addressed scenario; *Exponential Mapping* (EM) and *Point Signatures* (PS) as representatives of Signature-based methods, the former since it is a very recent approach, the latter given its importance in literature. All methods were implemented in C++ and are made publicly available together with the datasets (`www.vision.deis.unibo.it/SHOT`).

For a fair comparison, we use the same feature detector for all algorithms: in particular, we randomly extract a set of feature points from each model, then we extract their corresponding points from the scene, so that performance of the descriptors is not affected by errors of the detector. Analogously, for what concerns the matching stage, we adopt the same matching measure for all algorithms, *i.e.*, as proposed in [41], the Euclidean distance. We could also have evaluated the synergistic effect of description and matching for those methods that explicitly include a proposal for the latter, e.g. the tolerance band for PS. In turn, we did experiments on the whole dataset with the original EM and PS matching schemes, obtaining slightly worse performance for both. This, and the attempt to be as fair as possible, leaned us to use the same matching measure for all algorithms. However, we did not discard the characteristics of the descriptors that required a specific treatment during matching: in particular, since EM is a sparse descriptor, we compute the Euclidean distance only on the overlapping subset of EM descriptor pairs, as proposed by the authors; as for PS, we use the matching scheme proposed by the authors to disambiguate its not-unique local RF [14]. For each scene and model, we match each scene feature against all model features and we compute the ratio between the nearest neighbor and the second best (as in [54]): if the ratio is below a threshold a correspondence is established between the scene feature and its closest model feature.

According to the methodology for evaluation of 2D descriptors recommended in [61], we provide results in terms of *Recall* versus *Precision* curves. This choice is preferable compared to ROC curves (i.e.

True Positive Rate versus *False Positive rate*) when comparing descriptors due to the ambiguity in calculating the *False Positive Rate* [43]. We present three different experiments. Experiment 1 deals with 6 models (”Armadillo”, ”Asian Dragon”, ”Thai Statue”, ”Bunny”, ”Happy Buddha”, ”Dragon”) taken from the *Stanford 3D Scanning Repository* [2]. We build up 45 scenes by randomly rotating and translating different subsets of the model set so to create clutter[3]; then, similarly to [98], we add Gaussian random noise with increasing standard deviation, namely σ_1, σ_2 and σ_3 at respectively 10%, 20% and 30% of the average mesh resolution (computed on all models). In Experiment 2 we consider the same models and scenes as in Experiment 1, add noise (*i.e.* σ_1) and resample the 3D meshes down to 1/8 of their original point density by using MeshLab [4] Quadratic Mesh Collapse Decimation filter. For a fair comparison in this experiment, our implementation of SI -used throughout all the evaluation- normalizes each descriptor to the unit vector to make it more robust to density variations [18]. Finally, in Experiment 3 the dataset consists of scenes and models acquired in our lab by means of a 3D sensing technique known as *Spacetime Stereo* [21], [108]. In particular, we compare 8 object models against 15 scenes characterized by clutter and occlusions, each scene containing two models. Fig. 4.7 shows two scenes together with the models appearing in them. In each of the three experiments, 1000 feature points were extracted from each model. As for the scenes, in Exp. 1 and 2 we extract $n * 1000$ features per scene (n being the number of models in the scene) whereas in Exp. 3 we extract 3000 features per scene.

Throughout all the three experiments we used the same values for the parameters of the considered methods. In particular, we tuned the two parameters of each descriptor (*support radius* and *length of the descriptor*) based on a tuning scene corrupted with noise level σ_1 and built rotating and translating three Stanford models (”Bunny”, ”Happy Bud-

[2] `http://graphics.stanford.edu/data/3Dscanrep`

[3] 3 sets of 15 scenes each, containing respectively 3, 4 and 5 models

[4] `http://meshlab.sourceforge.net/`

dha", "Dragon"). The values resulting from the tuning process are reported in the last two columns of the Table in Fig. 4.8. It is worth noting that our tuning yielded comparable values of the support radius among the various methods, and that, for SI and PS, the resulting parameter values are coherent, as far as the order of magnitude is concerned, with those originally proposed by their authors (no indication about EM parameters is given in [66]). Yet, we used the finely tuned values instead of those originally proposed by the authors since the former yield higher performance in these experiments.

Results for the three Experiments are reported in Figure 4.5, 4.6 and 4.7, respectively. Experiment 1 focuses on robustness to noise. Given the reported results, it is clear that SHOT performs better than the other methods at all different noise levels on the Stanford dataset. We can observe that, comparing the two Signature methods, PS exhibits a higher robustness than EM. We address this mainly to the higher robustness of its local RF, as shown in Fig. 4.3. This, together with the good performance of SHOT, highlights the importance of deploying a robust local RF. As for SI, it appears to be highly susceptible to noise, its performance notably deteriorating as the noise level increases. This is due to the fact that this descriptor is highly sensitive to small variations in the normal estimation (i.e. SI Reference Axis), that here we compute as proposed in [41]. This is also consistent with the results reported in [27].

As for Experiment 2, it is clear that the point density variation is the most challenging nuisance among those accounted for in our experimental validation, causing a severe performance loss of all methods, even those specifically addressing it as EM. SHOT, PS and SI obtain comparable performance, nevertheless for high values of precision, that are typical working points for real applications, SHOT obtains the highest levels of Recall.

Experiment 3 shows that under real working conditions SHOT outperforms the other methods. It is worth noting that this experiment is especially focused on the descriptiveness of evaluated approaches, since the smoother shapes of the objects surfaces compared to those of the

Stanford models make the former harder to discriminate. Hence, results demonstrate the higher descriptiveness embedded in SHOT with respect to the other proposals.

In addition, we have compared the methods in terms of their computational efficiency and memory requirements. Since, as discussed in Sec. 4.1.2, descriptors based on multiple RFs, like PS, can not deploy efficient indexing to speed-up the matching stage, we use a full search strategy for all methods. Results are reported in Fig. 4.8. The two charts in the figure, showing the number of milliseconds per correspondence needed by the various methods using different support sizes, demonstrate the notable differences in computational efficiency between the algorithms. In particular, SI and SHOT run one order of magnitude faster than EM and almost two orders of magnitude faster than PS, with SI turning out consistently slightly faster than SHOT at each support size. As for EM, efficiency is mainly affected by the re-parametrization of the support needed to describe each feature point and to the large memory footprint (see next). With regards to PS, as discussed in Sec. (4.1.2) the use of multiple local RFs dramatically slows down the matching stage. These results are confirmed by the Table in the figure (first column), which reports the measured times required to match the scene to the models in Experiment 1 (i.e. 3000 scene features and 3000 models features) using the tuned parameter values. Here, the larger support needed by SI allows SHOT to run slightly faster. As for memory requirements, the reported descriptor length (third column) highlights the much higher memory footprint required by EM compared to other methods.

3D registration

As a practical application in a challenging and active research area, we demonstrate the use of SHOT correspondences to perform fully automatic 3D Reconstruction from Spacetime Stereo data. We merge 18 views covering a 360° field of view of one of the smooth objects used in Experiment 3 and 29 views of an object not use in the previous experi-

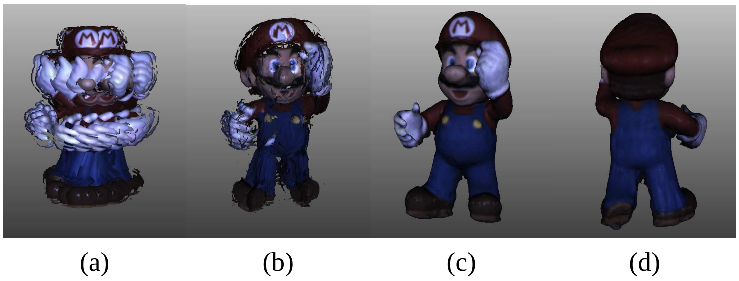


Figure 4.9: 3D Reconstruction from Spacetime Stereo views: (a) initial set of views (b) coarse registration (c) global registration frontal view (d) global registration rear view.

ments. We follow a 2 steps procedure:

1. we obtain a coarse registration by estimating the 3D transformations between every pair of views and retaining only those maximizing the global area of overlap;

2. we use the coarse registration as initial guess for a final global registration carried out using a standard external tool (*Scanalyze*).

In the first step, correspondences among views are established by computing and matching SHOT descriptors on 1000 randomly selected feature points. 3D transformations are estimated by applying a well known Absolute Orientation algorithm [36] on such correspondences and filtering outliers by RANSAC. Maximization of the area of overlap is achieved through the Maximum Spanning Tree approach described in [66]. As shown in Fig. 4.9 and Fig. 4.10, without any assumptions about the initial poses, SHOT correspondences allows for attaining a coarse alignment which is an accurate enough initial guess to successfully reconstruct the 3D shape of the object without any manual intervention. To the best of our knowledge, fully automatic 3D reconstruction from multiple Spacetime Stereo views has not been demonstrated yet.

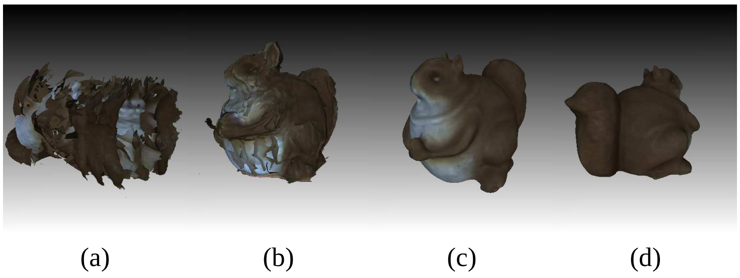


Figure 4.10: 3D Reconstruction from Spacetime Stereo views: (a) initial set of views (b) coarse registration (c) global registration frontal view (d) global registration rear view.

4.2 Color SHOT

In this section we show that the design of the SHOT descriptor can naturally and successfully be generalized to incorporate texture (Sec. 4.2.1) and that such an extension allows for improved performances on publicly available datasets (Sec. 4.2.2). This results in a particularly interesting approach for carrying out surface matching tasks based on the output of modern 3D sensors capable of delivering both shape and texture.

The majority of the proposals introduced in Sec 4.1.1 detect and describe a feature point by using shape data only. Recently, [106] has proposed the MeshDoG/HoG approach, which is the only 3D descriptor where texture information are taken into account. We will compare the performance of the generalized SHOT descriptor against this method.

4.2.1 A combined texture-shape 3D descriptor

To generalize the design of the SHOT descriptor so as to include multiple cues, we denote here as $SH_{G,f}(P)$ the generic signature of histograms computed over the spherical support around feature point P. This signature of histograms relies upon two different entities: G, a vector-valued point-wise property of a vertex, and f, the metric used to compare two of such point-wise properties. To compute a histogram of the signature, f is applied over all pairs (G_P, G_Q), with Q representing a generic vertex

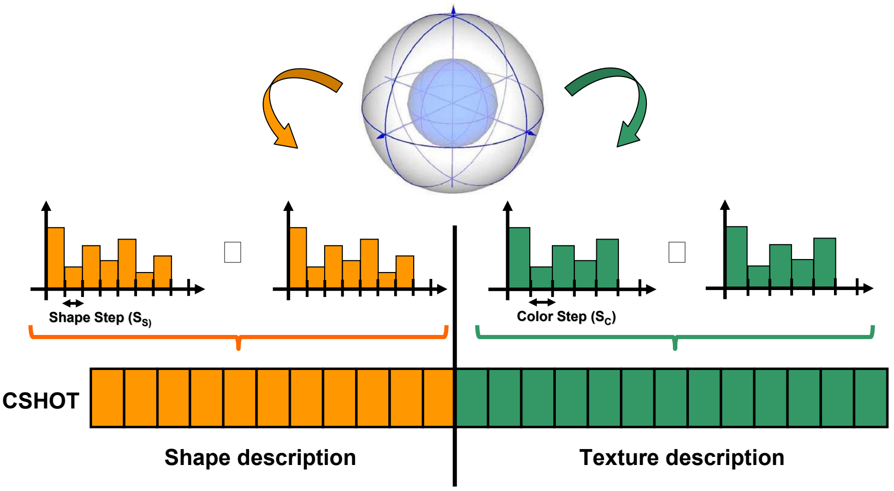


Figure 4.11: The proposed descriptor merges together a signature of histograms of normal orientations and of texture-based measurements.

belonging to the spherical support around feature point P. In the original SHOT formulation, G is the surface normal estimation, N, while $f(\cdot)$ is the dot product, denoted as $p(\cdot)$:

$$f(G_P, G_Q) = p(N_P, N_Q) = N_P \cdot N_Q \tag{4.6}$$

In the proposed generalization, m signatures of histograms relative to different *(property, metric)* pairs are computed on the spherical support and chained together in order to build the descriptor $D(P)$ for feature point P:

$$D(P) = \bigcup_{i=1}^{m} SH^{i}_{(G,f)}(P) \tag{4.7}$$

Although the formulation in (4.7) is general, we will hereinafter refer to the specific case of $m = 2$, so as to combine a signature of histograms of shape-related measurements together with a signature of texture-related measurements (Fig. 4.11). As for the former, we use the formulation of the original SHOT descriptor, *i.e.* vector H_P is represented by the surface normal estimation in P, N_P, while the operator $f()$ is the dot product, $p()$, as in (4.6). As for the latter, since we want here to embed texture information into the descriptor, we have to define a proper vector representing a point-wise property of the texture at each vertex and a suitable metric to compare two such texture-related properties. The overall descriptor, based on two signatures of histograms, will be dubbed hereinafter as Color-SHOT (CSHOT).

The most intuitive choice for a texture-based G vector is the RGB triplet of intensities associated to each vertex, referred to here as R. To properly compare RGB triplets, one option is to deploy the same metric as in SHOT, i.e. to use the dot product $p(R_P, R_Q)$. Alternatively, we have tested another possible metric based on the L_p norm between two triplets. In particular, we have implemented the operator based on the L_1 norm, referred to as $l(\cdot)$, which consists in the sum of the absolute differences between the triplets:

$$l\left(R_P, R_Q\right) = \sum_{i=1}^{3} \left|R_P(i) - R_Q(i)\right| \tag{4.8}$$

Moreover, we have investigated the possibility of using different color spaces rather than RGB. We have chosen the *CIELab* space given its well-known property of being more perceptually uniform than the RGB space[25]. Hence, as a different solution, vector G is represented by color triplets computed in this space, which will be referred to as C. Comparison between C triplets can be done using the metrics used for R triplets, i.e. the dot product $p(\cdot)$ or the L_1 norm $l(\cdot)$, leading to signatures of histograms relying, respectively, on $p\left(C_P, C_Q\right)$ and $l\left(C_P, C_Q\right)$.

In addition, we have investigated on the use of more specific metrics defined for the *CIELab* color space. In particular, we have deployed two metrics, known as *CIE94* and *CIE2000*, that were defined by the *CIE* Commission respectively in 1994 and 2000: for their definitions the reader is referred to [25]. These two metrics lead to two versions of operator $f(\cdot)$ which will be referred to, respectively, as $c_{94}(\cdot)$ and

$c_{00}(\cdot)$. Hence, two additional signatures of histograms can be defined based on these two measures, denoted respectively as $c_{94}\left(C_P, C_Q\right)$ and $c_{00}\left(C_P, C_Q\right)$.

The CSHOT descriptor inherits SHOT parameters, i.e. the radius of the support and the number of bins in each histogram). However, given the different nature of the two signatures of histograms embedded in CSHOT, it is useful to allow for a different number of bins in the two histogram types. Thus, the CSHOT descriptor will have an additional parameter with respect to SHOT, indicating the number of bins in each texture histogram and referred to as Color Step (S_C, see Fig. 4.11).

4.2.2 Experimental Results

The 6 different versions defined in Section 4.2.1 for the novel CSHOT descriptor are now evaluated in a typical 3D object recognition scenario where one or more objects have to be found in a scene with clutter and occlusions. The experimental evaluation is aimed at determining which version performs best in terms of both accuracy and efficiency. Furthermore, the best versions will be compared against the original SHOT descriptor as well as the MeshHoG descriptor, so as to evaluate the benefits brought in by the proposed approach.

In all experiments, features points are first extracted from a scene and an object, then they are described and matched based on the Euclidean distance between descriptors. As for the feature extraction stage, we rely on the same approach as in Sec. 4.1.5, *i.e.* features are first randomly extracted from the object, then the corresponding features are extracted from the scene by means of available ground-truth information together with a set of additional features randomly extracted from clutter. All algorithms have been tested by keeping constant their parameters. In particular, all parameters that CSHOT shares with SHOT have been set the values introduced in Sec. 4.1.4. Such values have been also used here for the tests concerning the SHOT descriptor. As for the additional parameter used by CSHOT (S_C), it has been tuned for each CSHOT

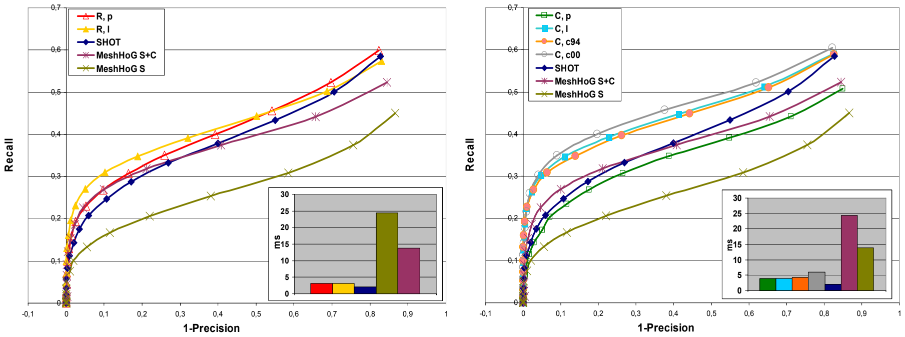


Figure 4.12: Comparison in terms of accuracy (big chart) and efficiency (small chart) between CSHOTs with different measures in the *RGB* (left chart) and *CIELab* (right chart) color spaces on *Dataset 1*. SHOT and two variants of MeshHoG are also reported.

version on a subset, made out of 3 scenes, of the *Spacetime Stereo* dataset introduced in Sec. 4.1.5. This subset has been used to tune also the radius and number of bins of the orientation histograms of MeshHoG, with the other parameters of the method kept as originally proposed in [106].

Comparison between color spaces and metrics

A first experimental evaluation has been carried out to identify the best CSHOT combinations for, respectively, the *RGB* and the *CIELab* color spaces. Results have been computed on a dataset composed of the 12 scenes not used for tuning of the *Spacetime Stereo* dataset. This subset, hereinafter referred to as *Dataset 1*, includes scenes with clutter and occlusions of the objects to be recognized.

Figure 4.12 shows the comparison between the evaluated measures respectively in the RGB (left chart) and CIELab (right chart) color spaces. As for the former, the two *(property, metric)* pairs being compared are: (R, p) and (R, l). As for the latter, four pairs are compared, i.e. : (C, p), (C, l), (C, c_{94}), (C, c_{00}). Each comparison is carried out in terms of accuracy (big chart) and efficiency (small chart). As for the former, results are provided in terms of *Precision vs. Recall* curves computed on the output of the descriptor matching process carried out between the fea-

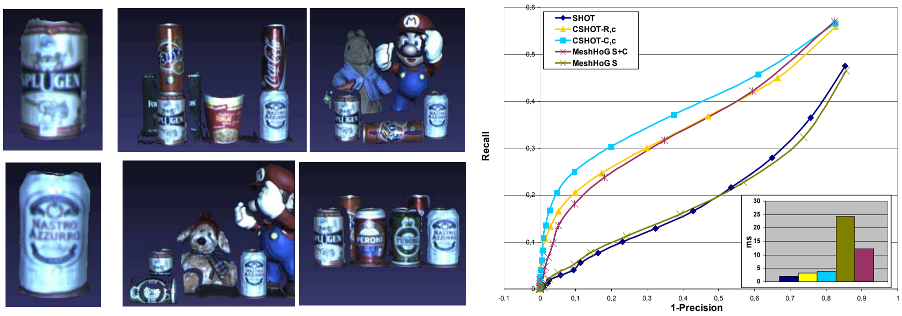


Figure 4.13: Left: Two models and four scenes of *Dataset 2*. Right: Comparison in terms of accuracy (big chart) and efficiency (small chart) between the 2 best versions of CSHOT, SHOT and two variants of MeshHoG on *Dataset 2*.

tures extracted from the objects and those extracted from the scenes. Each object-scene pair of the dataset is then averaged to give out the final charts shown in the figure. As for efficiency, results are provided as the average amount of time (*ms*) needed to compute one correspondence between the scene and the object.

As for the *RGB* space, (R, l) proves to be more accurate than (R, p), and only slightly less efficient. As for the *CIELab* space, (C, l), (C, c_{94}) and (C, c_{00}) notably outperform (C, p), with (C, l) being slightly more accurate and more efficient than (C, c_{94}), and with (C, c_{00}) being by far the least efficient one. Hence, the two CSHOT versions that turn out more favorable in terms of the accuracy-efficiency trade-off are, respectively, (R, l) for the *RGB* space, and (C, l) for the *CIELab* space.

Comparison with SHOT and MeshHoG

We will now comment on the comparison between the two best CSHOT versions and the SHOT and MeshHoG descriptors, so as to assess the benefits brought in by the combined deployment of texture and shape in the proposed extension as well as to compare its overall performance with respect to state-of-the-art methods. We tested two versions of MeshHoG: one using only shape, as done by SHOT, and one deploying shape and texture. For shape-only MeshHoG, we used the mean curvature as feature. As reported in the experimental results section of [106] (Sec 6.1), the use of both shape and texture can be achieved by juxtaposing two MeshHoG descriptors, computed respectively using as feature the mean curvature and the color. Conversely to what reported in [106], on our dataset the shape-and-texture version of MeshHoG provides slightly better performance than the texture-only version: thus, it is the one included in our comparison.

The two charts in Fig. 4.12 include the results yielded on *Dataset 1* by SHOT and the two considered variants of MeshHoG . In addition, Fig. 4.13 reports a further comparison carried out between the same proposals on another dataset. This dataset, referred to here as *Dataset 2*, comprises 8 models and 16 scenes(2 models and 4 scenes of this dataset are shown on the left side of the Figure). *Dataset 2* differs from *Dataset 1* because the former includes objects having very similar shapes but different textures (i.e. different types of cans). Hence, it helps highlighting the importance of relying also on texture for the goal of 3D object recognition in cluttered scenes. Similarly to the previous experiment, results are given both in terms of accuracy (big chart) and efficiency (small chart).

Several observations can be made on these charts. First of all, on both dataset, the two best versions of CSHOT, i.e. (R, l) and (C, l) , notably outperform SHOT and the shape-only version of MeshHoG in terms of accuracy, with the gap in performance being more evident on *Dataset 2*, where the algorithms that rely only on shape fail since they do not hold enough discriminative power to cope with the traits of the dataset. The results on both datasets confirm the benefits of including texture information in the descriptor. Secondly, on both datasets the CSHOT descriptor based on (C, l) proves to be more effective than that relying on (R, l) as well as than the shape and texture version of MeshHoG, thus allowing for state-of-the-art performance on the considered datasets. Finally, as for efficiency, the CSHOT descriptor based on (C, l) is approximately twice as slow as SHOT and one order of magnitude faster than

MeshHoG.

4.3 Object Category Recognition with 3D ISM

In the last decade the main effort on recognition of object categories has been devoted to categorizing classes of objects from images [73], one of the most prominent approaches being the application to image features of the Bag-of-Words paradigm, previously used for text categorization and document analysis. In particular, this approach, typically referred to as *Bag-of-Features* (BoF) or *Bag-of-Visual-Words* (BoVW), represents image categories as histograms (”bags”) of feature descriptors [19, 82, 84]. To account for efficiency, histograms are not built on descriptors themselves but on an alphabet of descriptors, typically termed ”codebook”, obtained via clustering or vector quantization [73].

BoF methods turned out to be particularly effective even though, unlike some more recent proposals, they discard geometrical relationships between object parts. Among those leveraging geometric structure, one of the most successful proposals is Implicit Shape Model (ISM) [50], that encodes spatial relationships by means of a probabilistic Generalized Hough Transform in a 3-dimensional space representing scale and translation. Moreover, the use of geometrically well-localized information allows these methods to be deployed also as detectors of specific object categories in presence of clutter, occlusion and multiple object instances. Typical object categories of interest have been pedestrians, faces, humans, cars [50].

The increasing availability of large databases of 3D models has fostered a growing interest towards computer vision and machine learning techniques capable of processing 3D point clouds and meshes. One of the most investigated tasks so far has been shape retrieval (see [39, 94] for surveys) which aims at finding the most similar 3D models in the database to a given query model inputted by the user. Another well investigated topic concerns 3D object recognition [27, 41]. Only very recently the first methods aimed at 3D object categorization have been proposed in literature. They mainly extend the BoF paradigm to the 3D scenario by representing categories as histograms of codewords obtained from local shape descriptions of 3D features [52, 67, 97].

In this last part of our work on 3D data we investigate on how to deploy Implicit Shape Modeling for the categorization of meshes. Although in the reminder of this paper we will focus only on categorization, it is worth noting that this approach holds the potential to solve within the same framework the problem of simultaneous localization and classification of objects in cluttered scenes, even in presence of multiple instances, *i.e.* to be uses as a category detector able to initialize a tracker.

4.3.1 3D Implicit Shape Model

The basic idea idea underlying Implicit Shape Models is to perform object category recognition and instances localization based on a non-parametric probability mass function of the position of the object center. These probability functions come from a probabilistic interpretation of the voting space of a Generalized Hough Transform algorithm. Votes are casted by local features that are matched against a codebook learned, together with votes, from a set of training examples. When applied to 3D data, we identify the general form of an algorithm training a 3D ISM as follows (Fig. 4.14):

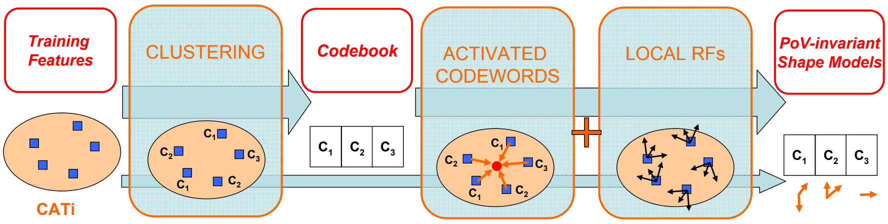


Figure 4.14: Overview of the training stage of 3D ISM.

- local features are detected and described from the 3D training data.
- for each category C_i

  - all features belonging to C_i are clustered to create the codebook of C_i
  - for each training feature $f_j^{C_i}$ of category C_i
    - $f_j^{C_i}$ is matched against the codebook of C_i according to a *codeword activation strategy*.
    - each activated codeword adds to the ISM of C_i the position of $f_j^{C_i}$ with respect to the object center. Each feature $f_j^{C_i}$ needs to incorporate a repeatable local Reference Frame (RF), and votes are expressed with respect to such local RF of $f_j^{C_i}$.

Then, a generical 3D ISM recognition procedure may be decomposed in the following steps (Fig. 4.15):

- local features are extracted and described from the 3D input data.
- for each feature f_j and each category C_i
  - f_j is matched against the codebook of C_i according to a *codeword activation strategy*.
  - each activated codeword casts its set of votes for the Hough Space of C_i in its ISM.
  - votes are rotated and translated so as to be expressed in the local RF of the input features before voting, thus obtaining *Point-of-View (PoV) independent votes*. The magnitude of the vote is set according to a *vote weighting strategy*.
- in case of categorization of 3D database entries, the category yielding the global maximum among all the Hough spaces is selected as output; in case of detection in a cluttered scene, local maxima of each category above a threshold are selected as category instance hypotheses for a further verification stage and/or pose estimation.

This scheme exhibits two main differences with respect to the use of ISM for detection of object categories in 2D images. First of all, since

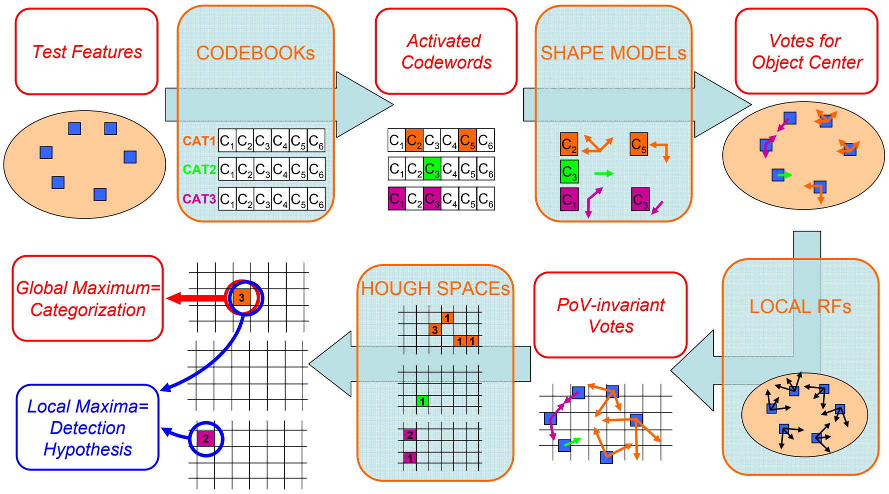


Figure 4.15: Overview of 3D ISM for Categorization and Detection.

the sensor produces metric data, there is no need for scale invariance: in the 2D case, when casting votes for the object center, the object scale is treated as a third dimension in the voting space. With 3D data we can cast votes for object hypotheses directly in the coordinates space, which is again a 3D dimensional space. The second difference regards the use of PoV-independent votes, that leads to a PoV-independent detector. In the original ISM proposal, objects of the same category under different point of views are regarded as instances of different, unrelated categories. It is worth pointing out that the use of PoV-independent votes is not just a nice extension that allows for more flexibility of the final method, it is indeed mandatory when using 3D ISM to categorizes 3D database entries, for these cannot be assumed to be expressed within the same global RF.

As noted before most of the proposals in the field of 3D local features do not include a fully defined local RF. Once more this demonstrates the importance that our SHOT descriptor defines a full 3D, unambiguous local reference frame. We thus use SHOT features as the base of our 3D ISM. This is also another test of the quality of the proposed features, which demonstrate good performance even in 3D object categorization, an experiment that was not proposed in Sec. 4.1.5.

In the previous overview of the method we have highlighted the main design decisions that need to be taken to define a 3D ISM, i.e. the codeword activation strategy and the vote weighting strategy. In the following we address, by discussion and experiments, the possible alternatives for these design choices together with other major issues related to codebook size and composition. It is worth noting that, although we have conducted experiments using 3D data only, all our reasoning is independent from data dimensionality. Therefore, we expect the observations drawn from our analysis to be beneficial also for the case of standard 2D ISMs.

4.3.2 Codebook

Codebook size

Codebooks are widely used for 2D and 3D object categorization (e.g. [85] [97] [52]). The reason behind their use is efficiency, both in terms of memory occupancy of the codebook and computational time for codeword activation. They are not expected to have any positive impact on the generalization abilities of the algorithms. They are usually built by applying some standard clustering algorithms, like *k*-means, on the features extracted from the training data. Little attention, however, has been paid to the loss in discriminative power of the codebook after size reduction. Furthermore, research in the field of Approximate Nearest Neighbor provides efficient methods to solve the codeword activation problem even in high dimensional spaces and with large databases [65]. Finally, the cost of storing a set of descriptors for each training model of the currently publicly available 3D datasets is nowadays definitely affordable by off-the-shelf machines. Based on the above considerations, we investigated on the actual importance of building a codebook to successfully perform object category recognition in 3D data.

The chart in Fig. 4.16 shows the outcome of an experiment carried out on the Aim@Shape Watertight dataset (see Sec. 4.3.5 for more details about the dataset and the experimental methodology). We used

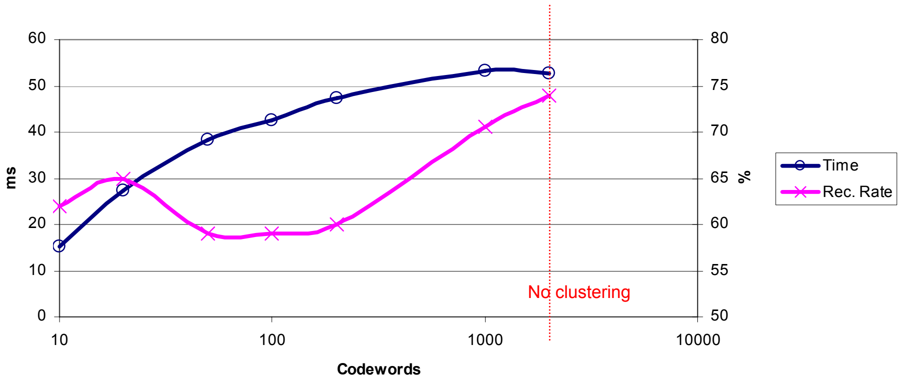


Figure 4.16: Impact of codebook size on mean recognition rate and mean recognition time

half dataset for training and half for testing, i.e. ten models for training and ten for testing for each category. 200 mesh vertexes were randomly selected on each training model obtaining 2000 features as training set for each category. We then performed *k*-means on this set, varying *k* logarithmically from 10 to 2000. We used such codebooks to categorize the test set. The best mean recognition rate is obtained with 2000 codewords, i.e. using the plain training data without any clustering. Loss in efficiency is minimal, for instance using 100 codewords the mean time to categorize one test model is about 42 ms, whereas using the plain training set as codebook it slightly increases to about 52 ms. Memory occupancy, of course, scales linearly with codebook size and, for the considered dataset, when using no clustering is less than 57MB. Therefore, based on the indication of this and other similar experiments, in the following we use as ”codebook” the whole training data, without carrying out any clustering on them.

Sharing codewords among categories

In the original ISM proposal, the case of simultaneous recognition of multiple categories is solved by running a detector for each category, endowed with its own codebook built from training data belonging to its category. We refer to this configuration as ISM with *separated code-*

books: codebooks of different categories are independently built and used. In the context of categorization of DB entries, we have investigated on another possible configuration, that we refer to here as ISM with *global codebook*: a codebook is created from the training data belonging to all categories and then used by all ISMs. The Shape Model of each category is still built during the training stage by considering only the training data belonging to that category. However, denoting with SM_i the Shape Model of category C_i, not only those originated by the training data of C_i, but all the codewords in the codebook, regardless of the categories of the features that generated them, can participate to SM_i, provided that they are similar - according to the codeword activation strategy - to any of the training features of C_i. Therefore, this scheme endows the ISM paradigm with a broader capability of generalization: whilst the separated codebooks configuration is able to generalize at an intra-class level, by letting features observed in different training instances of the same class collaborate to the detection of an instance during testing, the global codebook configuration lets ISM generalize also at an *inter-class* level. It allows features observed in training examples of different categories to reinforce the hypothesis that an instance of category C_i is present. In other words, it builds a ”universal” codebook of all the likely features given the training data, and then associates a spatial location for a specific category to all those that are ”similar” to the training features of such category, regardless of the labels of the training data that originated that codeword.

It is worth highlighting that memory requirements of both configurations are equal: although a global codebook requires C times more space than a separated codebook, with C the number of categories, only one instance of it has to be stored in memory since it can be shared among all the C 3D ISM required by our proposal. Query time scales logarithmically with the size of the codebook: since codewords in the global codebook are C times those of the separated codebooks, query time is increased by $\log C$, a limited amount for typical number of categories in publicly available 3D databases (i.e. less than 30).

4.3.3 Codeword Activation Strategy

The codeword activation strategy proposed for the deployment of ISM in the case of 2D data [50] is the *cutoff threshold*: codewords are activated, and, thus, cast their votes, if their distance from the test feature is below a threshold. An alternative approach is represented by the *k*-NN activation strategy: the closest *k* codewords to the test feature are activated, regardless of their distance. We consider the latter strategy more suitable to the task of categorization, the reason being twofold. First of all, in those parts of the feature space characterised by a high codeword density, *k*-NN activates generally less features than the cutoff strategy, only the *k* most similar ones. By increasing the number of votes casted by each test feature in the Hough space we may expect to sharpen the peak corresponding to a true instance of the class, but also to generate spurious peaks in the voting space, by randomly accumulating wrong votes in the same bin. In such parts of the feature space, the *k*-NN strategy acts as a filter that aims at reducing the probability of adding noise into the Hough space, while it hopefully retains the ability to let the correct hypothesis emerge, by selecting only the most similar codewords. Secondly, in those parts of the feature space with a low density or even absence of codewords, *k*-NN still activates *k* codewords, whereas the cutoff strategy cast very few votes, if any. Indeed, being the threshold generally chosen as small as to prevent generation of false peaks, the cutoff strategy generally tends not to activate any codeword in low density regions of the feature space. Obviously, the codewords activated by the *k*-NN strategy can be really different from the test data. Still, given the training set, they are the most similar at hand: if we have to generalize from the training examples to attempt to classify the current input, they appear a reasonable choice. The same reasoning does not hold when using 3D ISM to detect instances in cluttered scenes: in such a case, a high distance from any codeword is likely to indicate that the test feature comes from clutter and hence should not cast votes, such behavior being correctly modeled by the cutoff strategy. Yet, when reasoning in absence of clutter, as it is the case of categorization of entries of a 3D database, the k-NN strategy offers an adaptive behavior with respect to the training data that seems more suitable to the task.

4.3.4 Votes Weighting Strategy

In [50], the vote weight for each pair (test feature, vector in the shape model) is given by the product of a match weight and an occurrence weight

$$w = p\left(o_n, x|C_i, l\right) * p\left(C_i^* | f_k\right) = \frac{1}{|M|} * \frac{1}{|Occ\left[i\right]|} \tag{4.9}$$

with M being the set of codewords activated by the test feature f_k and $Occ[i]$ being the set of vectors in the Shape Model associated with codeword i.

The rationale behind this choice is tightly coupled with the use of the original ISM for detection in cluttered scenes. In presence of clutter, there is an obvious trade off between increasing the number of true detections and limiting the number of false detections. The choice of the vote weighting strategy operated in [50] goes in this direction. If a feature activates more codewords than another feature and/or if such codewords can be observed in more feasible positions with respect to the object center than other codewords, then this feature will be regarded as less distinctive since it likely generates more spurious votes in the Hough Space. By keeping low the weight, i.e. the confidence, on the position of the object center for the votes of such features, the original ISM tries to choose a good working point to optimize the above mentioned trade-off, by keeping below the detection threshold such spurious local maxima of the voting space. We refer to this vote weighting strategy as *Localization Weights (LW)*.

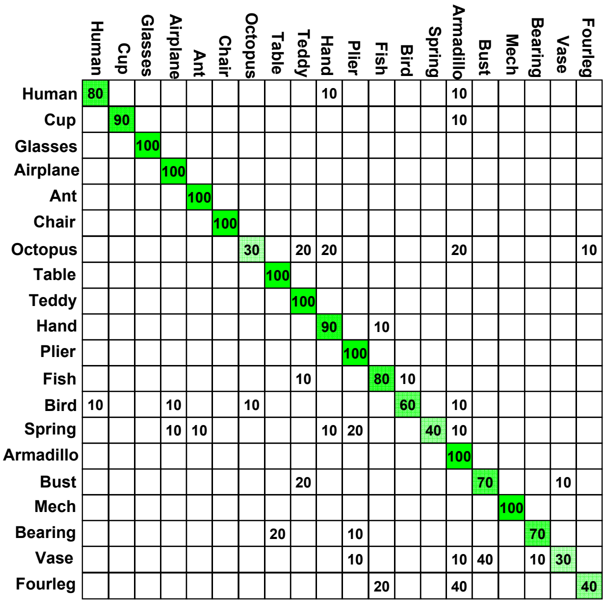
Again, in absence of clutter the scenario is different. Recall from Sec. 4.3.1 that we propose to select as output the category yielding the global maximum among all the Hough spaces. Therefore, in this case the emphasis for each 3D ISM should be on supporting as much as possible its best hypothesis. This means that spurious local maxima are not relevant for categorization, as long as they do not hide the true global maximum. Since we can reasonably expect that the geometrically consistent bin will likely provide the strongest peak in the voting space, there is no reason to try to weaken local maxima by acting on the vote weight. On the other hand, using the original ISM vote weighting strategy may uselessly reduce the strength of the global maximum only because features that casted vote for it have also casted votes for wrong locations, and this can lead to a wrong selection of the correct category in the final competition among each global maximum of all categories. Hence, in the case of categorization, we have investigated on the use of the same constant weight for all features and codewords. Hereinafter, we will denote this vote weighting strategy as *Categorization Weights (CW)*.

4.3.5 Experimental Results

We have tested our proposals on the Aim@Shape Watertight (ASW) dataset, previously used for the evaluation of 3D object categorization algorithms such as [97], and on the Princeton Shape Benchmark (PSB) [83], already used for 3D categorization in [52]. Since meshes in the PSB dataset exhibit a high variance in metric dimensions, even within the same class, to define a Hough Space suitable for all meshes, we normalize models before using them for testing or training. Specifically, we translate the model barycenter into the origin, compute the Eigenvalue Decomposition (EVD) of the scatter matrix of each model to find its principal axes, we scale the model down or up by a scale factor given by $1/X_{max} - X_{min}$, with X_{max},X_{min} the maximum and minimum coordinates of the mesh along the first principal axis, and finally rotate the model to align it with its principal axes. It is important to note that, due to the sign ambiguity inherent to the EVD [9], we still need PoV-independent votes to achieve correct categorization. This normalization allows also for an important simplification: we can define the Hough Space just around the barycenter, *i.e.* the origin: any hypothesis for the object center laying far away from the barycenter will clearly be a spurious peak in the voting space. This improves the effectiveness of our method, by discarding peaks in the a priori wrong regions of the voting space, and the its efficiency, since it reduces the memory footprint needed to store the Hough Space. In particular, we used a Hough Space consisting of one squared bin, centered in the origin and with a side of 0.2. In all the experiments with both datasets we randomly extract 200 feature points from each training model and 1000 feature points from each testing model, and we describe them using SHOT with 16 spatial sectors (8 on the tangent plane and 2 concentric spheres) and 10 bins for the normal histograms. We diminish the number of spatial divisions, and therefore the dimensionality of the descriptor with respect to that used in the previous experimental results because clustering operates better in lower dimensionality spaces. We do not perform any multi scale description, we use just a single support radius, equal to 0.25 and 0.45 for the AWS and the PSB dataset, respectively. As discussed in section 4.3.2, we use a plain codebook composed by all training descriptors.

The Aim@Shape Watertight dataset contains 20 categories, each including 20 models. We tested our performance on this dataset according to two methodologies. First, we divided the dataset in a training and a testing set by taking the first 10 models of each category as training set and the rest as testing set. With this configuration we studied the influence of the previously discussed design issues. Then, we also performed Leave-One-Out cross validation as done in [97], to be able to compare our results with such related work. Of course, the first test is more challenging, since significantly less training data is available to learn category shapes.

Results for the first series of experiments are reported in Fig. 4.18. We compared the performance of all the combinations of the proposed design decisions, i.e. global codebook (GC) vs. separated codebooks (SC), LW vs. CW and *k*-NN vs. cutoff with different values. The best recognition rate for this dataset is 79% and is obtained using 1-NN as Codeword Activation Strategy and a global codebook. In such configu-



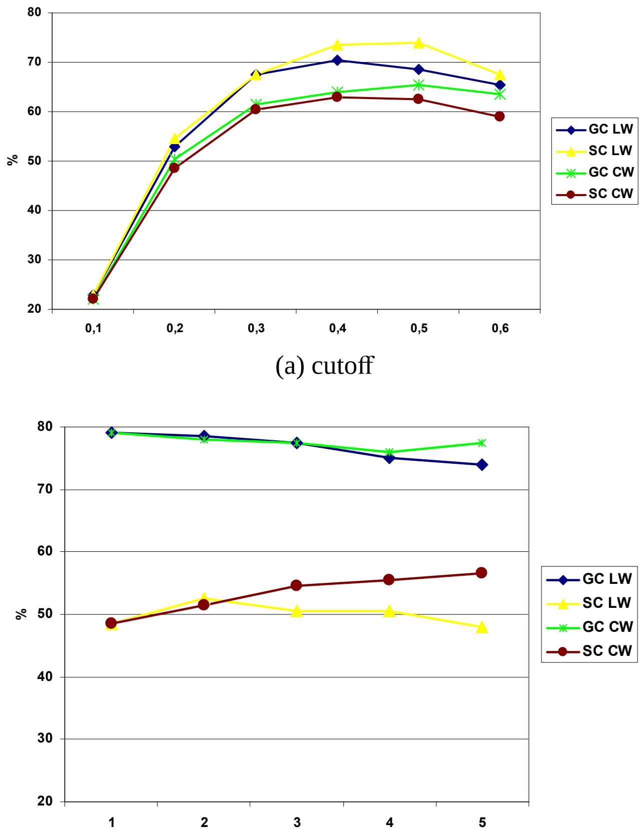
	Human	Cup	Glasses	Airplane	Ant	Chair	Octopus	Table	Teddy	Hand	Plier	Fish	Bird	Spring	Armadillo	Bust	Mech	Bearing	Vase	Fourleg
Human	80									10					10					
Cup		90													10					
Glasses			100																	
Airplane				100																
Ant					100															
Chair						100														
Octopus							30		20	20					20					10
Table								100												
Teddy									100											
Hand										90		10								
Plier											100									
Fish									10			80	10							
Bird	10			10			10						60		10					
Spring				10	10					10	20			40	10					
Armadillo															100					
Bust									20							70			10	
Mech																	100			
Bearing								20			10							70		
Vase											10				10	40		10	30	
Fourleg												20			40					40

Figure 4.17: Confusion Matrix for Aim@Shape Watertight, 1-NN Codeword Activation Strategy and CW Votes Weighting Strategy. The rows represent the test categories of the input model, the columns the output of the 3D ISM.

ration LW is the same as CW, since each codeword has zero or one vote. Fig. 4.17 reports the confusion matrix for such case.

In the case of the Leave-One-Out cross validation, [97] reports a mean recognition rate of 87.25%. Using 2-NN as Codeword Activation Strategy, a global codebook and CW as Votes Weighting Strategy, we have obtained **100**%.

The PSB dataset comes with a hierarchical categorization and a predefined division in training and testing sets. We use such categorization and such division. To compare our results against those in [52] we use the categorization level named Coarse 2, although it defines quite abstract meta-categories, such as ”Household”, which includes electric guitars, guns as well as stairs, or ”-1”, that stands for ”all other models in the dataset”. Clearly this dataset is more challenging than ASW, the intra-class and the inter-class variability being definitely higher.

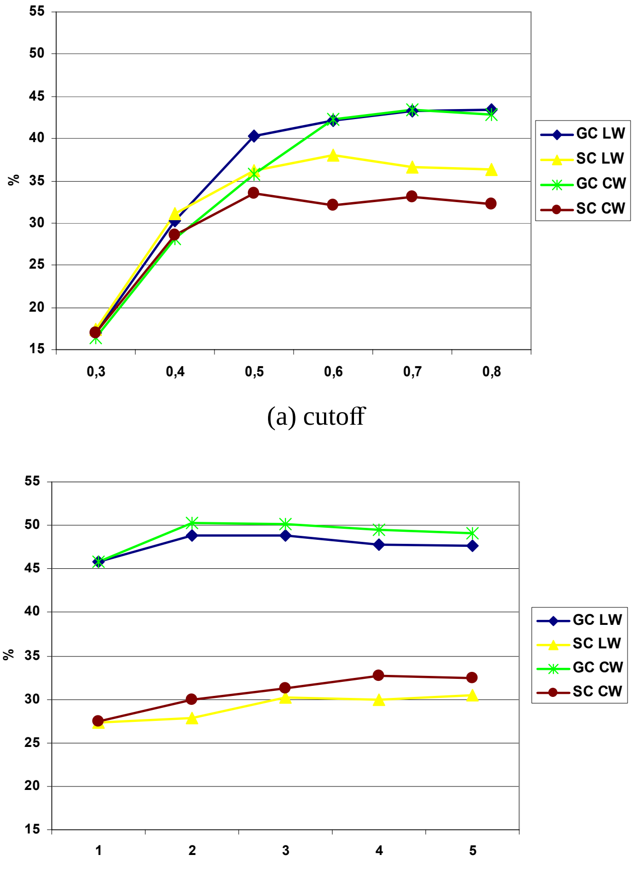


(a) cutoff

(b) k-NN

Figure 4.18: Mean recognition rate as a function of varying cutoff and k-NN values on Aim@Shape Watertight.

Results are reported in Fig. 4.19. We compared the same combinations as in the previous experiment. The best recognition rate for this dataset is 50.2% and is obtained using 2-NN as Codeword Activation Strategy, a global codebook and the CW Votes Weighting Strategy. [52] reports a mean recognition rate of 55%. It is worth noting that, in addition to the previously mentioned difficulties, the PSB dataset presents also a highly variable point density among the models. As it has been noted in the experimental comparison on the SHOT descriptor (Sec. 4.1.5), point density variation is not well tolerated by current 3D descriptors. This was explicitly accounted for in [52], where all PSB



(a) cutoff

(b) k-NN

Figure 4.19: Mean recognition rate as a function of varying cutoff and k-NN values on the PSB coarse 2 dataset.

meshes were resampled to a constant number of vertexes, uniformly distributed in the meshes. We have not implemented such resampling yet, that could likely improve our performance.

4.3.6 Discussion

The most evident outcome of our investigation is definitely the fact that the Codeword Activation Strategy and codebook composition play a significant role on the performance of 3D ISM for categorization. In both datasets *k*-NN with global codebook consistently outperforms the cutoff threshold with both kinds of codebook composition, regardless of the choice of *k*. This confirms two intuitions:

- that the intrinsic adaptation to codewords density in the feature space provided by *k*-NN is more suitable for database entries categorization, i.e. in absence of clutter, since it enhances ISM generalization ability;

- that the global codebook, when compatible with the application constraints on memory occupancy and computation time, endows ISM with higher, inter-class generalization power.

Experiments also reveal a tight coupling between the use of *k*-NN and the global codebook: *k*-NN with separated codebooks exhibits unsatisfactory performance, even with respect to the cutoff strategy. With the global codebook the *k* nearest neighbor codewords for a test feature are the same for each tested category, i.e. they represent the overall *k* most similar features throughout those belonging to all categories seen in the training stage, what then differs for the different categories is how these codewords vote in the different ISMs. In particular, it is worth pointing out that, differently from the case of separated codebooks, it happens that some of the codewords have no associated votes in the ISM of a specific category. This happens when a codeword is not similar to any training data of that category. Therefore, many of the *k* activated codewords will likely vote only for a subset of the categories, so that votes accumulation in the Hough Space has more chances to let the true category emerge, being required to filter out a limited amount of wrong votes. In other words, this configuration balances the impact of codebook (i.e. of features similarity) and shape model (i.e. of geometrical structure) and results in good recognition rates. With separated codebooks, instead, the *k* nearest neighbors are different in different codebooks, so that in several of them the activated codewords may be very dissimilar to the test feature. Moreover, since there are no codewords without votes in this configuration, all the activated codewords will cast votes in their shape models. This configuration, therefore, tends to diminish the importance of feature similarity and relies almost completely on shape models being able to select the correct category. This increases the probability of generating wrong, spurious peaks in the voting space.

The vote weighting strategy does not play a role as important as the other two design decisions. Nevertheless, as far as the *k*-NN codeword activation strategy is concerned, the Categorization Voting obtains consistently slightly better performance in both datasets and with both kind of codebooks. This provides experimental evidence to the reasoning of Sec. 4.3.4.

As for the experiments on the cutoff threshold strategy, whilst on the PSB dataset the global codebook is still the favorable option, and there is little difference between the votes weighting strategies, in the case of the ASW dataset the decisive factor for obtaining higher performance seems to be the LW strategy whereas, unlike in the *k*-NN case, the codebook options seem to have quite a minor impact. We ascribe the latter to the cutoff strategy intrisecally balancing feature similarity and geometrical structure, for dissimilar codewords, given the cutoff threshold, cannot cast votes at recognition time also when the separated codebook is used. On the other hand, it is quite more difficult to explain the higher performance of LW on this dataset. The higher performance of LW seems to suggest that in the ASW dataset wrong categories are supported in the voting space by less distinctive codewords, whose vote weights are indeed diminished by using LW.

The Confusion Matrix in Fig. 4.17 evidences how, beside gross errors that must be ascribed to the difficulty of the task, several errors are somehow reasonable for an algorithm that tries to categorize objects based only on 3D shape only. For instance, the category ”Octopus”, for which our proposal fails to recognize the majority of test models, is confused with ”Hand”, ”Armadillo” and ”Fourleg”, i.e. with categories that present sort of ”limbs” in configurations similar to those assumed by the models in the ”Octopus” category. The 40% of ”Fourleg” test models are wrongly categorized as ”Armadillo”, which, again, in some training models appears in a Fourleg-like pose. All the wrongly assigned test models of ”Bearing” are labeled as ”Table” or ”Plier”, which have parts (the legs, the handles) that are shaped as bearings. Provided that this dataset can be successfully categorized by using only shape when enough training data can be deployed, as our 100% result in the Leave-One-Out test demonstrates, the mostly reasonable errors in the Confusion Matrix show that our proposal is able to learn a plausible, although less specific, model for the category shape in presence of less training data.

Conclusions

This dissertation has presented the research activity concerning adaptive visual tracking carried out during the Ph.D. course. In particular, three main contributions related to adaptive tracking have been presented: adaptive transition models, adaptive appearance models and an adaptive Bayesian loop for tracking based on change detection in case of static cameras. Moreover, our work on category detection in 3D data has been presented.

As far as adaptive transition models are concerned, a new approach to build an adaptive recursive Bayesian estimation framework has been introduced, both from a theoretical point of view and in terms of its instantiation in the case of linear transition and measurement models and Gaussian noise. The proposed SVK filter has been shown to outperform a standard Kalman solution, requiring less parameters to be arbitrarily (and possibly wrongly) tuned. In the linear and Gaussian scenario, an interesting future investigation concerns the evaluation of the proposed approach against comparable solutions for adaptive Kalman filtering (*i.e.* Covariance Matching Techniques and [109]).

We also see this work, as all the contributions of this thesis, as a step toward a general and parameters free tracking system. Endowing this vision, another interesting future work will be directed to the insertion of algorithms for automatic on-line selection of SVR parameters. Finally, the instantiation of our proposal also in the case of non linear and non Gaussian tracking, in particular by modifying it in order to be beneficially used also with particle filters, would be a great contribution to foster its applicability and adoption.

As far as adaptive appearance models are concerned, our contribution has been twofold: we presented a critical review and classification of the most significant, recently proposed algorithms that deal with model adaptation; we casted the problem of model update as a Recursive Bayesian Estimation problem. Preliminary experimental results, where our proposal was compared on challenging sequences against many state of the art trackers, both adaptive and non adaptive, are encouraging. The main extension for our proposal would be to define a proper method to compare different features, in order to use the particle filter framework to perform also on-line probabilistic feature selection. Moreover, the proposed importance density and observation likelihoods are just one possible instantiation of this novel framework. They can be modified and extended in several ways: to make them more robust to tracker misalignments, by exploiting the full posterior PDF on the state instead of the current estimation only; to make them more robust to occlusions by deploying more stable schemes than the sliding window and consequently modifying the PDFs evaluation; to make them fully compliant with the particle filtering framework, by not fully relying on the current frame during the proposal density sampling and, hence, allowing for a proper observation likelihood to be defined.

An adaptive Bayesian loop for tracking based on change detection in case of static cameras has been proposed. On-line training of a binary Bayesian classifier based on background-frame pairs of intensities has been proposed to perform change detection robustly and efficiently in presence of common sources of disturbance such as illumination changes, camera gain and exposure variations. The ability of such algorithm to learn a model of admissible intensity variations frame by frame allows it to obtain high sensitivity without sacrificing specificity. Importantly, this promising trade-off is yielded without penalizing efficiency. Based on this novel change detection algorithm, a principled framework to model the interaction between Bayesian change detection and tracking have been presented. By modeling the interaction as marginalization of the joint probability of the tracker state and the change mask, it is possible to obtain analytical expressions for the PDFs of the tracker observation likelihood and the change detector prior. Benefits of such interaction have been discussed with experiments on publicly available datasets targeting visual surveillance and automatic analysis of sport events, where the proposed method outperformed two standard solutions for visual tracking. Several interesting extensions are possible: adapt the probabilistic reasoning on change maps to the case of particle filters; extend the proposed Bayesian algorithm to color-based change detection; take into account in the loop the number and the position of multiple targets and also their appearance, in the spirit of BraMBLe [38] but without requiring a foreground model; experiment with multiple sources of measurements, such as color histograms, providing for them, too, a fully specified observation likelihood.

As for the categorization of 3D data, our proposal encompasses the deployment of Implicit Shape Models in combination with a novel proposal for 3D description, dubbed SHOT. We have devised the general structure of a 3D ISM and identified and discussed three design decisions that could improve the performance of the method when used for categorization. Experimental results on two well known and large datasets demonstrate that the combination of the *k*-NN codeword activation strategy and the use of a global codebook built from the training data of all categories is more effective for categorization than the standard ISM approach. Votes weighting strategy, on the other hand, does not seem to play such an important role for overall performance. The proposed optimal configuration compares favorably with the state of the art in 3D data categorization, obtaining similar results in one case and outperforming current proposals on the other dataset.

We have tested also the SHOT descriptor on its own. The results validate the intuition that the synergy between the design of a repeatable local RF and the embedding of an hybrid signature/histogram nature into a descriptor allows for achieving at the same time state-of-the-art robustness and descriptiveness. Remarkably, our proposal delivers such notable performances with high computational efficiency.

Starting from SHOT, we have presented a general formulation for multi-cue description of 3D data by signatures of histograms. We have then proposed a specific implementation of this formulation, CSHOT, that realizes a joint texture-shape 3D feature descriptor. CSHOT has been shown to improve the accuracy of SHOT and to obtain state-of-the-art performance on data comprising both shape and texture. By means of experimental evaluation, different combinations of metrics and color spaces have been tested: the L_1 norm in the *CIELab* color space turns out to be the most effective choices.

As for future work, the obvious next step is to deploy 3D ISM to detect category instances in 3D data and initialize a tracker. 3D ISM may be used also to continuously guide a tracker in a tracking-by-detection approach. As for the SHOT descriptor, we plan to investigate on how to improve robustness to point density variations. Comparing our proposal with other relevant methods and on larger datasets is another important prosecution for this work.

Bibliography

[1] Adam, A., E. Rivlin, and I. Shimshoni (2006). Robust Fragments-based Tracking using the Integral Histogram. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) - Volume 1*, pp. 798–805. IEEE Computer Society Washington, DC, USA.

[2] Akagunduz, E. and I. Ulusoy (2007). 3D object representation using transform and scale invariant 3D features. In *Proc. of the International Conference on Computer Vision (ICCV),* pp. 1–8. IEEE Computer Society, Washington, DC, USA.

[3] Arulampalam, S., S. Maskell, N. Gordon, and T. Clapp (2001). A tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing 50,* 174–188.

[4] Avidan, S. (2005). Ensemble tracking. In *International Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 494–501.

[5] Babenko, B., M.-H. Yang, and S. Belongie (2009). Visual tracking with online multiple instance learning. In *Proc. of the International Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 983–990. IEEE Computer Society Washington, DC, USA.

[6] Bay, H., A. Ess, T. Tuytelaars, and L. J. V. Gool (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding 110*(3), 346–359.

[7] Blum, A. and T. Mitchell (1998). Combining labeled and unlabeled data with co-training. In *Proc. of the eleventh annual conference on Computational learning theory (COLT),* pp. 92–100. ACM New York, NY, USA.

[8] Breitenstein, M. D., F. Reichlin, B. Leibe, E. Koller-Meier, and L. van Gool (2009). Robust tracking-by-detection using a detector confidence particle filter. In *Proc. of the International Conference on Computer Vision (ICCV),* pp. 1515–1522. IEEE Computer Society Washington, DC, USA.

[9] Bro, R., E. Acar, and T. Kolda (2008). Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics 22*, 135–140.

[10] Calonder, M., V. Lepetit, C. Strecha, and P. Fua (2010). Brief: Binary robust independent elementary features. In *Proc. of the European Conference on Computer Vision (ECCV) - Part IV*, Heraklion, Greece, pp. 778–792. Springer-Verlag, Berlin, Heidelberg.

[11] Cao, L. and Q. Gu (2002). Dynamic Support Vector Machines for non-stationary time series forecasting. *Intelligent Data Analysis 6*, 67–83.

[12] Chen, H. and B. Bhanu (2007). 3d free-form object recognition in range images using local surface patches. *International Journal of Pattern Recognition and Artificial Intelligence 28*(10), 1252–1262.

[13] Chu, W., S. Keerthi, and C. J. Ong (2004, Jan.). Bayesian Support Vector Regression using a unified loss function. *IEEE Transactions on Neural Networks 15*(1), 29–44.

[14] Chua, C. S. and R. Jarvis (1997). Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision (IJCV) 25*(1), 63–85.

[15] Collins, R. T., A. J. Lipton, and T. Kanade (1999). A system for video surveillance and monitoring. Technical report, Robotics Institute at Carnegie Mellon University.

[16] Collins, R. T., Y. Liu, and M. Leordeanu (2005, oct). Online selection of discriminative tracking features. *IEEE transactions on pattern analysis and machine intelligence 27*(10), 1631–43.

[17] Comaniciu, D., V. Ramesh, and P. Meer (2003). Kernel-based object tracking. *IEEE Transactions Pattern Analysis and Machine Intelligence (PAMI) 25*(5), 564–575.

[18] Conde, C., L. Rodríguez-Aragón, and E. Cabello (2006). Automatic 3d face feature points extraction with spin images. *International Conference on Image Analysis and Recognition (ICIAR) 4142*, 317–328.

[19] Csurka, G., C. Bray, C. R. Dance, and L. Fan (2004). Visual categorization with bags of keypoints. In *Proc. of. European Conference of Computer Vision - Workshop on Statistical Learning in Computer Vision (ECCV)*, Lecture Notes In Computer Science (LNCS), pp. 1–22. Springer-Verlag, London.

[20] Dalal, N. and B. Triggs (2005). Histograms of oriented gradients for human detection. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 886–893. IEEE Computer Society Washington, DC, USA.

[21] Davis, J., D. Nehab, R. Ramamoothi, and S. Rusinkiewicz (2005). Spacetime stereo : A unifying framework for depth from triangulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 27*(2), 1615–1630.

[22] D'Orazio, T., M. Leo, N. Mosca, P. Spagnolo, and P. L. Mazzeo (2009). A semi-automatic system for ground truth generation of soccer video sequences. In *Proc. of Sixth International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 559–564. IEEE Computer Society Washington, DC, USA.

[23] Elgammal, A., D. Harwood, and L. Davis (1999). Non-parametric model for background subtraction. In *Proc. of the International Conference on Computer Vision (ICCV)*, pp. 751–767. IEEE Computer Society, Washington, DC, USA.

[24] Elhabian, S. Y., K. M. El-Sayed, and S. H. Ahmed (2008). Moving object detection in spatial domain using background removal techniques - state-of-art. *Recent Patents on Computer Sciences 1*, 32–54.

[25] Fairchild, M. (2005). *Color Appearance Models*. John Wiley & Sons Ltd.

[26] Freeman, W. T. and E. H. Adelson (1991, September). The design and use of steerable filters. *IEEE transactions on pattern analysis and machine intelligence 13*(10), 891–906.

[27] Frome, A., D. Huber, R. Kolluri, T. Bülow, and J. Malik (2004). Recognizing objects in range data using regional point descriptors. In *Proc. of the European Conference on Computer Vision (ECCV) - Volume III*, Lecture Notes In Computer Science (LNCS), pp. 224–237. Springer-Verlag, London.

[28] Gao, J., S. Gunn, C. Harris, and M. Brown (2 January 2002). A probabilistic framework for SVM regression and error bar estimation. *Machine Learning 46*, 71–89(19).

[29] Grabner, H. and H. Bischof (2006). On-line boosting and vision. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) - Volume 1*, pp. 260–267. IEEE Computer Society Washington, DC, USA.

[30] Grabner, H., C. Leistner, and H. Bischof (2008). Semi-supervised on-line boosting for robust tracking. In *Proc. of the tenth European Conference on Computer Vision (ECCV) - Part I*, Lecture Notes In Computer Science (LNCS), pp. 234–247. Springer-Verlag, Berlin, Heidelberg.

[31] Grossberg, S. (1988). *Competitive learning: from interactive activation to adaptive resonance*, pp. 243–283. Norwood, NJ, USA: Ablex Publishing Corp.

[32] Haritaoglu, I., D. Harwood, and L. S. Davis (2000). W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 22*, 809–830.

[33] Harville, M. (2002, July). A framework for high-level feedback to adaptive, per-pixel, mixture-of-Gaussian background models. In *Proc. of the seventh European Conference on Computer Vision (ECCV) - Part III*, Lecture Notes In Computer Science (LNCS), pp. 543–560. Springer-Verlag, London.

[34] Harville, M. and D. Li (2004). Fast, integrated person tracking and activity recognition with plan-view templates from a single stereo camera. In *Proc. of the computer society conference on Computer vision and pattern recognition (CVPR) - Volume 2*, pp. 398–405. IEEE Computer Society, Washington, DC, USA.

[35] Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle (1992). Surface reconstruction from unorganized points. In *Proc. of the Compute Graphics Conference (SIGGRAPH)*, pp. 71–78.

[36] Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A (JOSA A) 4*(4), 629–642.

[37] Isard, M. and A. Blake (1998). Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV) 29*(1), 5–28.

[38] Isard, M. and J. MacCormick (2001, July). Bramble: A bayesian multiple-blob tracker. In *Proc. of the International Conference on Computer Vision (ICCV) - Volume 2*, pp. 34–41. IEEE Computer Society, Washington, DC, USA.

[39] Iyer, M., S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani (2005). Three dimensional shape searching: state-of-the-art review and future trends. *Computer Aided Design 5*(15), 509–530.

[40] Jepson, A. D., D. J. Fleet, and T. F. El-Maraghi (2003). Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*(10), 1296–1311.

[41] Johnson, A. and M. Hebert (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 21*(5), 433–449.

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

[43] Ke, Y. and R. Sukthankar (2004). Pca-sift: A more distinctive representation for local image descriptors. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition Workshops(CVPR) - Volume 2*, pp. 506–513. IEEE Computer Society Washington, DC, USA.

[44] Koenderink, J. and A. Doorn (1992). Surface shape and curvature scales. *Image Vision Computing 8*, 557–565.

[45] Kwon, J. and K. M. Lee (2009). Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive Basin Hopping Monte Carlo sampling. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1208–1215. IEEE Computer Society Washington, DC, USA.

[46] Kwon, J. and K. M. Lee (2010). Visual tracking decomposition. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 1269–1276. IEEE Computer Society Washington, DC, USA.

[47] Lanza, A., L. Di Stefano, and L. Soffritti (2009). Bayesian order-consistency testing with class priors derivation for robust change detection. In *Proc. of Sixth International Conference on Advanced Video and Signal Based Surveillance (AVSS),* pp. 460–465. IEEE Computer Society Washington, DC, USA.

[48] Lanza, A. and L. D. Stefano (2006). Detecting changes in grey level sequences by ML isotonic regression. In *Proc. of the International Conference on Advanced Video and Signal-based Surveillance (AVSS),* pp. 1–4. IEEE Computer Society, Washington, DC, USA.

[49] Lee, K.-C. and D. Kriegman (2005). Online Learning of Probabilistic Appearance Manifolds for Video-Based Recognition and Tracking. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) - Vol. 2,* pp. 852–859. IEEE Computer Society Washington, DC, USA.

[50] Leibe, B., A. Leonardis, and B. Schiele (2008, May). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision 77*(1-3), 259–289.

[51] Lin, C.-J. and R. C. Weng (2004). Simple probabilistic predictions for Support Vector Regression. Technical report, Department of Computer Science, National Taiwan University,.

[52] Liu, Y., H. Zha, and H. Qin (2006). Shape topics: a compact representation and new algorithms for 3d partial shape retrieval. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) - Volume 2,* pp. 2025–2032. IEEE Computer Society Washington, DC, USA.

[53] Lou, J., H. Yang, W. Hu, and T. Tan (2002). An illumination-invariant change detection algorithm. In *Proc. of the 5th Asian Conference on Computer Vision (ACCV) - Volume 1,* pp. 13–18.

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

[55] Lu, L. and G. D. Hager (2007). A nonparametric treatment for location / segmentation based visual tracking. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8. IEEE Computer Society Washington, DC, USA.

[56] Matas, J., O. Chum, M. Urba, and T. Pajdla (2002). Robust wide baseline stereo from maximally stable extremal regions. In *Proc. of British Machine Vision Conference (BMVC)*, pp. 384–396.

[57] Matthews, I., T. Ishikawa, and S. Baker (2004, June). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*(1), 810–815.

[58] Mehra, R. (1972, Oct). Approaches to adaptive filtering. *IEEE Transactions on Automatic Control 17*(5), 693–698.

[59] Mian, A., M. Bennamoun, and R. Owens (2006). A novel representation and feature matching algorithm for automatic pairwise registration of range images. *International Journal of Computer Vision (IJCV) 66*(1), 19–40.

[60] Mian, A. S., M. Bennamoun, and R. A. Owens (2010). On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International Journal of Computer Vision (IJCV) 89*(2-3), 348–361.

[61] Mikolajczyk, K. and C. Schmid (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 27*(10), 1615–1630.

[62] Mikolajczyk, K., T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool (2005, oct). A Comparison of Affine Region Detectors. *International Journal of Computer Vision (IJCV) 65*(1-2), 43–72.

[63] Mitra, N. J., A. Nguyen, and L. Guibas (2004). Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry and Applications 14*(4–5), 261–276.

[64] Mittal, A. and V. Ramesh (2006). An intensity-augmented ordinal measure for visual correspondence. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) -*

Volume 1, pp. 849–856. IEEE Computer Society, Washington, DC, USA.

[65] Muja, M. and D. G. Lowe (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *Proc. of International Conference on Computer Vision Theory and Application (VISSAPP)*, pp. 331–340. INSTICC Press.

[66] Novatnack, J. and K. Nishino (2008). Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images. In *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 440–453. Springer-Verlag, Berlin, Heidelberg.

[67] Ohbuchi, R., K. Osada, T. Furuya, and T. Banno (2008). Salient local visual features for shape-based 3d model retrieval. In *Proc. of the Int. Conf. on Shape Modeling and Applications (SMI)*, pp. 93–102.

[68] Ohta, N. (2001). A statistical approach to background subtraction for surveillance systems. In *Proc. of the International Conference on Computer Vision (ICCV) - Volume 2*, pp. 481–486. IEEE Computer Society, Washington, DC, USA.

[69] Ojala, T., M. Pietikainen, and T. Maenpaa (2002, July). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE transactions on pattern analysis and machine intelligence 24*(7), 871–987.

[70] Oussalah, M. and J. De Schutter (2000). Adaptive Kalman filter for noise identification. In *Proc. of the 25th International Conference on Noise and Vibration Engineering (ISMA)*.

[71] Ovsjanikov, M., J. Sun, and L. Guibas (2008). Global intrinsic symmetries of shapes. *Computer Graphics Forum 5*, 1341–1348.

[72] Oza, N. C. (2001, Sep). *Online Ensemble Learning*. Ph. D. thesis, The University of California, Berkeley, CA.

[73] Pinz, A. (2005). Object categorization. *Foundation and Trends in Computer Graphics and Vision 1*(4), 255–353.

[74] Platt, J. C. (1999). Fast training of Support Vector Machines using Sequential Minimal Optimization. *Advances in kernel methods: support vector learning*, 185–208.

[75] Poggio, T., S. Mukherjee, R. Rifkin, A. Rakhlin, and A. Verri (2001). b. *CBCL Paper 198/AI Memo 2001-011*.

[76] Pontil, M., S. Murkerjee, and F. Girosi (1998). On the noise model of Support Vector Machine Regression. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

[77] Prati, A., I. Mikic, M. M. Trivedi, and R. Cucchiara (2003). Detecting moving shadows: Algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*(7), 918–923.

[78] Pérez, P., C. Hue, J. Vermaak, and M. Gangnet (2002). Proc. of the color-based probabilistic tracking. In *Proceedings of the seventh European Conference on Computer Vision (ECCV) - Part I*, Lecture Notes In Computer Science (LNCS), pp. 661–675. Springer-Verlag, London.

[79] Ristic, B., S. Arulampalam, and N. Gordon (2004). *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House.

[80] Ross, D. A., J. Lim, R.-S. Lin, and M.-H. Yang (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision (IJCV) 77*(1-3), 125–141.

[81] Schweighofer, G. and A. Pinz (2006, Dec.). Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*(12), 2024–2030.

[82] Serre, T., L. Wolf, and T. Poggio (2005). A new biologically motivated framework for robust object recognition. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society Washington, DC, USA.

[83] Shilane, P., P. Min, M. Kazhdan, and T. Funkhouser (2004). The princeton shape benchmark. In *Proc. of Shape Modeling International (SMI)*, pp. 167–178. IEEE Computer Society, Washington, DC, USA.

[84] Sivic, J., B. Russell, A. Elfros, and Z. Zisserman (2005). Discovering objects and their location in images. In *Proc. of the International Conference on Computer Vision (ICCV) - Volume 1*, pp. 370–377. IEEE Computer Society, Washington, DC, USA.

[85] Sivic, J. and A. Zisserman (2006). Video google: Efficient visual search of videos. In *Toward Category-Level Object Recognition*, Lecture notes in computer science, pp. 127–144. Springer-Verlag, Berlin, Heidelberg.

[86] Smola, A. J. and B. S. Olkopf (1998). A tutorial on Support Vector Regression. Technical report, Statistics and Computing.

[87] Somanath, G. and C. Kambhamettu (2010). Abstraction and generalization of 3d structure. In *Proc. of the Asian Conference of Computer Vision (ACCV) - Part III*, Lecture Notes in Computer Science, pp. 483–496. Springer.

[88] Song, X., J. Cui, H. Zha, and H. Zhao (2008). Vision-based multiple interacting targets tracking via on-line supervised learning. In *Proc. of the tenth European Conference on Computer Vision (ECCV) - Part III*, Lecture Notes In Computer Science (LNCS), pp. 642–655. Springer-Verlag, Berlin, Heidelberg.

[89] Stalder, S., H. Grabner, and L. van Gool (2009). Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *Proc. of the International Conference on Computer Vision (ICCV) - Workshop on On-line Learning for Computer Vision.*, pp. 1409. IEEE Computer Society Washington, DC, USA.

[90] Stauffer, C. and W. E. L. Grimson (1999). Adaptive background mixture models for real-time tracking. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) - Volume 2*, pp. 246–252. IEEE Computer Society, Washington, DC, USA.

[91] Stein, F. and G. Medioni (1992). Structural indexing: Efficient 3-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 14*(2), 125–145.

[92] Sun, Y. and M. A. Abidi (2001). Surface matching by 3d point's fingerprint. *IEEE International Conference on Computer Vision (ICCV) 2*, 263–269.

[93] Tang, F., S. Brennan, Q. Zhao, and H. Tao (2007). Co-tracking using semi-supervised support vector machines. In *Proc. of the International Conference on Computer Vision (ICCV)*, pp. 1–8. IEEE Computer Society Washington, DC, USA.

[94] Tangelder, J. W. H. and R. C. Veltkamp (2004). A survey of content based 3d shape retrieval methods. In *Proc. of the Shape Modeling International Conference (SMI)*, pp. 145–156. IEEE Computer Society Washington, DC, USA.

[95] Taycher, L., J. W. F. Iii, and T. Darrell (2005, January). Incorporating object tracking feedback into background maintenance framework. In *Proc. of the Workshop on Motion and Video Computing (WACV/MOTION) - Volume 2*, pp. 120–125. IEEE Computer Society, Washington, DC, USA.

[96] Thomas, A., V. Ferrari, B. Leibe, T. Tuytelaars, and L. van Gool (2007). Depth-from-recognition: Inferring metadata by cognitive feedback. In *Proc. of the International Conference on Computer Vision (ICCV)*, pp. 1–8. IEEE Computer Society, Washington, DC, USA.

[97] Toldo, R., U. Castellani, and A. Fusiello (2009). A bag of words approach for 3d object categorization. In *Proc. of the the forth International Conference on Computer Vision/Computer Graphics Collaboration Techniques (MIRAGE)*, pp. 116–127. Springer-Verlag, Berlin, Heidelberg.

[98] Unnikrishnan, R. and M. Hebert (2008). Multi-scale interest regions from unorganized point clouds. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition Workshops(CVPR)*, pp. 1–8. IEEE Computer Society Washington, DC, USA.

[99] Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc.

[100] Viola, P. and M. Jones (2002). Robust real-time object detection. *International Journal of Computer Vision 57*(2), 137–154.

[101] Weng, S.-K., C.-M. Kuo, and S.-K. Tu (2006). Video object tracking using adaptive Kalman filter. *Journal of Visual Communication and Image Representation 17*(6), 1190–1208.

[102] Xie, B., V. Ramesh, and T. Boult (2004, feb). Sudden illumination change detection using order consistency. *Image and Vision Computing 22*(2), 117–125.

[103] Yang, M., L. Fengjun, X. Wei, and G. Yihong (2009). Detection driven adaptive multi-cue integration for multiple human tracking. In *Proc. of the International Conference on Computer Vision (ICCV)*, pp. 1554–1561. IEEE Computer Society Washington, DC, USA.

[104] Yilmaz, A., O. Javed, and M. Shah (2006, dec). Object Tracking: A Survey. *ACM Computing Surveys 38*(4).

[105] Yu, Q., T. B. Dinh, and G. Medioni (2008). Online tracking and reacquisition using co-trained generative and discriminative trackers. In *Proc. of the European Conference on Computer Vision (ECCV) - Part II*, Lecture notes in computer sciences (LNCS), pp. 678–691. Springer-Verlag, Berlin, Heidelberg.

[106] Zaharescu, A., E. Boyer, K. Varanasi, and R. P. Horaud (2009, June). Surface feature detection and description with applications to mesh matching. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 373–380. IEEE Computer Society Washington, DC, USA.

[107] Zelniker, E. E., T. M. Hospedales, S. Gong, and T. Xiang (2009). A unified approach for adaptive multiple feature tracking for surveillance applications. In *Proc. of the British Machine Vision Conference (BMVC)*.

[108] Zhang, L., B. Curless, and S. Seitz (2003). Spacetime stereo: Shape recovery for dynamic scenes. In *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition(CVPR) - Volume 2*, pp. 367–374. IEEE Computer Society Washington, DC, USA.

[109] Zhang, Y., H. Hu, and H. Zhou (2005, June-3 July). Study on adaptive Kalman filtering algorithms in human movement tracking. In *Proc. of the IEEE International Conference on Information Acquisition (ICIA)*, pp. 11–15.

[110] Zhao, W., R. Chellappa, P. Phillips, and A. Rosenfeld (2003). Face recognition: A literature survey. *ACM Computing Survey*, 399–458.

[111] Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Proc. of the International Conference on Computer Vision Workshops (ICCV)*, pp. 689–696. IEEE Computer Society, Washington, DC, USA.

Publications related to this work

- A. Lanza, S. Salti, L. Di Stefano, *On-Line Training of a Binary Bayesian Classifier for Robust and Efficient Background Subtraction, submitted to ICIP 2011* .

- F. Tombari, S. Salti, L. Di Stefano, *A combined intensity-shape descriptor for texture-enhanced 3D feature matching, submitted to ICIP 2011* .

- S. Salti, F. Tombari, L. Di Stefano, *A Performance Evaluation of 3D Keypoint Detection*, The 1st IEEE Joint 3DIM/3DPVT Conference (3DIMPVT), Hangzhou, China, 16-19 May, 2011.

- F. De Crescenzio, M. Fantini, F. Persiani, L. Di Stefano, P. Azzari, S. Salti, *Augmented Reality for Aircraft Maintenance Training and Operations Support*, Computer Graphics and Applications, IEEE, vol. 31, no. 1, pp. 96-101, January-February 2011.

- S. Salti, F. Tombari, L. Di Stefano, *On the use of Implicit Shape Models for recognition of object categories in 3D data*, The 10th Asian Conference on Computer Vision (ACCV), Queenstown, New Zealand, 8-12 November, 2010.

- S. Salti, A. Lanza, L. Di Stefano, *Bayesian Loop for Synergistic Change Detection and Tracking*, The 10th International Workshop on Visual Surveillance (VS), Queenstown, New Zealand, 8 November, 2010.

- F. Tombari, S. Salti, L. Di Stefano, *Unique Shape Context for 3D Data Description* , ACM Int. Workshop on 3D Object Retrieval @ ACM MM 2010, Firenze, Italy, 25-29 October, 2010.

- F. Tombari, S. Salti, L. Di Stefano, *Unique Signatures of Histograms for Local Surface Description*, The 11th European Conference on Computer Vision (ECCV), Heraklion, Crete, Greece, 5-11 September, 2010.

- S. Salti, L. Di Stefano, *On-line learning of the Transition Model for Recursive Bayesian Estimation*, The 2nd International Workshop on Machine Learning for Vision-based Motion Analysis (MLVMA) @ ICCV 2009, Kyoto, Japan, October 2009.

- S. Salti, L. Di Stefano, *SVR-based jitter reduction for markerless Augmented Reality*, International Conference on Image Analysis and Processing (ICIAP), Vietri sul Mare (SL), Italy, September 2009.