3D IMAGE PROCESSING OF VEHICULAR TRAJECTORIES IN ROUNDABOUTS

Matteo Matteucci, Politecnico di Milano, DEI, Milano, 20133, Italy

Davide Rizzi, Politecnico di Milano, DEI, Milano, 20133, Italy

Andrea Romanoni, Politecnico di Milano, DEI, Milano, 20133, Italy

Lorenzo Mussone, Politecnico di Milano, BEST, Milano, 20133, Italy, mussone@polimi.it

ABSTRACT

Vehicular trajectory analysis is a challenging task both for safety and capacity concerns in road transportation. In particular, vehicular trajectories on roundabouts should be carefully designed in order to achieve desired performance. Therefore, the analysis of what happens on existing roundabouts is a key analytical tool in propelling the drafting of new design norms and in correcting building errors. Difficulties in analysing, by visual tracking, vehicle trajectories on roundabouts are known; for instance, different perspective effects on vehicle outlines during circulation, due to the way cameras are pointed, negatively affect precision both for vehicle position and speed estimation in 2D tracking. Especially when flow includes heavy vehicles, it is not possible to perform trajectory reconstruction based on the consecutive projection of vehicle image on the road plane and a model based approach is required. The research described in this paper overcomes the above mentioned problems and faces the trajectory tracking in roundabouts by estimating the true 3D position of each vehicle. Two different algorithms are investigated both based on a model based smoothing Montecarlo approach: the Viterbi algorithm and the Particle Smoother. The model used in this research for vehicles is a parallelepiped whose dimensions are directly estimated from images inferring vehicular class: cars, trucks or motorbikes. Both algorithms are tested and compared on data collected in one working roundabouts with two different set-ups. 3D tracking presents a higher complexity to be implemented since the object to be reconstructed needs to be modelled to retrieve its position and orientation from its 2D projection on the image plane. On the other hand, performance is enhanced and precision in trajectory and speed reconstruction (especially with the Particle Smoother algorithm) is not far from that obtainable by a RTK-GPS system. This new approach allows a better reconstruction of flow features in roundabouts and hence provides researchers with a very accurate tool for analysing roundabout performance, applicable also to other types of road infrastructures

Keywords: Vehicular trajectory, Roundabouts, 3D Visual tracking, Montecarlo smoothing, sample-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 could be used, for instance, to detect infractions, road accidents, and it could also provide useful information about traffic distribution. Moreover, when the scenario is a roundabout intersection, such a tracking system could provide useful data to improve the design of the roundabout itself.

One approach for traffic monitoring is visual vehicle tracking. Tracking is the process of recognizing moving objects and estimating their trajectory. Vehicle tracking could be carried out using different sensors and approaches: in this paper we use a multi camera vision-based system, which relies on the analysis of videos from the roundabout where vehicles pass, hence the name of visual-tracking. Visual-tracking is quite an active research topic in the computer vision community, where, in general, tracked object are not only vehicles, but could be e.g. people, human posture, face or head.

Most of existing visual tracking systems for vehicles, propose a 2D approach, called 2D tracking; this method identifies moving vehicles on the image plane and it tracks their trajectories on it. Although in some applications this type of estimation might be sufficient, to fully understand vehicle interaction with the roundabout intersection and with each other, we need to estimate vehicle trajectories with high accuracy with respect to a 3D world reference system; this process is called 3D tracking.

The standard approach used to obtain a 3D trajectory from a 2D tracking system is to project the 2D vehicle trajectory from the image plane, estimated with 2D tracking, on the roundabout plane as done for instance in Mussone et al. (2011a and 2011b). In Figure 1 a sketched example of this process for a single frame is shown. Figure 2a shows the image of a small truck and Figure 2b shows the corresponding region on the image plane, namely the corresponding *blob*. The centroid of this blob on the image plane is used to approximate the vehicle position in 3D by taking the intersection between the viewing ray passing through the blob centroid and the ground plane: C_{err} is the projected center on the roundabout plane. The main drawback of this approach is the high sensitivity to perspective deformation and the effect of the unknown height of vehicles, especially when they are heavy trucks. Due to the angling of the camera, some estimated trajectories could be significantly different from real ones, e.g. in Figure 1 the estimated center C_{err} is far from the real vehicle center C. Moreover, the presence of shadows introduces a bias in the center of the blob which depends on the position of the vehicle in the roundabout and this bias is time variant.

To overcome these issues, in this paper we present a model-based 3D tracking algorithm. This class of algorithm gives a trajectory estimation in 3D world coordinates, by exploiting the knowledge of a model for the tracked object; in particular, to represent vehicles of different shapes and dimensions we use a set of models with variable dimensions and we detect automatically which model should be used for the current vehicle. The approach was implemented first for a single camera case, and was then extended to multi-camera cases as well.

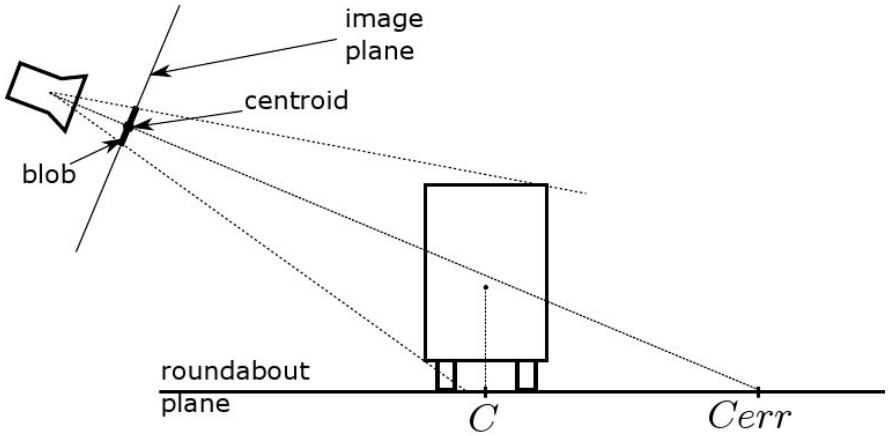

Figure 1: Back-projection error; due to the height of the vehicle and the perspective transformation, the estimated position of the vehicle on the ground plane differs significantly from the real one.

(a) A vehicle image.

(b) The corresponding blob of a) defined by the white region.

Figure 2: Example of a blob extracted from the current frame through background subtraction (Migliore et al., 2006).

In the next section of the paper, a literature overview on visual-tracking is presented, focusing our attention on 3D model-based tracking algorithms; we also present a Bayesian smoothing formalization of the tracking problem. Then, in the third section two tracking algorithms are described implementing a Montecarlo approximation of the Bayesian smoother: we explain the vehicle model management, how proposed algorithms work and how likelihood probability is calculated for single and multi camera cases. In the final section experimental results for the two tracking algorithms on simulated and real scenarios are reported.

2. TRACKING APPROACHES OVERVIEW

As already mentioned, visual-tracking algorithms can be classified in 2D tracking and 3D tracking algorithms. The former give a trajectory estimate on the image plane which is generally quite a good approximation of object trajectory; however they do not allow for an accurate 3D position estimation with respect to a global reference frame. Conversely, 3D tracking algorithms are able to reconstruct object trajectories in world coordinates achieving a higher level of accuracy. 3D estimates give a clearer idea of what is going on in the observed scene, but this entails a more complex task.

2.1 3D model-based tracking

To simplify the tracking task, all studies in literature assume that the camera calibration is known (e.g. Hartley and Zisserman, 2004); then, they usually assume the Ground Plane Constraint, i.e., a vehicle always lies on the road plane, so tracking is executed on this plane in order to diminish the vehicle degrees of freedom to be estimated from 6 to 3. Most of 3D tracking systems make use of an object model; this is why 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 fits the object measure extracted from the images.
Lepetit and Fua (2005) presented a 3D tracking algorithm classification according to the method used to compare model pose with object measures extracted from images. The first class follows the so called edge-based approach. These algorithms project the object model on the image plane using the camera model to obtain a geometric shape, typically this results in a polygon. Then, starting from an initial guess, a rigid transformation, i.e., a rotation and translation, is found to minimize the distance between projected segments of the model and image edges (Harris and Stennet, 1990; Zhang et al., 2010; Lou et al., 2005; Yang et al., 2001). This method has the advantage of being robust to light changes, but a relevant drawback occurs during the minimization step whenever the algorithm stops on local minima.
A second class of algorithms is based on the comparison between the projection of the model on the image plane and the image region occupied by the tracked object. This is the so called region-based approach, and regions are typically extracted via background subtraction (Stauffer and Grimson, 1999; Elgammal et al., 2000; Pousa et al., 2005). These regions are usually referred to as blobs (e.g. in Figure 2b) and, in this case, to estimate the vehicle pose, some algorithms minimize a metric, as for the edge-based case (Brox et al., 2005), others calculate a convenient score for a set of hypothesized model poses (Buch et al., 2009; Shi and Tomasi, 2007). The former approach aims at diminishing significantly the number of local minima compared to the edge-based method while the latter eliminates them completely; in this paper we use the second approach. The main drawback of the region-based methods is due to background subtraction not always being robust to noise while edge extraction is.
The approaches introduced so far to evaluate model likelihood assume the model pose is given for each frame. Other approaches (Basu et al., 1996; Brox et al., 2004; Munoz et al., 2005) rely on a different principle assuming that the object pose is perfectly known only when tracking begins. This initial pose is used as a model pose initialization and, at each frame, object motion is estimated iteratively on the image plane through the use of a set of 2D motion vectors. Then, these vectors are projected in 3D coordinates, and the model is virtually moved from the last position according to 3D motion vectors implied by the 2D motion vectors and a new position is estimated.
Optical-flow-based methods (Basu et al., 1996; Brox et al., 2004) estimate 2D motion vectors through optical-flow (Horn and Schunck, 1981; Lucas and Kanade, 1981). This approach shows a high sensitivity to lighting changes. In feature-based methods, as in (Munoz et al., 2005), motion vectors are computed using frame-by-frame feature correspondences, e.g. KLT (Lucas and Kanade, 1981; Shi and Tomasi, 2007; Tomasi and Kanade, 1991), SIFT (Lowe, 1999) or SURF (Bay et al., 2006). This second approach is more robust to light changes than the optical-flow based one, but it requires objects which have a textured surface, i.e., a surface which has some irregularities easily noticeable through KLT, SURF or SIFT, so that feature

matching between images becomes effective. Both approaches, and in general 2D motion based approaches, have the advantage of not relying on a pose estimation for each frame, but on the other hand they present a drifting issue, partially solved in (Brow et al., 2004): errors in vectors estimation are summed up over time so that after a long tracking period errors may become considerably high.
Combining the advantages of different approaches, some researchers proposed hybrid methods as well. In (Haag and Nagel, 1999; De Carlo and Metaxas, 2000; Ambardekar et al., 2008) hybrid solutions combining edge-based and optical-flow-based approaches are presented, e.g. (Haag and Nagel, 1999) combines global information on motion from optical-flow, and local information from edges. In (Vacchetti et al., 2004) a feature-based method works in conjunction with an edge-based one. In (Brox et al., 2010) the authors present a hybrid approach which uses optical flow, feature and region-based approach simultaneously.
In this paper we propose two model-based 3D tracking algorithms and, following the previous classification their approach is region-based; in particular they use blobs extracted by background subtraction and the trajectory extracted on the image plane to implement a Montecarlo based Bayesian smoother to recover the 3D trajectory of the tracked vehicles.

2.2 Bayesian tracking and smoothing

Most of the previous algorithm approach tracking such as a Bayesian filtering problem achieving a robust estimation through well-known iterative estimation tools such as Kalman Filters or Particle Filters. In the following we provide a brief discussion of those techniques which have been used extensively in the paper.
Let s_t be the vehicle state and z_t some measurement at a given time t. s_t components are, for instance, the position of the vehicle on the image plane or its 3D position on the road plane and, optionally, velocity and orientation. A measurement z_t for the vehicle could be blob centroid or the blob itself. A tracking system looks for a convenient succession of vehicle states s_1,..., s_t..., s_T, which represents the vehicle trajectory. A Bayesian tracker is an algorithm which estimates this succession by using the a-posteriori probability through Bayesian statistic, i.e., the estimation tool used relies on an equation derived from the Bayes' theorem:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \quad (1)$$

where A and B are two generic unknowns.
According to the Bayesian filtering framework (Fox et al., 2003), tracking aims at the maximization 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 $z_{1:t+1}=\{z_1;\dots ; z_t; z_{t+1}\}$. Usually the Markov hypothesis is assumed, i.e., a state depends only from the previous state and the applied control, and this makes it possible to derive relatively simple tracking equations.
The Markov assumption in the vehicular case is quite reasonable. If we know the previous state of a vehicle, that is, its position orientation and velocity at time *t-1*, we can have a relatively good estimate of the vehicle state at time *t* propagating the state through a constant velocity vehicle motion model, without any need for considering the past system states. Then, to correct this estimate of the vehicle state we only need the current measurement. 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 define the so called *prediction equation*:

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

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 state s_t. Then, this equation is updated with the current measurement through the Bayes' theorem, in this way:

$$\begin{aligned} Bel(s_{t+1}) &= \alpha_{t+1}\, p(z_{t+1}|s_{t+1})\, p(s_{t+1}|s_1, z_{1:t}) = \\ &= \alpha_{t+1}\, p(z_{t+1}|s_{t+1})\; Bel^{-}(s_{t+1}) \end{aligned} \tag{3}$$

where $p(z_{t+1}|s_{t+1})$ is the likelihood of observation z_{t+1} in state s_{t+1} and $\alpha_{t+1} = p(z_{t+1}|z_{1:t})^{-1}$ is a normalization constant.

It is worth mentioning that the previous approach allows an iterative computation of the a-posteriori probability for 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 states in the trajectory, the entire state sequence $s_{1:T}$ is estimated using all the measurements. In the case of vehicle tracking, the vehicle trajectory is estimated 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:

$$p(s_{1:T}|z_{1:T}) = p(z_{1:T}|s_{1:T})\; p(s_{1:T}). \tag{4}$$

In the general case, i.e., both for filtering and smoothing, Bayesian tracking algorithms estimate the a-posteriori probability assuming different probability distributions. The easiest way is to assume that all probability distributions are Gaussian. This is often a good approximation, and, if the tracked system has a linear dynamic and a linear measurement we obtain the Kalman filter formulas. For non linear systems, either because of measurement or because of the motion model, we can use the Extended Kalman Filter (EKF) which provides an estimation like the Kalman filter by linearizing the system. Under linearity and Gaussian noise, the Kalman Filter provides an optimal solution efficiently. A smoothed version of the Kalman Filter also exists and is called a 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 posterior probability as a parametric distribution, but it estimates it using a set of samples. More formally, let π be a generic distribution, and $X_i \sim \pi$ an independent samples with $i = 1, ..., N$. If δ_{X_i} is Dirac function centered in X_i, $\hat{\pi}$ such that:

$$\hat{\pi} = \frac{1}{N} \sum_{1}^{N} \delta_{X_i} \tag{5}$$

is an approximation of π. In Figure 3a a set of samples representing the dotted-line Gaussian distribution is shown; sample concentration increases as probability increases.

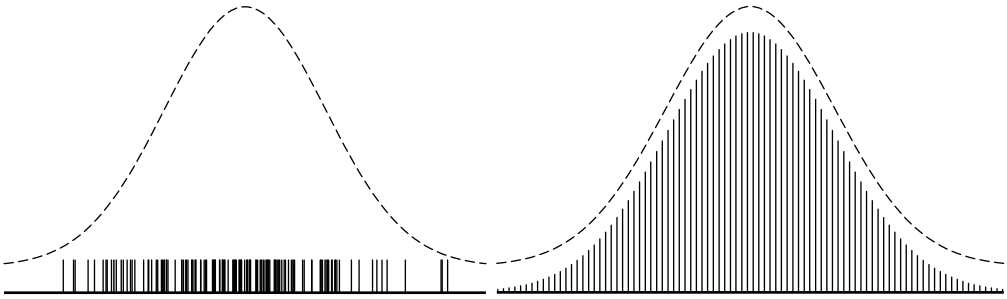

Figure 3: Two sample-based approaches to represent a function. The dotted line represents the Gaussian distribution to be sampled. In (a) a group of samples are proportionally distributed according to their density; in (b) the samples are uniformly distributed but they are weighted in such a way that they represent the dotted Gaussian distribution

One widespread tool that uses a sample-based approach is the Particle Filter. Starting from a given initialization, the Particle Filter uses a probabilistic state transition model to generate an estimate of the current state distribution. Then it calculates, for each sample state, the likelihood of the current measurement obtaining a weight for each sample. The samples, each of them with its own weight, represent the current posterior probability distribution as exemplified in Figure 3b.
Using this approach, i.e., approximating a state distribution with a set of samples, it is possible to compute measurement likelihood in a straightforward way. Indeed for each sample it is sufficient to compute a score and this can easily be done as we propose in the blob case in Section 3.3. Computing a proper measurement function as in the case of a Kalman Filter would have not been so straightforward.
In the next Section 3 two tracking algorithms to implement trajectory smoothing using a sample based approach are described. The first algorithm computes the most likely sequence of states for the given trajectory using the Viterbi algorithm (Forney, 1973) and it implements in an exact way equation (4); the second approach system is based on the use of two Particle Filters each using equation (3).

3. TWO METHOD FOR SAMPLE BASED 3D SMOOTHING

As opposed to the typical 3D tracking algorithm, the proposed methods do not directly process images, but they use 2D vehicles trajectories extracted from the 2D tracking algorithm presented in (Mussone et al., 2011). The 2D tracking algorithm recognizes vehicles by background subtraction (see Migliore et al., 2006), 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. Resulting trajectories are made up of the history of blob centroids on the image sequence and this is the input for our methods. In Figure 4 an example of 2D trajectory extracted from the image plane is shown.
As we have already mentioned , the straightforward way to estimate vehicle trajectory in the 3D world from a 2D trajectory on the image plane is to back-project each blob centroid position from the image plane to the roundabout plane as in Figure 2. It is clear that the back-projected point, C_{err}, gives a poor estimate of the real vehicle center, C. This error is affected by the vehicle height and the camera perspective, thus a robust way to overcome this problem needs to consider the true vehicle size during the tracking process.

Despite its limitation, we decided to bootstrap the 3D smoothing from the 2D trajectory on the image plane for three reasons. First of all, the 2D tracking algorithm implements data association, i.e., tries to work out which blob belongs to which vehicle. This is a challenging task, so, starting from 2D trajectory the proposed methods should not deal with data association problems. A second reason for using 2D tracking is that a 2D trajectory on the image plane is a reasonable initialization for the proposed systems and this resolves the local minima issue which affects several 3D models based on tracking systems. Finally, using a 2D tracking algorithm output allows us to know the entire trajectory of each vehicle, so we can adopt a smoothing approach to tracking instead of the classical filtering. By using further information appropriately we can make a more accurate estimate.

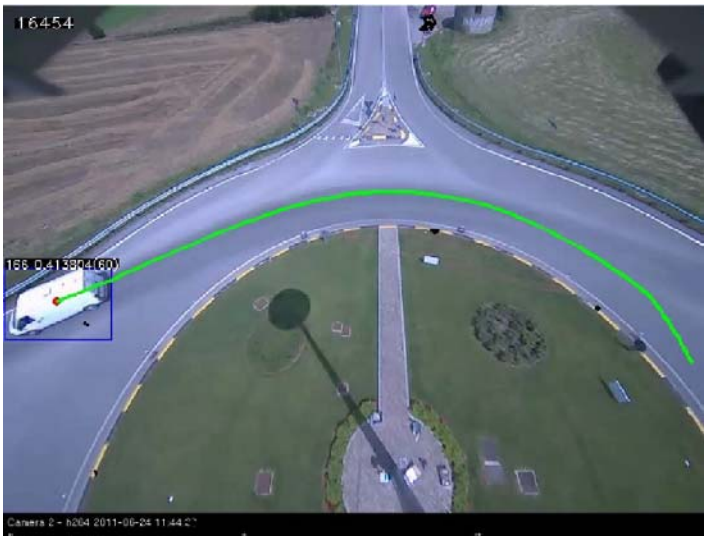

Figure 4: 2D vehicle trajectory extracted on the image plane using the algorithm in (Mussone et al., 2011).

The proposed algorithms use a vehicle model so they have an estimate of the vehicle height; however we need to specify a model (or several models) for the vehicles. Models could be of different shapes or dimensions depending on the level of detail we want to reach. Taking into account different shapes of existing vehicles, we decided to approximate all of them through a simple parallelepiped, but with different dimensions according to the vehicle class. In particular, a set of parallelepipeds with variable dimensions is used. In fact, since vehicle dimensions can vary radically, a model with fixed dimension can cause problems as described in (Song and Nevatia, 2007). To define the model dimensions three main classes of vehicles have been identified: cars, trucks and motorcycles. For each class a reference parallelepiped is defined, setting a reference length value L_c and two ratio values $r_h^c = L_c / L_h$ and $r_d^c = L_c / L_d$, respectively for height and width calculation.

For each pose hypothesis, a set of sample values of model length is extracted from a Gaussian distribution centred on the reference length L_c of each class, these are called class-scale samples. From each of these samples we infer the model depth and height using r_h^c and r_d^c values. By doing this a set of models for each class is created. A different standard deviation for each class is used for sample extraction since car dimensions vary less than truck dimensions, and more than motorcycle ones. Figure 5 shows how the parallelepiped

model should be projected on to vehicle image: the model must ideally wrap the tracked vehicle.

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

Figure 5: Example of expected result when the correct parallelepiped model is projected on to the image plane.

In the following sections the two 3D tracking algorithms proposed based on sampling and smoothing are described: the Viterbi-based algorithm and the Particle Smoother. Since the likelihood of measurements is computed in the same way for both algorithms, it is explained as follows. Considering that vehicles in the roundabout (should) always proceed forward, we implemented a transition state model 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 biasing the state transition in the forward direction. Since this wasn't within the specific scope of the paper, we suggest that interested readers should go into more depth regarding this choice and how to calculate the transition state model, $p(s_{t+1}|s_t)$, in (Romanoni, 2012).

3.1 The Viterbi-Based Algorithm

For each 2D vehicle trajectory and for each class-scale sample, similarly to what is done in (Song and Nevatia, 2007), the Viterbi-Based Algorithm (VBA) performs two steps: the frame-by-frame sampling and the estimate of the most likely trajectory using the Viterbi algorithm (Forney, 1973).

In the frame-by-frame sampling step, for each frame, 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 $(x_t; y_t, \theta_t)$ and a diagonal covariance structure reflecting the independence of the three components (independence is used 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 Figure 6. Each of these samples represents a vehicle state, i.e., vehicle position and orientation on the roundabout plane at time t. This step relies entirely on the 2D trajectory estimated by the 2D tracker and on the samples generated from the class scale sampling.

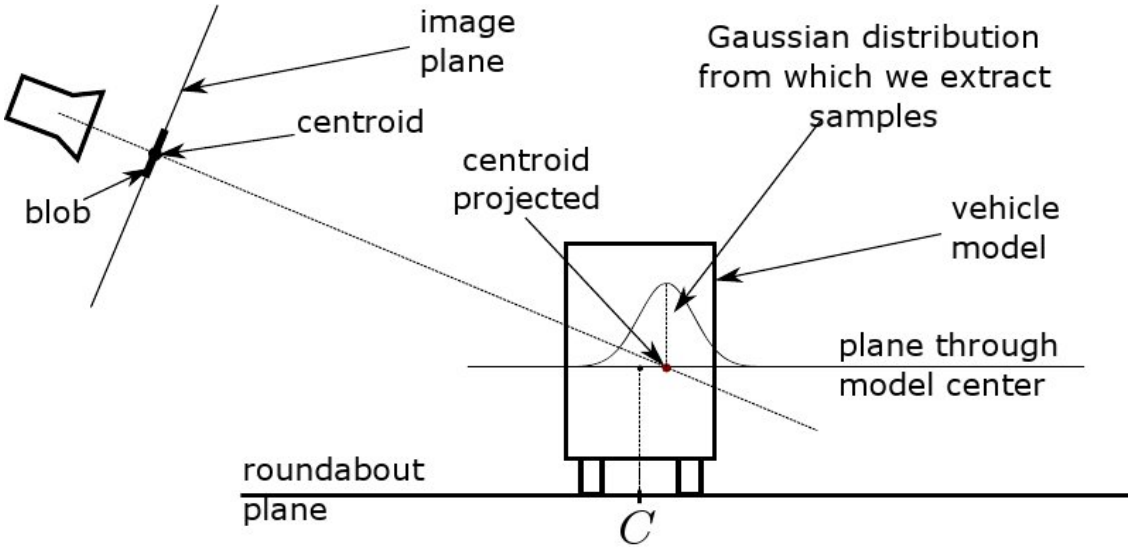

Figure 6: Example of 2D vehicle center projected on the plane passing through model center.

After extracting the samples for each frame and for each class-scale model, the algorithm chooses the most likely class-scale sample that defines model dimensions. This is chosen at this time in the algorithm since we want to find the most likely trajectory keeping fixed the class-scale. In principle, the selection of the most likely trajectory could be done for each class-scale sample, but this would require to repeat the Viterbi search multiple times (once for each class-scale sample) affecting the performance significantly and resulting in a mirror (if any) improvement.

To decide which class-scale should be used, we compute the likelihood of all vehicle state samples and, for each frame, the mean of the sample likelihood for each class-scale is computed; the class-scale sample which gives the highest sum of mean probabilities is chosen (an alternative choice could have been selecting the class-scale which has, in the highest number of frames, the highest mean likelihood). Only vehicle state samples corresponding to the class-scale sample chosen are used in the next step according to the Viterbi algorithm.

The Viterbi algorithm is used to extract the most likely sequence of samples along the trajectory. Let s_t be a state sample extracted at time $1 \leq t \leq T$. The Viterbi algorithm finds the succession of samples which maximizes equation (4) in this way:

$$\begin{aligned} \tilde{s}_{1:T} &= \arg\max_{s_{1:T}} \left\{ p\left(s_{1:T} \middle| z_{1:T}\right) \right\} = \arg\max_{s_{1:T}} \left\{ p\left(z_{1:T} \middle| s_{1:T}\right) p\left(s_{1:T}\right) \right\} = \\ &= \arg\max_{s_{1:T}} \left\{ \prod_{t=1}^{T} p\left(s_{t+1} \middle| s_t\right) p\left(z_{t+1} \middle| s_{t+1}, s_t\right) \right\} \end{aligned} \tag{6}$$

and, according to the Markov hypothesis :

$$\begin{aligned} \tilde{s}_{1:T} &= \arg\max_{s_{1:T}} \left\{ \prod_{t=1}^{T} p\left(s_{t+1} \middle| s_t\right) p\left(z_{t+1} \middle| s_{t+1}, s_t\right) \right\} = \\ &= \arg\min_{s_{1:T}} \left\{ \sum_{1}^{T} \left\{ -\log\left(p\left(s_{t+1} \middle| s_t\right)\right) - \log\left(p\left(z_{t+1} \middle| s_{t+1}, s_t\right)\right) \right\} \right\} \end{aligned} \tag{7}$$

In practical implementations of the Viterbi algorithm, states/samples are represented in a graph as in Figure 7: 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 Equation (6), i.e., $-\log(p(s_{t+1}|s_t))-\log(p(z_{t+1}|s_{t+1}, s_t))$.

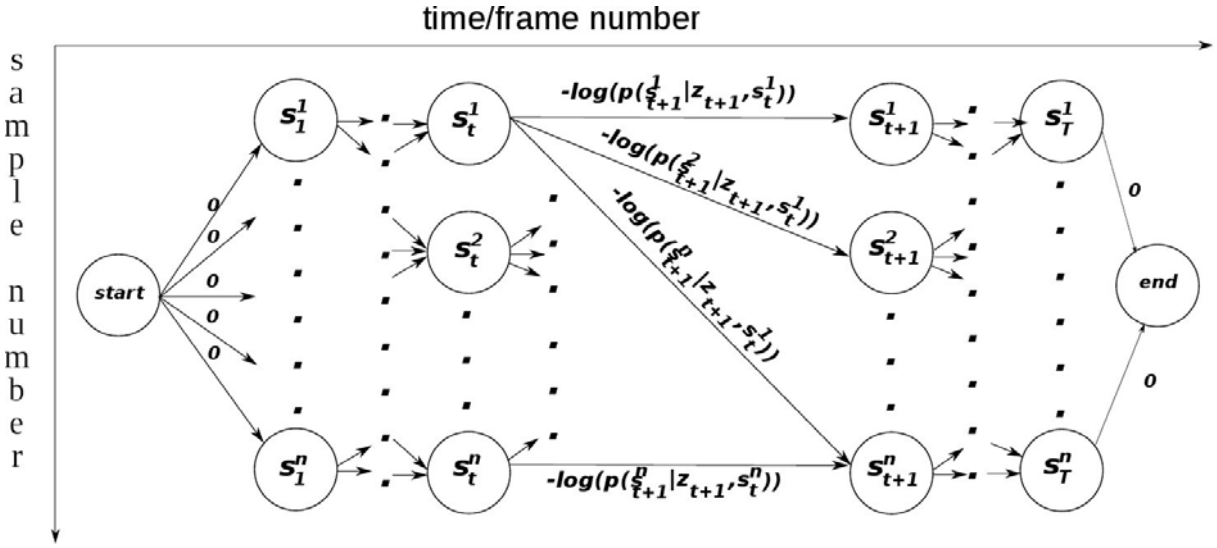

Figure 7: Graph built to apply Viterbi algorithm.

Therefore, to find the best path according to Equation (6), we look for the shortest path from a fictitious starting node, connected to samples at time *1*, to a fictitious node end, connected with each sample at time *T*. The nodes in the graph correspond to the vehicle state samples extracted during the previous step, and a Dijkstra algorithm (Dijkstra, 1959) is used to find the shortest path in the graph: for each considered arc two terms must be calculated: the logarithm of the likelihood of the arc-ending state and the state transition from the arc-beginning state to the arc-ending state. The shortest path extracted contains a state for each time *t*. In this way the succession of vehicle states which represent the most likely trajectory on the roundabout plane is obtained.

3.2 The Particle Smoother algorithm

The second algorithm proposed is a Particle Smoother which relies on two Particle Filter trackers. The Particle Filter is a tool which estimates the state distribution at time *t* using a sample-based approach, and implementing Bayesian tracking as in Equation (3).
At each iteration *t*, the Particle Filter estimates $p(s_{t+1} \mid z_{1:t+1})$ with a sampled distribution; at each time, a set of random state samples, called particles, represents $p\left(s_{1:t+1} \middle| z_{1:t+1}\right)$ as:

$$p\left(s_{1:t+1} \middle| z_{1:t+1}\right) \approx \sum_{i=1}^{N_s} w_{t+1}^{i}\ \delta\left(s_{1:t+1} - s_{1:t+1}^{i}\right), \tag{8}$$

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

$$w_{t+1}^{i} \propto w_{t}^{i} \frac{p\left(z_{t+1} \middle| s_{t+1}^{i}\right) p\left(s_{t+1}^{i} \middle| s_{t}^{i}\right)}{p\left(s_{t+1}^{i} \middle| s_{t}^{i}, z_{t+1} \middle|\right)}. \tag{9}$$

The main problem of Particle Filters using this approach, is due to particle degeneracy. Particles tend to collapse around a single state value. To solve this issue, usually, a particle filter uses a resampling stage. At each iteration, after new weights computation, the resampling algorithm resamples a new set of particles according to current approximated probability distribution of s_{t+1}. This new set replaces old particles, and each particle is

assigned the same weight. More details about Particle Filter are available in (Arulampalam et al., 2002).

In our work, particles represent the vehicle state at time *t*, i.e., vehicle position, orientation and velocity on the roundabout plane. We implement a Particle Filter for each model defined by each class-scale sample (see the algorithm in Figure 8). These Particle Filters are initialized through a set of particles extracted from the first frame of the 2D trajectory the same way we did in the Viterbi algorithm. Then we propagate the sample and use the standard Particle Filter machinery. After this forward pass of the vehicle trajectory, the algorithm chooses which class-scale sample is the best to represent the tracked vehicle, i.e., for each time *t* it calculates 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.

At this point, starting from the last group of particles, a second Particle Filter tracks the vehicle backward, using only the model whose dimensions have been calculated from the class-scale sample previously chosen. Taking into consideration the theoretic result about which particles tends to wrap around the real state, the second recursion is the aimed at diminishing particles variance and makes the estimate more accurate for the states in the first part of the trajectory. This backward recursion acts as a smoother although it does not properly optimize the integral of equation (4). Note that, in this algorithm, the 2D trajectory is used only for filter initialization and for data association.

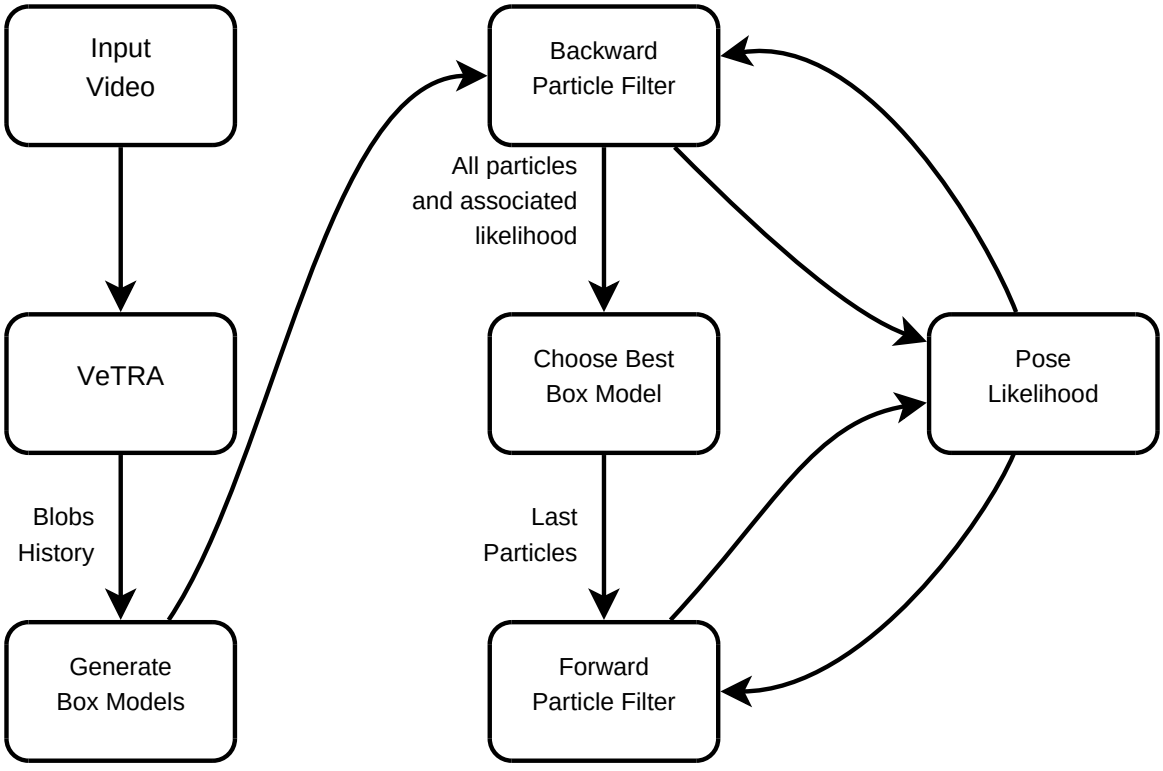

Figure 8: Flow chart for the Particles Smoother algorithm.

3.3 Likelihood calculation

In both the methods that we propose, a key step is the computation of the likelihood of samples. Let focus on the measures be 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:

1. Project on the image plane the model located in the roundabout plane according to s_t (the red polygon in Figure 9).

2. Compute blob area (A_{blob}), visible model projection area (A_{mv}), and overlap area between the blob and the model projection ($A_{overlap}$).

3. Compute the two errors e_1 and e_2 (Figure 9), as:

$$e_1 = A_{blob} - A_{overlap} \tag{10}$$

$$e_2 = A_{mv} - A_{overlap}: \tag{11}$$

4. Define the score term:

$$e = \frac{\lambda_1 e_1 + \lambda_2 e_2}{A_{blob}} \tag{12}$$

where λ_1 and λ_2 are weight for e_1 and e_2 such that $\lambda_1 + \lambda_2 = 1$.

Likelihood is then defined as:

$$p(z_t | s_t) = \begin{cases} 1 - e, & if\ 1 - e > 0 \\ 0, & if\ 1 - e \leq 0. \end{cases} \tag{13}$$

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. Figure 10 shows the difference between A_{mv} and the area of the entire projected model, A_m.

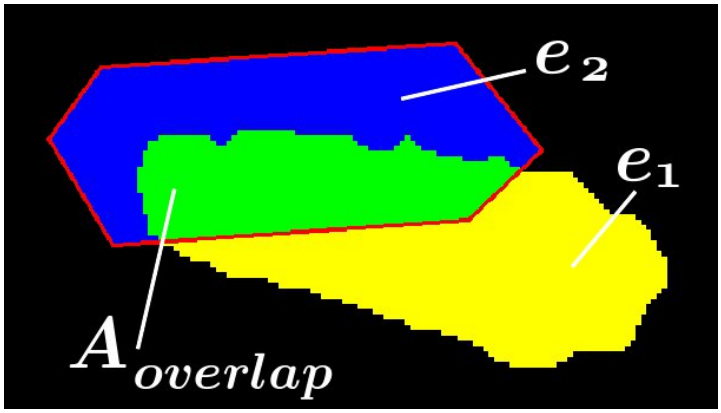

Figure 9: Example of likelihood calculation: e_1 measures how much of the predicted model is not reflected by the blob, e_2 measures how much of the blob is not captured by the projected model.

Figure 10: Example showing difference between A_{mv} and A_m.

In a multi-camera setting, a vehicle could be seen by more than one camera. Likelihood calculation is the only one stage of our algorithms, in which we have to consider the presence of more than one camera since measurements are blobs from the same vehicle generated on the image planes of different cameras.

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

$$p(z_t|s_t) = \frac{1}{p(C_1)\dots p(C_{n_c})}\left(p(C_1)p_1(z_t^1|s_t) + \dots + p(C_{n_c})p_{n_c}(z_t^{n_c}|s_t)\right) \qquad (14)$$

The camera probability $p(C_j) = A_{mv}^j / A_m^j$ is designed taking into account different situations. In the case of only two cameras, they have the same probability equal to 0.5; when both cameras see the whole vehicle, or more precisely the model located according to s_t. When a camera does not see the model, its probability is nil l. In the case of a vehicle which is only partially observed by the cameras, the probability is proportional to observed model percentage.

4. EXPERIMENTAL RESULTS

As stated in Section 3, the aim of 3D tracking is to provide precise estimates of vehicle positions. In particular, it should provide a better estimate compared to that obtained back-projecting a 2D trajectory on the roundabout plane. The two algorithms we have presented (the Viterbi Based Algorithm and the Particle Smoother) have been thus tested in two scenarios where the ground is known: a simulated scenario and a real one.

In the simulated case we created a movie where a parallelepiped model virtually drives through a hypothetical roundabout. This is an ideal case because blobs representing the vehicle coincide exactly with the parallelepiped used by the tracking. Errors due to noise in image processing and 2D tracking are much reduced and have almost no relevance. Proposed algorithms errors are then evaluated comparing the poses (here synonymous with vehicle states) estimated by the trackers with the poses from which the movie was created. Results obtained in this way give an idea of the maximum t performance our methods can attain.

Expected errors are mainly due to reduced sample size, likelihood definition/computation, and approximations with respect to the optimal smoothing in the case of the Particle Smoother.
In the real case vehicles circulating in a roundabout are taped by two cameras but vehicle positions are not known a priori. Therefore, a vehicle equipped with an RTK-GPS device and an inertial sensor is used to collect a precise estimate of vehicle position and orientation. In this way, we are able to evaluate the performance of the algorithms by comparing 3D tracking results for the equipped vehicle and the data obtained from the two devices, both in the single and in the multiple camera cases (in fact two cameras with overlapping fields were used in the field survey).
The comparison between poses requires, to obtain a perfect comparison, that the two poses have been taken at the same time. However, synchronization between RTK-GPS devices, inertial sensors and video cameras is not perfect. In addition, the sensors and the cameras working frequencies may be different (as occurred in the experiment), so that the estimate provided by sensors and our algorithm cannot be considered perfectly synchronous. In the multi-camera case this imperfect synchronization between the two considered video cameras could be another source of error. Just to give an idea about the reason for this, consider that the first camera captures a vehicle at time t, but the corresponding frame for the second camera is captured at time $t + \varepsilon$; though ε is a low value, this introduces an error. As a matter of facts, the blobs from the two cameras are not captured precisely at the same time although they concur to compute the likelihood for the same model.
Table 1 reports tracking errors for the 3D pose reconstructed back-projecting the 2D tracking result, the output of the Viterbi Based Algorithm (VBA) and the result of the Particle Smoother (PS) with one camera and in a multi-camera set up. Errors are calculated for x and y coordinates of vehicle pose; we report also the error module $d = |x + y|$, 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 $\boldsymbol{x}$ vector $MAD = median\left(\left|x_i - median(\mathrm{x})\right|_{\forall i}\right)$, and IQR is the Inter Quartile Range, $IRQ = Q_{\frac{3}{4}} - Q_{\frac{1}{4}}$ where $Q_{\frac{1}{4}}$ is the first quartile and $Q_{\frac{3}{4}}$ is the 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 they tend to compensate each other being their sum for z a little lower; only a few degrees for the orientation θ of the error have been observed. In the simulated case the two algorithms reach a very satisfying accuracy, consistent with the aims of many types of analyses on vehicle trajectory.
As expected, we confirm with the numbers in Table 1 that the estimates of the proposed algorithms for 3D tracking are significantly more precise than those obtained back-projecting the 2D trajectory (2D rows). Particle Smoother gives generally better results than VBA; this can be interpret by the fact that VBA extracts state samples independently from previous or next state at each time while Particle Smoother generates a set of particles-samples starting from previous particles through the state transition model and this increases its accuracy. Using more samples for the VBA algorithm could probably reduce this discrepancy from the expected result. Also without synchronization errors between cameras, in multi-camera case Particle Smoother should have always better performances than VBA.

Table 1: Tracking errors of vehicle pose by using Simulated and real scenarios.

		Median	MAD	IQR
x [m]	***Simulated Scenario***			
	VBA	0.036	0.042	0.090
	PS	−0.027	0.022	0.050
	Real Scenario			
	2D tracking	0.292	0.338	**0.348**
	VBA Single Camera	−0.180	0.236	0.468
	PS Single Camera	−0.171	**0.171**	**0.348**
	VBA Multicamera	0.154	0.305	0.512
	PS Multicamera	**−0.056**	0.261	0.475
y [m]	***Simulated Scenario***			
	VBA	−0.039	0.055	0.113
	PS	0.004	0.023	0.046
	Real Scenario			
	2D tracking	−0.317	0.284	0.575
	VBA Single Camera	−0.136	0.316	0.500
	PS Single Camera	0.058	0.135	0.277
	VBA Multicamera	**−0.041**	0.241	0.384
	PS Multicamera	0.094	**0.179**	**0.248**
d [m]	***Simulated Scenario***			
	VBA	0.090	0.054	0.059
	PS	0.046	0.040	0.056
	Real Scenario			
	2D tracking	n.a.	n.a.	n.a.
	VBA Single Camera	0.483	0.215	0.392
	PS Single Camera	**0.288**	**0.116**	0.187
	VBA Multicamera	0.343	0.215	0.288
	PS Multicamera	0.237	0.191	**0.125**
θ [deg]	***Simulated Scenario***			
	VBA	−4.137	1.281	2.478
	PS	0.803	1.176	2.459
	Real Scenario			
	2D tracking	n.a.	n.a.	n.a.
	VBA Single Camera	−3.664	5.123	10.227
	PS Single Camera	4.156	**2.446**	**4.875**
	VBA Multicamera	−3.759	3.845	7.718
	PS Multicamera	**2.401**	7.629	8.465

From a theoretical perspective, VBA is computationally less efficient than Particle Smoother, due to the shortest path research which has quadratic complexity in the number of frames; conversely, Particle Smoother has a linear complexity because it uses two Particle Filters which are, in turn, linear. Also at practical level Particle Smoother is more efficient than VBA. In fact, execution times reported in Table 2, as function of the number of frames used to describe the trajectory, show values three times shorter for Particle Smoother than VBA. Whereas Particle Smoother uses 250 state samples for each frame, VBA needs 500 samples to reach comparable results. Then, both algorithms use a set of 30 class-scale samples, that is,

ten 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 development environment, which offers high flexibility and ease of implementation but at the cost of a lack of efficiency. It should be noticed that most of the computation could be performed in a parallel manner on modern computer architectures.
It must be underline that the aim of experimental setups was mainly to test the accuracy of the tracking of vehicles reached by the proposed algorithm, and, therefore, we focused on the tracking of vehicles to assess this accuracy and we did not investigate the accuracy in the estimate of vehicle dimensions.
However, the two estimates are strictly correlated. In both simulated and real scenarios the parallelepiped dimensions can be handled in quite an accurate way. However, it must be stressed that we tested the algorithm in real cases only with a light vehicle, equipped with an RTK-GPS system. No quantitative and numeric estimation of algorithm performance in the specific case of motorcycles or trucks can be given but we can add some qualitative considerations based on image analysis, i.e., the model projections against the recorded image.
Results obtained for motorcycles are generally accurate. In fact, even if the blob of a motorcycle is small, and it is more affected by noise, the parallelepipeds chosen by the algorithm for tracking are usually very similar each other (the variance of its dimensions is small); consequently, the pose estimation is accurate. Being the class-scale samples extracted from a Gaussian with a small variance, it is more likely to get the real dimension of the currently tracked motorcycle. Dimension estimation for heavy vehicles is more difficult, since their dimensions, in particular length, have a large variance as a consequence of a wide range of heavy vehicles. A light truck is around 6-7 meters long, while an articulated lorry may reach 16.5 meters. So it is difficult to extract the exact dimensions during the parallelepiped dimension choice. Moreover, the shape of articulated lorries has a degree of freedom on a curve which cannot be accurately modeled by a (rigid) parallelepiped.

Table 2: Execution times (in seconds).

	Number of frames			
	45	*106*	*230*	*442*
Viterbi Algorithm (VBA)	255	1456	3642	6598
Particle Smoother Algorithm (PS)	87	624	1141	2817

CONCLUSIONS

We presented two algorithms to implement a 3D tracking system using a model-based / region-based approach. As opposed to widespread methods which track objects directly from images, a 2D trajectory estimate on the image plane, extracted by an existing 2D tracking algorithm, is used. The use of this first estimation brings some benefits to the algorithms with no loss of generality. If we aim at removing the 2D tracking phase, we should implement a proper data association mechanism. Moreover, we have a meaningful initialization for our tracking systems and, instead of the usual filtering algorithms, the availability of the entire trajectory makes it possible to use a smoothing approach which produces better and smoother estimates.

The two proposed algorithms (the Viterbi-based and the Particle Smoother ones) use a sample-based approach to model vehicle state distribution, so we can evaluate observation likelihood in a more straightforward way compared to the calculations needed for Kalman approaches. Both were implemented for single camera and multi-camera cases. They were tested in simulated and in real scenarios, and experimental results show better performances than the 2D tracking initialization algorithm; the Particle Smoother is the best both in terms of speed and accuracy. Future research will concern the independence of these algorithms from the 2D tracking implementation.
To reach this independence, we need to implement a data association system embedded in our 3D tracking algorithm. Then we need to have an initialization module independent from the trajectory estimated on the image plane from the 2D algorithm. Considering this last observation, we should remember that VBA grounds the sampling at each frame on the 2D initialization, while Particle Smoother uses only the first point of the 2D trajectory to initialize the first filter. We expect that by making Particle Smoother independent it will be easier to implement than VBA.
Using a model-based approach will allow shadow estimation and occlusion management too. In fact, shadows could be estimated by projecting model shadows on a roundabout plane; occlusion could be managed by avoiding models in incoherent positions, e.g., models intersecting each other.

REFERENCES

Ambardekar, A. A., M. Nicolescu, and G. Bebis, G. (2008). Efficient Vehicle Tracking and Classification for an Automated Traffic Surveillance System. In: Signal and Image Processing. ACTA Press, Calgary, AB, Canada.

Arulampalam, M. S., and G. Maskell. (2002). A tutorial on particle filters for online nonlinear non-Gaussian Bayesian tracking. In: IEEE Transactions on Signal Processing. 50.2, pp. 174-188. IEEE Computer Society Press, Washington. doi: 10.1109/78.978374.

Basu, S., I. Essa, and Pentland, A. (1996). Motion regularization for model-based head tracking. In: Proceedings of the 13th International Conference on Pattern Recognition. Vol. 3. , pp. 611-616. IEEE Computer Society Press, Washington. doi: 10.1109/ICPR.1996.547019.

Bay, H., T. Tuytelaars, and L: Van Gool (2006). SURF: Speeded Up Robust Features. In: Computer Vision ECCV 2006. Vol. 3951, pp. 404-417. Springer, Berlin / Heidelberg. doi: 10.1007/11744023ffi32.14.

Brox T., Bruhn, A., Papenberg, N., Weickert, J. (2004). High Accuracy Optical Flow Estimation Based on a Theory for Warping. In: Computer Vision - ECCV 2004. Vol. 3024, pp. 25-36. Springer, Berlin / Heidelberg. doi: 10.1007/978-3-540-24673-2ffi3.

Brox, T., B. Rosenhahn and J. Weickert (2005). Three-Dimensional Shape Knowledge for Joint Image Segmentation and Pose Estimation. In: DAGM-Symposium. Vol. 3663, pp. 109-116. Lecture Notes in Computer Science. Springer, Berlin / Heidelberg. doi: 10.1007/11550518ffi14.

Brox, T., B. Rosenhahn, J. Gall and D. Cremers (2010). Combined Region and Motion-Based 3D Tracking of Rigid and Articulated Objects. In: IEEE Transactions on Pattern

Analysis and Machine Intelligence, 32.3, pp. 402-415. IEEE Computer Society Press, Washington. doi: 10.1109/TPAMI.2009.32.

Buch, N., F. Yin, J. Orwell, D. Makris and S. Velastin, S. (2009). Urban Vehicle Tracking Using a Combined 3D Model Detector and Classifier. In: Knowledge-Based and Intelligent Information and Engineering Systems. Vol. 5711, pp. 169-176. Springer Berlin / Heidelberg, 2009. doi:10.1007/978-3-642-04595-0ffi21.

DeCarlo, D. and D. Metaxas (2000). Optical Flow Constraints on Deformable Models with Applications to Face Tracking. International Journal of Computer Vision, Vol. 38. Issue. 2, pp. 99-127. Springer Berlin / Heidelberg. doi: 10.1023/A:1008122917811.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik. Springer Berlin, Heidelberg. Vol. 1. pp. 269-271. doi: 10.1007/BF01386390

Elgammal, A., D. Harwood and L. Davis (2000). Non-parametric Model for Background Subtraction. In: Computer Vision, ECCV 2000. Vol. 1843, pp. 751-767. Springer, Berlin / Heidelberg. doi: 10.1007/3-540-45053-Xffi48.

Forney Jr., G.D. (1973). The Viterbi algorithm. In: Proceedings of the IEEE, Vol. 61, No. 3, pp. 268-278. IEEE Computer Society Press, Washington. doi: 10.1109/PROC.1973.9030.

Fox D., J. Hightower, Lin Liao, D. Schulz and G. Borriello (2003). Bayesian filters for location estimation. In: IEEE Pervasive Computing, Vol. 2, Issue 3, pp. 24-33.

Haag, M. and H.H. Nagel (1999). Combination of Edge Element and Optical Flow Estimates for 3D-Model-Based Vehicle Tracking in Traffic Image Sequences. International Journal of Computer Vision. Vol. 35. No.3, pp. 295-319. doi: 10.1023/A: 1008112528134.

Harris, C. and C. Stennet (1990). RAPiD - A video-rate object tracker. In: British Machine Vision Conference. Oxford Brookes University Press, Oxford. pp. 73-77.

Hartley, R. and A. Zisserman (2004). Multiple View Geometry in Computer Vision. Second Edition, Cambridge University Press, Cambridge, ISBN: 0521540518.

Horn, B. K.P. and B.G. Schunck (1981). Determining optical flow. Artificial Intelligence, Vol. 17. Issues 1-3. pp. 185-203. Elsevier, Amsterdam. doi: 10.1016/0004-3702(81)90024-2.

Lepetit, V. and P. Fua (2005). Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. In: Foundations and Trends in Computer Graphics and Vision, pp. 1-89. Now Publishers Inc. Boston. doi: 10.1561/0600000001.

Lou, J., T. Tan, W. Hu, H. Yang and S.J. Maybank (2005). 3-D model-based vehicle tracking. In: Image Processing, IEEE Transactions on Vol. 14, No. 10, pp. 1561-1569. IEEE Computer Society Press, Washington. doi: 10.1109/TIP.2005.854495.

Lowe, D.G. (1999). Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision. Vol. 2, pp. 1150-1157. IEEE Computer Society Press, Washington. IEEE Computer Society Press, Washington. doi: 10.1109/ICCV.1999.790410. 15

Lucas, B. D. and T. Kanade (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In: Proceedings of Imaging Understanding Workshop, pp. 121-130.

Migliore, D. A., M. Matteucci and M. Naccari (2006). 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. ACM, New York, NY, USA. pp. 215-218. doi: 10.1145/1178782.1178815.

Munoz, E., J.M. Buenaposada and L: Baumela (2005). Efficient model-based 3D tracking of deformable objects. In: 10th IEEE International Conference on Computer Vision. Vol. 1, pp. 877-882. IEEE Computer Society Press, Washington. doi: 10.1109/ICCV.2005.84.

Mussone, L., M. Matteucci, M. Bassani and D. Rizzi (2011a). Traffic Analysis in Roundabout Intersections by Image Processing. In: Proceedings of the 18th IFAC World Congress. Vol. 18. Elsevier, doi: 10.3182/ 20110828-6-IT-1002.02626, ISBN: 978-3-902661-93-7

Mussone L., Matteucci M., M. Bassani, D. Rizzi (2011b). An innovative method for the analysis of vehicle movements in roundabouts based on image processing, J. Adv. Transp., DOI: 10.1002/atr.184.

Ponsa, D., A. Lopez, J.Serrat, F. Lumbreras and T. Graf (2005). Multiple vehicle 3D tracking using an unscented Kalman. In: Proceedings of Intelligent Transportation Systems, pp. 1108-1113. IEEE Computer Society Press, Washington. doi: 10.1109/ITSC.2005.1520206.

Romanoni. A. (2012). Ricostruzione 3D delle traiettorie veicolari da immagini di una o più tele- camere". MA thesis. Politecnico di Milano.

Shi, J. and C. Tomasi (1994). Good features to track. In: Computer Vision and Pattern Recognition. Proceedings CVPR, pp. 593-600. IEEE Computer Society Press, Washington. doi: 10.1109/CVPR.1994.323794.

Song, X. and R. Nevatia (2007). Detection and Tracking of Moving Vehicles in Crowded Scenes. In: IEEE Workshop on Motion and Video Computing. WMVC '07. p. 4. IEEE Computer Society Press, Washington. doi: 10.1109/WMVC.2007.13.

Stauffier, C. and W.E.L. Grimson (1999). Adaptive background mixture models for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Vol. 2. pp.246-252. IEEE Computer Society Press, Washington. doi: 10.1109/CVPR.1999.784637.

Tomasi, C. and T. Kanade (1991). Detection and tracking of point features. Technical report, Carnegie Mellon University.

Vacchetti, L., V. Lepetit and P. Fua (2004). Combining Edge and Texture Information for Real- Time Accurate 3D Camera Tracking. In: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality. ISMAR '04. pp. 48-57. IEEE Computer Society Press, Washington. doi: 10.1109/ISMAR.2004.24.

Yang, H., J. Lou, H. Sun, W. Hu, T. Tan (2001). Efficient and robust vehicle localization. In: Proceedings International Conference on Image Processing. Vol. 2, pp. 355-358. IEEE Computer Society Press, Washington. doi: 10.1109/ICIP.2001.958501.

Zhang, Z., K. Huang, T. Tan and Y. Wang (2010). 3D Model Based Vehicle Tracking Using Gradient Based Fitness Evaluation under Particle Filter Framework. In: 20th International Conference on Pattern Recognition (ICPR). pp. 1771-1774. IEEE Computer Society Press, Washington. doi: 10.1109/ICPR.2010.437.