Poster Abstract: VeLoc: Finding Your Car in the Parking Lot

Mingmin Zhao, Ruipeng Gao, Jiaxu Zhu, Tao Ye, Fan Ye, Yizhou Wang, Kaigui Bian, Ming Zhang
Department of Electrical Engineering and Computer Science
Peking University, Beijing, China
{zhaomingmin, gaoruipeng, zhujiaxu, yetao, yefan, Yizhou.Wang, bkg, mzhang}@pku.edu.cn

Abstract

We present VeLoc, a smartphone-based vehicle localization approach that tracks the vehicle's parking location using the embedded accelerometer and gyroscope sensors. It harnesses constraints imposed by the map and landmarks (e.g., speed bumps) recognized from inertial data, employs a Bayesian filtering framework to estimate the location of the vehicle. We have conducted experiments in 3 parking lots of different sizes and structures, using 3 vehicles and 3 kinds of driving styles. We find that VeLoc can always localize the vehicle within 10m, which is sufficient for the driver to trigger a honk using the car key.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human Information Processing*; I.5.1 [**Pattern Recognition**]: Models—*Neural Nets*

Keywords

Vehicle Localization, Indoor Localization, Inertial Tracking, Robotic Navigation, Virtual Landmarks

1 Introduction

Remembering where a vehicle was parked has proven a hassle in large parking structures. Existing RF signature based indoor localization technology is not applicable where such signals may not be available such as at underground parking lots. Instrumenting additional sensors may solve the problem but at the cost of significant overheads in time, money and human efforts.

VeLoc is a vehicle localization system that utilizes accelerometer and gyroscope sensors in the smartphone to provide accurate vehicle localization. It does not rely on GPS or RF signals, neither require any additional sensors to instrument the parking ground.

Realizing such an inertial-based solution, however, involves non-trivial challenges. First, the driver may place the phone in arbitrary positions and road conditions may jolt the phone to change its position on the course. How to estimate the pose (i.e., the relative orientation of the smartphone to the vehicle) despite all such uncertainty and disturbances? Second, inferring the vehicle's traveling distance or even trajectory is still difficult due to the lack of periodic acceleration patterns, which are fundamental in step-counting techniques [10, 13]. Also, identifying which types of landmarks can be reliably recognized despite different parking lots, vehicles and driving styles, and how, remain open questions. Finally, since both the pose and landmark detection results contain errors, how can one track the location of a vehicle accurately, such that the driver can always find the vehicle?

As shown in Figure 1, VeLoc contains three components to deal with the above challenges. A *pose estimation* module estimates the pose of the smartphone inside the vehicle. A *landmark detection* module finds unique patterns corresponding to different types of landmarks and classify them reliably. A *location estimation* module determines the vehicle's final location from the map and detected landmarks.

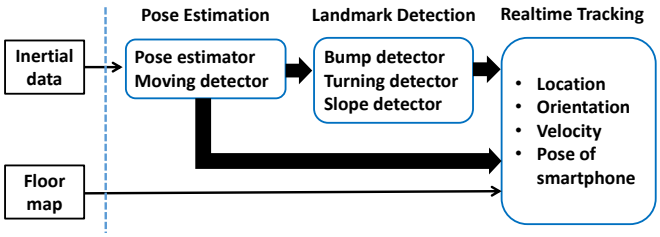
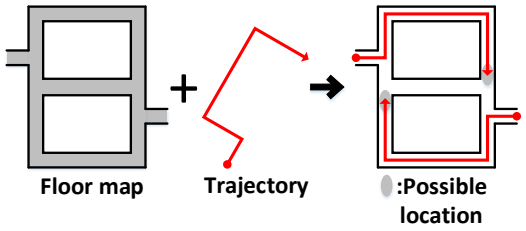


Figure 1. Three components of VeLoc. Inertial data is used to compute the smartphone's pose in the vehicle, then it is further processed to detect certain landmarks during driving, and the augmented particle filter harnesses constraints from landmark detection and the floor map for vehicle localization.

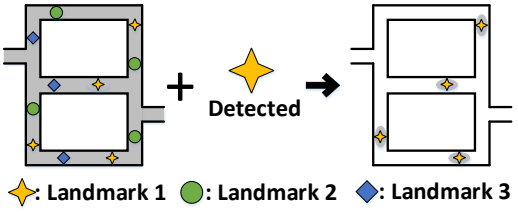
The intuition behind VeLoc is that although a noisy tra-

SenSys'14, November 3–6, 2014, Memphis, TN, USA.

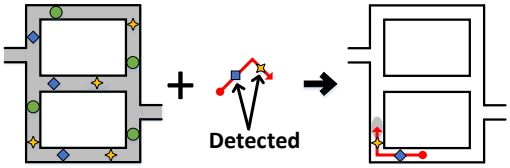
jectory does not, by itself, reveal a vehicle's location, using the constraints imposed by the parking structures map and detected landmarks is able to help vehicle tracking(shown in Figure 2).



(a) Localization using constraints imposed by the map.



(b) Localization using detected landmarks.



(c) Localization using both map constraints and detected landmarks.

Figure 2. Intuition of localization. (a) trajectories can be used for localization since only a few paths on the map could accommodate the trajectory. (b) once a landmark is detected, only a few positions marked with the same kind of landmark are possible. (c) using both map constraints and detected landmark could narrow down the uncertainty more quickly.

2 Pose Estimation

The sensor readings from a smartphone measure the physical quantity in the local coordinate system of the phone, e.g., the acceleration components along the phone's width, length and depth directions (x', y' and z' as shown in Figure 3). Because in most cases the phone's axes are not exactly aligned with respective axes of the vehicle, a transformation is needed to derive the acceleration or attitude angles of the vehicle from those of the phone.

In this section, we describe how to estimate the directions of the vehicle's axes ((x,y,z) in Figure 3) in the phone's coordinate system. First, we find the direction of the vehicle's z axis from the gravity, assuming that the vehicle is on a level surface. Next we detect the direction of the vehicle's y axis from its acceleration and de-acceleration when moving along straight lines. After y and z axes are decided, the x axis is determined in the phone's coordinate system as well.

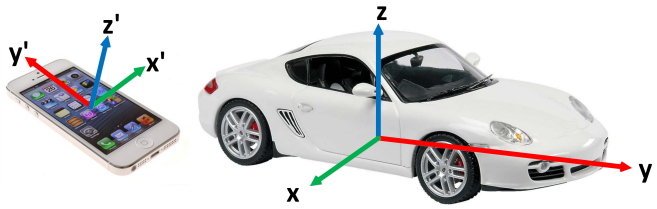


Figure 3. Coordinate systems of a vehicle (x,y,z) and a smartphone (x',y',z').

2.1 Z-axis Estimation

The z-axis of the vehicle and the gravity are on exact opposite directions when the vehicle is on level ground. Thus we can tell the z-axis direction from that of the gravity. When the phone remains static or moves at constant speed, the only force and thus acceleration on the phone is the gravity. When the phone has other acceleration, various techniques can be used [21] to extract the components of the gravity on the phone's three axes at reasonable accuracy (e.g., iOS has an API to obtain those three components). Since vehicle movements will cause small, random disturbances on x and z axes acceleration even when moving along level, straight lines, we further average the extracted gravity samples within a time window. Thus such disturbances cancel out themselves on opposite directions of the same axis.

Given the z-axis direction in the phone's coordinate system, the angle the vehicle makes a turn on the xy plane can be derived from the gyroscope data of the phone after a transformation. Thus we can tell whether the vehicle is moving along straight lines or making a turn.

2.2 Moving Detection

To estimate the y-axis direction, we need to detect wether the vehicle is moving after knowing that it is not making a turn. To this end, we calculate the variance of accelerometer readings on three axes of the smartphone's coordinate system averaged within a sliding window. Theoretically, the accelerometer readings remain unchanged and the variance is zero if the vehicle is static or moving at a constant speed along a straight line. In practice, due to inevitable vibrations, small accelerations always happen and the variance cannot be zero. We set a threshold and consider the vehicle static if the variance is smaller than the threshold. We find that this works very well for moving detection. Figure 4 shows the normalized variance distribution during static and moving states. We can see that the overlapping portion is small compared to total areas.

2.3 Y-axis Direction Estimation

Now we estimate the direction of the y-axis in the smartphone's coordinate system. During straight driving at constant speed, theoretically, the accelerometer readings are

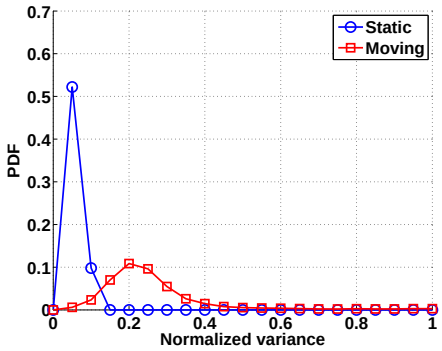


Figure 4. Distribution of normalized standard deviation during static and moving states.

non-zero on the y-axis only (assuming gravity is removed). Thus the direction of acceleration is that of the y axis in the phone's coordinate system. In practice, there are some noise on the x-axis because roads cannot be perfectly smooth.

Suppose that the direction of acceleration on the xy plane is represented by a unit vector $(cos\theta, sin\theta)$ on the xy-plane of the vehicle. Given a set of accelerometer samples, we first project them onto the xy-plane of the vehicle. This can be done because the z axis' direction is known. Denote the projected results $(x_i, y_i), i = 1, ..., k$. We want to find the θ such that the total deviation of each of them from the true y axis direction is minimized:

$$\sum_{i=1}^{k} (x_i cos\theta + y_i cos\theta)^2 \quad (1)$$

$$= \quad asin2\theta + bcos2\theta + c \quad (2)$$

$$= \quad \sqrt{a^2+b^2} sin(2\theta+\gamma) + c \quad (3)$$

where $a = \sum_{i=1}^{k} x_i y_i, b = \sum_{i=1}^{k} \frac{x_i^2 - y_i^2}{2}, c = \sum_{i=1}^{k} \frac{x_i^2 + y_i^2}{2}, tan\gamma = \frac{b}{a}$.

This equation has a closed-formed solution:

$$\theta_1 = \frac{1}{4}\pi - \frac{\gamma}{2} \quad \theta_2 = \frac{5}{4}\pi - \frac{\gamma}{2} \quad (4)$$

In the solution, we can see that there are two optimal solutions to θ, representing the forward and backward directions of the vehicle, denoted by θ_1 and θ_2. Next we decide which is the true direction of the y-axis of the vehicle (i.e., the forward direction).

2.4 Forward Direction Estimation

First we suppose that θ_1 is the correct direction. Then we can get the estimated (x, y, z) directions in the smartphone's coordinate system and project acceleromater readings to this coordinate system of the vehicle. In a short time period when the vehicle starts moving from static, there should be more positive acceleration on the y-axis direction.

We use a sliding time window starting at the beginning of the trace. Once we detect that the vehicle starts moving in this time window, we calculate the proportion of positive accelerometer readings within this time window such as:

$$\rho = \frac{1}{k} \sum_{i=1}^{k} y_i^3 \quad (5)$$

We use cubic form rather than linear form to better filter small noises caused by vibration of the vehicle, and focus on the real accelerometer readings caused by the acceleration of the vehicle.

However, this value is insufficient to determine whether θ_1 or θ_2 is the correct direction. Then, we put this value into sigmoid function and derive a probability of how θ_1 likely equals θ:

$$p(\theta = \theta_1) = \frac{1}{1 + e^{-\rho}} \quad (6)$$

The particle filter (in Section 4) then uses this probability as a prior to accurately converge on the correct direction—θ_1 or θ_2. This method helps narrow down the uncertainty caused by two opposite directions.

3 Landmark Detection

Landmarks along a road in certain environments are highly likely to cause distinguishable patterns in the observed inertial sensor data when a vehicle passes over them. Typical landmarks in a parking structure include bumps, turns, and slopes. In this section, we present three robust algorithms for detecting these three types of landmarks using inertial sensor data.

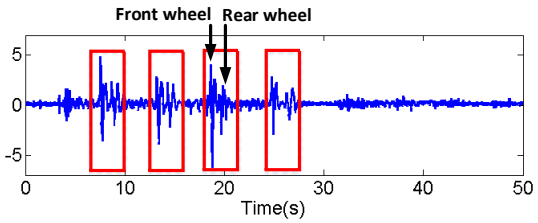
3.1 Bump Detection

Bumps are used to limit the vehicle speed inside parking lots for the sake of safety. When a vehicle pass over a bump, it causes a jolting that generates a detectable pattern in the inertial sensor data. Note that other landmarks such as drainage outlets or joint part of fire shutters, may also cause similar jolting patterns as bumps and thus are categorized as bumps as well in this paper.

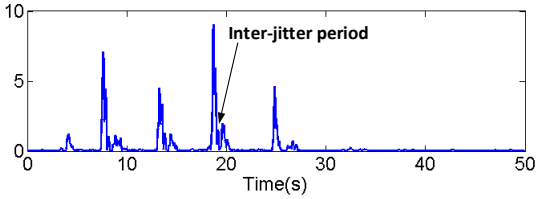
Acceleration-based detection. When a vehicle passes over a bump, jitters occur in the signals of both accelerometer and gyroscope sensors of a smartphone inside the vehicle. Acceleration along the z-axis (the vertical direction) can best characterize such jitters.

Some previous work [6] calculate the standard deviation of the z-axis acceleration, and use a pre-determined threshold to detect bump-passing. Figure 5(a) shows the acceleration on the z-axis for a car driving over four bumps along a straight road, and the standard deviation is shown in Figure 5(b). The first jitter at the 4th second is from a sudden change of the phone's pose, which is less significant than other four caused by the bumps.

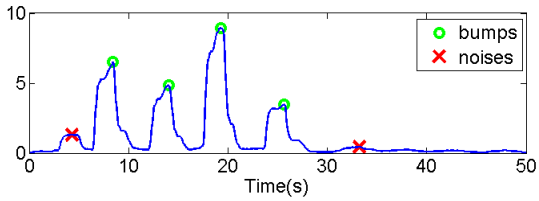
However, due to the variation of parking lots structures, vehicle models and driving styles, it is difficult to find a universal threshold that fits all cases. Moreover, occasional phone pose changes during driving may also cause false alarm (e.g, the one at the 4th second in Figure 5 (b)).



(a) Acceleration along the z-axis with four jitters.



(b) Standard deviation with four high peaks

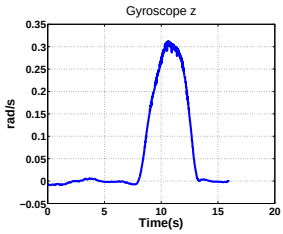


(c) Convoluted standard deviation with four distinguishable peaks

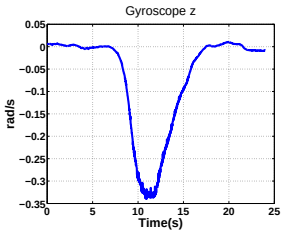
Figure 5. Bump detections: (a) four jitters (circled ones in the figure) in the acceleration on the z-axis; (b) four high peaks in the standard deviation of the z-axis acceleration; (c) four distinguishable peaks after applying convolution to the standard deviation with a Gaussian filter, where green (or red) markers indicate the local maximums for detected bumps (or false alarms).

Proposed bump detection algorithm. We propose a robust bump detection algorithm that is less sensitive to the choice of the threshold and occasional pose changes of the phone.

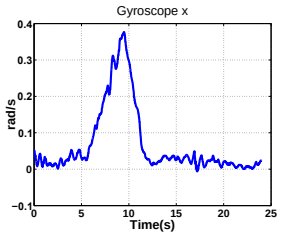
We observe that when driving straight, each bump can incur two consecutive car jolts when its front wheels pass over a bump followed by rear ones. Note that the first jolt usually is more significant than the second as the phone is often placed in the front of a car, and it is also because that the vehicle's speed usually has been reduced when the front wheels pass over the bump. Or a bump may cause four jitters at a turning corner because each of wheels' pass over the bump at different times. These consecutive jitters lead to two or four peaks in the standard deviation of the z-axis acceleration. We smooth the standard deviation curve by convolving it with a Gaussian filter. The two jitters caused by a bump in Figure 5(b) is merged into a single distinguishable peak in Figure 5(c). The time stamp of the peak is the time when the center of the vehicle is passing over the bump. After the convolution, it is much easier to distinguish jolts caused by bumps from those incurred by sudden phone pose changes, because the latter does not have subsequent followers.



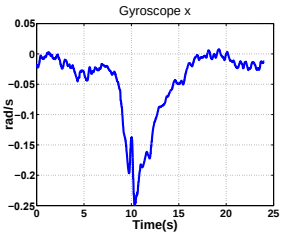
(a) Left turn



(b) Right turn



(c) Up slope



(d) Down slope

Figure 6. Inertial sensing data used for turn and slope detection. (a)(b) show turns using gyroscope signal along the z-axis. (c)(d) show slopes using gyroscope signal along the x-axis.

3.2 Turn and Slope detection

Both turns and slopes can be detected by gyroscope signals when rotational velocities are measured: horizontal turns result in rotational velocity along the z-axis as shown in Figure 6(a)(b), and slopes lead to rotational velocity along the x-axis as shown in Figure 6(c)(d). Based on this observation, VeLoc calculates the accumulated rotation angle as the detection statistic,

$$C(t;\mathbf{s},k) = |\sum_{\tau=t-k}^{t} \mathbf{s}(\tau)|, \tag{7}$$

where $\mathbf{s}(\tau)$ is the gyroscope reading at time τ of an axis, and k defines the size of the data collection window. The magnitude of C depends on the angle that the vehicle turns.

- **A turn** is detected at time t if and only if

$$C(t;\mathbf{s}_z,k) > \frac{\pi}{6} \tag{8}$$

  where $\mathbf{s}_z$ is the gyroscope signal along the z-axis.

- **A slope** is detected at time t if and only if

$$C(t;\mathbf{s}_x,k) > \frac{\pi}{18}, \tag{9}$$

  where $\mathbf{s}_x$ is the gyroscope signal along the x-axis. Both the thresholds are chosen empirically.

4 Realtime Tracking

In the vehicle tracking problem, a vehicle's initial position and its relative movement w.r.t. each of its previous states are two fundamental issues to be concerned. The motion trajectory of a vehicle can, in principle, be computed by integrating the inertial sensor measurements over time given its initial position. However, due to noise and error accumulation

during the integration, the dead-reckoned trajectories [15,23] diverge from the truth rapidly (the prediction error grows cubically in time). The drift incurred by the inertial movement unit of the phone will typically exceed 100 meters after one minute of operation [23].

In this section, we will introduce a set of algorithms in VeLoc that are designed to simultaneously harness constraints imposed by the parking structure map and detected landmarks for localization a vehicle. Intuition and probabilistic explanation is presented in Section 4.1, followed by algorithm details in Section 4.2. Finally, Section 4.3 describes how the algorithms are used to track vehicles in the parking lots.

4.1 Intuition and Probabilistic Framework

Although a noisy trajectory does not, by itself, reveal a vehicle's location, using the constraints imposed by the parking structures map and detected landmarks is able to help vehicle tracking. This is because (i) there may be only a few paths on the map that could accommodate the trajectory as shown in Figure 7(a). (ii) A detected landmark (e.g., a bump, turning or slope) can further pin down the positions where a vehicle might be (as shown in Figure 7(b)) since there are limited number of such landmarks on the map. Hence, using the synergy of both constraints is able to trace a vehicle more accurately (as shown in Figure 7(c)).

The problem of cubically growing error in dead-reckoned trajectories [15, 23] can also be resolved by map constraints and detected landmarks. These information provides an opportunity to re-calibrate a vehicle to correct location before the estimation error grows rapidly.

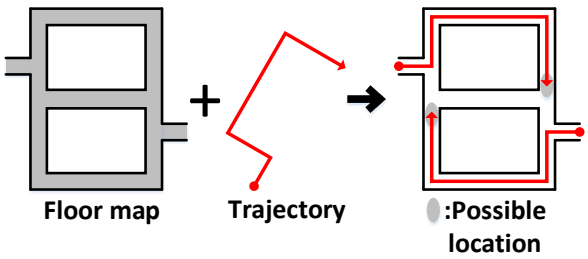
In VeLoc, we represent the uncertainty of a vehicle's states as a probability distributions over the state space. (A vehicle state includes its location, heading direction and velocity.) The tracking problem is posed as a sequential Bayesian filtering problem under the sequential importance re-sampling framework implemented by an Augmented Particle Filtering method [14].

We use s_t to represent the state of the vehicle at time t, the probability distribution of vehicle states is denoted as $b(s_t)$. Bayesian filtering possesses the following two essential steps: time update and measurement update.

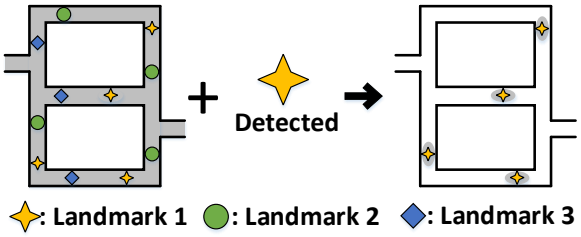
Time update is to estimate the state at time t given the previous state. Different from the dead-reckoning, which simply updates the previous state with the movement, sequential Bayesian Filtering framework models the noise in movement with a probability distribution which defines the possible transitions from one state to the next as $p(s_t|s_{t-1},m_t)$, where m_t is the movement at time t. Thus, the prior is obtained as

$$\overline{b}(s_t) = \int p(s_t|s_{t-1},m_t)b(s_{t-1})ds_{t-1}. \quad (10)$$

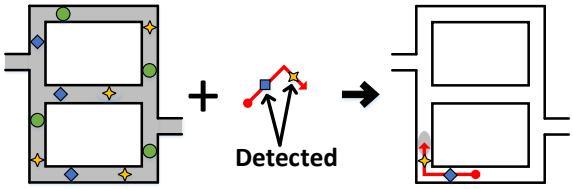
Measurement update is to update the probability distribution of the current state given the current measurement m_t. Using the Bayes rule, the posterior distribution over the state



(a) Localization using constraints imposed by the map.



(b) Localization using detected landmarks.



(c) Localization using both map constraints and detected landmarks.

Figure 7. Intuition of localization. (a) shows that trajectories can be used for localization since only a few paths on the map could accommodate the trajectory. (b) shows that once a landmark is detected, only a few positions marked with the same kind of landmark are possible. (c) shows that using both map constraints and detected landmark could narrow down the uncertainty more quickly.

can be calculated as

$$b(s_t) = \eta_t p(m_t|s_t)\overline{b}(s_t), \quad (11)$$

where η_t is a normalization factor.

4.2 Augmented Particle Filter

Particle filter is an alternative nonparametric implementation of the sequential Bayesian filter. The key idea of the particle filter is to represent a distribution by a set of samples drawn from the distribution. The samples are called "particles" denoted as:

$$\mathcal{S}_t := s_t^{(1)}, \ldots, s_t^{(M)},$$

where M denotes the number of particles in the particle set $\mathcal{S}_t$. Each particle $s_t^{(i)} (1 \le i \le M)$ is a concrete instantiation of the state at time t.

Augmented particles are particles in an augmented state space. In VeLoc, the state of a particle is a five-tuple defined as (x, y, θ, v, d), where x, y are position of a vehicle on a 2D floor plan, θ is the heading direction of the vehicle and v is the velocity along the y-axis of the vehicle and d is a binary variable, which determines the pose of the phone from two possible poses as described in Section 2.4.

Time update in our algorithm is performed by generating a hypothetical state $s_t^{(m)}$ at time t from its ancestor $s_{t-1}^{(m)}$ and the current measurement $m_t = (\omega_t, a_t)$, where ω is the rotational velocity along the z-axis of the vehicle and a is the acceleration along the y-axis of the vehicle. This step involves sampling from the transition distribution $p(s_t|s_{t-1}, m_t)$. We use Gaussian distribution $\mathcal{N}(s_t; \mu_t, \Sigma_t)$ to represent the transition distribution. The mean μ_t is defined as

$$\mu_t = \begin{pmatrix} x_{t-1} + v_{t-1}\Delta t \cdot \cos\theta_{t-1} \\ y_{t-1} + v_{t-1}\Delta t \cdot \sin\theta_{t-1} \\ \theta_{t-1} + \omega_t \Delta t \\ v_{t-1} + a_t \Delta t \\ d_{t-1} \end{pmatrix} \quad (12)$$

, and Σ_t is diagonal matrix modeling measurement noise.

Measurement update in our algorithm first computes the so-called "importance weights" (denoted as $w_t^{(m)}$) of each particle $s_t^{(m)}$ according to $w_t^{(m)} \propto p(m_t|s_t^{(m)})$. In VeLoc, the weight w_t is computed as

$$w_t := \prod_{i=0}^{3} w_{ti}, \quad (13)$$

The sum of the weights at each time t is normalized to one. And each w_{ti} is described as follows.

- **Constraints imposed by the map.** w_{t0}=1 if the position of the particle is accessible, otherwise 0.
- **Detected landmarks.** For each w_{ti} of the i-th type of landmarks (i=1,2,3), w_{ti}=$\mathcal{N}(D_i(x_t, y_t); 0, \sigma_i^2)$ if a landmark is detected, otherwise $w_{ti} = 1$. $D_i(x_t, y_t)$ is the distance to the closest landmark of the same type and σ_i^2 are parameter controlling scale of the distance.

The key idea of the particle filtering algorithm is *sequential importance re-sampling* that draws with replacement of M particles from the current set of particles according to their importance weights. VeLoc uses Low variance sampler [18] to fulfill the re-sampling task. It is worth mentioning that since re-sampling is the most time-consuming part of the algorithm, VeLoc does not perform re-sampling after every time step, instead, it maintains the importance weight for each particle, and the weights updated multiplicatively until the next re-sampling. In Veloc, re-sampling is performed every 10 time steps.

4.3 Vehicle State Estimation

Using augmented particle filter as described above, VeLoc can estimate the state of the vehicle, including its location. To illustrate how VeLoc works, we use an example scenario to drive the process of realtime tracking. Figure 8 shows the map of an underground parking lot with an actual route denoted by 7 key points (i.e., A to G as shown with time stamps).

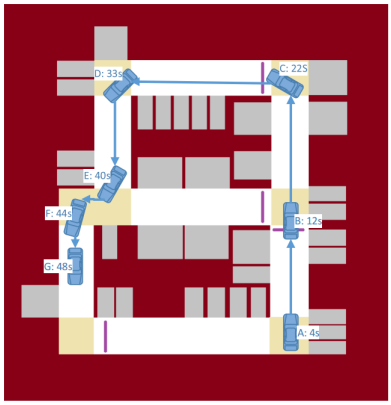


Figure 8. Map of the example scenario shown with an actual route denoted by 7 key points and the times when the vehicle passes them. The vehicle starts at A and stops at G.

Experiment with the initial position and heading direction of the vehicle. We first show in Figure 9 the positions of all the particles as time goes by. Since the initial position and heading direction are known, Figure 9(a) shows that all the particles are initialized around the entrance. Figure 9(b) and (c) show that the particles diverge due to growing uncertainties caused by noises in measurements. Figure 9(d) shows that particles slightly come closer after a speed bump detection reduces the uncertainty. After a left turn, the uncertainty is greatly reduced and particles come very close (Figure 9(e)-(g)). In Figure 9(f), particles with too great or too slow speeds will hit either the front wall or side wall, and eventually disappear. Only those with appropriate speed around that of the ground truth will be able to make the left turn and continue traveling along the top isle. Figure 9(h)-(k) show how three more turns each cause more concentration of the particles. Figure 9(l) shows the final position of all the particles and the average position is shown with a red point. Compared to the actual route (Figure 8), VeLocE produces a quite accurate final location.

Experiment with only the initial heading direction of the vehicle. Figure 10 shows the process if the initial position is unknown but the initial heading direction is somehow known (e.g, using compass to find which of the four directions imposed by the parking lot aisles is the most likely one). As shown in Figure 10(a), particles are initialized everywhere in the parking lot. As the vehicle starts moving up, constraints imposed by the map filter out many particles that would hit walls when moving up (shown in Figure 10(b)). Figure 10(c) shows more particles are eliminated when a landmark (i.e., speed bump) is detected, and only

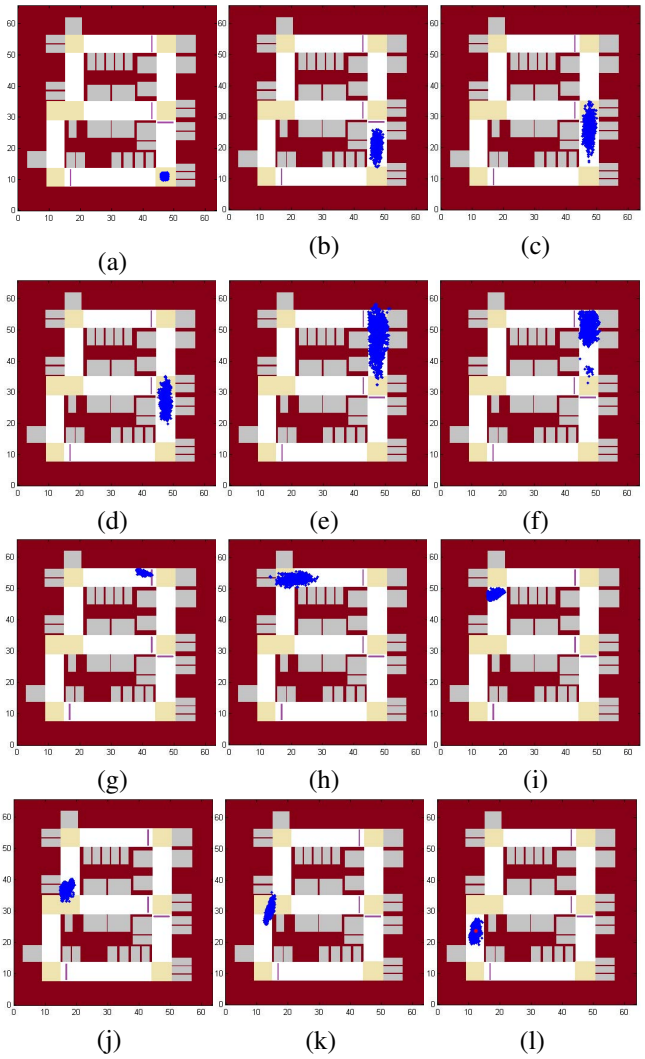


Figure 9. Particles over time with initial position and heading direction of the vehicle.

those just passing around a speed bump can survive. Figure 10(d)(e) show that the detected left turn eliminate many particles and the remaining ones cluster around the corner. Figure 10(e) looks similar to Figure 9(g), showing that VeLoc succeeds in estimating the state of the vehicle without the initial position. From there the process is similar to the previous one. Figure 10(f) shows the final position of all the particles and the average position is shown with a red point, which is very accurate as well. We can also trace back to determine the initial position of the vehicle which is not known at the beginning.

Experiment with neither the initial position nor heading direction of the vehicle. Figure 11 shows the results if neither the initial position nor the heading direction is unknown. VeLocE is still able to estimate the true state of the vehicle but it may take a litter longer to converge. As shown in Figure 11(a), particles are initialized everywhere in the

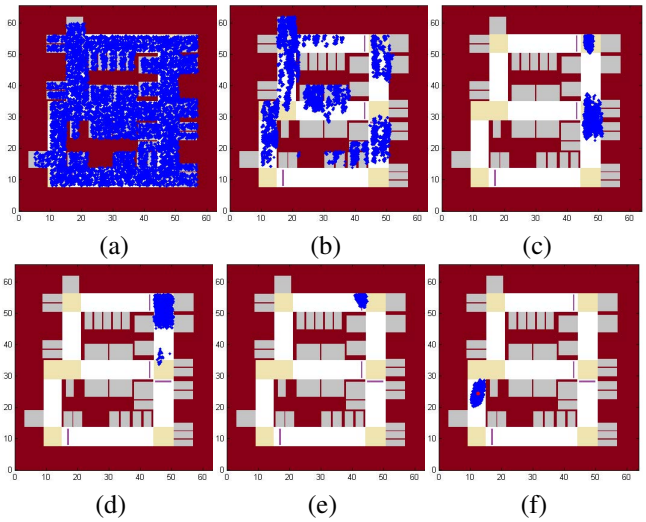


Figure 10. Particles over time with only the initial heading of the vehicle.

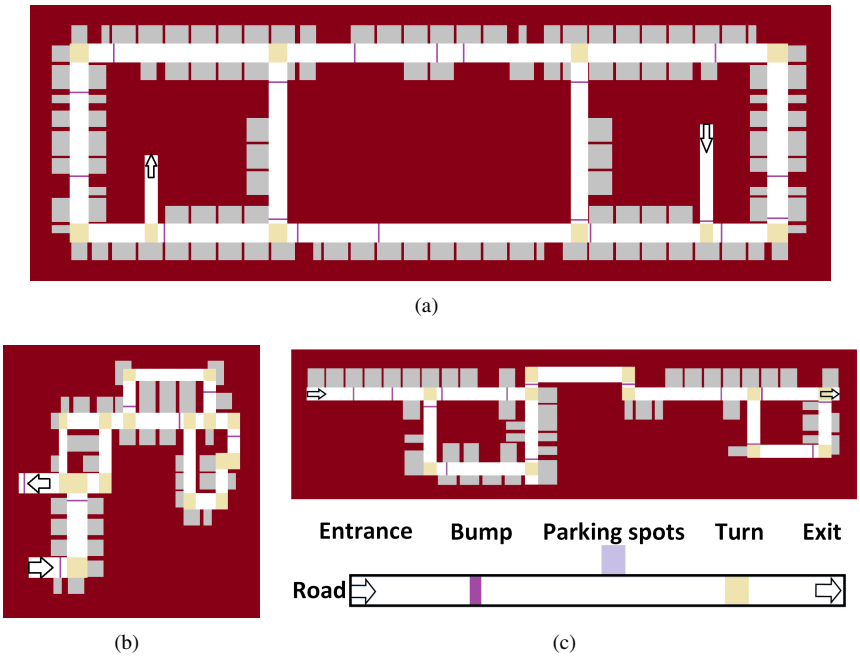
parking lot. Constraints imposed by the map (i.e., walls) filter out some particles but compared to Figure 10(b) they are still widely distributed (shown in Figure 11(b)). This is because the lack of heading direction gives more particles the chance to survive. Figure 11(c) shows how particles converge quickly when a landmark of speed bump eliminate those without a bump nearby. However, compared to Figure 10(c), they are still distributed more widely. In Figure 11(d), particles traveling in horizontal directions are eliminated because the vehicle keeps moving upwards, but those traveling vertically become more dispersed. Figure 11(e) shows that how a left turn detected remove most of the particles and leaving only close to the true location, the upper right corner. Figure 11(f) again shows the final positions of all the particles and their average position shown with a red point. From this example, we can see that VeLocE can produce very accurate results even both the initial position and the heading direction are unknown.

5 Performance Evaluation

5.1 Methodology

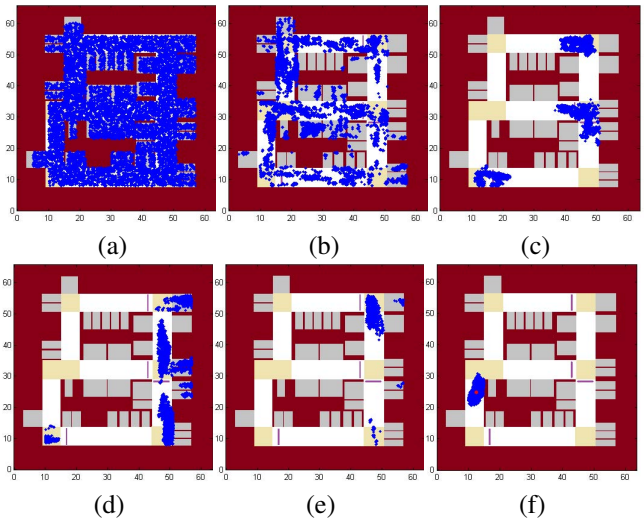
We use smartphones to collect motion sensor readings on a vehicle in three underground parking lots, and their size are $250m \times 90m$, $80m \times 90m$, and $180m \times 50m$, respectively. The floor plan of those three parking lots are shown in Figure 12. Each parking lot has one entrance and one exit, and there are 298, 68, 79 parking spots, 19, 7, 12 bumps, 10, 14, 11 turns and 4, 2, 2 slopes in each parking lot, respectively. During experiments, we conduct 20 vehicle traces in each parking lot with 4 iPhones with different poses to simultaneously collect inertial sensor data during the driving. We design a mould to fix 4 iPhones with different poses as shown in Figure 13. The mould is placed flat inside the vehicle with direction shown in Figure 13.

As to evaluate our system's robustness, we invite three



(a) (b) (c)

Figure 12. Floor map of three underground parking lots: (a) parking lot 1: $250m \times 90m$ **with 298 parking spots, 19 bumps and 10 turns. (b) parking lot 2:** $80m \times 90m$ **with 68 parking spots, 7 bumps and 14 turns. (c) parking lot 3:** $180m \times 50m$ **with 79 parking spots, 12 bumps and 11 turns.**



(a) (b) (c) (d) (e) (f)

Figure 11. Particles over time with neither the initial position or initial heading direction of the vehicle.

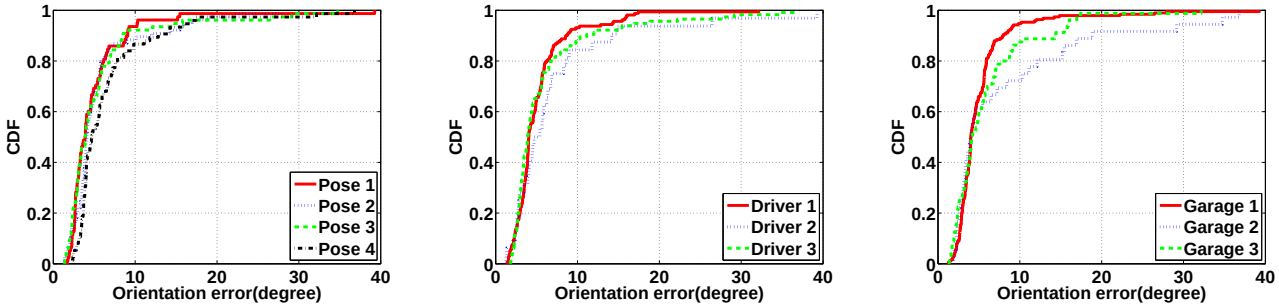
volunteers to drive their own cars in one parking lot, and their car's cost is around 10 thousand, 20 thousand and 30 thousand dollars, respectively. During each driving, we start the data collection app developed by us on 4 phones to record their accelerometer and gyroscope readings before vehicle enters a parking lot, and ends app after vehicle stops at a parking spot.

We measure each pose's position in the mould as ground truth for the pose estimation algorithm. The landmarks come across during the drive are recorded as ground truth for the landmark detection algorithm. Also, the final position where the vehicle stops is recorded as ground truth for the localization algorithm.

5.2 Evaluation of Pose Estimation

Here we evaluate the performance of pose estimation algorithms, namely the accuracy of estimated smartphone's pose inside vehicle. We measure the orientation error between vehicle's estimated orientation and its ground truth orientation both in smartphone's coordinate system. We use the control variate method to evaluate the effect of different smartphone poses, driving styles and parking lots.

Figure 14(a) shows the orientation error with different smartphone poses. We could observe that the 90-percentile error is around 9 degrees, and the largest error is less than 40 degrees. Figure 14(b) presents the effect of driving styles, namely drivers, on our pose estimation. We could observe that despite different drivers and performance of cars, we achieve similar pose estimation accuracy, also around 9 degrees at 90-percentile. Figure 14(c) illustrates that our algo-



(a) (b) (c)

Figure 14. Pose estimation error in different scenarios: (a) 4 different poses. (b) 3 different vehicles and drivers in one parking lot. (c) 3 different parking lots.

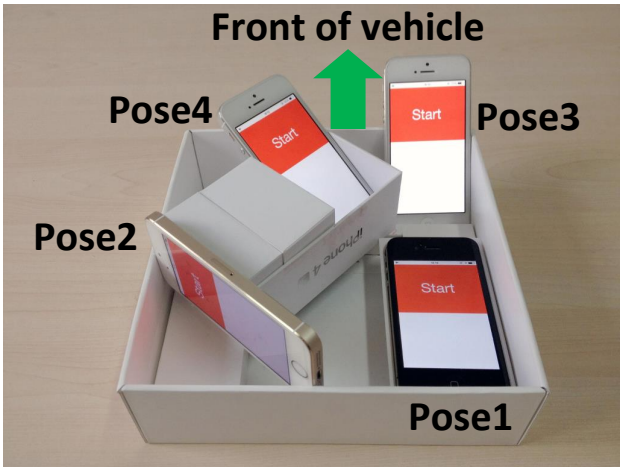


Figure 13. Mould is with 4 iPhones.

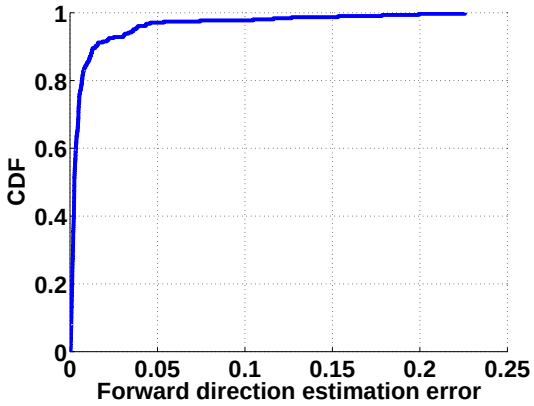


Figure 15. The distribution of estimated probability of the backward direction.

rithm works well in different parking lots, they all achieve 13 degrees accuracy at 90-percentile, and Parking lot 2 has the largest error of 16 degrees, due to its short straight road which we use to compute the vehicle's orientation.

In Section 3.4, we estimated the probability of the forward and backward directions of the vehicle, $p(\theta = \theta_2)$ and $p(\theta = \theta_2)$. And we also know which one is the real forward direction as we have recorded the ground truth. Then we can calculate the distribution of estimated probability of the backward direction,i.e., the estimation error. As showed in Figure 15, it achieve more than 99% accuracy at 90 percentile and a maximum error of 23%.

Figure 16 shows the pose estimation accuracy with different time windows. We could see that the 90-percentile pose estimation error is reduced when increasing the time window for pose estimation algorithms. Additionally, the localization error stays stable when time window is larger than 0.5s, which reflects realtime performance of our system.

5.3 Evaluation of Landmark Detection

Here we evaluate the performance of landmark detection using the precision and recall as metrics. As to measure the precision and recall of landmark detection, we set breakpoints where a certain landmark is announced to be detected, and we check whether it corresponds to a correct landmark on the floor map at that time stamp. We also compute how many landmarks the vehicle goes through from floor map as its ground truth number of landmarks.

Since our calibration of vehicle localization mainly relies on the landmark detection, we prefer higher precision rather than recall. This is because if we omit a certain landmark, we'll just miss a chance for recalibration. On the contrary, the detection of a nonexistent landmark will lead our particles to somewhere unknown.

Table 1 shows the recall and precision for different landmarks. We observe that turn detection has detected all the turns correctly. Bump detecion has the lowest precision of 87%, and recall of 83%. This is because gyroscope sensor is much more precise than accelerometer sensor on smartphone, and there are many activities can be confused with bumps.

Table 2 shows the precisions with different poses of smartphone, we can observe that all precisions are quite high, while twos are relatively lower. This is because that in some positions, the smartphones are also sensitive to the jolting

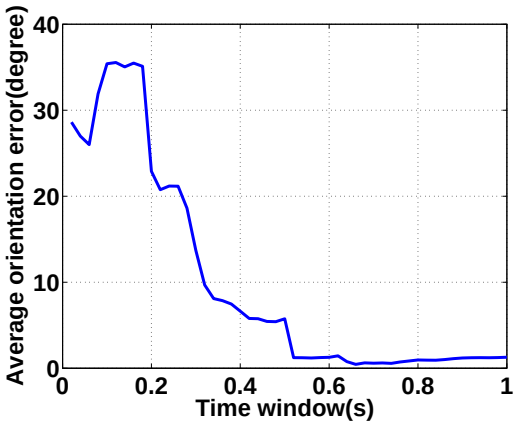


Figure 16. Orientation error of pose estimation for different time windows.

of the car, which may falsely be detected as a bump. Table 3 presents the effect of driving styles on our landmark detection. We achieve similar landmark detection precision, around 92%. Table 4 illustrates that we achieve quite different recall in different parking lots since the recall is effected by the property of landmarks.

5.4 Evaluation of tracking

Figure 17 shows the vehicle localization accuracy in different scenarios. The 90-percentile localization error is around 10m for all 4 poses, and the maximum errors are about 30m, shown in Figure 17(a). Additionally, as Figure 17(b) shows, different drive styles achieve localization accuracy around 10m at 90-percentile. Figure 17(c) shows the localization error in 3 parking lots, which are different since those parking lots have different shape and size.

Table 1. Landmark detection performance for different landmarks

	Bump	Turn	Slope
Precision	87%	100%	97%
Recall	83%	100%	95%

Table 2. Performance of landmark detection with different poses

	Pose 1	Pose 2	Pose 3	Pose 4
Precision	93%	88%	85%	93%
Recall	88%	89%	87%	85%

Table 3. Performance of landmark detection with different drivers

	Driver 1	Driver 2	Driver 3
Precision	93%	92%	91%
Recall	89%	90%	87%

Table 4. Performance of landmark detection with different garages

	Garage 1	Garage 2	Garage 3
Precision	94%	90%	92%
Recall	78%	93%	91%

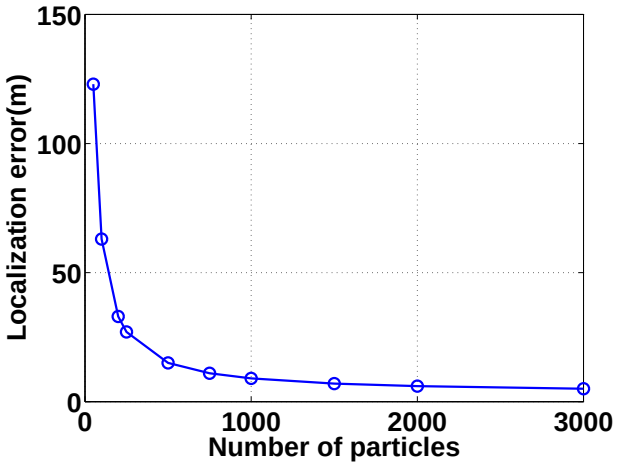


Figure 18. Performance of different particle numbers.

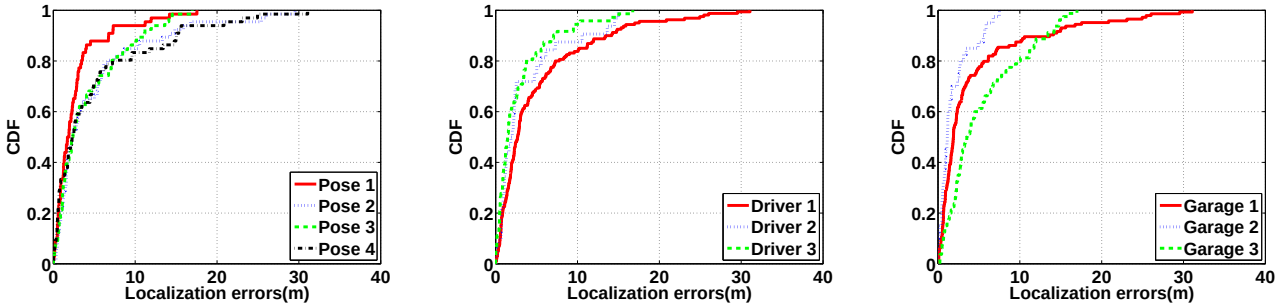
Additionally, we evaluate the performance of the localization algorithm with different particle numbers. As we can see in the Figure 18, localization errors shrink along with the increasing particle number and almost converge when then its number exceeds a certain threshold, namely 2000 particles.

6 Related Work

We present a brief introduction of the related work below to distinguish components of Veloc from existing technologies for estimating the phone pose, detecting the landmarks, monitoring the states of a robot, a pedestrian or a vehicle, etc.

Phone pose estimation. It is impractical to assume that the smartphone inside a vehicle has a known position or it is placed stationary. It is necessary to periodically estimate the phone pose in the vehicle. Wang et al. found it possible to align the smartphone's and vehicle's coordinate systems using gravity, accelerometer and gyroscope sensors [21]. Specifically, it aligns z-axis of the vehicle with its gravity direction; gyroscope is used to determine whether the vehicle is driving straight, then accelerometer readings are extracted as vehicle driving direction, namely y-axis of the vehicle. However, this approach fails to consider the case when the vehicle runs on a slope, and simply extracting accelerometer readings as vehicle direction is not robust.

Virtual landmark detection. In robot localization system, a robot is assumed to be exploring the space of interest that has various landmarks (e.g., barcode pasted on walls or a particular pattern painted on the ceiling). The equipped sensors on a robot, such as laser-based ranging and cameras, are used to detect these artificially placed landmarks. However, for smartphone-based indoor localization, it is the smartphone that is carried around by a user to explore the s-



(a) (b) (c)

Figure 17. Vehicle localization errors in different scenarios: (a) 4 different poses. (b) 3 different cars and drivers in one parking lot. (c) 3 different parking lots.

pace of interest. It is impossible to have robot sensors (e.g., laser ranging) on a commodity phone. Thus, researchers refer to virtual landmarks which are essentially ambient signatures or recognized patterns/activities that are perceivable by smartphone sensors [1, 16], and UnLoc [20] is believed to be the first to apply virtual landmarks towards deadreckoning. In VeLoc, we use sensor measurements to detect all kinds of road anomaly and turnings as landmarks. In addition, sensor measurements also provide cues for estimating the state of vehicles. For instance, different patterns between a immobile vehicle and a moving one can be view as a measurement of the velocity of the vehicle.

Robotic localization. SLAM is a popular technique in robotics which allows the robot to acquire a map of its environment while simultaneously localizing itself relative to this map [12]. Recently, WiFi-SLAM [7] was proposed to utilize the WiFi signal strength as the input of SLAM. Unlike SLAM, VeLoc assumes the availability of a map and the problem to be addressed is equivalent to the robot localization problem of determining the pose of a robot relative to a given map of the environment [18].

The early work on robot localization problem used Kalman Filters which is thought to be the earliest tractable implementations of the Bayes filter for continuous spaces. Subsequent work has been based on Markov localization, which is a better match in practice since it allows the robot's position to be modeled as multi-modal and non-Gaussian probability density functions. Of particular interest to us is the Monte Carlo Localization (MCL), or particle filtering based approach [8, 19]. Instead of representing the distribution by a parametric form, particle filters represent a distribution by a set of samples drawn from this distribution. Those particles are then evolved based on the action model and the measurements [18]. Again, robot localization typically depends on using explicit environment sensors, such as laser range finders and cameras. Moreover, the rotation of the robot wheels offer a precise computation of displacement.

Unlike robot localization systems, VeLoc is independent of laser ranging or cameras, but it uses smartphone sensors to compute the displacement and direction of vehicles and detect the virtual landmarks that are only specifically available in parking lots.

Dead-reckoning. Dead reckoning using inertial sensors is a well explored approach to monitor the states of a moving object or a pedestrian. However, conventional sensors used in these applications are very expensive. Recently, it is attractive to use smartphone sensors in indoor environments [4], since consumer mobile devices are increasingly being equipped with sensors such as accelerometer, gyroscope, magnetometer and barometer. However, directly applying this approach in indoor environments is non-trivial since many factors cause fluctuations in acceleration, resulting in erroneous displacements.

Many methods have attempted to mitigate the accumulation of error. Foot-mounted sensors have been shown effective in reducing the error [15, 23]. However, the accumulation of error remains when a smartphone pose is unknown. Outdoor localization schemes like CompAcc [5] employ a periodic GPS measurement to recalibrate the users location. UnLoc [20] provides an option to replace GPS with virtual indoor landmarks that can be detected using existing sensing modalities for calibration. Dead-reckoning techniques have been widely used in mobile computing community with the purpose of addressing the indoor localization problem [14].

To prevent the error accumulation, VeLoc simultaneously harnesses constraints imposed by the map and environment sensing. The only external input for VeLoc is a map of the indoor space of interest. Since the map of a place do not change for several months or years, no repeated manual efforts for calibration are required by VeLoc. In addition, the need for special-purpose hardware and infrastructure is avoided to make VeLoc more practical for the real-world use.

Estimation of vehicle states. There have been many active research efforts in using smartphones' embedded sensors (1) to monitor the states of vehicles (e.g. dangerous driving alert [10], car speaker [24] and CarSafe [25]); (2) to inspect the road anomaly/condition (e.g., Pothole Patrol [6] and Nericell [11]); and (3) to detect traffic accidents (Nericell [11] and WreckWatch [22]).

The vehicle speed is a critical input for implementing these applications. It is easy to calculate the outdoor vehicle speed by using the phone GPS [9, 17], while the GPS signal is weak or even unavailable at indoor parking lots. Some alternative solutions leverage the phone's signal strength to estimate the vehicle speed [2, 3].

7 Discussion

The start of the trace. It is possible that the car is not stationary when the VeLoc application starts, or that it does not start exactly at the entrance of the parking lot. In these cases, VeLoc can employ two heuristics to determine what time should be considered the start of the trace by examining when: (1) the smartphone loses its GPS signal, which is usually the time it enters a parking structure such as an underground lot; and (2) the first (or certain) landmark is detected, which can be used to back trace the time when the car enters the structure (as illustrated in Section 4.3).

Complex parking structures. There are many other types of complex parking structures such as a multi-storey parking lot with up/down-slope parking spaces. VeLoc can support vehicle tracking in these complex parking structures: (1) the turn and slope detection algorithms can jointly determine which storey the vehicle is located; (2) the pose estimation module can tell whether the vehicle is parking on the slope and calibrate the coordinate systems accordingly.

Other extreme cases. The smartphone may keep jittering when the vehicle is on a jerky road, or the driver puts the phone in his pocket. These extreme cases require VeLoc to estimate the phone pose in real time. To address this problem, VeLoc employs a sliding time window technique, which shows that it can track the pose quite accurately in just one second (Figure 16). This avoids large errors that may arise due to the latency in pose estimation.

8 Conclusion and Future Work

In this paper we describe VeLoc that can track the vehicle's movements and estimate the final parking location using the smartphone's inertial sensor data only. It does not depend on GPS or WiFi signals which may be be available in environments such as underground parking lots, or require additional sensors to instrument the environment. VeLoc first estimate the pose of the smartphone relative to the vehicle, so that inertial sensor data can be transformed into the coordinate system of the vehicle. It then detects landmarks such as speed bumps, turns and slopes, and combine them with the map information to estimate the vehicle's location using a probabilistic model. Experiments in three parking structures have shown that VeLoc can track the parking locations to within 4 parking spaces, which is enough for the driver to trigger a honk using the car key.

Currently VeLoc depends on accurate parking structure maps to reduce the uncertainty in the vehicle location. Since such maps are not always available, we plan to study how to obtain the map information, and track the vehicle when only incomplete and/or inaccurate map is available. This further extend VeLoc's capability in the real world.

9 References

[1] M. Azizyan, I. Constandache, and R. Roy Choudhury. Surroundsense: Mobile phone localization via ambience fingerprinting. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, MobiCom '09, pages 261–272, New York, NY, USA, 2009. ACM.

[2] G. Chandrasekaran, T. Vu, A. Varshavsky, M. Gruteser, R. Martin, J. Yang, and Y. Chen. Tracking vehicular speed variations by warping mobile phone signal strengths. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 213–221, March 2011.

[3] G. Chandrasekaran, T. Vu, A. Varshavsky, M. Gruteser, R. P. Martin, J. Yang, and Y. Chen. Vehicular speed estimation using received signal strength from mobile phones. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, Ubicomp '10, pages 237–240, New York, NY, USA, 2010. ACM.

[4] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury. Did you see bob?: Human localization using mobile phones. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, pages 149–160, New York, NY, USA, 2010. ACM.

[5] I. Constandache, R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[6] 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, pages 29–39, New York, NY, USA, 2008. ACM.

[7] B. Ferris, D. Fox, and N. D. Lawrence. Wifi-slam using gaussian process latent variable models. In *IJCAI*, volume 7, pages 2480–2485, 2007.

[8] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999:343–349, 1999.

[9] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 15–28, New York, NY, USA, 2008. ACM.

[10] J. Lindqvist and J. Hong. Undistracted driving: A mobile phone that doesn't distract. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, HotMobile '11, pages 70–75, New York, NY, USA, 2011. ACM.

[11] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: Using mobile smartphones for rich monitoring of road and traffic conditions. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 357–358, New York, NY, USA, 2008. ACM.

[12] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pages 593–598, 2002.

[13] S. Nawaz, C. Efstratiou, and C. Mascolo. Parksense: A smartphone based sensing system for on-street parking. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom '13, pages 75–86, New York, NY, USA, 2013. ACM.

[14] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 293–304, New York, NY, USA, 2012. ACM.

[15] P. Robertson, M. Angermann, and B. Krach. Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, Ubicomp '09, pages 93–96, New York, NY, USA, 2009. ACM.

[16] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 155–168, New York, NY, USA, 2011. ACM.

[17] 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*, SenSys '09, pages 85–98, New York, NY, USA, 2009. ACM.

[18] S. Thrun, W. Burgard, D. Fox, et al. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2005.

[19] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1):99–141, 2001.

[20] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 197–210, New York, NY, USA, 2012. ACM.

[21] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin. Sensing vehicle dynamics for determining driver phone use. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 41–54, New York, NY, USA, 2013. ACM.

[22] J. White, C. Thompson, H. Turner, B. Dougherty, and D. C. Schmidt. Wreckwatch: Automatic traffic accident detection and notification with smartphones. *Mob. Netw. Appl.*, 16(3):285–303, June 2011.

[23] O. Woodman and R. Harle. Pedestrian localisation for indoor environments. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, pages 114–123, New York, NY, USA, 2008. ACM.

[24] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin. Detecting driver phone use leveraging car speakers. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 97–108, New York, NY, USA, 2011. ACM.

[25] C.-W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de Oca, Y. Cheng, M. Lin, L. Torresani, and A. T. Campbell. Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 461–462, New York, NY, USA, 2013. ACM.