A comparison of two Monte Carlo algorithms for 3D vehicle trajectory reconstruction in roundabouts [☆]

Andrea Romanoni [a,*], Lorenzo Mussone [b], Davide Rizzi [a], Matteo Matteucci [a]

[a] *Politecnico di Milano, DEIB, Via Ponzio, 34/5, Milano, 20133, Italy*
[b] *Politecnico di Milano, ABC, Via Bonardi, 9, 20133, Milano, Italy*

Visual vehicular trajectory analysis and reconstruction represent two relevant tasks both for safety and capacity concerns in road transportation. Especially in the presence of roundabouts, the perspective effects on vehicles projection on the image plane can be overcome by reconstructing their 3D positions with a 3D tracking algorithm. In this paper we compare two different Monte Carlo approaches to 3D model-based tracking: the Viterbi algorithm and the Particle Smoother. We tested the algorithms on a simulated dataset and on real data collected in one working roundabout with two different setups (single and multiple cameras). The Viterbi algorithm estimates the Maximum A-Posteriori solution from a sample-based state discretization, but, thanks to its continuous state representation, the Particle Smoother overcomes the Viterbi algorithm showing better performance and accuracy.

Keywords:Vehicular trajectory, 3D visual tracking, Monte Carlo smoothing, Model-based tracking

1. Introduction

Vehicular monitoring is one of the most relevant research topics in the intelligent transportation systems field. A system capable of estimating vehicle position and its dynamics on the road is important, for instance, to detect infractions as well as road accidents [1], and it could also provide useful information about traffic distribution [2]. In the context of vehicular monitoring, roundabouts represent a uniquely challenging scenario for their complexity, both in terms of vehicular trajectories (which are different between vehicles of the same class and very different between vehicles of different classes) and in terms of simultaneous occlusions of more vehicles, especially occurring in multi-lanes circulatory roadways and with heavy vehicles.

One approach to traffic monitoring is the visual vehicle tracking [3], i.e., the process of recognizing moving objects and estimating their trajectory from a video sequence. Most of the existing visual vehicle tracking systems propose a 2D approach (2D tracking hereafter): these systems identify moving vehicles on the image plane, e.g., by identifying their blobs (see Fig. 1) via background subtraction [4], and they track their trajectories on this plane [5,2]. Although, in some applications, this type of estimate might be sufficient to fully understand the vehicle behavior, in many cases, especially in roundabout intersections, we need to estimate vehicle trajectories with high accuracy and with respect to a 3D world reference system; the latter process is called *3D tracking*.

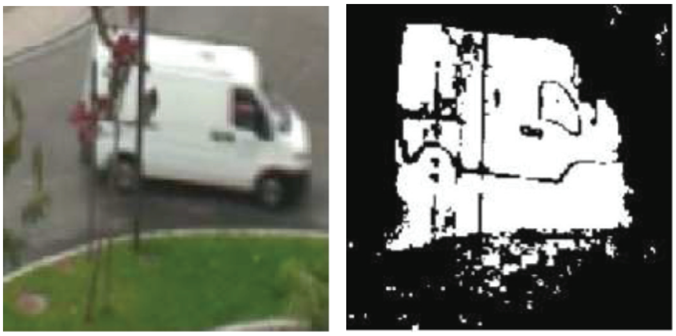

(a) *Vehicle.* (b) *Blob.*

Fig. 1. Blob extracted through background subtraction.

The straightforward approach to reconstruct a 3D trajectory projects the 2D vehicle positions — approximately the centroids of the blobs estimated with 2D tracking — from the image plane on the road plane as in Fig. 2, where C_{wrong} is the intersection of the centroid viewing ray with the road ground plane, see [6,2]. The main drawbacks of this approach are the high sensitivity to perspective deformations and the effect of the unknown height of vehicles, especially when they are heavy trucks. An example of that latter is reported in Fig. 2 where the estimated center C_{wrong} is far from the real vehicle center C.

Article history:
Received 14 March 2014
Available online xxx

[☆] This paper has been recommended for acceptance by R. Davies.
[*] Corresponding author. Tel.: +39 02 2399 4031.
E-mail address: andrea.romanoni@polimi.it (A. Romanoni).

Published Journal Article available at http://dx.doi.org/10.1016/j.patrec.2014.09.003

To overcome these issues, some researchers have proposed to use 3D model-based tracking algorithms and this is the approach we focus on in this paper. This class of algorithms gives a trajectory estimation in 3D world coordinates by representing the tracked object with one or more models, for instance in our implementation we have used a set of parallelepipeds with variable dimensions and we infer (computationally) which model should be used for the current vehicle.

The two most common approaches to 3D model-based vehicle tracking are named *edge-based* and *region-based*, according to the features used to recognize and track a vehicle. The latter, i.e., the region-based, has shown more flexibility and robustness [7] and for this reason we focus our comparison on this class of algorithms. The most suitable way to deal with region-based 3D tracking is by means of a Monte Carlo estimation [8,9], since it natively applies the concept of hypotheses scoring, very useful when comparing the 3D vehicle model back-projected on the image plane against the region occupied by the vehicle. Moreover Monte Carlo estimation does not rely on the strong Gaussian and unimodal assumptions of the common Kalman estimation.

An abridged version of this paper was presented in the conference paper [10]. Here we propose an analysis of two Monte Carlo approaches to region-based 3D tracking by comparing a Viterbi algorithm and a Particle Smoother. The former deals with a discrete representation of the state to provide the Maximum A-Posteriori (MAP) estimate, while the latter approximates the MAP solution through its sample-based distribution; at the end of our analysis we show that, thanks to its continuous state representation, the Particle Smoother gives better results and with a lower computational cost. While in [10] we have focused on the relevance of 3D tracking with respect to the 2D one in a roundabout setting, in this paper we focus on the comparison of two 3D tracking algorithms, and we discuss on the reasons why a Monte Carlo approach fits well the region-based 3D tracking.

In Section 2, we present a literature overview on 3D model-based tracking algorithms. In Section 3 we present a (Bayesian smoothing) formalization of the tracking and smoothing problem. Then, in Section 4 we describe the two Monte Carlo tracking algorithms and how they implement the Bayesian smoothing. In the same section we explain also the vehicle model management, the algorithm functioning and how the likelihood is computed in the single and multiple camera cases. In Section 5 we illustrate the experimental results for the two tracking algorithms on simulated and real scenarios, while Section 6 concludes the paper.

2. 3D model-based tracking

The visual tracking process aims at estimating the state of an object from a sequence of images. The classical computer vision tracking involves the estimation of the object position on the image plane, hence it is named 2D tracking. A lot of approaches to 2D tracking have been presented, see [11]; the most successful ones learn the appearance of the object and track it leveraging on the learned description. The main differences among the various algorithms lie on what kind of features they learn and on how they model the tracked object: for instance [12] and [13] learn the color histogram of the object; [14] learn an eigenbasis representation; [15] model the object with SIFT features; and a very recent and successful approach uses sparse coding to represent the objects, see [16].

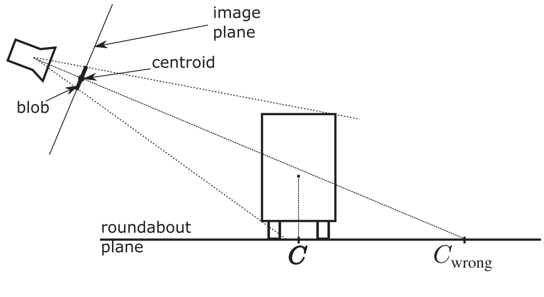


Fig. 2. 2D to 3D projection of vehicle position: the estimate C_{wrong} differs significantly from the real position C.

Another approach to object tracking is named 3D tracking and it estimates the sequence of 3D positions [7]. Especially in the vehicle tracking scenarios this approach represents a widespread method; indeed in the comprehensive review of vehicular trackers [17] most of the analyzed systems estimate the 3D position and usually by means of a 3D model. To simplify the tracking task, all studies in 3D tracking literature assume that the camera calibration is known, see [18]; then, they usually assume the Ground Plane Constraint, i.e., a vehicle always lies on the road plane, and tracking is executed on this plane in order to diminish the vehicle degrees of freedom to be estimated from 6 to 3. Most of these 3D tracking systems make use of an object model, indeed they are called model-based tracking systems. Briefly, vehicle position is estimated, frame-by-frame, by looking for the best model position and orientation which fit the object measure extracted from the images.

The authors in [7] classify the 3D tracking algorithm in: edge-based, region-based, optical flow-based and feature-based. In the vehicle tracking literature the edge-based and region-based represent the most common approaches. They both project the vehicle model from the estimated pose on the image plane, but they differ in the choice of the metrics adopted to evaluate the current estimate.

The *edge-based* algorithms compare the model projection with the image edges; starting from the current estimate, they look for the roto-translation that minimizes the distance between projected segments of the model and the image edges [19,20]. This method has the advantage of being robust to light changes and to image noise, but relevant failures may occur during the minimization step, since the algorithm often stops on local minima.

The *region-based* algorithms compare the model projection with the image region occupied by the tracked object, usually referred to as blob (see Fig. 1), typically extracted by background subtraction [4] or frame-by-frame difference [21]. To estimate the vehicle pose, some region-based algorithms minimize a metric, as for the edge-based case [22], while other algorithms calculate a convenient score for a set of hypothesized model poses [23,8]. The former approach aims at diminishing significantly the number of local minima compared to the edge-based method while the latter almost eliminates them.

Even if the edge-based approach is more robust to light changes and image noise, the region-based one is a more adequate choice for vehicle tracking: it is flexible, since it does not require an exact model of the vehicle; it is robust to the local minima issue; and it relies on background subtraction, a well known module implemented in most video surveillance systems.

Both the edge-based and the region-based approaches usually adopt the Kalman Filter [24,23,25] to perform model-based 3D tracking; but an increasing number of researchers have adopted a Monte Carlo approach [8,9] where the vehicle state is represented by a set of weighted samples.

In the region-based algorithms, the most effective way to deal with the comparison between blob and model projection is the computation of an overlap score; with the Kalman Filter this score cannot be used, and the common solution is to back-project on the road plane the blob centroid, then compare it with the Kalman state prediction. This process has two main limitations: by using just the back-projected blob centroid we neglect a lot of information coming from the blob dimension and shape, moreover we cannot directly compute the fitting of the 3D vehicle model to the measurement. Conversely, a Monte Carlo approach natively weights hypotheses with a likelihood score and this can be derived from the overlap score (see Section 4.4).

Moreover, the Kalman estimation relies on the strong assumption that the underlying process is unimodal and Gaussian, while with Monte Carlo sampling we are able to estimate multi-modal distributions, and, it may be possible to estimate the 3D trajectory of more than one vehicle at time, in a similar fashion to [8], although in this paper we use a single estimator for each vehicle.

Because of the aforementioned advantages we are mostly interested in region-based Monte Carlo approaches and in this paper we compare two estimators: the Viterbi algorithm and the Particle Smoother. These estimators are Bayesian smoothers, i.e., they make use of the entire trajectory of a single vehicle to estimate its 3D trajectory; for this reason in the following, we provide an introduction to such Bayesian formalism.

3. Bayesian tracking and smoothing

We start introducing tracking as a Bayesian filtering problem, then we extend the formalization to Bayesian smoothing. Let s_t be the vehicle state, in our case the 3D vehicle pose, and z_t some measurement at a given time t, in our case the vehicle blob.

According to the Bayesian filtering framework [26], tracking aims at the maximization through Bayesian statistics of the so called posterior probability, or belief, for the system state, i.e., $Bel(s_{t+1}) = p(s_{t+1}|z_{1:t+1})$, where $z_{1:t+1} = \{z_1, \ldots, z_t, z_{t+1}\}$. Usually the Markov hypothesis is assumed, i.e., a state depends only from the previous state and, optionally, the applied control (in our case we omit the control term). The Markov assumption in the vehicular case is quite reasonable and very common in a tracking algorithm [27].

If $z_{1:t+1}$ are the measurements up to time $t+1$ and $s_{1:t}$ are the estimated states up to time t we want to estimate:

$$\begin{aligned} Bel(s_{t+1}) &= p(s_{t+1}|z_{1:t+1}) \\ &= \alpha p(z_{t+1}|s_{t+1}, z_{1:t})p(s_{t+1}|z_{1:t}) \\ &= \alpha p(z_{t+1}|s_{t+1})p(s_{t+1}|z_{1:t}), \end{aligned} \tag{1}$$

where $p(z_{t+1}|s_{t+1})$ is the *likelihood* of observation z_{t+1} in state s_{t+1} and $\alpha = 1/P(z_{t+1}|z_{1:t})$ is a normalization constant. Moreover, we have:

$$\begin{aligned} p(s_{t+1}|z_{1:t}) &= \int p(s_{t+1}, s_t|z_t)ds_t \\ &= \int p(s_{t+1}|s_t)Bel(s_t)ds_t, \end{aligned} \tag{2}$$

where $p(s_{t+1}|s_t)$ is the state *transition model* and represents the probability to reach the state s_{t+1} starting from the state s_t.

It is worth mentioning that the approach described previously allows an iterative computation of the a-posteriori probability of the positions in the vehicle trajectory and each of them is obtained exploiting all the measures acquired up to that time.

A different approach to Bayesian tracking is called *smoothing*. In this case, if T is the number of all the positions in the trajectory, the entire states sequence $s_{1:T}$ is estimated using all the measurements. In the case of smoothing, the vehicle trajectory is estimated at each frame using past, present and future blobs. In this case we aim at maximizing the posterior probability $p(s_{1:T}|z_{1:T})$ for the whole sequence, and using Bayes theorem this becomes:

$$p(s_{1:T}|z_{1:T}) \propto p(z_{1:T}|s_{1:T})p(s_{1:T}). \tag{3}$$

In the general case, the states and the measurements have different probability distributions, but, both in filtering and in smoothing, the classical Bayesian tracking algorithms estimate the a-posteriori probability assuming that that all the distributions are Gaussians. This is often a good approximation, and, if the tracked system has a linear dynamic, a linear measurement model and a Gaussian initial state, these assumptions enable the application of the classical Kalman Filter formulas. For non linear systems, we can use the Extended Kalman Filter (EKF) which provides an approximation of the Kalman Filter by linearizing the system according to the current estimates. Under linearity and Gaussian noise assumptions, the Kalman Filter provides an optimal solution efficiently. A smoothed version of the Kalman Filter also exists and is called Kalman smoother.

When non-linearities are significant and the Gaussian assumption does not hold, non-parametric approaches are used. A non-parametric approach does not model the posterior probability as a parametric distribution, e.g., a Gaussian distribution, but it estimates the posterior using, for instance, a set of samples. More formally, let π be a generic distribution, and $X_i \sim \pi$ independent samples with $i = 1 \ldots N$. If δ_{X_i} is Dirac function centered in X_i, then $\hat{\pi}$ such that:

$$\hat{\pi} = \frac{1}{N}\sum_{i=1}^{N}\delta_{X_i} \tag{4}$$

is a non-parametric, sample-based, approximation of π.

By approximating a state distribution with a set of samples, it is often possible to compute measurement likelihood in a straightforward way: for each sample it is sufficient to compute a score as described in Section 4.4.

In the next section we describe two tracking algorithms to implement trajectory smoothing using a sample based approach. The first one computes the most likely sequence of states for the given trajectory using the Viterbi algorithm [28], and it implements in an exact way equation (3); the second approach is based on the use of two Particle Filters each using equation (7), and it provides a simpler but effective approximation of (3).

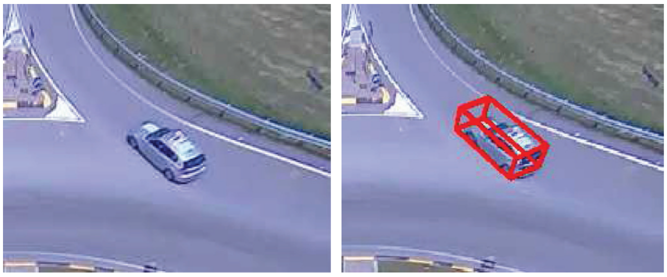
4. Two methods for sample-based 3D smoothing

As opposed to typical 3D tracking algorithms, the methods compared in the following do not directly process images, but they make use of the 2D vehicles trajectories extracted from the 2D tracking algorithm presented in [2]. The 2D tracking algorithm recognizes the vehicles by background subtraction [29], therefore vehicle measurement corresponds to blobs. For each vehicle, a 2D Extended Kalman Filter uses the blob centroids to estimate its trajectory on the image plane. The resulting trajectories are composed by the history of blob centroids on the image sequence and this is the input for the two algorithms presented here. By doing this, we compare the smoothing and 3D trajectory reconstruction methods starting from the same data and, more relevantly, from the same data association.

The implemented algorithms use a model-based approach. Since vehicle dimensions can vary significantly, a model with fixed dimension can cause problems, as described in [8], then we model the existing vehicles through a parallelepiped with different dimensions. We model three main classes of vehicles: cars, trucks and motorcycles. For each class we define a reference parallelepiped, by setting a reference length value L_c and two ratio values $r_{hc} = H_c/L_c$ and $r_{wc} = W_c/L_c$, respectively for height (H_c) and width (W_c) computation. We create a set of models for each class by extracting a set of sample values of model length from a Gaussian distribution centered on the reference length L_c of each class, these are called class-scale samples. From each of these samples we infer the model height and width from the values r_{hc} and r_{wc}. We fixed all the parameters as in Table 1; we have chosen reasonable values according to the dimensions of existing vehicles. We use a different standard deviation for each class, since car dimensions vary less than truck dimensions, and more than motor-

Table 1
Reference dimensions and variances of the models.

	$L_c(m)$	$H_c(m)$	$W_c(m)$	σ_l^2	σ_h^2	σ_w^2
Car	4	1.4	1.7	0.5	0.15	0.15
Truck	15.0	3	2.0	2.5	0.5	0.5
Motorcycle	2.0	1.0	0.5	0.25	0.3	0.3



(a) *Vehicle to track.* (b) *Projected model on the image plane.*

Fig. 3. Example of expected result when the correct parallelepiped model is projected on to the image plane.

cycle ones. Fig. 3 shows an example of the parallelepiped model projected on the image plane: the model must ideally wrap the tracked vehicle.

In the following subsections we describe the two compared algorithms: the Viterbi-based algorithm and the Particle Smoother. Both algorithms compute the likelihood in the same way (we postpone its description to Section 4.4). Moreover, considering that vehicles in the roundabout (should) always proceed forward, the transition state model slightly different from the common Gaussian motion model used in literature. Instead of a Gaussian propagation model, we make use of the log-normal distribution to model the motion difference with respect to the previous pose and thus we bias the state transition in the forward direction as in [30,10].

4.1. *Sampling a 3D state from a 2D measurement*

Both the Viterbi and the Particle Smoother need to extract the 3D sample states from the 2D measurements of the vehicle; the Viterbi Algorithm performs this sampling stage frame-by-frame, while the Particle Smoother uses it as the initialization of the smoothing process.

In the sampling from 2D to 3D, the current blob and the next blob centroids are projected on a plane parallel to the roundabout and passing through the center of the vehicle model under evaluation. Let the resulting points be, (x_t, y_t) and (x_{t+1}, y_{t+1}) respectively. Let now θ_t be the orientation of the vector from (x_t, y_t) to (x_{t+1}, y_{t+1}); the algorithm extracts a set of n samples from a trivariate Gaussian distribution having mean (x_t, y_t, θ_t) and a diagonal covariance structure reflecting the independence of the three components (we set experimentally $\sigma_x^2 = 0.5$m, $\sigma_y^2 = 0.5$m, $\sigma_\theta^2 = \pi/12$ rad). Independence is a reasonable assumption to simplify the tuning of the system, but a non-diagonal covariance matrix could be used as well. A simplified example of this operation is shown in Fig. 4; each of these samples represents a vehicle state, i.e., vehicle position and orientation on the roundabout plane at time t.

4.2. *The Viterbi-based algorithm*

For each 2D vehicle trajectory and for each class-scale sample, the Viterbi-based algorithm (VBA) performs two steps: a frame-by-frame sampling and the estimate of the most likely trajectory, i.e., the sequence of samples, using the Viterbi algorithm [28].

In the frame-by-frame sampling, the algorithm extracts a set of sample for each frame and for each class-scale sample according to the algorithm described in Section 4.1. After this extraction step, we apply the Viterbi algorithm for only the most likely class-scale sample: we compute the likelihood (Section 4.4) of all the vehicle state samples; then, for each class-scale sample, we compute the mean likelihood of the most likely samples for each frame; finally, we choose the class-scale sample which gives the highest sum of mean probabilities.

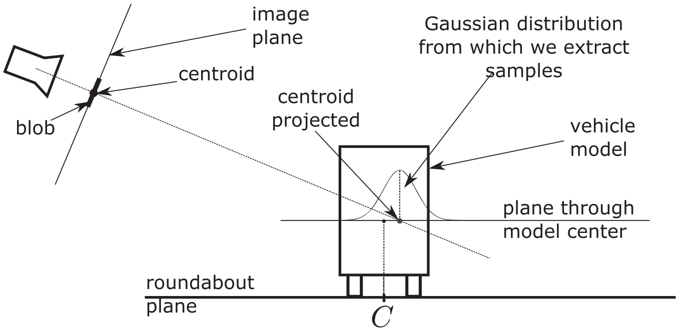


image plane
Gaussian distribution from which we extract samples
centroid
centroid projected
blob
vehicle model
plane through model center
roundabout plane
C

Fig. 4. 2D vehicle centroid projected on the plane passing through the 3D model center. We extract the pose samples from the Gaussian centered on this projection for each class-scale sample.

We use the Viterbi algorithm to compute the most likely sequence of samples along the trajectory, extracted for the chosen class-scale. The Viterbi algorithm finds the Maximum A-Posteriori (MAP) estimate given a discretized state space: in our case the state space is represented by the set of all possible 3D poses of the vehicle, and the discretization is performed thanks to the previous sampling stage.

Let s_t be a state sample extracted at time $1 \le t \le T$. The Viterbi algorithm finds the succession of samples which maximizes Eq. (3) as:

$$\bar{s}_{1:T} = \arg\max_{s_{1:T}} \{p(s_{1:T}|z_{1:T})\}, \tag{5}$$

and, according to the Bayes rule and the Markov hypothesis:

$$\begin{aligned} \bar{s}_{1:T} &= \arg\max_{s_{1:T}} \prod_{t=1}^{T} \{p(s_{t+1}|s_t)p(z_{t+1}|s_{t+1})\} \\ &= \arg\min_{s_{1:T}} \sum_{t=1}^{T} \{-\log(p(s_{t+1}|s_t)) - \log(p(z_{t+1}|s_{t+1}))\}, \end{aligned} \tag{6}$$

where we compute the likelihood term $p(z_{t+1}|s_{t+1}, s_t)$ as explained in Section 4.4, and the transition probability $p(s_{t+1}|s_t)$ coincides with the log-normal motion model that we use within the particle filter.

In practical implementations of the Viterbi algorithm, the states, i.e., the samples, are represented in a graph as in Fig. 5: each node represents a state sample and each sample at time t is connected to samples at time $t+1$ through an arc weighted according to Eq. (6), i.e., $-\log(p(s_{t+1}|s_t)) - \log(p(z_{t+1}|s_{t+1}, s_t))$. Therefore, to find the best path according to Eq. (6), we look for the shortest path from a fictitious starting node, connected to samples at time 1, to a fictitious ending node, connected with each sample at time T; this can be efficiently done through the Dijkstra algorithm [31]. In this way, we obtain the succession of vehicle states, i.e., the most likely trajectory.

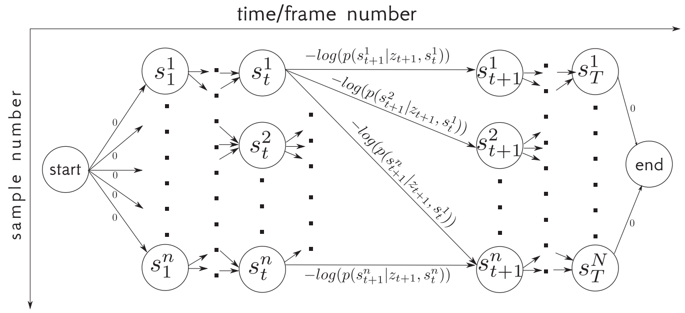

time/frame number
sample number
start
end
0
s_1^1
s_1^n
s_t^1
s_t^2
s_t^n
s_{t+1}^1
s_{t+1}^2
s_{t+1}^n
s_T^1
s_T^N
$-log(p(s_{t+1}^1|z_{t+1}, s_t^1))$
$-log(p(s_{t+1}^2|z_{t+1}, s_t^1))$
$-log(p(s_{t+1}^n|z_{t+1}, s_t^1))$
$-log(p(s_{t+1}^n|z_{t+1}, s_t^n))$

Fig. 5. Graph built to apply Viterbi algorithm.

4.3. The Particle Smoother algorithm

The second algorithm compared is a Particle Smoother which relies on two Particle Filter trackers. Differently from the Viterbi algorithm, which directly looks for the MAP solution, the Particle Filter, and, in turn, the Particle Smoother estimate the distribution of the Maximum A-Posteriori solution through a set of samples. The Particle Filter deals with a continuous state, therefore we do not need to discretize it beforehand with the frame-by-frame sampling as in the previous algorithm.

At each time t, the Particle Filter estimates $p(s_{t+1}|z_{1:t})$ (Eq. (1)) with a sampled distribution; a set of random state samples, called particles, represents $p(s_{1:t+1}|z_{1:t+1})$ as:

$$p(s_{1:t+1}|z_{1:t+1}) \approx \sum_{i=1}^{N_s} w_{t+1}^{i}\delta(s_{1:t+1} - s_{1:t+1}^{i}), \tag{7}$$

where w_{t+1}^{i} is the weight associated to the i-th particle. The weights are updated at time $t+1$ according to this equation:

$$w_{t+1}^{i} \propto w_{t}^{i}\frac{p(z_{t+1}|s_{t+1}^{i})p(s_{t+1}^{i}|s_{t}^{i})}{p(s_{t+1}^{i}|s_{t}^{i}, z_{t+1})}. \tag{8}$$

The main issue of Particle Filters is the particle degeneracy, i.e., the particles tend to collapse around a single state value. To solve this issue, after weights computation, the Particle Filter resamples a new set of particles according to approximated probability distribution of s_{t+1}. This new set replaces old particles, and each new particle has now the same weight. More details about Particle Filter are available in [27].

In our work, the particles represent the vehicle state at time t, i.e., vehicle position and orientation on the roundabout plane. We implemented a Particle Filter for each model defined by each class-scale sample (see the algorithm in Fig. 6). These Particle Filters are initialized through a set of particles extracted from the first frame of the 2D trajectory according to the steps described in Section 4.1, then we apply the standard Particle Filter iterations. After this forward pass of the vehicle trajectory, the algorithm chooses the best class-scale sample to represent the tracked vehicle, i.e., for each time t it computes the mean probability of particles for each class-scale sample and, then, it chooses the class-scale sample for which the sum of those means is maximum. Consequently, we select the model whose dimensions have been calculated from the class-scale sample chosen.

At this point, we apply a backward recursion that turns the Particle Filter into a Particle Smoother. We start from the last group of particles estimated by the forward Particle Filter and we change the sign of the orientations. Then, we apply a second instance of the same Particle Filter that tracks the vehicle backward by considering the blob measurements in reverse order. We do not need to change the orientation of the vehicle parallelepiped model since it is symmetric under rotations about its centroid.

Taking into account the theoretical result about which particles tends to wrap around the real state, the second recursion aims at diminishing particles variance and it especially makes the estimate more accurate for the states in the first part of the trajectory where the forward Particle Filter had not yet collected sufficient information to converge. This backward recursion acts as a smoother; a detailed formalization of the Backward Particle Smoother is in [32, p. 167].

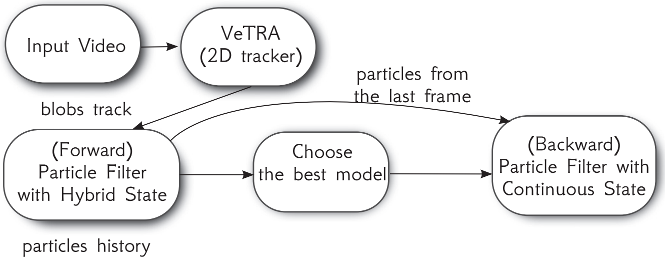

Input Video
VeTRA (2D tracker)
blobs track
particles from the last frame
(Forward) Particle Filter with Hybrid State
Choose the best model
(Backward) Particle Filter with Continuous State
particles history

Fig. 6. Flow chart for the Particles Smoother algorithm.

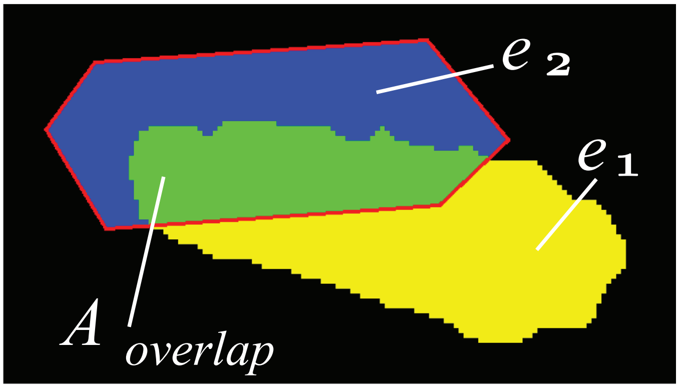


Fig. 7. Likelihood calculation: e_1 counts the pixels of the blob not covered by the projected model; e_2 counts the pixels of predicted model that do not correspond to the blob.

4.4. Likelihood calculation

In both the methods that we investigated in this paper, a key step is the computation of the samples likelihood. Let focus on measures z_t which are the blobs extracted through background subtraction and on the state s_t representing vehicle pose on the roundabout plane. Likelihood calculation follows these steps, analogous to [8] and [23]:

1. Project on the image plane the model located in s_t on the roundabout (the red polygon in Fig. 7).
2. Compute blob area (A_{blob}), visible model projection area (A_{mv}), and overlap area between the blob and the model projection ($A_{overlap}$) (see Figs. 7 and 8).
3. Compute the two errors $e_1 = A_{blob} - A_{overlap}$ and $e_2 = A_{mv} - A_{overlap}$ (see Fig. 7).
4. Define the score term: $e = [\lambda_1 e_1 + \lambda_2 e_2]/A_{blob}$, where λ_1 e λ_2 weight e_1 and e_2 such that $\lambda_1 + \lambda_2 = 1$.

Likelihood is then defined as:

$$p(z_t|s_t) = \begin{cases} 1 - e, & \text{if } 1 - e > 0 \\ 0, & \text{if } 1 - e < 0. \end{cases} \tag{9}$$

Notice that only the visible part of the model projection concurs to compute the likelihood. So, A_{mv} is not the entire area of the projected model, but the projected area visible in the current camera image. Fig. 8 shows the difference between A_{mv} and the area of the entire projected model, A_m.

Leveraging on the flexibility of the region-based approach, we implemented the likelihood calculation in a multiple camera setting too, where a vehicle could be seen by more than one camera. Likelihood calculation is the only stage of our algorithms in which we have to consider the presence of more than one camera.

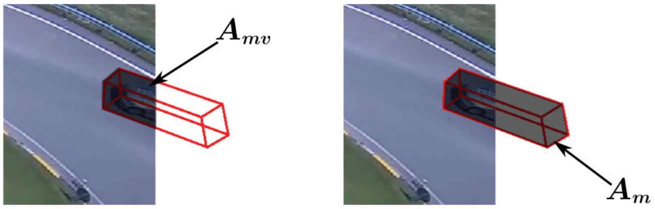


Fig. 8. Difference between A_{mv} and A_m (shaded area).

Let z_t^j be the blob of one vehicle perceived by the j-th camera (C_j), and n_c the number of cameras. Let $p_j(z_t^j|s_t)$ be the likelihood calculated for each camera j according Eq. (9); this value is weighted according to the probability of observing the vehicle from that camera, i.e., $p(C_j) = A_{mv}^j / A_m^j$, and, by doing so, the overall likelihood becomes:

$$p_{\mathrm{tot}}(z_t|s_t) = \frac{p(C_1)p_1(z_t^1|s_t) + p(C_{n_c})p_{n_c}(z_t^{n_c}|s_t)}{p(C_1) + \cdots + p(C_{n_c})}. \tag{10}$$

The camera probability $p(C_j) = A_{mv}^j / A_m^j$ takes into account different situations. For instance, with only two cameras, when both cameras see the whole vehicle, or, more precisely, the back-projected model, they have the same probability equal to 0.5; when one camera does not see the model, its probability is zero. In general the probability is proportional the observed model percentage.

5. Experimental results

We presented the Viterbi Based Algorithm and the Particle Smoother implementations to reconstruct the 3D trajectory of vehicles from a sequence of images. We compared the two algorithms by testing them in a simulated and in a real scenario where the ground truth is known. In the video http://youtu.be/8swc1nhxs14 we show the results in both cases.

For the *simulated* scenario we created six video sequences where a fixed-size parallelepiped model virtually drove through a hypothetical roundabout. We simulated two different trajectories: a uniform circular motion and a linear motion where the synthetic car starts, stops and leaves again. For each trajectory we use three different perspectives. The simulated scenario represents an ideal case because the blobs representing the vehicle coincide exactly with the parallelepiped used by the tracking: the errors due to noise ware significantly reduced and have almost no relevance with respect to a real case. Moreover, the results obtained in this experiment give an idea of the maximum performance the two methods can attain.

The *real* scenario came from a field survey where vehicles circulating in a roundabout were recorded by two cameras, but the true vehicle positions on ground are not a priori known; therefore, we equipped a vehicle with an RTK-GPS device and an inertial sensor and we collected an accurate estimate of the vehicle positions and orientations for nine transits of this vehicle. In this way, we have been able to evaluate performance both in the single and in the multiple camera cases.

To obtain a proper result, the comparison between poses requires the two poses to be taken at the same time. However, synchronization between RTK-GPS devices, inertial sensors and video cameras was not perfect. In addition, the sensors and the cameras working frequencies was different, so that the estimate provided by sensors and our algorithm cannot be considered perfectly synchronous. In the multiple camera case this imperfect synchronization between the two considered video cameras has been another source of error. Nevertheless, these issues affect in equal manner the two compared algorithm.

Table 2 reports tracking errors for the Viterbi Based Algorithm (VBA) and the Particle Smoother (PSA) with one camera and in a multiple camera setup both in simulated and real scenarios. The errors are computed for x and y coordinates of vehicle pose; we report the distance d between the estimated x, y and the ground truth, and its orientation θ, in degrees. The median, MAD and IQR values are presented. MAD is the median absolute deviation, i.e., for a certain x vector MAD $= median(|x_i - median(\mathbf{x})|_{\forall i})$, and IQR is the Inter Quartile Range, where IQR $= Q_{\frac{3}{4}} - Q_{\frac{1}{4}}$ where $Q_{\frac{1}{4}}$ is first quartile and $Q_{\frac{3}{4}}$ is third quartile of the error distribution. Both MAD and IQR represent an error dispersion index; the smaller they are, the better it is.

Errors are generally very low: a few centimeters for x and y coordinates of pose, and only a few degrees for orientation θ. In the simulated case the two algorithms reach a very satisfying accuracy, consistent with the aims of many types of analyses on vehicle trajectory. However, the PSA gives better results than VBA; this is due to the fact that VBA extracts state samples from each frame independently from previous or next state, while Particle Smoother generates a set of particles-samples starting from previous particles through the state transition model and this increases its density of sample in the useful parts of the search space (we report a more detailed discussion in Section 6).

Table 2
Tracking errors in the simulated and real scenarios.

			Median	MAD	IQR
$x(m)$	Simulated	VBA	0.036	0.042	0.090
		PSA	**−0.027**	**0.022**	**0.050**
	Real	VBA single camera	−0.180	0.236	0.468
		PSA single camera	−0.171	**0.171**	**0.348**
		VBA multicamera	0.154	0.305	0.512
		PSA multicamera	**−0.056**	0.261	0.475
$y(m)$	Simulated	VBA	−0.039	0.055	0.113
		PSA	**0.004**	**0.023**	**0.046**
	Real	VBA single camera	−0.136	0.316	0.500
		PSA single camera	0.058	0.135	0.277
		VBA multicamera	**−0.041**	0.241	0.384
		PSA multicamera	0.094	**0.179**	**0.248**
$d(m)$	Simulated	VBA	0.090	0.054	0.059
		PSA	**0.046**	**0.040**	**0.056**
	Real	VBA single camera	0.483	0.215	0.392
		PSA single camera	0.288	**0.116**	0.187
		VBA multicamera	0.343	0.215	0.288
		PSA multicamera	**0.237**	0.191	**0.125**
$\theta(deg)$	Simulated	VBA	−4.137	1.281	2.478
		PSA	**0.803**	**1.176**	**2.459**
	Real	VBA single camera	−3.664	5.123	10.227
		PSA single camera	4.156	**2.446**	**4.875**
		VBA multicamera	−3.759	3.845	7.718
		PSA multicamera	**2.401**	7.629	8.465

VBA is computationally less efficient than Particle Smoother, due to the shortest path search which has quadratic complexity in the number of frames; Particle Smoother has a linear complexity because it uses two Particle Filters which are, in turn, linear. We report execution times in Table 3 as a function of the number of frames in which the vehicle appears; the PSA is three time faster with respect to VBA. Moreover, PSA turned out easier to be implemented with respect to the VBA. It should be noted that Particle Smoother uses 250 state samples for each frame, VBA needs 500 samples to reach comparable results. Both algorithms use a set of 30 class-scale samples, that is, 10 samples for each of the three classes: car, truck and motorcycle. Execution times are considerably high because, at this stage of implementation, we preferred to use Matlab$^{\mathrm{TM}}$development environment, which offers high flexibility and ease of implementation, but at the cost of some reduced efficiency. It should be noticed that most of the computation could be performed in a parallel manner on modern computer architectures in a similar fashion both for PSA and for VBA.

The aim of both experimental setups was mainly to test the accuracy of the tracking of vehicles reached by the two compared algorithm. Therefore we did not investigate the accuracy in the estimate of vehicle dimensions. However, the two (pose and dimensions)

Table 3
Execution times (in seconds).

Number of frames	45	106	230	442
VBA	255	1456	3642	6598
PSA	87	624	1141	2817

estimates are strictly correlated; in both simulated and real scenarios the parallelepiped dimensions was handled in quite an accurate way.

6. Discussion and conclusions

In this paper, we compared two algorithms to implement a 3D tracking system using a model-based and region-based approach. In particular, we focus our experimental analysis on the roundabout scenario since it offers a challenging test-bed where perspective distortions and vehicle inter-occluding trajectories make the 3D tracking necessary to enable traffic monitoring and flow analysis.

In the first part of the paper we supported the choice of this approach. Model-based is the standard approach to 3D vehicle tracking: some researchers adopt an edge-based approach while others a region-based one. Even if it is less robust to illumination changes, we choose the latter for two main reasons: it is robust against local minima minimization issue, and it is flexible, since we have not to choose an accurate model of the tracked vehicle.

Then we implemented two Monte Carlo smoothing algorithms, i.e., the Viterbi-based (VBA) and the Particle Smoother (PSA) both for the single camera and the multiple camera cases, and we tested them both in simulated and in real scenarios. The experimental results showed that the PSA reaches better performance, both in terms of speed and accuracy.

From a theoretical perspective the VBA provides the Maximum A-Posteriori estimate of the vehicle trajectory, while the PSA only approximates the Maximum a Posteriori distribution (out of which the MAP estimate is computed) in a sample-based fashion. Therefore, before the comparison, we expected that VBA would have outperformed the PSA, at least in terms of extracting the best solution; the results overturn our belief.

The reason of this apparently counter-intuitive result has to be ascribed to the state discretization in the Viterbi algorithm: VBA finds the trajectory among the set of samples extracted from each frame, then it does not look for the MAP estimate in the whole continuous state space. To obtain the real MAP estimate we should have subsampled the space with an infinitesimal discretization, increasing the number of samples considerably. Indeed, we tested the VBA with different combinations of parameters and we found that the most sensitive parameter is the number of samples, but to have a fair comparison with the PSA we chose to limit the number of samples to 500 (already twice as much the number of samples of the PSA). Moreover, the PSA deals with a continuous state and extracts the samples from a continuous distribution at each frame. This lets the motion model, in the PSA, to focus the samples around the *real* state, while, in the Viterbi case, the motion model is only involved in the posterior probability evaluation.

References

[1] S. Kamijo, Y. Matsushita, K. Ikeuchi, M. Sakauchi, Traffic monitoring and accident detection at intersections, IEEE Trans. Intell. Transp. Syst. 1 (2000) 108–118.

[2] L. Mussone, M. Matteucci, M. Bassani, D. Rizzi, Traffic analysis in roundabout intersections by image processing, in: Proceedings of the 18th IFAC World Congress, vol. 18, 2011.

[3] R. Cucchiara, M. Piccardi, P. Mello, Image analysis and rule-based reasoning for a traffic monitoring system, IEEE Trans. Intell. Transp. Syst. 1 (2000) 119–130.

[4] A. Prati, I. Mikic, M.M. Trivedi, R. Cucchiara, Detecting moving shadows: algorithms and evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 25 (2003) 918–923.

[5] Y.K. Jung, Y.S. Ho, Traffic parameter extraction using video-based vehicle tracking, in: Intelligent Transportation Systems, Proceedings, 1999 IEEE/IEEJ/JSAI Interna-tional Conference on, IEEE, 1999, pp. 764–769.

[6] L. Mussone, M. Matteucci, M. Bassani, D. Rizzi, An innovative method for the analysis of vehicle movements in roundabouts based on image processing, J. Adv. Transp. 47 (6) (2013) 581–594.

[7] V. Lepetit, P. Fua, Monocular model-based 3D tracking of rigid objects: a survey, in: Foundations and Trends in Computer Graphics and Vision, pp. 1–89.

[8] X. Song, R. Nevatia, Detection and tracking of moving vehicles in crowded scenes, in: Motion and Video Computing, WMVC '07, IEEE Workshop on, 2007, p. 4.

[9] Z. Zhang, K. Huang, T. Tan, Y. Wang, 3d model based vehicle tracking using gradient based fitness evaluation under particle filter framework, in: Pattern Recognition (ICPR), 2010 20th International Conference on, IEEE, pp. 1771–1774.

[10] M. Matteucci, D. Rizzi, A. Romanoni, L. Mussone, 3d image processing of vehicular trajectories in roundabouts, in: World Conference on Transportation Research, pp. 1–11. Available at http://www2.wctr2013rio.com/publications/1077/index.html.

[11] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Comput. Surv. 38 (2006) 13.

[12] P. Pérez, C. Hue, J. Vermaak, M. Gangnet, Color-based probabilistic tracking, in: Computer Vision ECCV 2002, Springer, 2002, pp. 661–675.

[13] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (2003) 564–577.

[14] D.A. Ross, J. Lim, R.S. Lin, M.H. Yang, Incremental learning for robust visual tracking, Int. J. Comput. Vis. 77 (2008) 125–141.

[15] H. Zhou, Y. Yuan, C. Shi, Object tracking using sift features and mean shift, Computer Vis. Image Understanding 113 (2009) 345–352.

[16] S. Zhang, H. Yao, X. Sun, X. Lu, Sparse coding based visual tracking: review and experimental comparison, Pattern Recognit. 46 (2013) 1772–1788.

[17] N. Buch, S.A. Velastin, J. Orwell, A review of computer vision techniques for the analysis of urban traffic, IEEE Trans. Intell. Transp. Syst. 12 (2011) 920–939.

[18] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2000.

[19] J. Lou, T. Tan, W. Hu, H. Yang, S. Maybank, 3-D model-based vehicle tracking, IEEE Trans. Image Process. 14 (2005) 1561–1569.

[20] H. Yang, J. Lou, H. Sun, W. Hu, T. Tan, Efficient and robust vehicle localization, in: Image Processing, Proceedings, 2001 International Conference on, vol. 2, 2001, pp. 355–358.

[21] D.A. Migliore, M. Matteucci, M. Naccari, A revaluation of frame difference in fast and robust motion detection, in: Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks, VSSN '06, ACM, New York, NY, USA, 2006, pp. 215–218.

[22] T. Brox, B. Rosenhahn, J. Weickert, Three-dimensional shape knowledge for joint image segmentation and pose estimation, in: W.G. Kropatsch, R. Sablatnig, A. Hanbury (Eds.), DAGM-Symposium, vol. 3663, Lecture Notes in Computer Science, Springer, 2005, pp. 109–116.

[23] N. Buch, F. Yin, J. Orwell, D. Makris, S. Velastin, Urban vehicle tracking using a combined 3D model detector and classifier, in: J. Velásquez, S. Ríos, R. Howlett, L. Jain (Eds.), Knowledge-Based and Intelligent Information and Engineering Systems, vol. 5711, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2009, pp. 169–176.

[24] H. Kollnig, H.H. Nagel, 3d pose estimation by directly matching polyhedral models to gray value gradients, Int. J. Comput. Vis. 23 (1997) 283–302.

[25] K.H. Lee, J.N. Hwang, J.Y. Yu, K.Z. Lee, Vehicle tracking iterative by Kalman-based constrained multiple-kernel and 3-d model-based localization, in: Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, IEEE, pp. 2396–2399.

[26] D. Fox, J. Hightower, L. Liao, D. Schulz, G. Borriello, Bayesian filters for location estimation, IEEE Pervasive Comput. 2 (2003) 24–33.

[27] M. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Process. 50 (2002) 174–188.

[28] G.J. Forney, The viterbi algorithm, Proc. IEEE 61 (1973) 268–278.

[29] A. Romanoni, M. Matteucci, D.G. Sorrenti, Background subtraction by combining temporal and spatio-temporal histograms in the presence of camera movement, Mach. Vis. Appl. 25 (6) (2014) 1573–1584.

[30] A. Romanoni, Ricostruzione 3D delle traiettorie veicolari da immagini di una o più telecamere, Master's thesis, Politecnico di Milano, 2012.

[31] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271.

[32] S. Särkkä, Bayesian Filtering and Smoothing, vol. 3, Cambridge University Press, 2013.