SCISPACE
formerly Typeset

Open access • Journal Article • DOI:10.1007/S10514-009-9115-1

Model based vehicle detection and tracking for autonomous urban driving — Source link

Anna Petrovskaya, Sebastian Thrun

Institutions: Stanford University

Published on: 01 Apr 2009 - Autonomous Robots (Springer US)

Topics: Vehicle tracking system

Related papers:

- Are we ready for autonomous driving? The KITTI vision benchmark suite
- Simultaneous Localization, Mapping and Moving Object Tracking
- Autonomous driving in urban environments: Boss and the Urban Challenge
- An algorithm for tracking multiple targets
- Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion

Share this paper:

View more about this paper here: https://typeset.io/papers/model-based-vehicle-detection-and-tracking-for-autonomous-2ek6mdlvb8

Autonomous Robots (2009) 26: 123-139
(official version online at www.springerlink.com)

Model Based Vehicle Detection and Tracking for Autonomous Urban Driving

Anna Petrovskaya · **Sebastian Thrun**

Received: 8 November 2008 / Accepted: 6 March 2009 / Published online: 1 April 2009

Abstract Situational awareness is crucial for autonomous driving in urban environments. This paper describes the moving vehicle detection and tracking module that we developed for our autonomous driving robot Junior. The robot won second place in the Urban Grand Challenge, an autonomous driving race organized by the U.S. Government in 2007. The module provides reliable detection and tracking of moving vehicles from a high-speed moving platform using laser range finders. Our approach models both dynamic and geometric properties of the tracked vehicles and estimates them using a single Bayes filter per vehicle. We present the notion of motion evidence, which allows us to overcome the low signal-to-noise ratio that arises during rapid detection of moving vehicles in noisy urban environments. Furthermore, we show how to build consistent and efficient 2D representations out of 3D range data and how to detect poorly visible black vehicles. Experimental validation includes the most challenging conditions presented at the Urban Grand Challenge as well as other urban settings.

Keywords vehicle tracking · autonomous driving · urban driving · Bayesian model · particle filter · laser range finders

1 Introduction

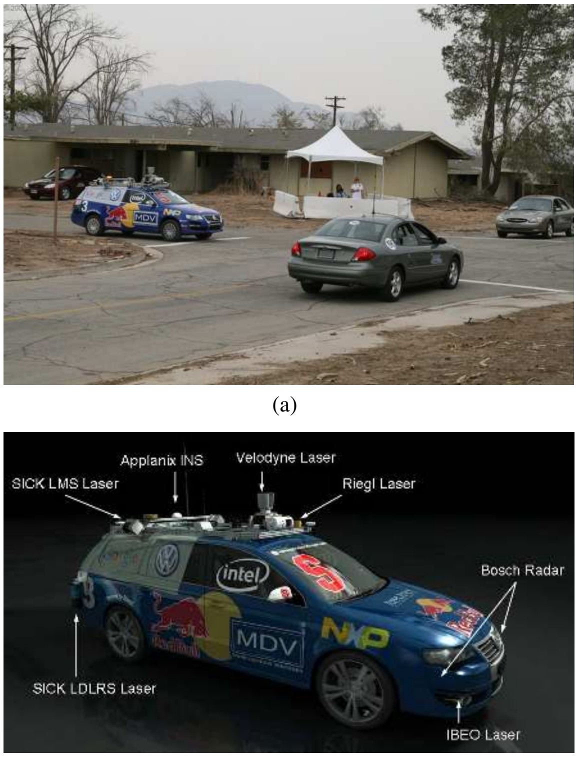
Autonomously driving cars have been a long-lasting dream of robotics researchers and enthusiasts. Self-driving cars promise to bring a number of benefits to society, including prevention of road accidents, optimal fuel usage, comfort and convenience. In recent years the Defense Advanced Research Projects Agency (DARPA) has taken a lead on encouraging research in this area and organized a series of competitions for autonomous vehicles. In 2005 autonomous vehicles were able to complete a 131 mile course in the desert (Buehler et al, 2007). In the 2007 competition, the Urban Grand Challenge (UGC), the robots were presented with an even more difficult task: autonomous safe navigation in urban environments. In this competition the robots had to drive safely with respect to other robots, human-driven vehicles and the environment. They also had to obey the rules of the road as described in the California rulebook (see DARPA (2007) for a detailed description of the rules). One of the most significant changes from the previous competition is the need for situational awareness of both static and dynamic parts of the environment. Several successful approaches have been developed in parallel by the UGC participants (Leonard et al, 2008; Urmson et al, 2008). Our robot, Junior, won second prize in the 2007 competition. An overview of Junior's software and hardware architecture is given in Montemerlo et al (2008). In this paper we describe the approach we developed for detection and tracking of moving vehicles.

Vehicle tracking has been studied for several decades. A number of approaches focused on the use of vision exclusively (Zielke et al, 1993; Dickmanns, 1998; Dellaert and Thorpe, 1998), whereas others utilized laser range finders (Zhao and Thorpe, 1998; Streller et al, 2002; Wang et al, 2007) sometimes in combination with vision (Wender and Dietmayer, 2008). We give an overview of prior art in Sect. 2.

This work was in part supported by the Defense Advanced Research Projects Agency under contract number HR0011-06-C-0145. The opinions expressed in the paper are ours and not endorsed by the U.S. Government.

A. Petrovskaya
Computer Science Department, Stanford University
E-mail: anya@cs.stanford.edu

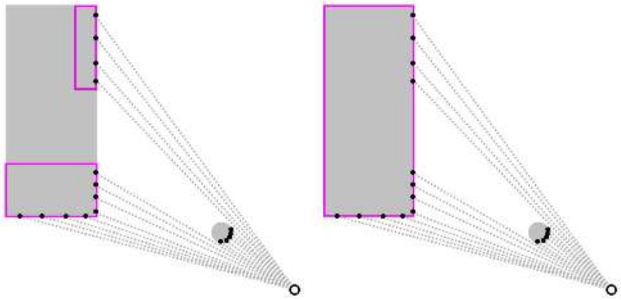
S. Thrun
Computer Science Department, Stanford University
E-mail: thrun@cs.stanford.edu



(a)

(b)

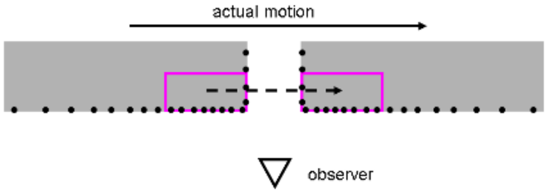
Fig. 1 (a) Our robot Junior (blue) negotiates an intersection with human-driven vehicles at the qualification event for the Urban Grand Challenge in November 2007. (b) Junior, is equipped with five different laser measurement systems, a multi-radar assembly, and a multi-signal inertial navigation system.



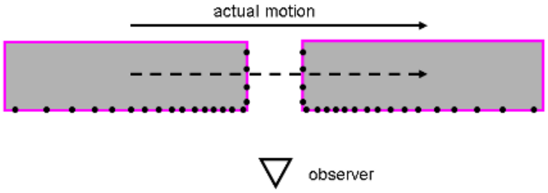
(a) without geometric model

(b) with geometric model

Fig. 2 Scans from vehicles are often split up into separate clusters by occlusion. Geometric vehicle model helps interpret the data properly. Purple rectangles group together points that have been associated together. In (b) the purple rectangle also denotes the geometric vehicle model. Gray areas are objects. Gray dotted lines represent laser rays. Black dots denote laser data points. (Best viewed in color.)



(a) without shape estimation



(b) with shape estimation

Fig. 3 Vehicles come in different sizes. Accurate estimation of geometric shape helps obtain a more precise estimate of the vehicle dynamics. Solid arrows show the actual distance the vehicle moved. Dashed arrows show the estimated motion. Purple rectangles denote the geometric vehicle models. Black dots denote laser data points. (Best viewed in color.)

For our application we are concerned with laser based vehicle tracking from the autonomous robotic platform Junior, to which we will also refer as the ego-vehicle (see Fig. 1). In contrast to prior art, we propose a model based approach which encompasses both geometric and dynamic properties of the tracked vehicle in a single Bayes filter. The approach eliminates the need for separate data segmentation and association steps. We show how to properly model the dependence between geometric and dynamic vehicle properties using *anchor point coordinates*. The geometric model allows us to naturally handle the disjoint point clusters that often result from partial occlusion of vehicles (see Fig. 2). Moreover, the estimation of geometric shape leads to accurate prediction of dynamic parameters (see Fig. 3).

Further, we introduce an abstract sensor representation, called the *virtual scan*, which allows for efficient computation and can be used for a wide variety of laser sensors. We present techniques for building consistent virtual scans from 3D range data and show how to detect poorly visible black vehicles in laser scans. To battle the low signal-to-noise ratio during rapid detection of vehicles in noisy urban settings, we introduce the notion of *motion evidence*, which allows us to quickly prune false positives caused by noise. Our approach runs in real time with an average update rate of 40Hz, which is 4 times faster than the common sensor frame rate of 10Hz. The results show that our approach is reliable and efficient even in the challenging traffic situations presented at the UGC.

2 Background

A number of vehicle tracking approaches have been developed over the past few decades (e.g. Zhao and Thorpe (1998); Streller et al (2002); Wang (2004); Wender and Dietmayer (2008)) including most recent developments by the UGC participants (Darms et al, 2008; Leonard et al, 2008). Typically these approaches proceed in three stages: data segmentation, data association, and Bayesian filter update. During data segmentation the sensor data is divided into meaningful pieces — usually line features (Zhao and Thorpe, 1998; Darms et al, 2008) or clusters (Wender and Dietmayer, 2008; Leonard et al, 2008). During data association these pieces are assigned to tracked vehicles. Next, a Bayesian filter update is performed to fit targets to the data.

The second stage — data association — is generally considered the most challenging stage of the vehicle detection and tracking problem because of the association ambiguities that arise. Typically this stage is carried out using variants of the multiple hypothesis tracking (MHT) algorithm (e.g. Streller et al (2002); Wang et al (2007)).

In the third stage, the filter update is usually carried out using variants of Kalman filter (KF), which is augmented by the interacting multiple model method in some cases (Zhao and Thorpe, 1998; Wang et al, 2007).

Although vehicle tracking literature primarily relies on variants of KF, there is a great body of multiple target tracking literature for other applications where parametric, sample-based, and hybrid filters are used. Blackman et al (2004) provides a summary. For example Särkkä et al (2007) uses a Rao-Blackwellized particle filter (RBPF) for multiple target tracking on simulated data. A popular alternative to MHT for data association is the joint probabilistic data association (JPDA) method, which is used by Schulz et al (2001) to track multiple targets from an indoor mobile robot platform.

The work included in this paper has been presented at two conferences: Petrovskaya and Thrun (2008a) focused on efficient detection of vehicles and Petrovskaya and Thrun (2008b) focused on model based tracking. In contrast to prior vehicle tracking literature, we utilize a model based approach, which uses RBPFs and eliminates the need for separate data segmentation and association stages. Our approach estimates position, velocity and shape of tracked vehicles.

Further, we propose techniques for fast and accurate moving vehicle detection, which is a prerequisite for vehicle tracking. In prior art, the detection problem has been solved by addition of vision sensors (e.g. Wender and Dietmayer (2008)), although visual classification does not help distinguish moving vehicles from stationary. Another approach is to sample frames at lower rates to overcome the low signal-to-noise ratio (Wang et al, 2007), although this increases the time it takes to detect a new moving vehicle. Other described approaches detect vehicles by scan shape (Zhao and Thorpe, 1998; Streller et al, 2002) or by location (Wang et al, 2007). Due to possible ambiguities in the range data, these approaches tend to have lower detection accuracy.

3 Representation

In this paper we shall assume that a reasonably precise pose of the ego-vehicle is always available. On our robot, the pose estimates are provided by the localization module, which is described in detail in (Montemerlo et al, 2008). Here we provide a brief summary. The robot is outfitted with an Applanix POS LV 420 inertial navigation system (INS) which provides pose localization with 1m accuracy. Due to periodic GPS measurement updates the INS pose estimate can suddenly shift by up to 1m. The sudden shifts are very undesirable for vehicle tracking as they greatly increase tracking uncertainty. For the purposes of vehicle tracking the ego-vehicle pose estimate should evolve smoothly over time. For this reason we implemented *smooth coordinates*, which provide a locally consistent estimate of the ego-vehicle motion by integrating the velocity estimates from the INS. Although the smooth pose estimate can drift over time, it does not experience sudden shifts. To map from smooth coordinates to globally consistent GPS coordinates, one simply needs to add an offset, which is periodically updated to reflect the mismatch between the smooth and GPS coordinate systems. A similar smooth coordinate system was independently developed by the MIT UGC team (Leonard et al, 2008). In the remainder of this paper all operations will be carried out in the smooth coordinate frame, which we will also call the world frame. The transformation from smooth to GPS coordinates will only be needed when dealing with global features, such as the digital road map.

Following the common practice in vehicle tracking (Dellaert and Thorpe, 1998; Dietmayer et al, 2001; Leonard et al, 2008), we will represent each vehicle by a separate Bayesian filter, and represent dependencies between vehicles via a set of local spatial constraints. Specifically, we will assume that no two vehicles overlap, that all vehicles are spatially separated by some free space, and that all vehicles of interest are located on or near the road.

3.1 Probabilistic Model and Notation

For each vehicle we estimate its 2D position and orientation $X_t = (x_t, y_t, \theta_t)$ at time t, its forward velocity v_t and its geometry G (further defined in Sect. 3.2). Also at each time step we obtain a new measurement Z_t. A dynamic Bayes network representation of the resulting probabilistic model is shown in Fig. 4. The dependencies between the parameters involved are modeled via probabilistic laws discussed in detail in Sects. 3.3 and 3.5. For now we briefly note that the velocity evolves over time according to

$$p(v_t | v_{t-1}). \tag{1}$$

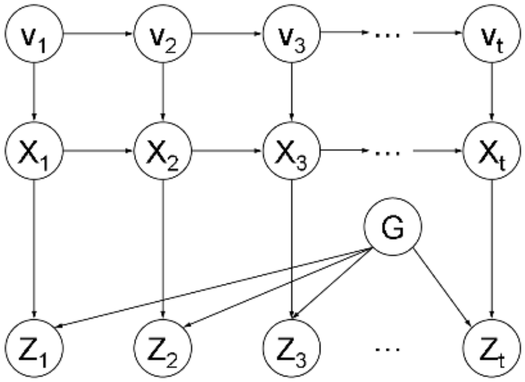


Fig. 4 Dynamic Bayesian network model of the tracked vehicle pose X_t, forward velocity v_t, geometry G, and measurements Z_t.

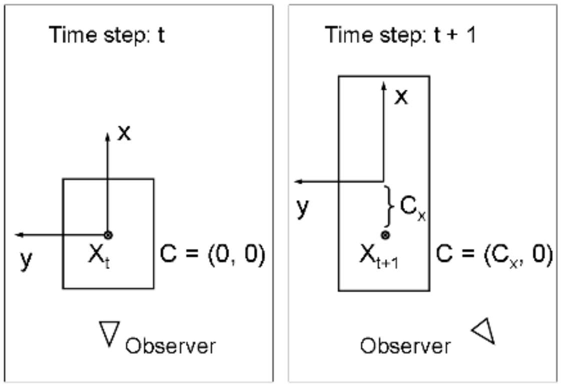


Fig. 5 As we move to observe a different side of a stationary car, our belief of its shape changes and so does the position of the car's center point. To compensate for the effect, we introduce local anchor point coordinates $C = (C_x, C_y)$ so that we can keep the anchor point X_t stationary in the world coordinates.

The vehicle moves based on the evolved velocity according to a dynamics model:

$$p(X_t | X_{t-1}, v_t). \tag{2}$$

The measurements are governed by a measurement model:

$$p(Z_t | X_t, G). \tag{3}$$

For convenience we will write $X^t = (X_1, X_2, ..., X_t)$ for the vehicle's trajectory up to time t. Similarly, v^t and Z^t will denote all velocities and all measurements up to time t.

3.2 Vehicle Geometry

The exact geometric shape of a vehicle can be complex and difficult to model precisely. For simplicity we approximate it by a rectangular shape of width W and length L. The 2D representation is sufficient because the height of tracked vehicles is not important for driving applications.

During vehicle tracking, the state variable X_t usually represents the position of the vehicle's center in the world coordinate frame. However, there is an interesting dependence between our belief about the vehicle's shape and its position. As we observe the object from a different vantage point, we change not only our belief of its shape, but also our belief of the position of its center point. Allowing X_t to denote the center point can lead to the undesired effect of obtaining a non-zero velocity for a stationary vehicle, simply because we refine our knowledge of its shape as Fig. 5 illustrates.

To overcome this problem, we view X_t as the pose of an *anchor point* whose position with respect to the vehicle's center can change over time. Initially we set the anchor point to be the center of what we believe to be the car's shape and thus its coordinates in the vehicle's *local* coordinate system are $C = (0, 0)$. We assume that the vehicle's local coordinate system is tied to its center with the x-axis pointing directly forward. As we revise our knowledge of the vehicle's shape, the local coordinates of the anchor point will also need to be revised accordingly to $C = (C_x, C_y)$. Thus, the complete set of geometric parameters is $G = (W, L, C_x, C_y)$.

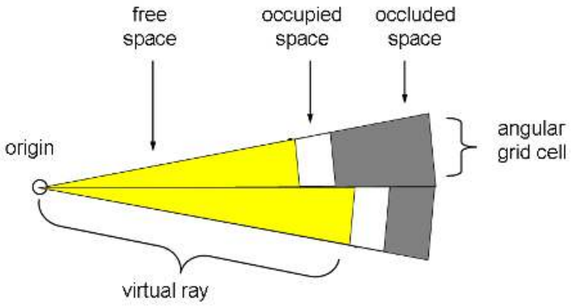
3.3 Vehicle Dynamics Model

In vehicle tracking literature it is common to use a constant velocity model (Dellaert and Thorpe, 1998), a constant acceleration model (Dietmayer et al, 2001), or a switching dynamics model (Wang, 2004; Darms et al, 2008). We use the constant velocity model and assume that velocity of each tracked vehicle stays constant for the duration of each time interval from $t-1$ to t. It also instantaneously evolves at each time step t via addition of random bounded noise based on maximum allowed acceleration a_{max} and the time delay Δt from the previous time step $t-1$. Specifically, we sample Δv uniformly from $[-a_{max}\Delta t,\ a_{max}\Delta t]$.

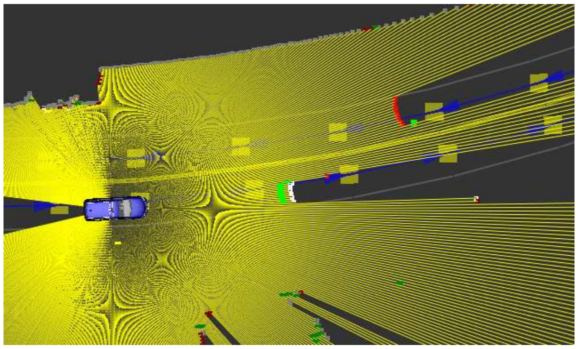
The pose evolves via linear motion (Thrun et al, 2005, Sec. 5.4) — a motion law that is often utilized when exact dynamics of the object are unknown. The motion consists of perturbing orientation by $\Delta\theta_1$, then moving forward according to the current velocity by $v_t \Delta t$, and making a final adjustment to orientation by $\Delta\theta_2$. Again we sample $\Delta\theta_1$ and $\Delta\theta_2$ uniformly from $[-d\theta_{max}\Delta t,\ d\theta_{max}\Delta t]$ for a maximum allowed orientation change $d\theta_{max}$.

3.4 Sensor Data Representation

In this paper we focus on laser range finders for sensing the environment. Recently these sensors have evolved to be more suitable for driving applications. For example IBEO Alasca sensors allow for easy ground filtering by collecting four parallel horizontal scan lines and marking which of the readings are likely to come from the ground (Ibeo Automobile Sensor GmbH, 2008). Velodyne HDL-64E sensors do not provide ground filtering, however they take a 3D scan of the environment at high frame rates (10Hz) producing



(a) anatomy of a virtual scan



(b) a virtual scan constructed from Velodyne data

Fig. 6 In (b) yellow line segments represent virtual rays. Colored points show the results of a scan differencing operation. Red points are new obstacles, green points are obstacles that disappeared, and white points are obstacles that remained unchanged or appeared in previously occluded areas. (Best viewed in color.)

1,000,000 readings per second (Velodyne Lidar, Inc., 2008). Given such rich data, the challenge has become to process the readings in real time. Vehicle tracking at 10 - 20Hz is desirable for driving decision making.

A number of factors make the use of raw sensor data inefficient. As the sensor rotates to collect the data, each new reading is made from a new vantage point due to ego-motion. Ignoring this effect leads to significant sensor noise. Taking this effect into account makes it difficult to quickly access data that pertains to a specific region of space. Much of the data comes from surfaces uninteresting for the purpose of vehicle tracking, e.g. ground readings, curbs and tree tops. Finally, the raw 3D data wastes a lot of resources as vehicle tracking is a 2D application where the cars are restricted to move on the ground surface. Therefore it is desirable to pre-process the data to produce a virtual sensor representation tailored for vehicle tracking.

Virtual sensors have been employed in the past for a wide range of applications. For example, in neuroimaging, virtual sensors have been created from fMRI data using machine learning techniques for diagnosis of mental processes in patients with brain injuries (Mitchell et al, 2002). In sensor networks, virtual sensors have been implemented to abstract data from multiple non-homogeneous sensors (Kabadayi et al, 2006). In geoscience, virtual sensors have been constructed using models trained on spectrally rich data to “fill in” unmeasured spectral channels in spectrally poor data for improved detection of clouds over snow and ice (Srivastava et al, 2005). In artificial intelligence and robotics, virtual sensors are commonplace in simulated environments, often used as a testbed for perception, planning and control algorithms (Thalmann et al, 1997; Gerkey et al, 2003).

To create a virtual sensor for our application, we construct a grid in polar coordinates — a *virtual scan* — which subdivides 360° around a chosen origin point into angular grid cells (see Fig. 6). In each angular grid cell we record the range to the closest obstacle within that cell. Hence each angular grid cell contains the following information: the space from origin up to the recorded range is free, at the recorded range – occupied, and beyond the recorded range – occluded. We will often refer to the cone of an angular grid cell from the origin up to the recorded range as a *ray* due to its similarity to a laser ray. We will also treat each angular grid cell as a single range measurement in the virtual scan.

Virtual scans simplify data access by providing a single point of origin for the entire data set, which allows constant time look-up for any given point in space. As we mentioned earlier, it is important to compute correct world coordinates for the raw sensor readings. However, once the correct positions of obstacle points have been computed, adjusting the origin of each ray to be at the common origin for the virtual scan produces an acceptable approximation. To minimize the error due to approximation, we select the common origin to be the average sensor pose during scan collection. Constructed in this manner, a virtual scan provides a compact representation of the space around the ego-vehicle classified into free, occupied and occluded. The classification helps us properly reason about what parts of an object should be visible as we describe in Sect. 3.5.

One important parameter of a virtual scan is the angular resolution. Although coarser resolutions can speed up computations because fewer rays need to be examined, it is desirable to set the resolution as fine as possible in order to capture more detail about objects at long range.[1] For this reason, we set the resolution as fine as possible in our implementation. For the IBEO lasers we set the resolution to 0.5°, which is the highest resolution the sensor provides.

For the purpose of vehicle tracking it is crucial to determine what changes take place in the environment over time. With virtual scans these changes can be easily computed in spite of the fact that ego-motion can cause two consecutive virtual scans to have different origins. The changes are computed by checking which obstacles in the old scan are cleared by rays in the new scan and vice versa. This computation takes time linear in the size of the virtual scan and only needs to be carried out once per frame. Fig. 6(b) shows results of a virtual scan differencing operation with red points denoting new obstacles, green points denoting obstacles that disappeared, and white points denoting obstacles that remained in place or appeared in previously occluded areas.

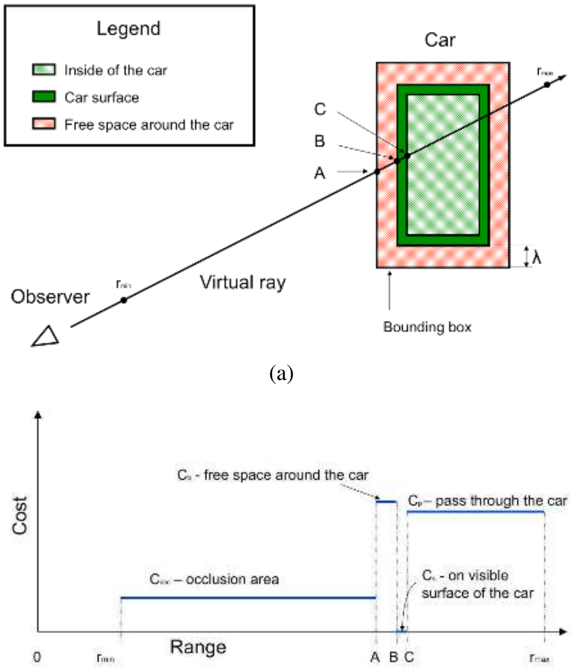
---

[1] In principle it is possible to get the best of both worlds by constructing several virtual scans of varying resolution for the same laser data. Lower resolution virtual scans can be used to examine close range objects, while higher resolution scans can be used for long range operations.

Virtual scans are a suitable representation for a wide variety of laser range finders. While this representation is easy to build for 2D sensors such as IBEO, 3D range sensors require additional considerations to produce consistent 2D representations. We describe these techniques in Sect. 6.

3.5 Measurement Model

This section describes the measurement model $p(Z|X,G)$ used in our approach. Here Z is a virtual scan representation of a single frame of range data from a laser range finder. To our knowledge, range scan likelihood models have not been proposed for vehicle tracking, as most of the vehicle tracking literature is concerned with tracking point targets — usually centers of clusters (Wang, 2004; Leonard et al, 2008) or features extracted from the range data (Streller et al, 2002; Wender and Dietmayer, 2008; Darms et al, 2008). In contrast to the prior art, we are able to provide a direct interpretation of the range measurements because we model geometry of the tracked vehicles. Measurement models for range finders in the presence of a geometric environment model have been proposed in mobile robot localization and mapping literature, where the environment is commonly represented by an occupancy grid map (see Thrun et al (2005, Ch. 6) for an overview). The two most common models are the independent beam model (IB) (Moravec, 1988; Burgard et al, 1996; Fox et al, 1999) and the likelihood field model (LF) (Thrun, 2001). The IB model treats each ray in the scan as an independent measurement of range to the closest obstacle along the ray corrupted by Gaussian noise. One drawback of the IB model is that rays are represented by lines. This assumption does not work well at longer ranges ($50-100$m) typical in outdoor environments. Outdoors it is better to represent rays by cones because the laser spot light is of non-negligible radius ($20-40$cm). Another drawback is that the IB model does not leave room for possible unmodeled occlusions of the geometric model — a very common scenario in vehicle tracking. The LF model also treats laser rays as independent of each other. The end point of each ray is compared to the closest obstacle point (not necessarily on the ray itself) under the assumption of Gaussian noise. The LF model is more appropriate for cone representation of rays. It also handles unmodeled occlusions very well. However the LF model allows rays to go through obstacles without any penalty. This is undesirable for vehicle tracking because rays going through a candidate vehicle provide strong evidence that these points may not belong to the same physical object.


Legend
▨ Inside of the car
🟩 Car surface
🟥 Free space around the car
Car
C
B
A
r_{max}
λ
Observer
r_{min}
Virtual ray
Bounding box
Cost
C_b - free space around the car
C_p – pass through the car
C_{occ} – occlusion area
C_s - on visible surface of the car
Range
0
r_{min}
A
B
C
r_{max}

(a)

(b)

Fig. 7 Measurement likelihood computations. (a) shows the geometric regions involved in the likelihood computations. (b) shows the costs assignment for a single ray. (Best viewed in color.)

Given a vehicle's pose X, geometry G and a virtual scan Z, we compute the measurement likelihood $p(Z|G,X)$ as follows. We position a rectangular shape representing the vehicle according to X and G. Then we build a bounding box to include all points within a predefined distance λ^2 around the vehicle (see Fig. 7). Assuming that there is an actual vehicle in this configuration, we would expect the points within the rectangle to be occupied or occluded, and points in its vicinity to be free or occluded because vehicles are spatially separated from other objects in the environment.

Like the IB and LF models for laser range finders, we consider measurements obtained along each ray to be conditionally independent of each other given vehicle pose and geometry. Thus, if we have a total of N rays in the virtual scan Z, the measurement likelihood factors as follows:

$$p(Z|G,X) = \prod_{i=1}^{N} p(z_i|G,X). \tag{4}$$

Following the IB and LF models, we use a Gaussian form for each ray's likelihood. Specifically, we model it as a zero-mean Gaussian of variance σ_i computed with respect to a

---

[2] We used the setting of $\lambda = 1m$ in our implementation.

cost c_i selected based on the relationship between the ray and the vehicle (η_i is a normalization constant):

$$p(z_i|G,X) = \eta_i \exp\Big(-\frac{c_i^2}{2\sigma_i^2}\Big). \tag{5}$$

The costs are set to constants that depend on the region in which the ray's end point falls (see Fig. 7 for illustration). c_{occ} is the cost for range readings that fall short of the bounding box and thus represent situations when another object is occluding the vehicle. c_b is the cost for range readings that fall short of the vehicle but inside of the bounding box. c_s is the cost for readings on the vehicle's visible surface which we assume to be of non-zero depth. c_p is used for rays that extend beyond the vehicle's surface. Assigning likelihood based on the region of space in which a ray's end point falls bears resemblance to the LF model. It is more appropriate for cone representation of rays than the IB model. Like the LF model our measurement model gives little penalty to occlusions by other objects, but unlike the LF model we assign a large penalty to rays passing through the candidate vehicle. We also enforce our assumption of free space around each vehicle by assigning a large penalty to rays that terminate in this region.

The domain for each range reading is between the minimum range r_{min} and the maximum range r_{max} of the sensor. Since the costs we select are piece-wise constant, it is easy to integrate the unnormalized likelihoods to obtain the normalization constants η_i. Note that for the rays that do not target the vehicle or the bounding box, the above logic automatically yields uniform distributions as these rays never hit the bounding box.

Note that the above measurement model naturally handles partially occluded objects including objects that are "split up" by occlusion into several point clusters (see Fig. 2). In contrast, these cases are often challenging for approaches that utilize separate data segmentation and correspondence methods.

4 Vehicle Tracking

Most vehicle tracking methods described in the literature apply separate methods for data segmentation and correspondence matching before fitting model parameters via extended Kalman filter (EKF). In contrast, we use a single Bayesian filter to fit model parameters from the start. This is possible because our model includes both geometric and dynamic parameters of the vehicles and because we rely on efficient methods for parameter fitting. We chose the particle filter method for Bayesian estimation because it is more suitable for multi-modal distributions than EKF. Unlike the multiple hypothesis tracking (MHT) method commonly used in the literature, the computational complexity for our method grows linearly with the number of vehicles in the environment because vehicle dynamics dictates that vehicles can only be matched to data points in their immediate vicinity. The downside, of course, is that two targets can in principle merge into one. In practice we have found that this happens rarely and only in situations where one of the targets is lost due to complete occlusion. In these situations target merging is acceptable for our application.

We have a total of eight parameters to estimate for each vehicle: $X = (x, y, \theta)$, v, $G = (W, L, C_x, C_y)$. Computational complexity grows exponentially with the number of parameters for particle filters. Thus, to keep computational complexity low, we turn to RBPFs first introduced in Doucet et al (2000). We estimate X and v by samples and keep Gaussian estimates for G within each particle. Below we give a brief derivation of the required update equations.

4.1 Derivation of Update Equations

At each time step t, we produce an estimate of a Bayesian belief about the tracked vehicle's trajectory, velocity, and geometry based on a set of measurements:

$$Bel_t = p(X^t, v^t, G|Z^t). \tag{6}$$

The derivation provided below is similar to the one used in Montemerlo (2003). We split up the belief into two conditional factors:

$$Bel_t = p(X^t, v^t|Z^t)\ \ p(G|X^t, v^t, Z^t). \tag{7}$$

The first factor encodes the vehicle's motion posterior,

$$R_t = p(X^t, v^t|Z^t). \tag{8}$$

The second factor encodes the vehicle's geometry posterior, conditioned on its motion,

$$S_t = p(G|X^t, v^t, Z^t). \tag{9}$$

The factor R_t is approximated using a set of particles; the factor S_t is approximated using a Gaussian distribution (one Gaussian per particle). Let us denote a particle by $q_m^t = (X^{t,[m]}, v^{t,[m]}, S_t^{[m]})$ and a collection of particles at time t by $Q_t = \{q_m^t\}_m$. We compute Q_t recursively from Q_{t-1}. Suppose that at time step t, particles in Q_{t-1} are distributed according to R_{t-1}. We compute an intermediate set of particles $\bar{Q}_t$ by sampling a guess of the vehicle's pose and velocity at time t from the dynamics model (described in detail in Sect. 3.3). Thus, particles in $\bar{Q}_t$ are distributed according to the vehicle motion prediction distribution,

$$\bar{R}_t = p(X^t, v^t|Z^{t-1}). \tag{10}$$

To ensure that particles in Q_t are distributed according to R_t (asymptotically), we generate Q_t by sampling from $\bar{Q}_t$ with replacement in proportion to importance weights given by

$w_t = R_t / \bar{R}_t$. Before we can compute the weights, we need to derive the update equations for the geometry posterior.

We use a Gaussian approximation for the geometry posterior, S_t. Thus we keep track of the mean μ_t and the co-variance matrix Σ_t of the approximating Gaussian in each particle: $q_m^t = (X^{t,[m]}, v^{t,[m]}, \mu_t^{[m]}, \Sigma_t^{[m]})$. We have:

$$\begin{aligned} S_t &= p(G|X^t, v^t, Z^t) \\ &\propto p(Z_t|G, X^t, v^t, Z^{t-1})\ \ p(G|X^t, v^t, Z^{t-1}) \\ &= p(Z_t|G, X_t)\ \ p(G|X^{t-1}, v^{t-1}, Z^{t-1}). \end{aligned} \tag{11}$$

The first step above follows from Bayes' rule; the second step follows from the conditional independence assumptions of our model (Fig. 4). The expression (11) is a product of the measurement likelihood and the geometry prior S_{t-1}. To obtain a Gaussian approximation for S_t, we linearize the measurement likelihood as will be explained in Sect. 4.3. Once the linearization is performed, the mean and the co-variance matrix for S_t can be computed in closed form because S_{t-1} is already approximated by a Gaussian (represented by a Rao-Blackwellized particle from the previous time step).

Now we are ready to compute the importance weights. Briefly, following the derivation in Montemerlo (2003), it is straightforward to show that the importance weights w_t should be:

$$w_t = R_t / \bar{R}_t = \frac{p(X^t, v^t|Z^t)}{p(X^t, v^t|Z^{t-1})} = \mathbb{E}_{S_{t-1}}[\ p(Z_t|G, X_t)\ ]. \tag{12}$$

In words, the importance weights are the expected value (with respect to the vehicle geometry prior) of the measurement likelihood. Using Gaussian approximations of S_{t-1} and $p(Z_t|G, X_t)$, this expectation can be expressed as an integral over a product of two Gaussians, and can thus be carried out in closed form.

4.2 Motion Inference

As we mentioned in Sect. 3.1, a vehicle's motion is governed by two probabilistic laws: $p(v_t|v_{t-1})$ and $p(X_t|X_{t-1}, v_t)$. These laws are related to the motion prediction distribution as follows:

$$\begin{aligned} \bar{R}_t &= p(X^t, v^t|Z^{t-1}) \\ &= p(X_t, v_t|X^{t-1}, v^{t-1}, Z^{t-1})\ p(X^{t-1}, v^{t-1}|Z^{t-1}) \\ &= p(X_t|X^{t-1}, v^t, Z^{t-1})\ p(v_t|X^{t-1}, v^{t-1}, Z^{t-1})\ R_{t-1} \\ &= p(X_t|X_{t-1}, v_t)\ p(v_t|v_{t-1})\ R_{t-1}. \end{aligned} \tag{13}$$

The first and second steps above are simple conditional factorizations; the third step follows from the conditional independence assumptions of our model (Fig. 4).

Note that since only the latest vehicle pose and velocity are used in the update equations, we do not need to actually store entire trajectories in each particle. Thus the memory storage requirements per particle do not grow with t.

4.3 Shape Inference

In order to maintain the vehicle's geometry posterior in a Gaussian form, we need to linearize the measurement likelihood $p(Z_t|G, X_t)$ with respect to G. Clearly the measurement likelihood does not lend itself to differentiation in closed form. Thus we turn to Laplace's method to obtain a suitable Gaussian approximation. The method involves fitting a Gaussian at the global maximum of a function. Since the global maximum is not readily available, we search for it via local optimization starting at the current best estimate of geometry parameters. Due to construction of our measurement model (Sect. 3.5), the search is inexpensive as we only need to recompute the costs for the rays directly affected by a local change in G.

The dependence between our belief of the vehicle's shape and its position (discussed in Sect. 3.2) manifests itself in a dependence between the local anchor point coordinates C and the vehicle's width and length. The vehicle's corner closest to the vantage point is a very prominent feature that impacts how the sides of the vehicle match the data. When revising the belief of the vehicle's width and length, we keep the closest corner in place. Thus a change in the width or the length leads to a change in the global coordinates of the vehicle's center point, for which we compensate with an adjustment in C to keep the anchor point in place. This way a change in geometry does not create phantom motion of the vehicle.

4.4 Initializing and Discontinuing Tracks

New tracks are initialized in areas where scan differencing detects a change in data that is not already explained by existing tracks. New tracks are fitted using the same measurement and motion models that we use for vehicle tracking (Sects. 3.5 and 3.3). The candidates are vetted for three frames before they can become "real tracks". Detection of new vehicles is the most computationally expensive part of vehicle tracking. In Sect. 5 we describe the techniques we used to achieve reliable vehicle detection in real time.

We discontinue tracks if the target vehicle gets out of sensor range or moves too far away from the road.[3] We also discontinue tracks if the unnormalized weights have been low for several turns. Low unnormalized weights signal that the sensor data is insufficient to track the target, or that our estimate is too far away from the actual vehicle. This logic keeps the resource cost of tracking occluded objects low, yet it still allows for a tracked vehicle to survive bad data or complete occlusion for several turns. Since new track acquisition only takes three frames, it does not make sense to continue tracking objects that are occluded for significantly longer periods of time.

---

[3] A digital street map was available for our application in the Road Network Definition Format (RNDF).

5 Vehicle Detection

Accurate moving vehicle detection in laser range data requires three frames. The first two frames are required to detect motion of an object. The third frame is required to check that the motion is consistent over time and follows the vehicle dynamics law. Thus for a 10Hz sensor the minimum vehicle detection time is 0.3 seconds.

Note that detection based on three frames allows for accurate results because we can observe two consecutive motion updates and verify that the observed motion is consistent with a moving vehicle. For some applications it may be acceptable to sacrifice accuracy in favor of faster detection based on just one or two frames. For example in Wang et al (2007) objects that appear in areas previously seen as empty are detected as "moving". Often this approach is adopted when the intention is to filter out moving obstacles to build a static map.

5.1 The Basic Detection Algorithm

Our vehicle detection method proceeds in three stages:

1. First a vehicle is fitted using importance sampling in an area where a change in the environment has been detected by scan differencing. The scoring is performed using the measurement model described in Sect. 3.5.
2. Next the vehicle's velocity is estimated by performing a particle filter update step and scoring using the measurement model in the next frame.
3. During the last stage, another particle filter update is performed and scored against a third frame.

5.2 Challenges in Vehicle Detection

The range data in outdoor urban environments contains large amounts of noise that adds up from a number of sources. The limitations of horizontal scan resolution (0.5° for IBEO and 0.1° for Velodyne) and vertical scan resolution (0.4° for Velodyne) produce $40-50$cm noise at 60m range. Another source of noise is the laser beam spot size, which can exceed the scan resolution (Sick Optics, 2003). Scanning the same vehicle at a slightly different height can result in $1-2$m range discrepancy. Additional noise comes from the virtual scan approximation (25cm at 60m range for 0.5° angular resolution) and the box model approximation of the vehicle's shape ($20-40$cm). Internal sensor construction, circuitry, and messaging time delays also produce noise, which is in general difficult to quantify. Studies have been performed for older sensors (Mäkynen, 2000; Blais, 2004), but this information is not yet available for the newer models of range finders. Finally, environmental factors such as dust and rain cause false readings many meters off the actual target.

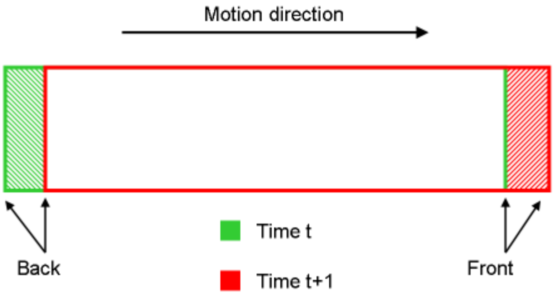


Fig. 8 Diagram representing forward motion of a bus. Green color represents the position of the bus at time t. Red color represents its position at time $t+1$. The green shaded area in the back of the bus frees up as the bus moves forward. The red shaded area in the front of the bus becomes occupied. Note that these changes are small compared to the overall area taken up by the bus, which remains occupied in both frames. (Best viewed in color.)

For the driving application we need to detect vehicles moving at 5mph to 35mph with a 10Hz sensor. Thus a vehicle moves $20-150$cm per frame. This signal can be easily overwhelmed by noise especially in the lower range of the velocities. The poor signal-to-noise ratio makes it difficult to accurately tell a moving object apart from noise in just three frames.

Although the signal is easier to detect if we use more than three frames, this solution is undesirable because it increases the detection time and takes up more computational resources. A more efficient approach, proposed by Wang et al (2007), is to sample the frames at a lower rate (e.g. 1Hz), so that the signal is prevalent over the noise. However, this method also increases the total time required for detection of a vehicle and therefore it is unsuitable for our application.

5.3 Motion Evidence

To overcome the poor signal-to-noise ratio, we turn to the method used by humans to detect moving vehicles in noisy data. Consider a long bus moving forward at 5mph (Fig. 8). From one frame to the next it travels 20cm - a negligible distance compared to the noise and overall size of the vehicle. Since the middle of the bus appears stationary, a human trying to discern motion will focus on the front and back of the bus, to see if there is at least a tad of motion.

To take advantage of the same method for vehicle detection, we define a score we call *motion evidence*. To compute this score, we consider the regions cleared by the vehicle as it moves. The cleared area behind the vehicle should be occupied in the prior frame and free in the current frame. Similarly, the area in front of the moving vehicle should be free in the prior frame and occupied in the current frame. Often we can only observe the front or the back of the vehicle, thus only half of the evidence is available due to self-occlusion. To allow for self-occlusion and partial occlusions by other objects we threshold the motion evidence score at 25%.

Note that the motion evidence score is different from the probabilities obtained by fitting a vehicle using a particle filter. The particle filter computes the probability that motion *"could have"* happened, whereas the motion evidence scores the motion that *"must have"* happened. In the bus example given above the motion evidence score would ignore the entire bus except 20cm in the front and in the back.

The motion evidence score can be computed for any pair of consecutive frames. In our approach we compute it for the first and the second pairs of frames and filter out vehicle candidates for which the score is below the threshold. Doing so provides a very dramatic decrease in false positives, without affecting the false negatives rate.

5.4 Optimizations

Since new vehicle detection is computationally expensive, we developed several optimizations to achieve reliable real time performance. We describe the optimization techniques below and evaluate their impact on the performance of vehicle detection in Sect. 7.2.

5.4.1 Road Masking

Since a digital road map is available in our application, one simple optimization is to restrict the search to the road regions. We do this by marking each data point as "close to road" or "far from road". Only the points near the road are considered for new vehicle detection. This optimization greatly improves the efficiency of the vehicle detection algorithm.

5.4.2 Cleared Area

As we already discussed above, a change in the data can be caused by either noise or motion. Ultimately the motion evidence score will help disambiguate motion from noise. However, the motion evidence score can only be used after the vehicle model has already been fitted to data. To make the search more efficient we would like to distinguish between noise and motion before performing any model fittings.

When a vehicle moves forward with a minimum velocity v_{min} for a time interval Δt, it clears an area of approximately $v_{min}\Delta t W$. Thus we can examine each data point to see if enough space has been cleared around it to allow for motion of a vehicle. If the vehicle is moving away from us, the cleared area will be in the current frame with respect to the prior frame. If the vehicle is approaching us, the cleared area will be in the prior frame with respect to the current frame. Thus we can find both types of cleared area by performing a symmetric clearing operation between the two frames.

Even though cleared area logic is not as powerful as the motion evidence score, it provides a significant speed-up when used as a fast data pre-processing step.

5.4.3 Scaling Series

The first step of vehicle detection involves fitting the geometric vehicle model to a virtual scan under conditions of large uncertainty: several meters in position and 360° in orientation of the vehicle. Using simple importance sampling with three state parameters makes the problem intractable within real time constraints.

To improve performance, we turn to Scaling Series, a method first proposed in Petrovskaya et al (2006) for a tactile localization application. In that application the number of parameters was also too large to perform an importance sampling step in real time in conditions of global uncertainty. They proposed the Scaling Series algorithm to efficiently produce a much more informed proposal distribution, one that is concentrated around the areas of high probability mass. We refer the reader to Petrovskaya et al (2006) for details on Scaling Series, but briefly, the algorithm works by performing a series of successive refinements, generating an increasingly informative proposal distribution at each step of the series. The successive refinements are performed by gradually annealing the measurement model from artificially relaxed to realistic.

For our problem, we applied the Scaling Series algorithm to choose the proposal distribution for the initial importance sampling step. We obtained measurement model relaxations by inflating the width, ω, of the vehicle surface region (see Fig. 7). The normal setting for ω is 0.25m. The most relaxed model was obtained by padding the region by 1m on the inside and outside, resulting in an ω setting of 2.25m. At this setting the vehicle surface region expands to consume the free space region, and thus the penalty c_b is not applied. However, even with this coarse model, the algorithm quickly rules out vehicle candidates placed more than 1m away from the actual vehicle location. The resulting high likelihood region includes a region of 1m radius around the true position of the vehicle. As ω is gradually annealed from 2.25m to 0.25m, the high likelihood region shrinks, resulting in a more and more informed proposal distribution.

In Sect. 7.2 we show that using this method, we obtained a very significant improvement in the reliability of the search and reduced the time it takes to detect a new moving vehicle by a factor of 10.

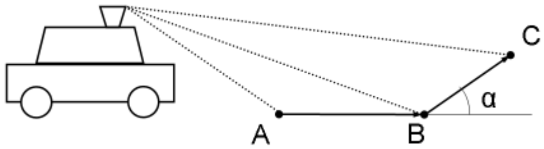


Fig. 9 We determine ground readings by comparing angles between consecutive readings. If A, B, C are ground readings, then α is close to 0 and thus $\cos\alpha$ is close to 1.

5.4.4 Backward Search

Since vehicle detection takes three frames, the minimum detection time is 0.3 seconds for a sensor with a frame rate of 10Hz. It turns out that if we only search forward in time, then the minimum detection time is 0.4 seconds for approaching vehicles because the first frame is only used to detect dynamic data points in the second frame. However, if we fit the vehicle in the second frame and then move it backwards in time, we can utilize the first frame as well. In this case we use frame number two for the initial vehicle fitting and frame number one for velocity estimation. As before, the third frame is used to check motion consistency.

6 Working with 3D Range Data

As we explained in Sect. 3.4, vehicle tracking is a 2D problem, for which compact 2D virtual scans are sufficient. However for 3D sensors, such as Velodyne, it is non-trivial to build consistent 2D virtual scans. These sensors provide immense 3D data sets of the surroundings, making computational efficiency a high priority when processing the data. In our experience, the hard work pays off and the resulting virtual scans carry more information than 2D sensor data.

To produce consistent 2D virtual scans, we need to understand which of the 3D data points should be considered obstacles. From the perspective of driving applications, we are interested in the slice of space directly above the ground and up to 2m high, as this is the space that a vehicle would actually have to drive through. Objects elevated more than 2m above ground — e.g. tree tops or overpasses — are not obstacles. The ground itself is not an obstacle (assuming the terrain is drivable). Moreover, for tracking applications, low obstacles such as curbs should be excluded from virtual scans because they can prevent us from seeing more important obstacles beyond them. The remaining objects in the 2m slice of space are obstacles for a vehicle, even if these objects are not directly touching the ground.

In order to classify the data into the different types of objects described above we first build a 3D grid in spherical coordinates. Similarly to a virtual scan, it has a single point of origin and stores actual world coordinates of the sensor readings. Just as in the 2D case, this grid is an approximation of the sensor data set because the actual laser readings in a scan have varying points of origin. In order to downsample and reject outliers for each spherical grid cell we compute the median range of the readings falling within.[4] This gives us a single obstacle point per grid cell. For each spherical grid cell, we will refer to the cone from the grid origin to the obstacle point as a virtual ray.

The first classification step is to determine ground points. For this purpose, we select a single slice of vertical angles from the spherical grid (i.e. rays that all have the same bearing angle). We cycle through the rays in the slice from the lowest vertical angle to the highest. For three consecutive readings *A*, *B*, and *C*, the slope between *AB* and *BC* should be near zero if all three points lie on the ground (see Fig. 9 for illustration). If we normalize *AB* and *BC*, their dot product should be close to 1. Hence a simple thresholding of the dot product allows us to classify ground readings and to obtain estimates of local ground elevation. Thus, one useful piece of information we can obtain from 3D sensors is an estimate of ground elevation. A similar ground estimation method was independently developed by the MIT Urban Challenge team (Leonard et al, 2008).

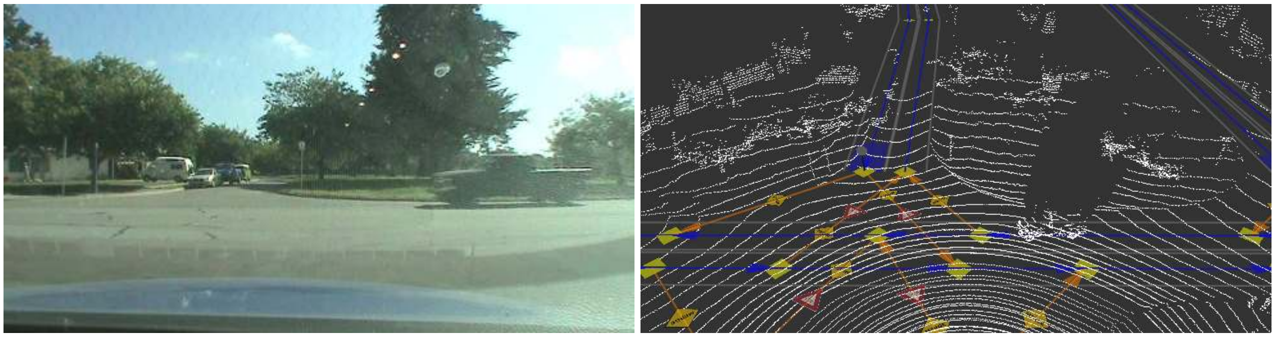
Using the elevation estimates, we can classify the remaining non-ground readings into low, medium and high obstacles, out of which we are only interested in the medium ones (see Fig. 10). It turns out that there can be medium height obstacles that are still worth filtering out: birds, insects and occasional readings from cat-eye reflectors. These obstacles are easy to filter because the *BC* vector tends to be very long (greater than 1m), which is not the case for normal vertical obstacles such as buildings and cars. After identifying the interesting obstacles we simply project them on the 2D horizontal plane to obtain a virtual scan.

6.1 Detection of Black Obstacles

Laser range finders are widely known to have difficulty seeing black objects. Since these objects absorb light, the sensor never gets a return. Clearly it is desirable to “see” black obstacles for driving applications. Other sensors could be used, but they all have their own drawbacks. Here we present a method for detecting black objects in 3D laser data. Figure 11 shows the returns obtained from a black car. The only readings obtained are from the license plate and wheels of the vehicle, all of which get filtered out as low obstacles. Instead of looking at the little data present, we can detect the black obstacle by looking at the absent data. If no readings are obtained along a range of vertical angles in a specific direction, we can conclude that the space must be occupied

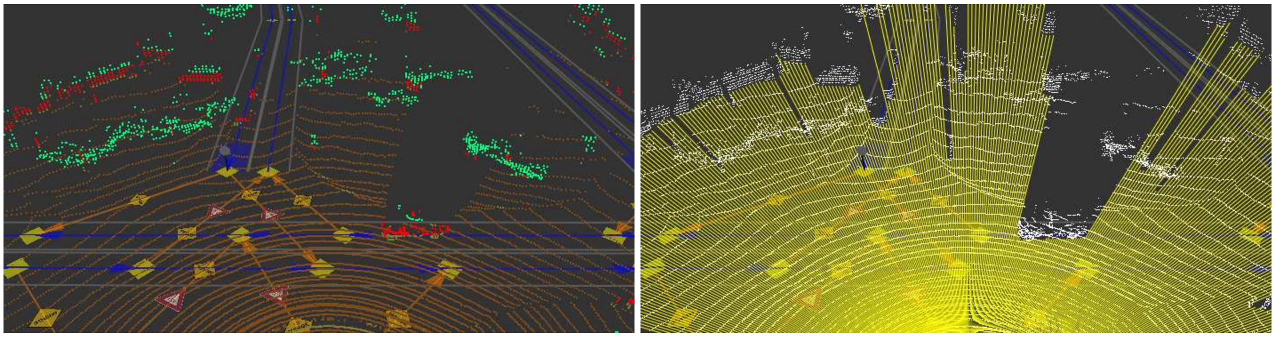
---

[4] In our implementation, the angular grid resolution for Velodyne based virtual scans is 0.5°, which results in three readings per angular grid cell on average — just enough to reject outliers.



(a) actual scene

(b) Velodyne data



(c) after classification

(d) generated virtual scan

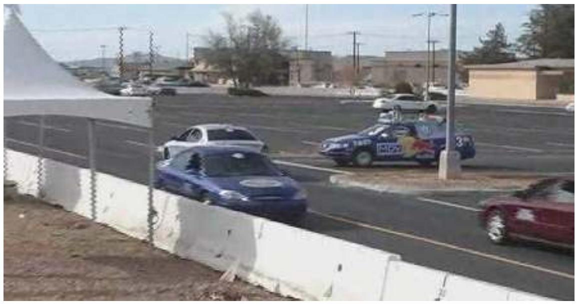
Fig. 10 In (c) Velodyne data is colored by type: orange — ground, yellow — low obstacle, red — medium obstacle, green — high obstacle. In (d) yellow lines denote the virtual scan. Note the truck crossing the intersection, the cars parked on a side of the road and the white van parked on a driveway. On the virtual scan all of these vehicles are clearly marked as obstacles, but ground, curbs and tree tops are ignored.

by a black obstacle. Otherwise the rays would have hit some obstacle or the ground. To provide a conservative estimate of the range to the black obstacle we place it at the last reading obtained in the vertical angles just before the absent readings. We note that this method works well as long as the sensor is good at seeing the ground. For the Velodyne sensor the range within which the ground returns are reliable is about 25 – 30m, beyond this range the black obstacle detection logic does not work.

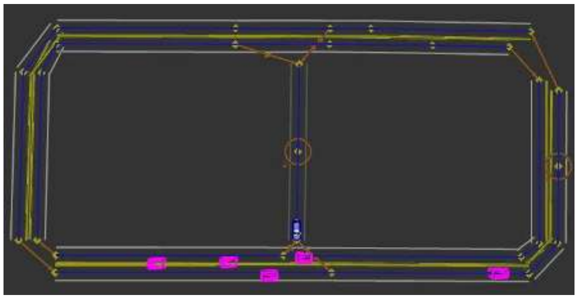
7 Experimental Validation

7.1 Tracking Results

The most challenging traffic situation at the Urban Grand Challenge was presented on course A during the qualifying event (Fig. 12) . The test consisted of dense human driven traffic in both directions on a course with an outline resembling the Greek letter θ. The robots had to merge repeatedly into the dense traffic. The merge was performed using a left turn, so the robots had to cross one lane of traffic each time. In these conditions, accurate estimates of positions and velocities of the cars are very useful for determining a gap in traffic large enough to perform the merge safely. Cars passed in close proximity to each other and to stationary obstacles (e.g. signs and guard rails) providing plenty of opportunity

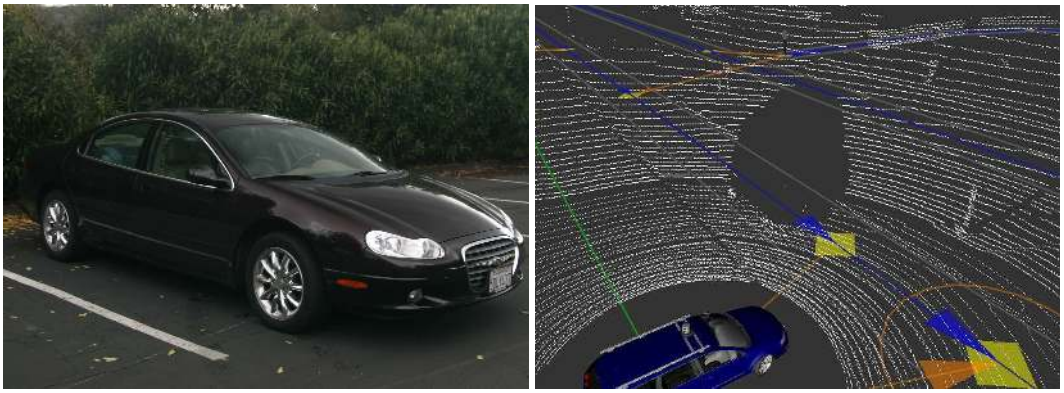


(a) traffic density



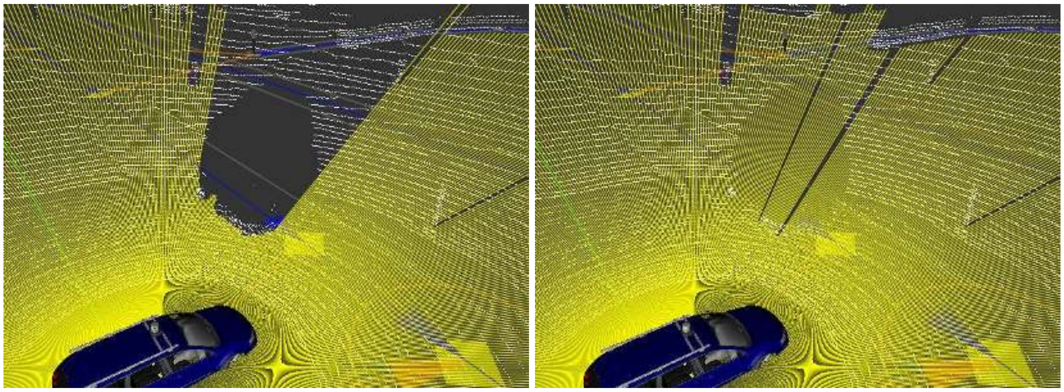
(b) course A outline

Fig. 12 Test conditions on course A at the Urban Grand Challenge. The test consisted of repeated merges into dense traffic (a) on a course with an outline resembling the Greek letter θ (b).



(a) actual appearance of the vehicle

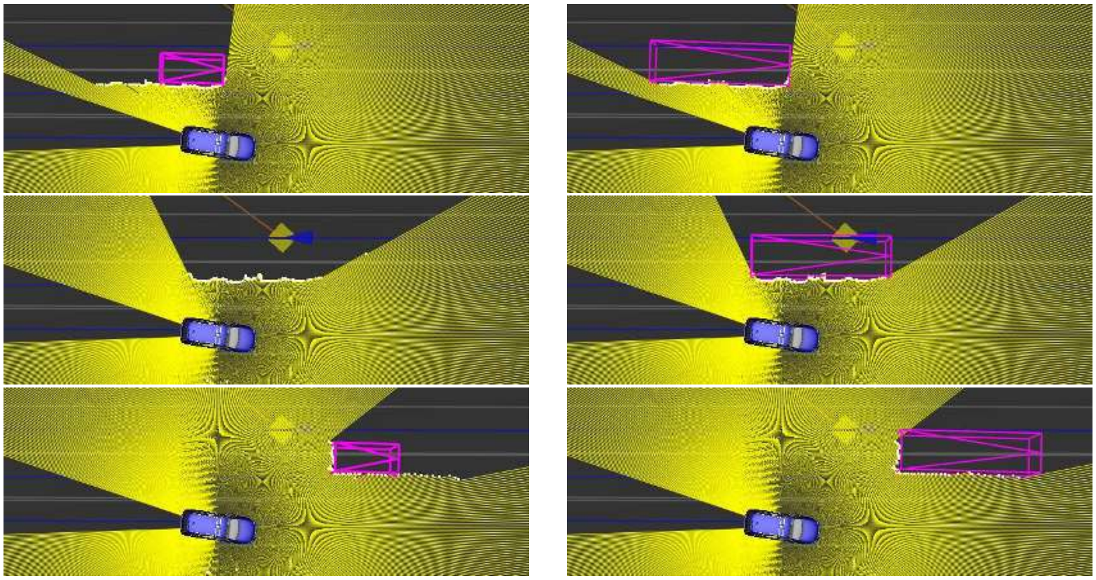
(b) the vehicle gives very few laser returns



(c) virtual scan with black object detection

(d) virtual scan without black object detection

Fig. 11 Detecting black vehicles in 3D range scans. White points represent raw Velodyne data. Yellow lines represent the generated virtual scans.



(a) without size estimation

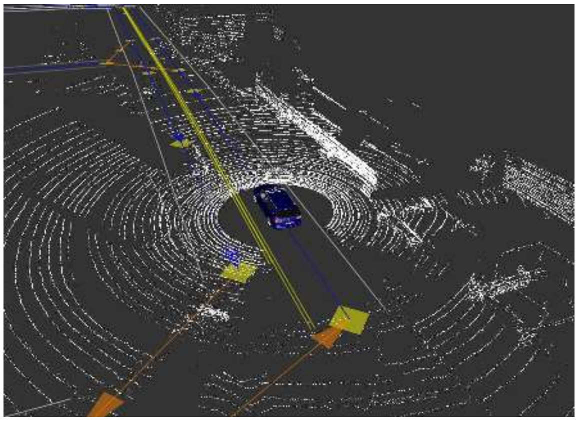
(b) with size estimation

Fig. 15 Size estimation on the example of a passing bus from a data set taken in Alameda. Without size estimation (a) the tracking results are poor because the geometric model does not fit the data well. Not only is the velocity estimated incorrectly, but the track is lost entirely when the bus is passing. With size estimation (b) the bus is tracked successfully and the velocity is properly estimated. (Best viewed in color.)

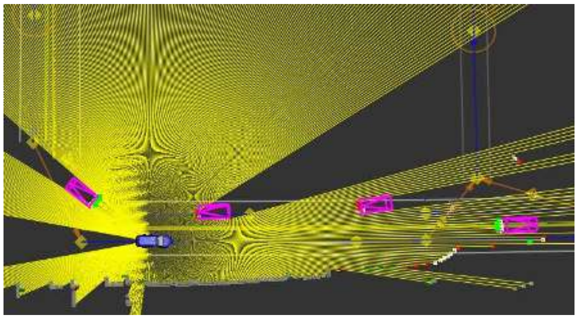
Table 1 Tracker performance on data sets from three urban environments. Max TP is the theoretically maximum possible true positive percent for each data set. TP and FP are the actual true positive and false positive rates attained by the algorithm.

Data Sets	Total Frames	Total Vehicles	Correctly Identified	Falsely Identified	Max TP (%)	TP (%)	FP (%)
UGC Area A	1,577	5,911	5,676	205	97.8	96.02	3.35
Stanford Campus	2,140	3,581	3,530	150	99.22	98.58	4.02
Alameda Day 1	1,531	901	879	0	98.22	97.56	0
Overall	5,248	10,393	10,085	355	98.33	97.04	3.3

(a) actual scene



(b) Velodyne data



(c) virtual scan and tracking results

Fig. 13 Tracking results on course A at the UGC. In (c) yellow line segments represent the virtual scan and red/green/white points show results of scan differencing. The purple boxes denote the tracked vehicles. (Best viewed in color.)

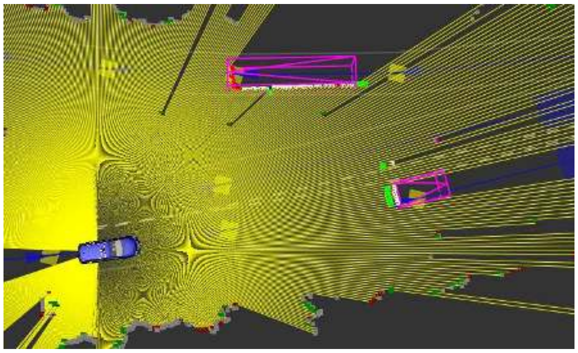
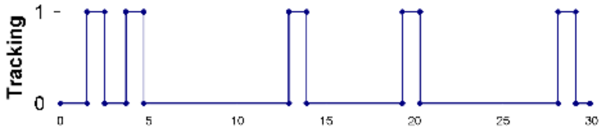


Fig. 14 Size estimation results on Stanford campus. Vehicles of different sizes are successfully estimated and tracked. (Best viewed in color.)

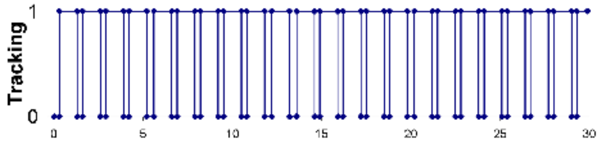
for false associations. Partial and complete occlusions happened frequently due to traffic density. Moreover, these occlusions often happened near merge points which complicated decision making.

During extensive testing, the performance of our vehicle tracking module has been very reliable and efficient (see Fig. 13). Geometric shape of vehicles was properly estimated (see Figs. 14 and 15), which increased tracking reliability and improved motion estimation. The tracking approach proved capable of handling complex traffic situations such as the one presented on course A of the UGC. The computation time of our approach averages at 25ms per frame, which is faster than real time for most modern laser range finders.

We also gathered empirical results of the tracking module performance on data sets from several urban environments: course A of the UGC, Stanford campus and a port town in Alameda, CA. In each frame of data, we labeled the vehicles a human is able to identify in the laser range data. The vehicles had to be within 50m of the ego-vehicle, on or near the road, and moving with a speed of at least 5mph. We summarize how the tracker performed on the labeled data sets in Tbl. 1. Note that the maximum theoretically possible true positive rate is lower than 100% because three frames are required to detect a new vehicle. On all three data sets the tracker performed very close to the theoretical bound. Overall the true positive rate was 97% compared to the theoretical maximum of 98%.



(a) standard PF



(b) Scaling Series

Fig. 16 Comparison of standard PF to Scaling Series for new vehicle detection. The horizontal axis denotes time in seconds. The vertical axis has two states: 0 — target is not tracked, 1 — target is tracked. To verify target acquisition, the code was specifically modified to discontinue tracking a target after 1 second. By construction of the algorithm, the minimum possible time spent in non-tracking state is 0.3 seconds. (a) standard PF has a long target acquisition time — too dangerous for autonomous driving. (b) Scaling Series method has nearly perfect acquisition time.

7.2 Detection Results

To evaluate the performance of the vehicle detection algorithm empirically, we forced the tracking module to drop each target as soon as it was detected. We then ran vehicle detection on data sets from three different urban environments: Area A of the Urban Grand Challenge qualifiers, the Stanford campus, and a port town in Alameda, CA (see Tbl. 2). In each frame of data we labeled all vehicles identifiable by a human in the range data. The vehicles had to be within 50m of Junior, on or near the road, and moving with a speed of at least 5mph. For each vehicle, we counted how many frames it took to detect it. We also counted false positives. Overall, all vehicles were detected in five frames or less and the false positive rate was 0.4%.

To evaluate motion evidence contribution, we ran the algorithm with and without motion evidence logic on labeled data sets. The use of motion evidence brought false discovery rate from 60% down to 0.4%. At the same time the rate of false negatives did not increase.

We used prerecorded data sets to evaluate performance gains from the optimization techniques. We compared the computation time of the algorithm with and without road masking. Road masking sped up the algorithm by a factor of eight. We also ran the algorithm with and without cleared area logic. The speed up from this optimization was approximately a factor of three. The backward search optimization reduced the minimum detection delay for oncoming traffic by 25%.

To evaluate improvements from Scaling Series, we used a 30 second data set of our ego-vehicle following another car. For evaluation purposes we modified the tracker to drop each target after tracking it for 1 second. Figure 16 presents comparison of results obtained using a standard particle filter and Scaling Series particle filter. Vehicle detection with the standard particle filter took 4.44 seconds on average and 13.7 seconds in the worst case, which can easily result in a collision in a real life situation. In contrast the Scaling Series particle filter took 0.32 seconds on average to detect the vehicle, with the worst case being 0.5 seconds. Thus, the Scaling Series approach performs very close to the theoretical minimum of 0.3 seconds.

Several videos of vehicle detection and tracking using the techniques presented in this paper are available at the website

http://cs.stanford.edu/people/petrovsk/uc.html

8 Conclusions

We have presented the vehicle detection and tracking module developed for Stanford's autonomous driving robot Junior. Tracking is performed from a high-speed moving platform and relies on laser range finders for sensing. Our approach models both dynamic and geometric properties of the tracked vehicles and estimates them with a single Bayes filter per vehicle. In contrast to prior art, the common data segmentation and association steps do not need to be carried out prior to the filtering step. The approach has proved reliable, efficient and capable of handling challenging traffic situations, such as the ones presented at the Urban Grand Challenge.

Our approach explicitly models tracked vehicle's geometric shape, which is estimated simultaneously with the vehicle's motion using an efficient RBPF method. The introduced anchor point notion allows us to correctly model the shape vs motion ambiguity, previously unaddressed in vehicle tracking literature. This reduces motion uncertainty and improves the estimation of vehicle dynamics.

Unlike prior vehicle tracking approaches, which relied on features for tracking, we introduced a direct measurement model for range scans. This approach eliminates the need for data segmentation and association steps. Moreover it naturally handles partial occlusions of the tracked vehicles, including situations where the vehicle scan is split up into multiple disjoint clusters by occlusion.

We presented a number of optimization techniques to improve accuracy and efficiency of vehicle detection. These techniques are largely independent of each other. To aid the design decisions of future vehicle tracking approaches, we provided an analysis of how each technique influences the end result.

We presented techniques for efficient manipulation of 3D data clouds and construction of 2D virtual sensor models. The method relies on a ground estimation technique,

Table 2 Vehicle detector performance on data sets from three urban environments. For each car we counted how many frames it took to detect it. By construction of the algorithm, at least three frames are required. We also counted the number of false detections. The '% Detected' columns give the percentages of cars detected by frame three, four and five. 'FP %' is the false positive rate attained by the vehicle detection algorithm.

	Total	Detected in Frame			False	% Detected by Frame			FP
Data Sets	Cars	3	4	5	Detections	3	4	5	%
UGC Area A	713	596	103	14	1	83.6	98.0	100.0	0.1
Stanford Campus	679	645	32	2	2	95.0	99.7	100.0	0.3
Alameda Day 2	532	485	45	2	5	91.2	99.6	100.0	0.9
Overall	1,924	1,726	180	18	8	89.7	99.1	100.0	0.4

which we expect to be applicable not only in urban environments but also in off-road settings with rugged terrain. The method purposefully ignores short obstacles in an effort to extract data useful specifically for vehicle tracking. As a result, detection and tracking of vehicles is unimpeded by curbs and short foliage present in urban settings, or even small rocks and rough features in completely off-road environments. It also ignores overhanging obstacles — such as trees, signs, and overpasses — if there is sufficient clearance for a vehicle to pass underneath. However, due to the fact that short obstacles are ignored, the presented data extraction method is not suitable for estimation of terrain drivability. For this reason, we used a separate method for detection of small hazards as described in Montemerlo et al (2008).

We also introduced a method for detection of poorly visible black objects in 3D range data. This method is applicable not only for vehicle tracking but also for static mapping and collision avoidance. Moreover it can be extended to dark object detection using 3D range scanners in indoor settings.

There is ample room for future work in the field of perception for autonomous urban driving. The presented approach does not model pedestrians, bicyclists, or motorcyclists — a prerequisite for driving in populated areas. Another promising direction for future work is fusion of different sensors, including laser, radar and vision.

Acknowledgements This research has been conducted for the Stanford Racing Team and would have been impossible without the whole team's efforts to build the hardware and software that makes up the team's robot Junior. The authors thank all team members for their hard work. The Stanford Racing Team is indebted to DARPA for creating the Urban Challenge, and for its financial support under the Track A Program. Further, Stanford University thanks its various sponsors. Special thanks also to NASA Ames for permission to use their air field.

References

Blackman S, Co R, El Segundo C (2004) Multiple hypothesis tracking for multiple target tracking. IEEE Aerospace and Electronic Systems Magazine 19(1 Part 2):5–18

Blais F (2004) Review of 20 years of range sensor development. Journal of Electronic Imaging 13:231

Buehler M, Iagnemma K, Singh S (2007) The 2005 Darpa Grand Challenge: The Great Robot Race. Springer Verlag

Burgard W, Fox D, Hennig D, Schmidt T (1996) Estimating the absolute position of a mobile robot using position probability grids. In: Proceedings of the National Conference on Artificial Intelligence, pp 896–901

Darms M, Rybski P, Urmson C, Inc C, Auburn Hills M (2008) Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In: Intelligent Vehicles Symposium, 2008 IEEE, pp 1197–1202

DARPA (2007) Urban challenge rules, revision oct. 27, 2007. URL http://www.darpa.mil/grandchallenge/rules.asp

Dellaert F, Thorpe C (1998) Robust car tracking using kalman filtering and bayesian templates. In: Proceedings of SPIE, vol 3207, p 72

Dickmanns E (1998) Vehicles capable of dynamic vision: a new breed of technical beings? Artificial Intelligence 103(1-2):49–76

Dietmayer K, Sparbert J, Streller D (2001) Model based object classification and object tracking in traffic scenes from range images. In: Proceedings of IV 2001, IEEE Intelligent Vehicles Symposium, pp 25–30

Doucet A, Freitas Nd, Murphy K, Russell S (2000) Rao-blackwellised filtering for dynamic bayesian networks. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, pp 176–183

Fox D, Burgard W, Thrun S (1999) Markov localization for mobile robots in dynamic environments. Journal of Artificial Intelligence Research 11(3):391–427

Gerkey B, Vaughan R, Howard A (2003) The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proceedings of the 11th International Conference on Advanced Robotics, pp 317–323

Ibeo Automobile Sensor GmbH (2008) IBEO AlascaXT brochure. URL http://www.ibeo-as.com/english/press_downloads_brochures.asp

Kabadayi S, Pridgen A, Julien C (2006) Virtual sensors: Abstracting data from physical sensors. In: Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks, IEEE Computer Society Washington, DC, USA, pp 587–592

Leonard J, How J, Teller S, Berger M, Campbell S, Fiore G, Fletcher L, Frazzoli E, Huang A, Karaman S, et al (2008) A perception-driven autonomous urban vehicle. Journal of Field Robotics 25(10):727–774, DOI 10.1002/rob.20262

Mäkynen A (2000) Position-sensitive devices and sensor systems for optical tracking and displacement sensing applications. PhD thesis, Department of Electrical Engineering, Oulu University, Oulu, Finland

Mitchell T, Hutchinson R, Just M, Newman S, Stefan R, Francisco N, Wang PX (2002) Machine learning of fmri virtual sensors of cognitive states. In: In The 16th Annual Conference on Neural Information Processing Systems, Computational Neuroimaging: Foundations, Concepts and Methods Workshop, 2002. 87, MIT Press

Montemerlo M (2003) Fastslam: A factored solution to the simultaneous localization and mapping problem with unknown data association. PhD thesis, Robotics Institute, Carnegie Mellon University

Montemerlo M, Becker J, Bhat S, Dahlkamp H, Dolgov D, Ettinger S, Haehnel D, Hilden T, Hoffmann G, Huhnke B, et al (2008) Junior: The Stanford entry in the Urban Challenge. Journal of Field Robotics 25(9):569–597

Moravec H (1988) Sensor fusion in certainty grids for mobile robots. AI Magazine 9(2):61

Petrovskaya A, Thrun S (2008a) Efficient techniques for dynamic vehicle detection. In: ISER, Athens, Greece

Petrovskaya A, Thrun S (2008b) Model based vehicle tracking for autonomous driving in urban environments. In: RSS, Zurich, Switzerland

Petrovskaya A, Khatib O, Thrun S, Ng AY (2006) Bayesian estimation for autonomous object manipulation based on tactile sensors. In: Robotics and Automation. Proceedings of IEEE International Conference, pp 707–714

Särkkä S, Vehtari A, Lampinen J (2007) Rao-blackwellized particle filter for multiple target tracking. Inf Fusion 8(1)

Schulz D, Burgard W, Fox D, Cremers A (2001) Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In: IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA, vol 2

Sick Optics (2003) LMS 200/211/220/221/291 laser measurement systems technical description. URL http://www.sickusa.com/Publish/docroot/Manual(s)/LMSTechnicalDescription.pdf

Srivastava A, Oza N, Stroeve J, Aeronaut N, Center S, Moffett Field C (2005) Virtual sensors: Using data mining techniques to efficiently estimate remote sensing spectra. IEEE Transactions on Geoscience and Remote Sensing 43(3):590–600

Streller D, Furstenberg K, Dietmayer K (2002) Vehicle and object models for robust tracking in traffic scenes using laser range images. In: The IEEE 5th International Conference on Intelligent Transportation Systems, 2002. Proceedings, pp 118–123

Thalmann D, Noser H, Huang Z (1997) Autonomous virtual actors based on virtual sensors. Lecture Notes In Computer Science 1195:25–42

Thrun S (2001) A probabilistic on-line mapping algorithm for teams of mobile robots. The International Journal of Robotics Research 20(5):335

Thrun S, Burgard W, Fox D (2005) Probabilistic Robotics. MIT Press

Urmson C, Anhalt J, Bagnell D, Baker C, Bittner R, Clark M, Dolan J, Duggins D, Galatali T, Geyer C, et al (2008) Autonomous driving in urban environments: Boss and the urban challenge. Journal of Field Robotics 25(8):425–466

Velodyne Lidar, Inc (2008) High definition lidar HDL-64E S2 specifications. URL http://www.velodyne.com/lidar/products/specifications.aspx

Wang C, Thorpe C, Thrun S, Hebert M, Durrant-Whyte H (2007) Simultaneous localization, mapping and moving object tracking. The International Journal of Robotics Research 26:889–916

Wang CC (2004) Simultaneous localization, mapping and moving object tracking. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA

Wender S, Dietmayer K (2008) 3d vehicle detection using a laser scanner and a video camera. Intelligent Transport Systems, IET 2(2):105–112

Zhao L, Thorpe C (1998) Qualitative and quantitative car tracking from a range image sequence. In: 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings, pp 496–501

Zielke T, Brauckmann M, von Seelen W (1993) Intensity and edge-based symmetry detection with an application to car-following. CVGIP: Image Understanding 58(2):177–190