CARLOC: Precisely Tracking Automobile Position*

Yurong Jiang†, Hang Qiu†, Matthew McCartney†, Gaurav Sukhatme†,
Marco Gruteser*, Fan Bai‡, Donald Grimm‡, Ramesh Govindan†
†University of Southern California
*Rutgers University ‡GM Global Research & Development
{yurongji,hangqiu,mmcartn,gaurav,ramesh}@usc.edu
gruteser@winlab.rutgers.edu
{fan.bai,donald.grimm}@gm.com

ABSTRACT

Precise positioning of an automobile to within lane-level precision can enable better navigation and context-awareness. However, GPS by itself cannot provide such precision in obstructed urban environments. In this paper, we present a system called CARLOC for lane-level positioning of automobiles. CARLOC uses three key ideas in concert to improve positioning accuracy: it uses digital maps to match the vehicle to known road segments; it uses vehicular sensors to obtain odometry and bearing information; and it uses crowd-sourced location estimates of roadway landmarks that can be detected by sensors available in modern vehicles. CARLOC unifies these ideas in a probabilistic position estimation framework, widely used in robotics, called the sequential Monte Carlo method. Through extensive experiments on a real vehicle, we show that CARLOC achieves sub-meter positioning accuracy in an obstructed urban setting, an order-of-magnitude improvement over a high-end GPS device.

Categories and Subject Descriptors

J.7 [**Computers in Other Systems**]: Consumer Products; H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Design; Experimentation; Performance; Algorithms

Keywords

GPS; Map; Accuracy

1. INTRODUCTION

As mobile devices have proliferated, they have become the defacto method for estimating the position of automobiles. The built-in GPS receiver in mobile devices provides positioning for navigation, but also for context-awareness; many apps now routinely use vehicle position to suggest nearby services or points of interest. As elements of autonomous driving start to appear in commercial offerings, the accuracy of positioning vehicles will become much more important.

However, it has long been known that smartphone GPS receivers have errors on the order of 10s of meters, especially in obstructed urban environments. It is precisely in these environments, unfortunately, where accurate positioning is most necessary because of the density of services or points of interest. An order of magnitude lower positioning error of automobiles would be able to position a vehicle with up to lane-level accuracy, which will likely enable much more accurate navigation, but also more precise context-awareness in urban environments [33].

Much research (Section 5) has explored how to enhance GPS position by fusing information from other sensors such as laser-range finders and inertial sensors and from other sources, such as digital maps. Intuitively, maps can be used to constrain vehicle trajectories, inertial sensors can be used for dead reckoning when GPS is unavailable, and laser range-finders can estimate distances to landmarks in the environment, which can then be used to get a position fix.

In this paper, we explore two dimensions in this design space that can help significantly improve positioning accuracy. First, we observe that modern automobiles have hundreds of sensors that govern the operation of their internal subsystems, and some of these sensors provide odometry and heading information. These can be used to improve the efficacy of matching a car's location to a digital map, and to model its motion. Second, car sensors can also provide enough information to detect *roadway landmarks* — roadway features such as potholes or speedbumps. If these can be reliably detected, then the *position estimates of other cars at these landmarks can be used to improve a car's position estimate.*

Contributions. In this paper, we design and evaluate a system called CARLOC (Section 3) that can continuously track the precise position of a vehicle, even in highly obstructed environments. CARLOC uses a collection of techniques, some of which are inspired by prior work on robot localization, while others use existing techniques by adapt them to use car sensors, and some are novel. Specifically, CARLOC uses a non-parametric probabilistic position representation, called a *particle filter*, as a uniform framework that is able to express various forms of information fusion. CARLOC matches a car's current position estimate to a road-segment on a map. This matching, whose accuracy we improve by leveraging the availability of vehicle sensors, can be used to truncate the position uncertainty to within the nominal road width of the matched segment. CARLOC then updates the particle filter using a well-known kinematic model, but uses car sensors to accurately estimate inputs to the kinematic model.

* The first author, Yurong Jiang, was supported by Annenberg Graduate Fellowship. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1330118.

SenSys'15, November 01–05, 2015, Seoul, Republic of Korea.

ACM ISBN 978-1-4503-3631-4/15/11/.../$15.00.
DOI: http://dx.doi.org/10.1145/2809695.2809725 .

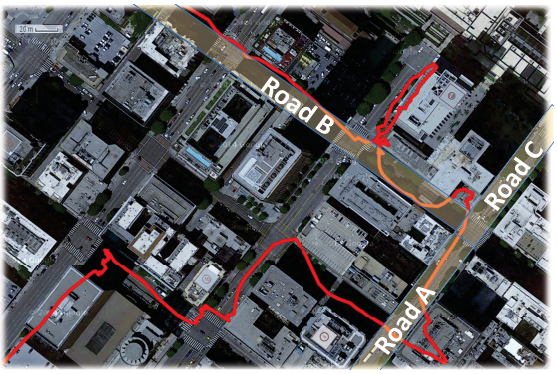

Figure 1—Portion of GPS Trace in City Downtown

A particularly novel contribution of CARLOC is the ability to enhance position estimates of a vehicle using crowd-sourced position estimates of roadway landmarks. To understand this, suppose a car hits a speed bump. If CARLOC is able to detect the speed bump, then *the car's particle filter at the instant the speed bump is encountered, is a probabilistic representation of the speed bump's position.* Suppose N cars pass over the same speed bump, the collection of all their particle filters at the speed bump represents a *crowd-sourced* collection of position estimates of the speed bump. Intuitively, one expects the distribution described by these crowd-sourced particles *to converge to the true location of the speed bump* as more and more vehicles contribute to the collection. CARLOC uses this observation and contains novel algorithms to detect three types of roadway landmarks (stop signs, speed bumps, and street corners) and to update particle filters.

Using extensive evaluations (Section 4) on roads with varying degree of satellite obstructions (and therefore various degrees of GPS availability and accuracy), we show that CARLOC has mean error of 2.7m in a highly obstructed downtown road, an order of magnitude improvement over commodity GPS, high-precision GPS receivers, differential GPS, and the closest prior work on GPS augmentation using mobile devices. In unobstructed environments, CARLOC's mean position error drops to 1.38m, while in partially obstructed environments, the mean error can vary between 1.1m and 2.2m. CARLOC's position error does not appear to depend on length of route, and a relatively small number of landmarks suffices to achieve significant accuracy. More important, each component of our design, and each optimization contributes significantly to the design.

2. BACKGROUND AND MOTIVATION

Positioning accuracy for automobiles. Over the last few years, the use of in-car navigation has increased significantly. This has been driven, in part, by the ubiquitous availability of free navigation apps on mobile devices such as smartphones and tablets. The commodity GPS receivers on these devices (that the navigation apps rely on) can be highly inaccurate in some settings. For example, Figure 1 shows GPS readings from a city downtown area, where the GPS signal reception is affected significantly by the obstructions caused by tall buildings, a well-known effect sometimes called the urban canyon effect.

To quantify the degree of error in GPS, we obtained smartphone GPS readings from nearly 200 miles of driving, on three different types of roads: *Urban roads*, e.g. a downtown road surrounded by tall buildings, *Shaded roads*, e.g. roads covered by trees, and *Opensky roads*, e.g. highway or major roads having an unobstructed view of the sky. Table 1 shows statistics for GPS errors from our traces. Although we obtain reasonably good GPS location accuracy on open sky roads, the accuracy degrades sharply on shaded and urban roads, with over 15 meters of error on average, and over 90 meters in some cases. This is consistent with other work that has observed similar errors in obstructed environments [4].

	Urban Area	Shaded Area	Opensky Area
Average Error (m)	24.3	15.3	4.7
Error STD (m)	5.5	3.2	1.6

Table 1—Measured GPS errors in three different areas

Why do we need highly accurate positioning? With these levels of inaccuracy, navigation apps may be led astray, and may give wrong turn-by-turn directions, which can lead to driver confusion. In this paper, we ask the question: *Is it possible to achieve lane-level positioning accuracy for automobiles even in highly obstructed environments?* In North America's interstate system, the nominal lane width is about 12 feet (3.6m), so our question translates to: Is it possible to achieve 3-4m accuracy, in the worst case, in obstructed environments?

Aside from more accurate (and therefore less confusing) navigation, precise positioning of vehicles can have many potential applications. Accurately positioning crowd-sourced detection of road features (e.g., potholes, rough roads etc.) can help municipalities target roadway improvements. Lane-level traffic flow analysis can help traffic agencies provision roadways; for example, an often clogged right lane might prompt the addition of a dedicated right turn lane. Moreover, insurance companies can track driver propensity to stay on fast lanes, or track violations of lane occupancy rules (e.g., on some roads, trucks are required to stay in the right lanes).

Possible Responses. One possible response is to hope that future GPS receivers will have enhanced accuracy in highly obstructed settings. As we demonstrate later, expensive GPS devices available on the market today are still susceptible to the urban canyon effect. This is not surprising, since GPS receivers will, in general, find it difficult to compensate for lack of visibility to satellites or for multipath effects.

For this reason, most prior work on precise positioning for automobiles (Section 5) has relied on information fusion: combining sensors of other modalities (like LIDAR), or other sources of information (such as digital maps), in order to augment or correct GPS readings.

Our approach. Our paper also uses this approach, but with a new twist: *we exploit the availability of sensors built-in to vehicles to improve positioning accuracy.*

Modern vehicles are equipped with several hundred physical and virtual (derived from physical) sensors on-board. These sensors provide the instantaneous internal state of all vehicular subsystems. From the industry standard CAN [26] bus and using the standard On-Board Diagnostics (OBD-II) port on cars, users can, in theory, access most of these sensors, such as: vehicle speed, steering wheel angle, throttle position, transmission lever position, and some inertial sensors [46, 25, 18]. These sensor readings are internally used to control subsystems of the vehicle, such as stability control and engine health monitoring.

While many of these sensors are proprietary, several tools [50] have been able to access these through reverse engineering. More recently, Ford and General Motors have made about 20 sensors available through their OpenXC platform and GM Developer Network respectively, so it's likely that, in the future, such information will be ubiquitously available. In collaboration with a major automotive manufacturer, we have obtained access to many internal car

sensors. In this paper, we explore whether, and how, these sensors can help precisely position an automobile.

Specifically, we use in-vehicle sensors to improve positioning accuracy in two ways. First, in-vehicle sensors can provide accurate *odometry* for precise dead reckoning. By contrast, prior work has used GPS-derived speed measurements for dead reckoning. Second, in-vehicle sensors can be used to precisely identify *roadway landmarks* (a pothole, or a speed bump). In turn, *crowd-sourced* position estimates of these landmarks can be used to fix a car's position.

3. THE DESIGN OF CARLOC

In this section, we describe the design of CARLOC. We begin with an overview of the overall design, which motivates specific design challenges that are then addressed in subsequent subsections.

3.1 Overview of CARLOC

Figure 2 depicts the various components of CARLOC. At a high-level, CARLOC models the current position of the vehicle *probabilistically*: intuitively, the position of the vehicle at any point in space is associated with a specific probability. The key idea then is to *update* or *refine* this probabilistic representation using information from various sources. Then, at any given point in time, the precise position of the vehicle is obtained by *actualizing* the probabilistic representation, as described later.

One way that CARLOC updates the probabilistic representation is by using vehicle sensors to obtain distance traveled and the heading of the car. This approach, often called *dead reckoning*, is, in CARLOC, more accurate because of its use of in-built vehicular sensors. These sensors are available at frequencies ranging from 10-100Hz, so they can provide accurate estimates of distance and heading over short timescales and distances. However, vehicle sensors by themselves are insufficient: sensor errors can, over time, cause position estimates to drift significantly from the true position.

A second way to update the probabilistic representation is to periodically obtain GPS readings. These readings are associated with estimates of error, which can be used to tighten the car's position estimates. However, as shown in Section 2, GPS errors in obstructed environments can increase the car's position uncertainty.

To overcome GPS errors, one can *spatially refine* the probabilistic position estimates using digital maps. Intuitively, a car is likely to be off a roadway with a very small or near-zero probability, and map-matching algorithms [29] use this observation to refine position estimates. Car sensors provide accurate estimates of speed and turns, and these can be used to enhance existing map-matching algorithms to increase positioning accuracy.

The last component of CARLOC is based on the observation that *roadway landmarks* mark consistent positions in the environment that be exploited to refine the car's probabilistic position estimate. Consider a speed-bump on a road: if a car passes over a speedbump, it can refine its own position estimates using position estimates of *other cars when they passed over the same speedbump*. This suggests that *crowd-sourced* position estimates of roadway landmarks can be used to improve a car's position estimates. CARLOC incorporates several novel algorithms that use vehicle sensors to identify roadway landmarks.

In designing these components of CARLOC, we faced the following challenges:

- Choosing the probabilistic model for position representation, since we needed a representation that would be amenable to update and refinement from a variety of sources of information, including maps, vehicle sensors, and crowd-sourced information.

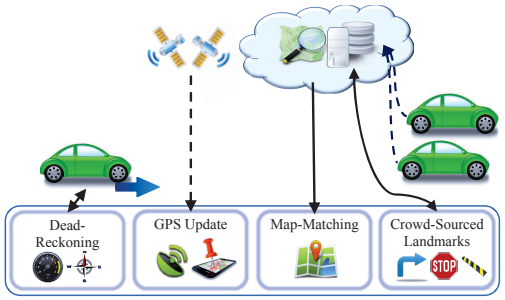

Figure 2—CARLOC Design

- Selecting a model that correctly represents GPS position uncertainty, so that the vehicle's positioning uncertainty could be appropriately updated.
- Designing a *motion model* for the vehicle that uses vehicle sensors to determine how a vehicle's position evolves over time; prior motion models have used GPS readings, but the vehicle sensors are available at much higher frequencies than GPS readings.
- Designing appropriate *map-matching* algorithms; although map-matching has been studied extensively, they rely on frequent GPS updates which can lead to matching errors in the face of significant GPS errors.
- Designing algorithms to detect landmarks from vehicle sensors, and to refine position estimates using crowd-sourced location information.

We discuss each of these challenges in the subsequent sections.

3.2 Probabilistic Representation of Position

A vehicle's position estimate has inherent uncertainty due to sensor noise. A common approach to dealing with this uncertainty is to use linearized models and assume Gaussian noise so that a Kalman filter can be applied. As prior work [12] has shown, however, vehicle positioning violates some of these assumptions; specifically, during a turn the a posteriori position distribution is non-Gaussian and non-linear filtering methods are required to solve the problem.

In this paper, we use a well-known non-parametric probabilistic model of position, based on Sequential Monte Carlo (SMC) methods. Our representation is commonly called a *particle filter* [34, 49]. A particle filter estimates the posterior density of a vehicle's position through predefined Bayesian recursion equations.

Concretely, the current position of the vehicle is represented by a set of particles. Each particle represents a probabilistic state vector, indicating the likelihood the vehicle is at this position. Thus, if we have N particles, each particle is associated with a state vector v_i (which contains its position and its orientation), and a probability or *weight* ω_i that determines the likelihood of the vehicle being in that state. At any given instant, the particle filter can be used to estimate the position of the vehicle as a weighted sum of the particles: $\frac{\Sigma_{i=1}^{N} v_i \omega_i}{\Sigma_{i=1}^{N} \omega_i}$.

This representation provides a uniform foundation for many of the kinds of fusion we are interested in. For example, vehicle odometry and bearing information can be used to update the v_i s, and the associated sensor errors can be used to update the ω_i s. Getting a GPS fix results in re-weighting the particles, and adding map constraints may require removing off-road particles from the filter. Finally, the positions of roadway landmarks can be represented as particle filters, so crowd-sourced landmark updates require merging particle filters (in a manner described later).

3.3 Map Matching

Many digital maps represent roads using *road segments* which are polyline representations of a road. Map-matching is the process of identifying the road segment corresponding to a given position. Map-matching has been used in prior work (Section 5) to improve vehicle positioning, and is in general known to be a hard problem because position errors can lead to errors in map-matching.

CARLOC builds upon a specific piece of prior work [29, 48] that models the map matching problem as a maximum likelihood path estimation on a Hidden Markov Model (HMM). In this work, the states in the HMM are the map-matched road-segments, and transitions occur when a vehicle turns from one road onto another. Given a GPS reading, one can estimate, given a model of GPS errors and using Bayes' rule, the posterior *observation probability* of the car being on a specific road segment. One can also estimate a *transition probability*; namely, at a given instant, and given a GPS reading, the probability that a transition has occurred from one road segment to another. With these observation and transition probabilities, [29] uses the Viterbi algorithm to find the maximum likelihood sequence of states (or road segments the car has traversed).

The Viterbi algorithm is fast enough that we can run map matching on a mobile device. To optimize the implementation, rather than run map-matching on every GPS update (1 per second in our implementation), we do so only when the GPS reading has deviated significantly from the last matched road segment.

CARLOC map-matching enhancements. We have modified the observation probability computations to increase its efficiency and robustness. For efficiency, we use the fact that modern GPS receivers report estimates of error: we then only search road segments that fall within these error bounds. For robustness, we avoid using GPS readings that are inconsistent with the heading (direction of motion) of the car.

Our changes to the transition probability calculations from [29] are more substantial. That work calculates transition probabilities by estimating the travel time from the last known update. One change we make is to use travel distance (as measured from car sensor readings, by integrating the instantaneous speed sensor) instead of travel time. However, travel distance isn't able to distinguish, in Figure 1, whether the car turned left from *Road A* to *Road B* or continued straight on *Road C*, since both outcomes would be equally likely.

CARLOC has additional information that can make the estimation more accurate — *car sensors that measure turns*. Specifically, using the steering wheel angle sensor, we can estimate the change in heading of the car $\Delta\theta_{j,k}$ from road segment j to road segment k. We also estimate the difference $\Delta l_{j,k}$ between the actual distance traveled and the projected distance traveled on the map (obtained by projecting GPS readings onto the road segments j and k). Using these two quantities, we can estimate the transition probability, at a given instant, of the event $\alpha_{j,k}$ of the vehicle transitioning from road segment j to road segment k from Bayes rule (M is the number of road segments considered):

$$P(\alpha_{j,k}|\Delta l_{j,k}, \Delta\theta_{j,k}) = \frac{P(\alpha_{j,k}|\Delta l_{j,k})P(\alpha_{j,k}|\Delta\theta_{j,k})}{\sum_{m=1}^{M} P(\alpha_{j,m}|\Delta l_{j,m})P(\alpha_{j,m}|\Delta\theta_{j,m})} \quad (1)$$

If we assume that the difference in angle, and the difference between the projected and traveled distance are both Gaussian with zero mean and standard deviations σ_a and σ_d respectively, then this becomes:

$$P(\alpha_{j,k}|\Delta l_{j,k}, \Delta\theta_{j,k}) = \frac{exp(-0.5(\frac{\Delta\theta_{j,k}}{\sigma_a})^2 - 0.5(\frac{\Delta l_{j,k}}{\sigma_d})^2)}{\sum_{m=1}^{M} exp(-0.5(\frac{\Delta\theta_{j,m}}{\sigma_a})^2 - 0.5(\frac{\Delta l_{j,l}}{\sigma_d})^2))} \quad (2)$$

Using the matched road segments. CARLOC uses the result of map-matching in three ways. First, when it starts up, CARLOC does not have a usable estimation of the vehicle's position so it projects the GPS reading onto the map-matched road segment as a position estimate.

Second, we use the map-matched road segments to filter erroneous GPS readings. Even though GPS readings come with an associated error, we have found that, especially in obstructed environments, the associated error bounds can under-estimate the actual error. To filter out these erroneous readings, we first project the GPS reading to the map-matched road-segment, then filter out readings whose projected distance to the road-segment is greater than the nominal road widths. In the map we use, Open Street Maps (OSM) [20], road segments have associated types such as residential or highway. We make conservative assumptions about the number of lanes in each type (e.g., 2 for residential and 6 for highway, in each direction), then use nominal lane width to compute road widths. If the GPS location does not fall within the road, we declare it invalid. We also use another optimization. For some roads, OSM depicts them using one road segment, for others two. If our projected GPS reading is closer to the road segment that is against the car heading, we drop that reading.

Finally, we use map-matching to update the weights on the particle filter. Intuitively, for each particle, let x be the projected distance of the particle from the outer edge of the map-matched road segment. For this calculation, we use the conservative road width estimate described in the previous paragraph. If $x > 0$ (the particle falls outside the road segment) we re-weight its probability inversely with x. Specifically, the new weight is calculated to be: $\frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{x^2}{2\sigma^2})$, where σ is the variance of all the particles' projected distance to road segment.

3.4 Motion Model

A motion model captures how the *pose* (position and orientation) of a car (or any moving object) evolves with time. Formally, the pose consists of 5 components, 3 of which estimate position (latitude, longitude and altitude) and 2 measure orientation (heading or yaw, and pitch). In this paper, we focus on modeling motion in a 2-dimensional plane; extending to 3 dimensions is left to future work.

Changes in position and orientation can be measured using GPS, but GPS measurements have error, and are sampled relatively infrequently (once a second). Instead, CARLOC *exploits the availability of car sensors* to dead-reckon the car's pose. Dead-reckoning requires the ability to estimate *displacements* and to estimate *changes in heading*.

Estimating displacement. In theory, one can estimate displacement using the vehicle speed and acceleration and using simple kinematic equations. Thus, if x_t represents the vehicle's position vector at time t, then we can write $x_t = x_0 + v_0 t + \frac{1}{2} a_0 t^2$ where v_0 represents the velocity and a_0 the acceleration. In vehicles, the *speed sensor* can be sampled at relatively high rate (10Hz), and if we assume constant acceleration between two samples of the speed sensor, the kinematic equation to update position becomes:

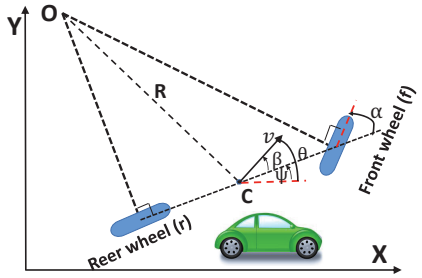

Figure 3—Kinematics of lateral vehicle motion

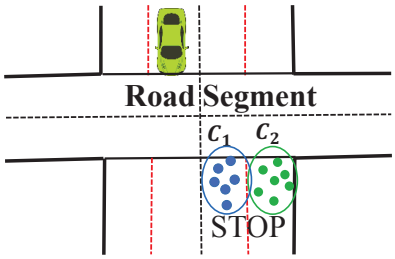

Figure 4—Multi-lane Stop Sign

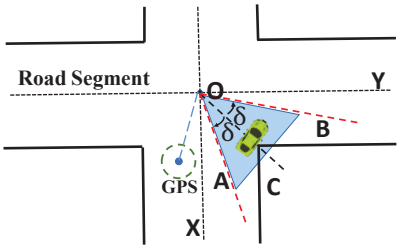

Figure 5—Street Corner Illustration

$x_t = x_{t-1} + \frac{1}{2}(v_t + v_{t-1})\Delta_t$, where Δ_t represents the sampling interval.[1]

Estimating change in heading. To estimate changes in heading, CARLOC also uses vehicle sensors. Modern vehicles expose two sensors that can help estimate changes in heading, the *steering wheel angle* and the *yaw rate* sensor. However, correctly estimating a change in heading needs to take the vehicle steering design into account. Vehicle steering follows the Ackermann geometry [1] which describes how turns are effected by steering.

A simplified version of the kinematics of lateral motion resulting from the steering system design is shown in Figure 3 [42]. In this figure, a turn angle of α on the front wheel results in an effective change in heading θ for the center of mass of the vehicle. θ consists of two components ψ and β. As [42] shows, β (the *slip angle*) can be estimated using vehicle geometry and estimates of α and ψ.

To estimate α, we use the steering wheel angle sensor reading from the vehicle and the empirical observation that there is a linear relationship between the steering wheel angle and the actual wheel angle α.

To estimate ψ, we continuously integrate the vehicle yaw rate sensor. However, we have found that errors in the yaw rate sensor can accumulate over time, so we correct for these errors by using filtered readings of GPS bearing. Our filtering uses three steps. First, we only take GPS readings that are consistent with map-matching (as described in Section 3.3). Next, we check if successive valid GPS readings are comparable to (within 10% of) the distance traveled as reported by the vehicle sensors. Finally, we check if the resulting bearing computed from the successive readings is consistent with (within 5% of) the heading change computed from the yaw rate sensors. If so, we use the GPS bearing to estimate ψ. Because the yaw rate sensor is sampled at a higher frequency than GPS, and because GPS can often be inaccurate, GPS bearing corrections occur infrequently relative to the calculations of ψ using the yaw rate sensor.

Updating the particle filter. In practice, the kinematics calculations can be affected by noise. Our particle filter representation is able to account for sensor noise as follows. Recall that each particle in the particle filter is associated with a pose vector x and a weight ω. When the vehicle moves, we update each particle's pose vector using the displacement and heading change calculations discussed above. To account for sensor noise, we assume that each particle's pose is independently affected by Gaussian noise in the speed and car sensor. We use nominal noise estimates for these sensors from the manufacturer datasheet.

[1]Rather than using a global geodetic coordinate system (e.g., latitude and longitude), we convert all poses to a local geodetic system East-North-Up (ENU [14]). This ignores the earth's curvature, but is easier to model, and has been used in the vehicular positioning literature [35, 9]. We omit the details of this conversion.

3.5 Location Update

Map-matching provides coarse location corrections by removing off-road particles. The motion model can provide fine-grain and accurate updates to particle locations but over small spatio-temporal scales. At larger time-scales, sensor errors can accumulate. As we show experimentally, these two methods alone do not achieve high positioning accuracy. So, CARLOC also uses GPS readings to update the particle filter.

Specifically, CARLOC uses GPS readings determined valid from map-matching (Section 3.3). Each GPS reading is associated with an accuracy range [6, 5] and the error distribution of GPS readings can be well approximated by a Rayleigh distribution [38]. When we obtain a valid GPS reading, we update each particle's weight according to Rayleigh distribution, based on the particle's distance to the GPS-reported location. Intuitively, particles that are far from the reported GPS location are assigned a lower weight or likelihood. When a vehicle is stopped, we might obtain multiple readings at the same location: in this case, we aggregate the error reported by those readings before re-weighting the particles.

We also apply several standard transformations to the particle filter. Recall that particles represent samples of positional probability distribution. With particle weight updates, particles need to resample occasionally to improve the probabilistic estimates. The resampling process adheres to the *Sampling Importance Re-sampling (SIR)* algorithm, by only resampling when the effective number of particles N_{eff} is less than the threshold (N_{th}). Assuming each particle has a weight of ω_i, it follows that $N_{eff} = \frac{1}{\sum \omega_i^2}$ [3, 13]. We set N_{th} to $\frac{2}{3}N$, where N is the number of particles. Moreover, an incorrect resample can cause particle diversity loss, so we also occasionally draw samples from the GPS position distribution, an approach called *sensor resetting* [31].

3.6 Crowd-sourced Landmark Positions

Given that GPS availability in obstructed urban environments is known to be poor, CARLOC uses an additional, novel positioning enhancement, *crowd-sourced landmarks*.

Suppose a car hits a speed bump. If CARLOC is able to detect the speed bump, then *the car's particle filter at the instant the speed bump is encountered is a probabilistic representation of the speed bump's position*. Suppose N cars pass over the same speed bump, the collection of all their particle filters at the speed bump represents a *crowd-sourced* collection of position estimates of the speed bump. Intuitively, one expects the distribution described by these crowd-sourced particles *to converge to the true location of the speed bump* as more and more vehicles contribute to the collection. Finally, a speed bump is an instance of a *roadway landmark*: this discussion applies to other roadway landmarks such as stop signs and street corners (at intersections).

CARLOC uses this observation to improve positioning accuracy. When a car detects a roadway landmark, it can check to see if crowd-sourced particles are available for the landmark. (These par-

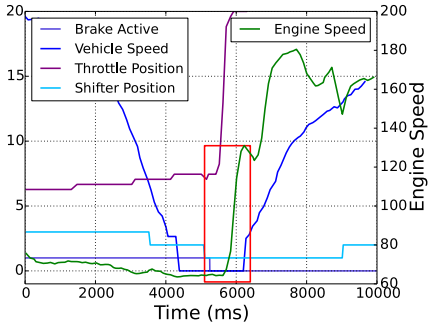

Figure 6—Stop Sign Landmark Detection

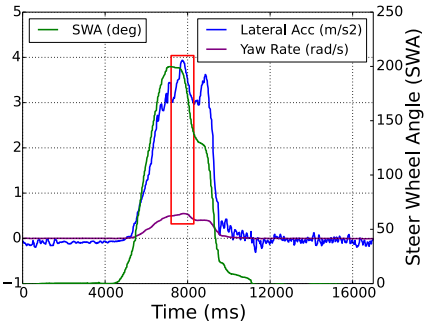

Figure 7—Street Corner Landmark Detection

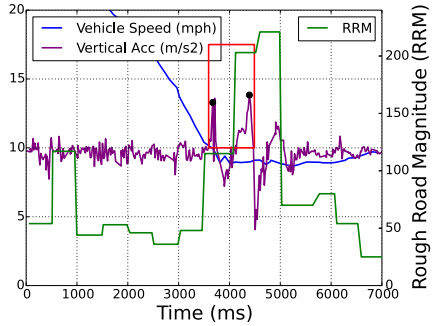

Figure 8—Speed Bump Landmark Detection

ticles can be maintained in a cloud database, and made available through a cloud service. To minimize network latency, relevant particle clouds can be pre-fetched before reaching a landmark. The detailed design of this service is beyond the scope of the paper). If they are available, the car can *resample its particle filter from the set of particles that include crowd-sourced particles* for the landmark and its own current particle filter.

This approach poses two challenges: (a) How can vehicles detect roadway landmarks? (b) How should a car's particle filter be updated using the crowd-sourced particles? We discuss answers to this question for three types of roadway landmarks below. Our detection algorithms use vehicle sensors to achieve accurate landmark detection.

Stop Signs. At a stop sign, there is usually a line drawn on the roadway surface. Drivers are supposed to stop at the line before proceeding into the intersection. Of course, not all drivers stop exactly at the line. CARLOC leverages the wisdom of the crowds: if most drivers stop at or near the line, their combined position distributions will be an accurate estimate of the average behavior of drivers when encountering a stop sign (e.g., stopping just a little before the stop sign) We make this intuition more precise below.

Detection: Our detection algorithm is based on the following observation: to leave the stop sign and enter the intersection, the driver usually releases the brake, steps on the gas pedal, as a result of which the engine speed increases, one or more gear shifts may occur, and the vehicle speed increases.

To detect this, CARLOC continuously samples the following vehicle sensors: (*Brake Active*,*Vehicle Speed*, *Throttle Position*, *Shifter Position* and *Engine Speed*). Figure 6 shows the timeseries of these sensors at a stop sign and pictorially depicts the algorithm: on the timeseries of each sensor, the algorithm applies a sliding window and attempts to discover a window that contains a brake pedal release, followed by a sharp increase in throttle position and engine speed, followed by an increase in engine speed. The particle filter **P** of the car at the time when there is a discontinuity in the vehicle speed timeseries (i.e., the time when the speed increases suddenly) marks an estimate of the location of the stop sign line. Once a car computes **P**, it can *store* **P** in a cloud service (discussed below), for later retrieval by other cars.

The precise algorithm is more complicated than this since it has to take into account many practical constraints. First, for any window that satisfies these features, we lookup the current position in an online database of stop signs [20] and our accumulated stop sign database, and only use the particle filter if it is found in the database. This eliminates false detections caused by a car stopping and then starting, say, after a delivery. This database is currently incomplete, but with time we expect its coverage to improve. For additional coverage, CARLOC uses other landmarks discussed below. Second, drivers may release the brake and roll through the stop sign, to account for which CARLOC uses a slightly larger window. Third, drivers may stop multiple times before reaching the stop sign line, since they may be queued up behind other cars. This behavior manifests itself as a sequences of detected windows, and we use the last window in the sequence. Finally, we crowd-sourcing to disambiguate traffic lights from stop signs; if some car traces don't stop at an intersection, but others do, that intersection has a traffic light, not a stop sign. We use this same technique to improve our stop sign database coverage; if every car stops near an intersection, that indicates the existence of a stop sign there.

Particle Storage and Update. When the particle filter **P** is uploaded to the cloud service, that service performs a processing step. Figure 4 shows a multi-lane road scenario, in which the cloud service can get aggregated data from cars stopping on each lane. The cloud service employs a clustering algorithm to cluster the particles based on lane width threshold; the resulting number of clusters determines the number of lanes on the road. Figure 4 shows two such clusters.

Now, suppose a car **A** wishes to update its particle filter when it reaches the stop sign. It downloads the cluster of particles closest to its current estimated position. These particles, together with its own, are then re-sampled with a probability proportional to the weight of each particle. Thus, more important particles are likely to be selected during re-sampling. These re-sampled particles then constitute the updated particle filter for **A**. In this way, if the crowd-sourced cluster of particles has converged to the actual location, the new particle filter will be closer to the car's precise position.

Street Corners. Street corners can be detected when a car performs a right turn.

Detection: At a right turn, the timeseries of the steering wheel angle sensor peaks (Figure 7). So, any maximum in the steering wheel angle (SWA) that is larger than some high threshold (90 degrees in our implementation) can indicate a street corner. We have found that this peak is fairly robust to a variety of turning behaviors. For example, even when drivers turn from the rightmost lane into the non-rightmost lane, this peak is observed.

To disambiguate other turns (for example, lane shifts at low speed which might also trigger the threshold), we correlate with two other car sensors: the lateral acceleration and the yaw rate[2] (Figure 7). During a turn, these three sensors all exhibit peaks, so CARLOC applies a sliding window to find a window in the trace that contains peaks of the three signals and then finds the average of the time of occurrence of these peaks. The particle filter of the car **P** at this time is used as an estimate of the position of the street corner and is crowd-sourced (uploaded to the cloud service). However, to disambiguate right turns at places other than intersections, we

[2]These sensors are noisy, so we use box smoothing [41] to smooth these time series.

use an online map of intersections to determine if the car's current estimated position is near an intersection; if not, **P** is not uploaded.

Particle Storage and Update: When **P** is uploaded to the cloud service, it filters outlier particles to improve accuracy. From road segment data in an online map, the location of the mid-point of the intersection can be determined, and CARLOC assumes that particles in **P** outside the shaded cone (between $\overline{OA}$ and $\overline{OB}$) in Figure 5 are unlikely to represent the car's true position, and assigns those particles very low weight.

When a car **A** reaches the street corner, it downloads the crowd-sourced particles from the cloud service and applies an update procedure identical to that for the stop sign. If a car stops before turning, then a particle update is performed only once (at the stop sign), rather than at both locations since the latter alternative could lose particle diversity (i.e., we might not have a good sample of the underlying distribution of position).

Speed Bumps. The last landmark we consider in this paper is the speed bump. CARLOC is careful to disambiguate potholes and speed bumps and uses several car sensors for this purpose.

Detection: The detection algorithm is best illustrated using Figure 8. This shows the measurements of three sensors recorded from a car traversing a speed bump: (*Rough Road Magnitude (RRM)*, *Vehicle Speed* and inertial sensor *Vertical Acceleration*). As it approaches the bump, the car slows down and the vertical acceleration sensor exhibits a peak. These two features are used to determine a potential speed bump. CARLOC then monitors the RRM sensor and performs peak detection within a window whose scale (shown as red box) approximates the vehicle's wheelbase. If two peaks (shown in black dots) of vertical acceleration and increase of RRM are sensed, we determine a potential speed bump has been observed.

CARLOC deems the car's particle filter **P** at the first peak to be an estimate of the speed bump's location. As with other landmarks, **P** is uploaded and stored in the cloud service.

Particle storage and update. CARLOC clusters speed bump particles as it does for stop sign particles. Updating another car's particle filter follows the same re-sampling procedure as discussed above.

4. EVALUATION

In this section, we assess the performance of CARLOC in obstructed and unobstructed environments over different trip lengths, and compare its performance against other alternatives including GPS positioning from commodity GPS receivers as well as higher precision GPS receivers, and with differential GPS. Our final goal in this section is to assess the efficacy of each of the components in CARLOC: dead reckoning, map-matching, and landmark-based position augmentation.

4.1 Methodology

Experimental Setup. Our evaluations are based on multiple traces collected by two different drivers over routes with different characteristics from the perspective of a GPS receiver: *obstructed* routes in a downtown urban canyon, *unobstructed* routes with a view of the open sky at all points, and *partially-obstructed* routes with obstructed sky visibility in some locations.

The routes are all of different lengths, and of different road types (with different lane widths), as discussed below. Each trace consists of several vehicle sensor readings obtained through our car sensing platform ([18, 25]). The sensors readings we collect are already available through the CAN bus, as described earlier, and our sensor collection software can sustain continuous collection nearly 40 car sensors, of which we use only a subset.

On each route, we collect multiple traces at different times of day, which helps avoid bias caused by a specific traffic pattern. In addition, we use a subset of these traces to obtain crowd-sourced landmark positions, and evaluate the remaining traces using these crowd-sourced positions. Our results are averaged over the evaluations on these remaining traces, and we quantify the variability of our results in terms of quartiles.

Comparisons. Across all routes, we compare CARLOC performance against using a GPS receiver on a Google Nexus 5 (labeled *SPGPS* on our graphs). In addition, on some routes, we also compare against using an expensive (>$200) high-precision GPS receiver, the ublox NEO-7P GPS (*HPGPS*). Finally, we also use a companion rover receiver, the ublox LEA-6T to obtain Differential GPS (DGPS [45]) position as well as Real-Time Kinematic (RTK [43]) based positioning. It is likely that future cars may be able to incorporate these high precision receivers.

The rover receiver estimates position based on two modes [35]. The first mode returns precise location values. In its DGPS mode the rover receiver utilizes corrections from a known base station. Both modes update location at 1Hz.

Our LEA-6T rover devices use ublox firmware 7.03 and the NEO-7P devices use version 1.00. For RTK and DPGS, we use a publicly available NTRIP caster base station within 10 miles of all our experiments. We obtained access to the base station's NTRIP stream and position information through the UNAVCO consortium [53]. This station is equipped with an advanced Ashtech antenna mounted on a hilltop. On our rover devices (LEA-6T), ubx-formatted GPS measurements are captured using u-center, the ublox driver for the device with settings obtained directly from the developers of RTKLIB [54, 43].

Metrics and Ground Truth. Our measure of CARLOC performance is *positioning error* measured by distance between CARLOC's position and ground truth. Obtaining ground truth is extremely hard for positioning in some places, and we resort to three approaches.

Closed-loop routes. In our partially obstructed routes, we start the route at a well-marked location (and empty metered parking spot) and return to the same location. Our measure of accuracy then is the difference between the start position and the end position as reported by CARLOC (or any of the candidate comparison algorithms). The start position is calculated using our high-precision GPS receiver. This approach has also been used by prior work [22, 51].

High-precision GPS receiver. On our unobstructed routes, we also continuously collect readings from the high precision GPS receiver and use these as ground truth. The reported accuracy of these devices is within one meter. This method enables us to determine accuracy along the entire route instead of just at the end.

Fiducials. On our obstructed routes, as we show below, the accuracy of the high precision receiver is not sufficient. So, we resort to using *fiducials* in the environment. As we drive on the obstructed route, we stop at several easy-recognizable points (or fiducials) along the right side of the car, e.g. sidewalk ramp exit, mailbox, etc. When we stop, we record the current timestamp and take a image from the passenger side to cover the car right side road to ramp distance. We then use these images to look up Google's satellite views, pin down the points recorded in the image, and then use the location of those points (as obtained from Google Maps) as ground truth (Figure 9).

To validate our fiducial-based ground truth collection, we applied the same methodology in several locations with an unob-

Figure 9—Static Measurement Setup

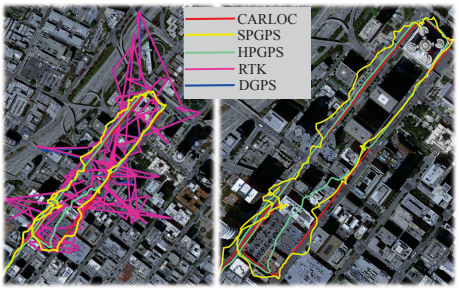

Figure 10—CARLOC and GPS Comparison in Downtown

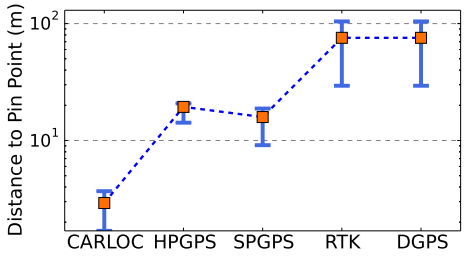

Figure 11—Map Pin Points Comparison

structed view of the sky, and at these locations we also used the high-precision reference GPS receiver to record position. We find that our pinned down points are within 1.2m of the reference receiver.

4.2 CARLOC on an Obstructed Route

Our obstructed route is a 2-mile loop in a downtown area surrounded by tall buildings (Figure 10). We collected a total of 12 traces along this loop using two different drivers at different times of day. On this route, we use only 4 landmarks: the 4 street corners shown in the figure. Of our traces, we use 8 to obtain crowd-sourced particles, and 4 to evaluate CARLOC and compare it against the high-precision GPS receiver, DGPS and RTK. As described above, we use the fiducials-based ground-truthing technique and we collected 15 different points as ground-truth.

Figure 11 shows the accuracy of each of these techniques with respect to the ground truth. The error bars represent the 25th and 75th percentile in our measurements. CARLOC has an average error of about 2.7m, with the smallest error being 0.6m as closest and the largest being 4.9m. Surprisingly, all of the alternatives have one to two orders of magnitude higher error. The high-precision receiver has an average error of 19.4m (min 7.7m, max 44m). The smartphone has an average/min/max error of 16m/1.2m/40.2m. This is slightly better than our high-precision GPS; this difference could either be within the margin of experimental error, or that smartphones have better dead-reckoning or GPS signal processing algorithms in software. Moreover, both DGPS and RTK, achieve really poor performance, with an average error of 75m, and a worst-case error of 200m.

Figure 10 depicts these results visually. In the figure on the left, it is evident that DGPS and RTK measurements span the entire area covered by the two-mile loop. In the figure on the right, the superior performance of CARLOC vis-a-vis the high-precision receiver and the smartphone are visually evident.

Why this pathological performance for the other alternatives? Clearly, both the high precision receiver and the smartphone GPS suffer from the urban canyon effect: the inability to see enough satellites affect their ability to get good position fixes. They are able to achieve reasonable performance primarily because of their use of dead reckoning filters. To understand why even the high-performance GPS receiver does not perform well, we examined the dilution of precision (DOP [30]) reported by the receiver. This measure of variability of GPS signals is much higher downtown (DTSHDOP) than along an unobstructed road with a clear view of the open sky (OSSHDOP) (Figure 12). This suggests that satellite availability and multipath effects degrade the performance of the high-performance receiver.

To further understand this performance, we discussed our findings with developers of RTKLib [43], a well-known open source software for processing GPS data which has been reported to have centimeter accuracy in many situations. As such, this forum includes many experts in GPS positioning performance. Our discussions corroborated our findings above that both DGPS and RTKLib suffer from lower satellite availability and multipath effects. It is well known that differential GPS cannot fix errors caused by differing multipath environments at the rover and basestation [10]. We have also verified that satellite availability is lower in our downtown trace (the rover receiver sees about 5 satellites) than in a trace from an unobstructed area (6-8 satellites). Finally, we notice far fewer location updates (once every 2.6s) from DGPS and RTK compared to using these on readings from an unobstructed trace (once every 1.1s). The developers of RTKlib believe these pathological errors can be improved with careful, route-specific parameter tuning, and we have left this to future work.

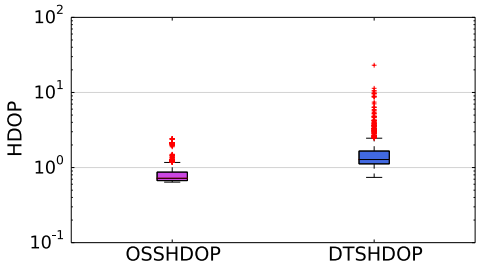

Figure 12—HDOP comparison for Open Sky Area and Downtown

4.3 CARLOC on an Unobstructed Route

We now quantify CARLOC performance along a 4.4km unobstructed route (Figure 13). Along this route, we treat the readings from the high-precision GPS receiver as ground truth, since, in this setting, its claimed accuracy is less than 1m [52]. We collected 8 traces along this route and used 5 of them to obtain crowd-sourced landmarks and 3 to evaluate accuracy. We obtained a total of 13 landmarks: 6 stop signs, 4 speed bumps and 3 street corners.

Table 2 shows CARLOC performance. When the error is computed across the entire trace (the *complete comparison*), CARLOC averages a 2.27m error with a minimum error of 0.14m and a maximum error of 4.51m. However, we believe this number is a little misleading because the high precision GPS receiver updates its position at a frequency of 1Hz, but CARLOC can track position changes every 10th of a second. So, these two readings can be off, on average by half of a tenth of a second in the worst-case. In that time, a car traveling at 45mph travels 1m, which can add to the error estimate.

To avoid this bias, Table 2 also reports error computed only at points where the car has stopped along the route (car sensors can tell us when the car has stopped). In this case (the *static comparison*), CARLOC has an average error of 1.38m and a maximum error of 2.4m. Finally, the smartphone GPS has a 3× higher worst-case error compared to CARLOC, and almost 2× higher average error.

Figure 13 pictorially depicts these results. Although the three alternatives are not visually distinguishable, the inset shows a part of the trace where the errors in the smartphone GPS are much more

	Mean (m)	Min (m)	Max (m)
Complete Comparison	2.27	0.14	4.51
Static Comparison	1.38	0.16	2.42
Smartphone GPS Comparison	4.19	0.14	15.83

Table 2—CARLOC, smartphone GPS to High-precision GPS Distance Statistics

evident: CARLOC's map matching and the motion model are able to compensate for inaccurate GPS, as a result of which it is able to much more closely follow the high-precision GPS receiver.

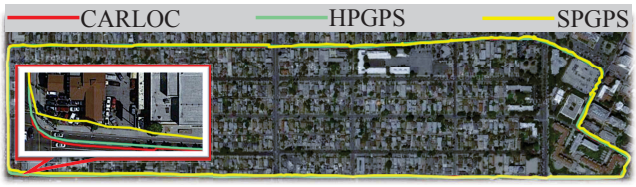

Figure 13—CARLOC, high-precision GPS and smartphone GPS in Open Sky Area

4.4 CARLOC on Partially-Obstructed Routes

In this section, we explore the performance of CARLOC on routes of different lengths. We also quantify the benefits of each of the components of CARLOC, and understand more closely how crowd-sourced landmarks help improve positioning.

It is hard to find unobstructed routes in metropolitan areas, so all our routes are partially obstructed. Because some of our routes are longer than our unobstructed route, it was logistically difficult to collect ground-truth using fiducials (which require significant manual effort), we used the closed-loop accuracy estimation technique described above.

Our routes range in length from 3.4km to 9.2km. For each route, we collect 15 traces, 10 of which we use for extracting crowd-sourced landmark locations, and 5 for evaluation. Along the longest of these routes, we have 19 landmarks: 5 stop signs, 10 street corners and 4 speedbumps.

Error as a function of distance. Over different distances, CARLOC is able to achieve mean error between 1.2 and 2.2m (Figure 14). The maximum errors for these 5 routes are 1.73m, 1.67m, 2.57m, 3.0m and 2.7m respectively. This is highly encouraging and suggests that lane-level precision might be achievable in most settings. Although there is a slight increase as a function of length, we believe this is largely due to difference in characteristics along the longer routes, rather than an increasing trend in CARLOC error as a function of distance. Indeed, there is no fundamental reason to believe that CARLOC error should increase with distance: any error accumulation with distance from, say the motion model would be corrected by GPS position fixes and crowd-sourced landmarks. The mean error along the longer 9.2km is slightly lower than the 7.6km trace primarily because the longer trace has two additional stop sign landmarks which improve CARLOC performance.

Contribution of different components of CARLOC. Using these traces, we are able to quantify the contribution of different components of CARLOC. Our motion model permits pure dead-reckoning (*DR*). To this, we consider the adding location updates from GPS (*DR GPS*). We also consider an alternative strategy which uses map-matching with dead-reckoning (*DR MAP*). Our final alternative strategy adds both map matching and location updates to dead-reckoning (*DR MAP GPS*).

Figure 14 shows the mean error using closed-loop error estimation for these different strategies and different route lengths.

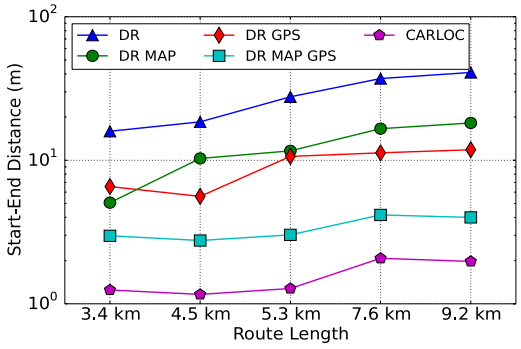

Figure 14—Different strategies' Start-End Distance

For the *DR*, we can clearly see an increase with length of trip, as expected: dead-reckoning error accumulates with distance. From the 15.9 m for 3.4 km to 40.9 m for 9.2 km trip, the errors grow almost linearly. When we add map-matching, we can see the Start-End distance for *DR MAP* drops to 10-26m. When GPS fixes are added to dead-reckoning, the errors also drop and appear to be independent of trace length. One would expect *DR GPS* to have similar error characteristics as the GPS receiver, since every position fix biases the location estimate towards the GPS position. *DR GPS* has errors of 6-10m, consistent with this expectation. Finally, with the addition of map-matching, the error of *DR MAP GPS* reduces to 3-4m across all route lengths. Finally, the addition of landmarks brings the error down to 1.2-2.2m as discussed above. These results suggest that each component plays a significant part in reducing the overall error, validating the design of CARLOC.

The Role of Landmarks. Why do landmarks perform well? How does the accuracy vary as a function of the number of landmarks along a route or the number of crowd-sourced particle filters used? [3]

How accurate are our landmarks? To measure the accuracy of our landmarks, we collected multiple traces on an unobstructed route on which we also collected measurements from our high precision GPS device (which we use as ground truth). Our route covers a total number of 9 right turns, 5 speed bumps and 4 stop signs.

We then ran our landmark detection algorithms over all traces. For each landmark in a trace, our algorithms determine the time t at which the landmark was detected (Section 3.6). In our traces, we find the nearest position reading from the high-precision GPS receiver within a small time interval Δt around t. (The high-precision GPS samples at 1Hz, which is too coarse since in 1 second the car can move several meters, so we restrict our search to a smaller window.) In our evaluations, we used 100ms for Δt for speed bumps and street corners and 300ms for stop signs, since the car stays longer at stop signs. Then, we define the *landmark error* as the difference between the estimated position of the landmark and the high-performance GPS receiver.

Figure 18 shows the statistics of landmark error for each type of landmark. The average error of three landmarks is around 2 meters. Stop signs have a minimum error of 0.6m and maximum of 2.9 m, which is encouraging. For street corners, the maximum error reaches 4m, mainly because 2 street corners are a bit obstructed, thus the results are biased by incorrect GPS readings. The minimum error can be as low as 0.3m. Speed bump errors can also reach around 4.1m, caused by a single outlier where the speed bump has a pothole just before it, which increases the error. However, the speed bump's minimum distance can reach as low as 0.15 m, which is very close to the center of the particle cloud. These results ex-

[3]We omitted the landmark detection accuracy evaluation due to space.

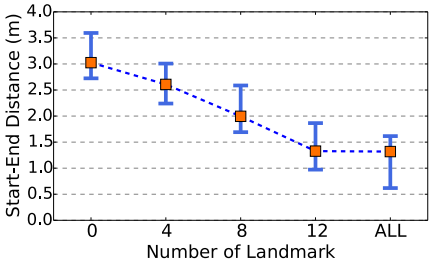

Figure 15—Start-End Distance with Number of Landmarks

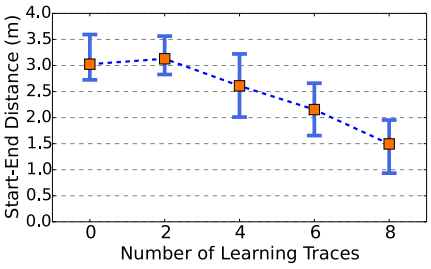

Figure 16—Start-End Distance with Number of Learning Trace

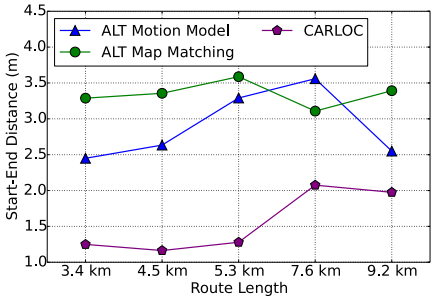

Figure 17—Map-matching and Motion Model Optimization Performance

	Precision	Recall
Stop Sign	0.89	0.95
Speed Bump	0.83	0.88
Right Turn	0.97	0.98

Table 3—Landmark Detection Accuracy

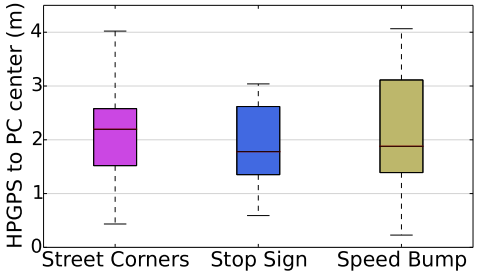

Figure 18—Landmark Error Statistics

plain why using crowd-sourced particles can improve positioning accuracy, but also indicate room for improvement in our detection algorithms.

How accurate are the landmark detection algorithms? We have described our detection algorithms in Section 3. We run our detection algorithms over all the traces we have and compare against the ground truth we collected. We summarize the precision and recall for each algorithm in Table 3. For stop sign, we apply crowd-sourcing to eliminate the outlier cases, like traffic lights. For speed bump, because of an severe pothole in our traces, our algorithm always treats it as speed bump. This brings down the overall accuracy. For right turn, we have near optimal performance. The main reason is the detection algorithm gets triggered by peak of multiple sensors, and we also apply the verification with map information, so the good accuracy performance is expected.

How many landmarks are enough? Figure 15 shows the CARLOC closed-loop error on our 5.3km loop. On this loop there are 15 landmarks, and the figure plots mean CARLOC error (and 25th and 75th percentiles) as a function of the number of landmarks used to calculate position. As expected, the error decreases as more landmarks are used in estimating position. However, beyond about 12 landmarks there is no improvement in the error, suggesting that of a relatively small number of landmarks along a route might be sufficient to achieve high accuracy.

What degree of crowd-sourcing is necessary? For the same route, Figure 16 shows the accuracy of using all 15 landmarks, but computed from an increasing number of traces. This shows what degree of crowd-sourcing is necessary. Interestingly, using 2 traces has higher error than using one trace. We found that this is because, in the second trace, the driver did not fully stop at the stop sign, inducing an error in landmark estimation. Moreover, the error drops linearly with the number of crowd sourced traces. This behavior is expected, but we don't see a flattening, suggesting that CARLOC accuracy can be improved by adding a higher degree of crowd-sourcing. We have left this to future work.

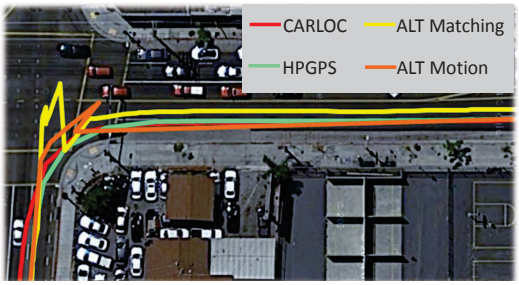

Figure 19—Map-matching and Motion Model Issues on Map View

4.5 Benefits of Optimizations

CARLOC optimizes map-matching (Section 3.3) and the motion model (Section 3.4). In this section, we quantify the benefits of these optimizations.

CARLOC enhances a previously proposed map-matching algorithm [29] to include car-sensor readings that provide distance and changes in heading. The availability of these readings motivates a more sophisticated transitional probability calculation for a Hidden Markov Model, and CARLOC incorporates this. What if CARLOC had used the original transitional probability calculation? Figure 17 shows how the performance of this *ALT Map Matching* strategy performs against CARLOC for our closed-loop tests. This strategy exhibits significantly higher error between 3-3.5m across all our route lengths, suggesting that the optimization is definitely beneficial. Figure 19 illustrates one situation where the optimization helps: CARLOC is able to track the turn correctly, but *ALT Map Matching*, because its transition probability calculation does not incorporate turns, is unable to do so.

Second, CARLOC incorporates an advanced motion model that computes the slip angle β. Instead, it could have simply used heading change computed from the yaw rate (ψ in Figure 3), which would have been a coarse approximation of the slip angle. This alternative motion model, named as *ALT Motion Model*, also has higher error than CARLOC and comparable error to *ALT Map Matching* (Figure 17). The reason for this inaccuracy is also depicted in Figure 19, which shows how *ALT Motion Model* is not able to track turns.

5. RELATED WORK

We are inspired by prior work in mobile sensing based position augmentation, improved GPS-based methods, and robot localiza-

tion. CARLOC sits in the unique point in the design space, with its use of vehicle sensors and crowd-sourced landmarks.

Mobile Sensing. The mobile sensing community has long explored approaches to use GPS position and other sensors to detect features on roadways (such as stop signs [7, 23] and potholes [27, 28, 57, 17, 15]). In contrast, CARLOC uses vehicle sensors to identify common roadway landmarks with the aim of improving positioning.

Closest to our work is SmartLoc [4], which estimates location and travel distance using inertial sensors on mobile devices. In obstructed environments, SmartLoc uses smartphone sensors to detect landmarks in the environment (like bridges and traffic lights), but these measurements are not crowd-sourced. CARLOC's use of vehicle sensors and crowd-sourced landmarks, together with advanced map matching, gives it an order of magnitude higher accuracy than the prior work. LaneQuest [2] uses probabilistic methods to estimate which lane a car is driving on, a qualitatively different problem than ours. LaneQuest, however, uses crowd-sourced anchors, but, unlike CARLOC, cannot leverage vehicle sensors to detect these. Similar to LaneQuest, [33] keeps track of relative location between cars, while CARLOC focuses on the problem of precisely positioning automobiles.

Several other pieces of work explore improvements to map matching: these can potentially be used to improve the accuracy of map-matching in CARLOC. Track [48] and CTrack [47] propose map-matching improvements using WiFi localization and cellular position respectively. AutoWitness [19] employs inertial sensor-based HMM and Viterbi Decoding to improve path estimation. Map-matching is just one of the components in CARLOC, and we use vehicle sensors to augment map-matching. Finally, [35] proposes fusing GPS and inertial measurements from custom hardware, and leverages DGPS for accurate vehicle position. In contrast, CARLOC does not require custom hardware, and our results show that in obstructed environments, DGPS can perform poorly.

GPS Enhancements. Much work has explored techniques to improve GPS positioning without fusion from other sensors. DGPS [45] and RTK [43, 37] use a base station and a rover receiver and are able to achieve high accuracy. More recent work has explored using DGPS [22, 21] but improving the positioning calculations; this work is able to achieve centimeter-level accuracy in unobstructed environments. Finally, a body of work has explored other improvements to DGPS and RTK [16, 44]. Unlike this class of work, CARLOC can achieve high accuracy in urban environments using a single commodity GPS receiver. High-precision GPS receivers [52] might well become available in future makes and models, but even these will require CARLOC-like fusion in obstructed or partially-obstructed urban environments.

Robot localization. Many of the techniques we use, such as the motion model and particle filters, are inspired by prior work on robot localization. Robot and vehicle localization have extensively explored fusion using information from various kinds of sensors: inertial sensors [55], stereo vision cameras [39], laser range finders [32, 11, 8, 24]. Unlike these, CARLOC explores the use of in-built vehicle sensors, and, in addition, crowd-sourcing landmark positions, in order to achieve high accuracy. Finally, other work has also explored map integration for position enhancement [40, 56, 36]; as we show, while maps and GPS can provide high accuracy, the use of crowd-sourced landmarks in CARLOC is necessary to get good results.

6. CONCLUSIONS AND FUTURE WORK

This paper presents CARLOC, a system for precisely tracking the position of an automobile. CARLOC builds upon prior work in probabilistic position estimation using map matching, but adds novel components: it uses sensors built into vehicles to augment map-matching and motion models, and crowd-sourced landmark estimates to improve positioning accuracy. CARLOC's mean error is on the order of 2m, suggesting the feasibility of lane-level positioning in the future.

Future work can explore several directions. CARLOC's motion model can be generalized to 3 dimensions to account for hilly roads. It may be possible that other alternatives like RTKLib can be tuned to achieve better performance, and it would be interesting to see how close such tuning comes to CARLOC's performance. Although CARLOC has high-accuracy and outperforms its competitors, its position tracking during turns can be improved (Figure 19). Furthermore, our current experiments are conducted with traces from 2 drivers. While our current experiments hint at the benefits of crowdsourcing, the impact of multiple drivers and cars needs further study. The landmark detection algorithms can be made more robust to different drivers' driving behaviors. CARLOC is designed to generalize to various landmarks: CARLOC can attempt to leverage additional roadway landmarks such as changes in the road surface texture, potholes, or discontinuities in lighting caused by entering a tunnel.

Finally, we propose to explore practical deployability of CARLOC. We envision this to be conceptually straightforward, since CARLOC uses in-built vehicle sensors, and needs a relatively simple cloud service for storing its particle cloud. In practice, CARLOC can be retrofitted into a car's existing navigation system as a firmware update.

7. REFERENCES

[1] Ackerman Steering Principle. http://www.rctek.com/technical/handling/ackerman_steering_principle.html.

[2] H. Aly, A. Basalamah, and M. Youssef. Lanequest: An accurate and energy-efficient lane detection system. *Proceedings of IEEE PerCom 2015*, 2015.

[3] N. Bergman. Recursive bayesian estimation: Navigation and tracking applications. dissertations no 579. *Linköping Studies in Science and Technology, SE-581*, 83, 1999.

[4] C. Bo, X.-Y. Li, T. Jung, X. Mao, Y. Tao, and L. Yao. Smartloc: Push the limit of the inertial sensor based metropolitan localization using smartphone. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013.

[5] J. Bornholt. *Abstractions and techniques for programming with uncertain data*. PhD thesis, Honors thesis, Australian National University, 2013.

[6] J. Bornholt, T. Mytkowicz, and K. S. McKinley. Uncertain< t>: A first-order type for uncertain data. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*. ACM, 2014.

[7] R. Carisi, E. Giordano, G. Pau, and M. Gerla. Enhancing in vehicle digital maps via gps crowdsourcing. In *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*. IEEE, 2011.

[8] F. Chausse, J. Laneurit, and R. Chapuis. Vehicle localization on a digital map using particles filtering. In *Proceedings of IEEE Intelligent Vehicles Symposium*. IEEE, 2005.

[9] P. Davidson, J. Collin, J. Raquet, and J. Takala. Application of particle filters for vehicle positioning using road maps. In *23rd International Technical Meeting of the Satellite Division of The Institute of Navigation, Portland, OR*, 2010.

[10] How Differential GPS works. http://www.trimble.com/gps_tutorial/dgps-how.aspx. 2011.

[11] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 2001.

[12] S. Dmitriev, A. Stepanov, B. Rivkin, and D. Koshaev. Optimal map-matching for car navigation systems. In *Proceedings of 6th International Conference on Integrated Navigation Systems, St. Petersburg*. DTIC Document, 1999.

[13] A. Doucet. On sequential simulation-based methods for bayesian filtering. 1998.

[14] East-North-Up Coordinates System. http://www.navipedia.net/index.php/Transformations_between_ECEF_and_ENU_coordinates.

[15] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: Using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08. ACM, 2008.

[16] J. Farrell and T. Givargis. Differential gps reference station algorithm-design and analysis. *IEEE Transactions on Control Systems Technology*, 2000.

[17] D. Festa, D. Mongelli, V. Astarita, and P. Giorgi. First results of a new methodology for the identification of road surface anomalies. In *Proceedings of IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, 2013.

[18] T. Flach, N. Mishra, L. Pedrosa, C. Riesz, and R. Govindan. Carma: towards personalized automotive tuning. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2011.

[19] S. Guha, K. Plarre, D. Lissner, S. Mitra, B. Krishna, P. Dutta, and S. Kumar. Autowitness: Locating and tracking stolen property while tolerating gps and radio outages. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 29–42. ACM, 2010.

[20] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing*, 2008.

[21] W. Hedgecock, M. Maroti, A. Ledeczi, P. Volgyesi, and R. Banalagay. Accurate real-time relative localization using single-frequency gps. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014.

[22] W. Hedgecock, M. Maroti, J. Sallai, P. Volgyesi, and A. Ledeczi. High-accuracy differential tracking of low-cost gps receivers. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013.

[23] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher. Smartroad: a crowd-sourced traffic regulator detection and identification system. In *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, pages 331–332. IEEE, 2013.

[24] A. S. Huang and S. Teller. Probabilistic lane estimation using basis curves. Robotics: Science and Systems (RSS), 2010.

[25] Y. Jiang, H. Qiu, M. McCartney, W. G. Halfond, F. Bai, D. Grimm, and R. Govindan. Carlog: a platform for flexible and efficient automotive sensing. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014.

[26] K. H. Johansson, M. Törngren, and L. Nielsen. Vehicle applications of controller area network. In *Handbook of networked and embedded control systems*. Springer, 2005.

[27] J. Karuppuswamy, V. Selvaraj, M. M. Ganesh, and E. L. Hall. Detection and avoidance of simulated potholes in autonomous vehicle navigation in an unstructured enviornment. In *Proceedings of Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision*, volume 4197, 2000.

[28] C. Koch and I. Brilakis. Pothole detection in asphalt pavement images. *Adv. Eng. Inform.*, 25(3), 2011.

[29] J. Krumm, E. Horvitz, and J. Letchner. Map matching with travel time constraints. Technical report, SAE Technical Paper, 2007.

[30] R. B. Langley. Dilution of precision. *GPS world*, 10(5):52–59, 1999.

[31] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'00)*. IEEE, 2000.

[32] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems*, volume 4, page 1. Citeseer, 2007.

[33] D. Li, T. Bansal, Z. Lu, and P. Sinha. Marvel: multiple antenna based relative vehicle localizer. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 245–256. ACM, 2012.

[34] J. S. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 1998.

[35] E. D. Martí, D. Martín, J. García, A. De La Escalera, J. M. Molina, and J. M. Armingol. Context-aided sensor fusion for enhanced urban navigation. *Sensors*, 2012.

[36] P. Merriaux, Y. Dupuis, P. Vasseur, and X. Savatier. Wheel odometry-based car localization and tracking on vectorial map. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1890–1891. IEEE, 2014.

[37] T. D. of Transportation. TxDOT survey manual - GPS RTK surveying. http://onlinemanuals.txdot.gov/txdotmanuals/ess/gps_rtk_surveying.htm. April 2011.

[38] A. Papoulis and S. U. Pillai. *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[39] I. Parra, M. Sotelo, D. F. Llorca, and C. Fernández. Visual odometry for accurate vehicle localization-an assistant for gps based navigation. In *17th International Intelligent Transportation Systems World Congress*, pages 1–6, 2010.

[40] A. U. Peker, O. Tosun, and T. Acarman. Particle filter vehicle localization and map-matching using map topology. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011.

[41] L. Qi, D. Sun, and G. Zhou. A new look at smoothing newton methods for nonlinear complementarity problems and box constrained variational inequalities. *Mathematical Programming*, 2000.

[42] R. Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[43] RTKLIB: An open source program package for GNSS positioning. http://www.rtklib.com/. 2011.

[44] E. M. d. Souza, J. F. G. Monico, and A. Pagamisse. Gps satellite kinematic relative positioning: analyzing and improving the functional mathematical model using wavelets. *Mathematical Problems in Engineering*, 2009.

[45] R. I. Technologies. What is differential GPS. http://www.roseindia.net/technology/gps/what-is-Differential-GPS.shtml. February 2008.

[46] The OpenXC Platform. http://openxcplatform.com.

[47] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, L. Girod, et al. Accurate, low-energy trajectory mapping for mobile devices. In *NSDI*, 2011.

[48] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[49] S. Thrun. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002.

[50] Torque Pro . https://play.google.com/store/apps/details?id=org.prowl.torque&hl=en.

[51] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006.

[52] Ublox Chips. http://www.ublox.com/en/.

[53] UNAVCO consortium. http://www.unavco.org/instrumentation/networks/status/pbo/overview/.

[54] B. Wiśniewski, K. Bruniecki, and M. Moszyński. Evaluation of rtklib's positioning accuracy usingn low-cost gnss receiver and asg-eupos. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 7(1), 2013.

[55] O. J. Woodman. An introduction to inertial navigation. *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, 2007.

[56] M. Yu. *Improved positioning of land vehicle in ITS using digital map and other accessory information*. PhD thesis, The Hong Kong Polytechnic University, 2006.

[57] X. Yu and E. Salari. Pavement pothole detection and severity measurement using laser imaging. In *Proceedings of IEEE International Conference on Electro/Information Technology (EIT)*, 2011.