Model Based Vehicle Tracking for Autonomous Driving in Urban Environments

Anna Petrovskaya and Sebastian Thrun
Computer Science Department
Stanford University
Stanford, California 94305, USA
{ anya, thrun }@cs.stanford.edu

***Abstract*— Situational awareness is crucial for autonomous driving in urban environments. This paper describes moving vehicle 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 tracking module provides reliable 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 also show how to build 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 UGC as well as other urban settings.**

I. 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. DARPA has organized a series of competitions for autonomous vehicles. In 2005, autonomous vehicles were able to complete a 131 mile course in the desert. In 2007 competition, the Urban Grand Challenge, 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. One of the most significant changes from the previous competition is that for urban driving, robots need to have situational awareness of both static and dynamic parts of the environment. Our robot won the second prize at the 2007 competition. In this paper we describe the approach we developed for tracking of moving vehicles.

Vehicle tracking has been studied for several decades. A number of approaches focused on the use of vision exclusively [1, 2, 3]. Whereas others utilized laser range finders sometimes in combination with vision [4, 5, 6, 7]. Typically these approaches perform data segmentation and data association prior to performing a filter update. Usually only position and velocity of each vehicle are tracked. The vehicle tracking literature almost universally relies on variants of Kalman filters, although particle filters and hybrid approaches have been widely used in other tracking applications [8, 9, 10].

For our application we are concerned with laser based vehicle tracking from our autonomous robotic platform Junior,

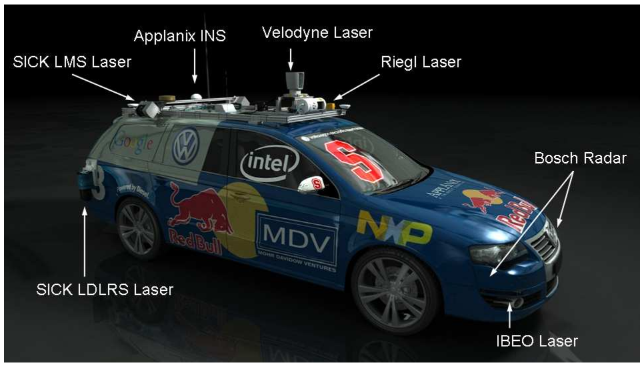


Fig. 1. Junior, our entry in the DARPA Urban Challenge. Junior is equipped with five different laser measurement systems, a multi-radar assembly, and a multi-signal inertial navigation system, as shown in this figure.

to which we will also refer as the ego-vehicle (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 naturally handles data segmentation and association, so that these pre-processing steps are not required. To properly model the dependence between geometric and dynamic vehicle properties, we introduce *anchor point coordinates*. Further, we introduce an abstract sensor representation we call the *virtual scan*, that allows for efficient computation and can be used for a wide variety of laser sensors. We present techniques for building virtual scans from 3D range data and show how to detect poorly visible black vehicles in laser scans. 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 challenging traffic situations presented at the Urban Grand Challenge.

II. Representation

A. Probabilistic model and notation

Our goal for this work is to track multiple vehicles in an urban environment. Our ego-vehicle has been outfitted with the Applanix navigation system that can provide pose localization with 1m error as well as produce a locally consistent pose estimates based on an inertial measurement unit (IMU). Hence we will leave ego-vehicle localization outside the scope of the paper. Instead we will assume that a reasonably precise pose of the ego-vehicle is always available.

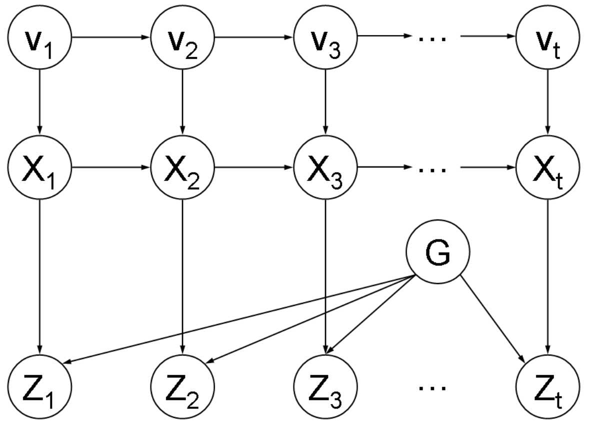


Fig. 2. Dynamic Bayesian network model of the tracked vehicle pose X_t, forward velocity v_t, geometry G, and measurements Z_t.

From the theoretical standpoint, multiple vehicle tracking entails a single joint probability distribution over the state parameters of all of the vehicles. Unfortunately, such a representation is not practical because it quickly becomes intractable as the number of vehicles grows. Also, since the number of vehicles is unknown and variable, it is in fact challenging to model the problem in this way. We note that dependencies between vehicles are strong when vehicles are close together, but become extremely weak as the distance between vehicles increases. Hence it is wasteful to model dependencies between vehicles that are far from each other. Instead, following the common practice in vehicle tracking, we will represent each vehicle with 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 there is a free space of at least 1m around each vehicle and that all vehicles of interest are located on or near the road. This representation is efficient because its complexity grows linearly with the number of vehicles. It also easily accommodates a variable number of tracked vehicles.

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. II-B). Also at each time step we obtain a new measurement Z_t. See Fig. 2 for a dynamic Bayes network representation of the resulting probabilistic model. The dependencies between the parameters involved are modeled via probabilistic laws discussed in detail in Sects. II-C and II-E. For now we briefly note that the velocity evolves over time according to

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

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

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

The measurements are governed by a measurement model:

$$p(Z_t|X_t, G).$$

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.

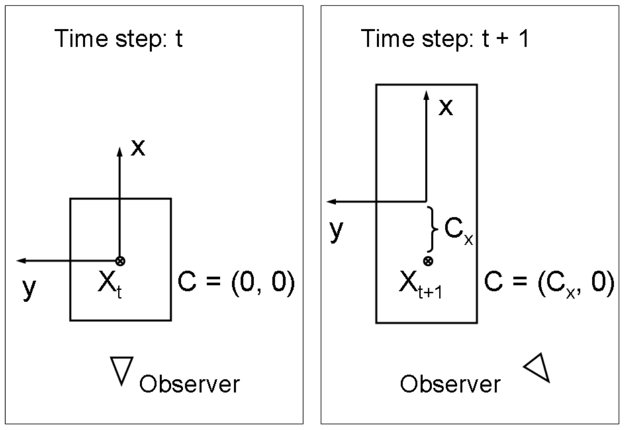


Fig. 3. As we move to observe a different side of a stationary car, our belief of its shape changes and so does the position of the car's center point. To compensate for the effect, we introduce local anchor point coordinates $C = (C_x, C_y)$ so that we can keep the anchor point X_t stationary in the world coordinates.

B. 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 the vehicles is not important for driving applications.

For vehicle tracking it is common to track the position of a vehicle's center within the state variable X_t. However, there is an interesting dependence between our belief about the vehicle's shape and position (Fig. 3). 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.

To overcome this problem, we view X_t as the pose of an *anchor point* who's 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 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 $G = (W, L, C_x, C_y)$.

C. Vehicle dynamics model

Given a vehicle's velocity v_{t-1} at time step $t-1$, the velocity evolves via addition of random bounded noise based on maximum allowed acceleration a_{max} and the time delay Δt between time steps $t-1$ and t. Specifically, we sample Δv uniformly from $[-a_{max}\Delta t,\ a_{max}\Delta t]$.

The pose evolves via linear motion - 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 $d\theta_{max}$.

D. Sensor data representation

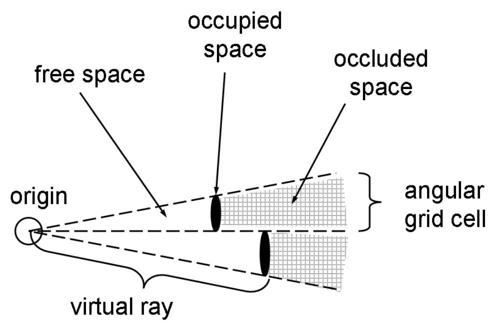
In this paper we focus on laser range finders for sensing the environment. Recently these sensors have evolved to be more suitable for driving applications. For example IBEO Alasca sensors allow for easy ground filtering by collecting four parallel horizontal scan lines and marking which of the readings are likely to come from the ground. Velodyne HDL-64E sensors do not provide ground filtering, however they take a 3D scan of the environment at high frame rates (10Hz) thereby producing 1,000,000 readings per second. 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 representation tailored for vehicle tracking.

To expedite computations, we construct a grid in polar coordinates - a *virtual scan* - which subdivides 360° around a chosen origin point into angular grids (Fig. 4). In each angular grid we record the range to the closest obstacle. Hence each angular grid contains information about free, occupied, and occluded space. We will often refer to the cone of an angular grid from the origin until the recorded range as a *ray* due to its similarity to a laser ray.

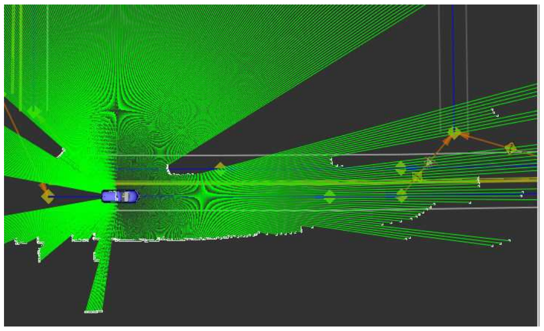
Virtual scans simplify data access by providing a single point of origin for the entire data set, which allows constant time look-up for any given point in space. As we mentioned earlier it is important to compute correct world coordinates for the raw sensor readings. However, once the correct positions 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. 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. II-E.

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. 4(d) shows results of a virtual scan differencing operation with red points denoting new obstacles, green points denoting obstacles that disappeared,

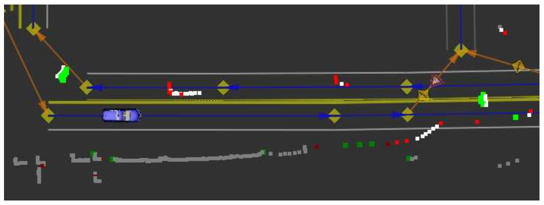


(a) schematic of a virtual scan

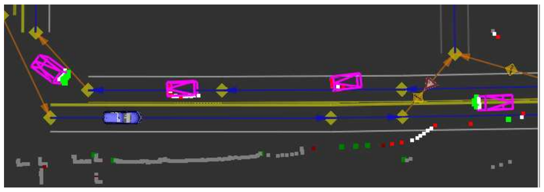
(b) actual scene



(c) virtual scan

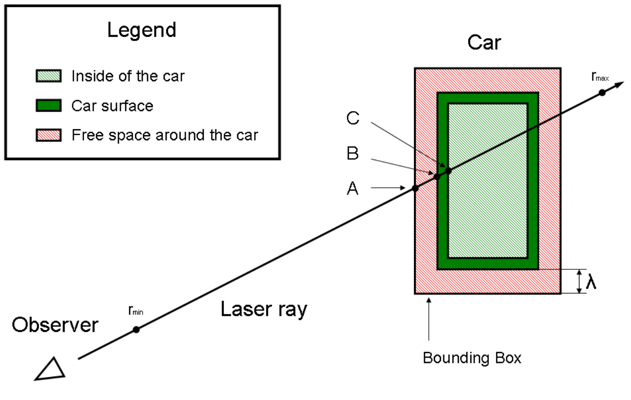


(d) scan differencing

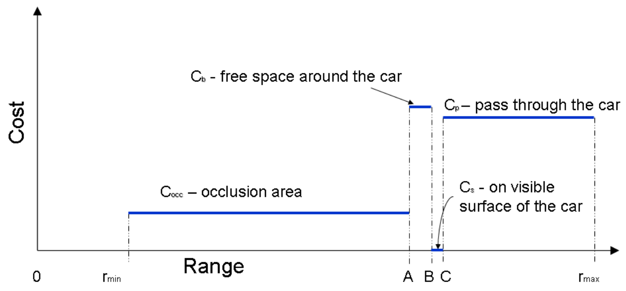


(e) tracking results

Fig. 4. Virtual scan construction. In (c) green line segments represent virtual rays. In (d) red points are new obstacles, green points are obstacles that disappeared, and white points are obstacles that remained unchanged. In (e) the purple boxes denote the tracked vehicles. (Best viewed in color.)



(a)



(b)

Fig. 5. Measurement likelihood computations. (a) shows the geometric regions involved in the likelihood computations. (b) shows the costs assignment for a single ray.

and white points denoting obstacles that remained in place or appeared in previously occluded areas.

E. Measurement model

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[1] around the vehicle (see Fig. 5). 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.

Following the common practice for modeling laser range finders, we consider measurements obtained along each ray independent of each other. Thus if we have a total of N rays in the virtual scan Z, the measurement likelihood factors as follows:

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

We model each ray's likelihood as a zero-mean Gaussian of variance σ_i computed with respect to a cost c_i selected based on the relationship between the ray and the vehicle (η_i is a normalization constant):

$$P(z_i|G,X) = \eta_i \; \exp\{ \, -\frac{c_i^2}{\sigma_i^2} \, \}.$$

The costs and variances are set to constants that depend on the region in which the reading falls into (see Fig. 5 for illustration). c_{occ}, σ_{occ} are the settings for range readings that fall short of the bounding box and thus represent situations when another object is occluding the vehicle. c_b and σ_b are the settings for range readings that fall short of the vehicle but inside of the bounding box. c_s and σ_s are the settings for readings on the vehicle's visible surface (that we assume to be of non-zero depth). c_p, σ_p are used for rays that extend beyond the vehicle's surface.

The domain for each range reading is between minimum range r_{min} and 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. In contrast these cases are often challenging for approaches that utilize separate data segmentation and correspondence methods.

III. Inference

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 in our case two targets can in principle merge into one. In practice we have found that it 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 Rao-Blackwellized particle filters (RBPFs) first introduced in [11]. 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.

A. 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

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

geometry based on a set of measurements:

$$Bel_t = p(X^t, v^t, G|Z^t).$$

The derivation provided below is similar to the one used in [12]. We split up the belief into two conditional factors:

$$Bel_t = p(X^t, v^t|Z^t) \;\; p(G|X^t, v^t, Z^t).$$

The first factor encodes the vehicle's motion posterior:

$$R_t = p(X^t, v^t|Z^t).$$

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

$$S_t = p(G|X^t, v^t, Z^t).$$

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). We denote a particle by $q^t_m = (X^{t,[m]}, v^{t,[m]}, S^{[m]}_t)$ and a collection of particles at time t by $Q_t = \{q^t_m\}_m$. 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. II-C). Thus, particles in $\bar{Q}_t$ are distributed according to the vehicle motion prediction distribution:

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

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: $q^t_m = (X^{t,[m]}, v^{t,[m]}, \mu^{[m]}_t, \Sigma^{[m]}_t)$. We have:

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

The first step above follows from Bayes' rule; the second step follows from the conditional independence assumptions of our model (Fig. 2). The expression (1) 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. III-C. 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 [12], it is straightforward to show that the importance weights w_t should be:

$$w_t = R_t/\bar{R}_t = \frac{p(X^t, v^t|Z^t)}{p(X^t, v^t|Z^{t-1})} = I\!E_{S_{t-1}}[\; p(Z_t|G, X_t) \;].$$

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.

B. Motion inference

As we mentioned in Sect. II-A, 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:

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

The first and second steps above are simple conditional factorizations; the third step follows from the conditional independence assumptions of our model (Fig. 2).

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.

C. 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. II-E) 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 position (discussed in Sect. II-B) 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.

D. Initializing new tracks

Before vehicle tracking can begin, we need to initialize new vehicle tracks. Detection of new vehicles is the most expensive part of vehicle tracking. However a number of optimizations can be made to achieve detection in real time, including spatial constraints and sensor data analysis. Detailed description of our vehicle detection algorithm is given in [13]. Here we provide a brief summary.

To detect new vehicles, we search the area within sensor range of our ego-vehicle to find good matches using the measurement model described in Sect. II-E. A total of three (3) frames are required to acquire a new tracking target. The

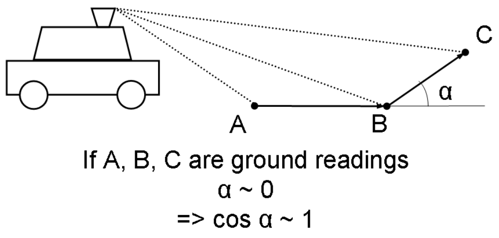


Fig. 6. We determine ground readings by comparing angles between consecutive readings.

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 vehicle dynamics laws described in Sect. II-C.

E. Discontinuing tracks

Under certain conditions it is desirable to discontinue tracking of a target. We discontinue tracks if the target vehicle gets out of sensor range or moves too far away from the road (a digital street map was available for our application). Additionally we implemented logic that merged hypothesis of two particle filters if the tracked targets were too close together. However, it turned out that this condition occurs only very rarely.

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.

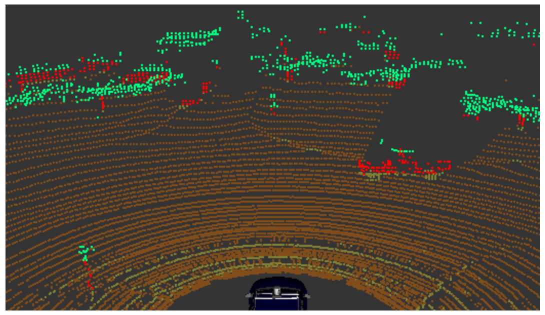
IV. Implementation and Results

A. Building virtual scans from 3D range data

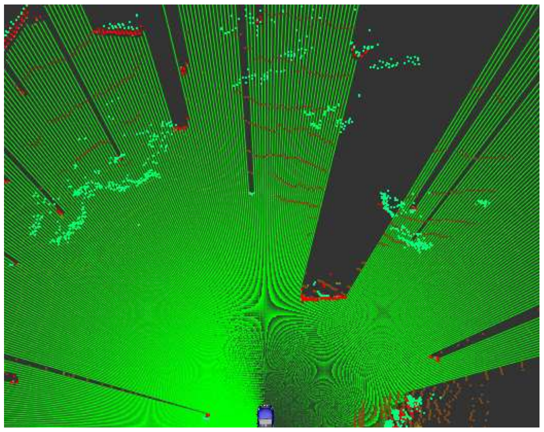
As we explained in Sect. II-D, vehicle tracking is a 2D problem, for which efficient 2D virtual scans are sufficient. These virtual scans are easy to build for 2D range sensors with ground filtering, such as IBEO. However for 3D sensors, such as Velodyne, it is a less trivial task. These sensors provide immense 3D data sets of the surroundings, making computational efficiency a high priority when processing the data. However, in our experience, the hard work pays off and the resulting virtual scans carry more information than for 2D sensors.

Given a 3D data set, which of the 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 about 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 otherwise 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.

(a) actual scene



(b) Velodyne data after classification



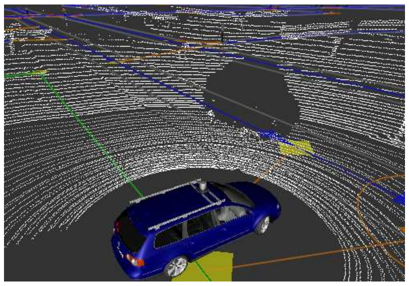
(c) generated virtual scan

Fig. 7. In (b) Velodyne data is colored by type: orange - ground, yellow - low obstacle, red - medium obstacle, green - high obstacle. Note the white van parked at a distance in (a) and (c).

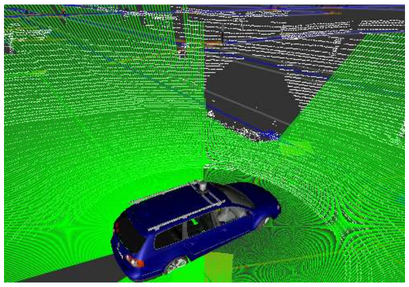
In order to classify the data into the different types of objects described above we first build a 3D grid in spherical coordinates. Similarly to a virtual scan, it has a single point of origin and stores actual world coordinates of the sensor readings. Just as in the 2D case, this grid is an approximation 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 it. 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

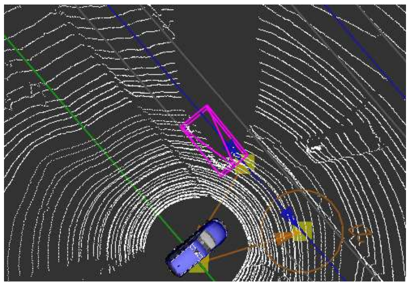
(a) actual appearance of the vehicle



(b) the vehicle gives very few laser returns



(c) generated virtual scan after black object detection



(d) successful tracking of the black vehicle

Fig. 8. Detecting black vehicles in 3D range scans. White points represent raw Velodyne data. In (c) green lines represent the generated virtual scan. In (d) the purple box denotes the estimated pose of the tracked vehicle.

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. 6 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.

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. 7). 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.

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 8 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 that is present, we can detect the presence of a black obstacle by looking at the data that is absent. If no readings are obtained along a range of vertical angles in a specific direction, we can conclude that the space must be occupied 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.

B. *Tracking results*

The most challenging traffic situation at the Urban Grand Challenge was presented on course A during the qualifying event (Fig. 9(a) and Fig. 9(b)) . 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 that 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 for false associations. Partial and complete occlusions happened frequently due to the traffic density. Moreover these occlusions often happened near merge points which complicated decision making.

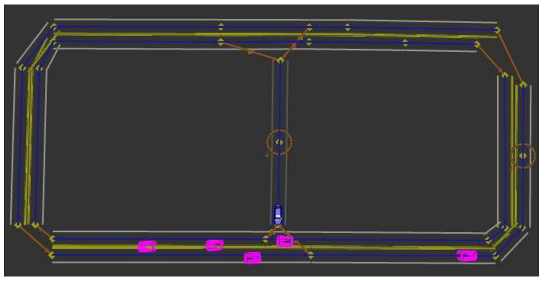
During extensive testing the performance of our vehicle tracking module has been very reliable and efficient (see Fig. 4 and Fig. 9). It proved capable of handling complex traffic situations such as the one presented on course A. The computation time of our approach averages out at 25ms per frame, which 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. For each frame of data we counted how many 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 the tracker's performance in Fig. 10. Note that the maximum theoretically possible true positive rate is lower

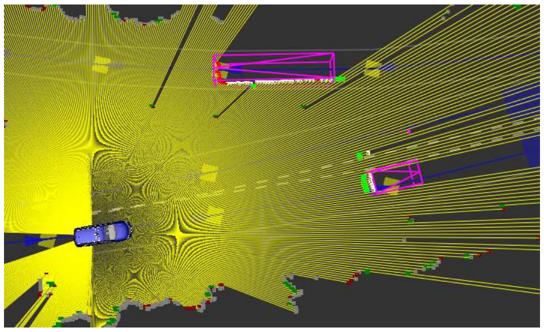
Datasets	Total Frames	Total Vehicles	Correctly Detected	Falsely Detected	Max TP (%)	TP (%)	FP (%)
Area A	1,577	5,911	5,676	205	97.8	96.02	3.35
Stanford	2,140	3,581	3,530	150	99.22	98.58	4.02
Alameda	1,531	901	879	0	98.22	97.56	0
Overall	5,248	10,393	10,085	355	98.33	97.04	3.3

Fig. 10. 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.

(a) test conditions on course A at the UGC



(b) Junior at intersection on course A



(c) vehicle size estimation on Stanford campus

Fig. 9. Test conditions and results of tracking. Purple boxes represent tracked vehicles. In (c) yellow lines represent the virtual scan.

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%.

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

http://cs.stanford.edu/$\sim$anya/uc.html

V. 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 are carried out as part of the filter itself. The approach has proved reliable, efficient and capable of handling challenging traffic situations, such as the ones presented at the Urban Grand Challenge.

Clearly there is ample room for future work. The presented approach does not model pedestrians, bicyclists, or motorcyclists, which is a prerequisite for driving in populated areas. Another promising direction for future work is fusion of different sensors, including laser, radar and vision.

Acknowledgment

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

[1] T. Zielke, M. M. Brauckmann, and W. von Seelen. Intensity and edge based symmetry detection applied to car following. In *ECCV, Berlin, Germany*, 1992.

[2] E. Dickmanns. Vehicles capable of dynamic vision. In *IJCAI, Nagoya, Japan*, 1997.

[3] F. Dellaert and C. Thorpe. Robust car tracking using kalman filtering and bayesian templates. In *Conference on Intelligent Transportation Systems*, 1997.

[4] L. Zhao and C. Thorpe. Qualitative and quantitative car tracking from a range image sequence. In *Computer Vision and Pattern Recognition*, 1998.

[5] D. Streller, K. Furstenberg, and K. Dietmayer. Vehicle and object models for robust tracking in traffic scenes using laser range images. In *Intelligent Transportation Systems*, 2002.

[6] C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research, Sep 2007; vol. 26: pp. 889-916*, 2007.

[7] S. Wender and K. Dietmayer. 3d vehicle detection using a laser scanner and a video camera. In *6th European Congress on ITS in Europe, Aalborg, Denmark*, 2007.

[8] D. Schulz, W. Burgard, D. Fox, and A. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *ICRA*, 2001.

[9] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE AE Systems Magazine*, 2004.

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

[11] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-blackwellised filtering for dynamic bayesian networks. In *UAI, San Francisco, CA*, 2000.

[12] M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2003.

[13] A. Petrovskaya and S. Thrun. Efficient techniques for dynamic vehicle detection. In *ISER, Athens, Greece*, 2008.