sensors

MDPI

Article

VeLoc: Finding Your Car in Indoor Parking Structures

Ruipeng Gao [1,*], Fangpu He [1] and Teng Li [2]

[1] School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China; 15301039@bjtu.edu.cn
[2] Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244, USA; tli01@syr.edu
* Correspondence: rpgao@bjtu.edu.cn

Received: 12 April 2018; Accepted: 29 April 2018; Published: 2 May 2018

Abstract: While WiFi-based indoor localization is attractive, there are many indoor places without WiFi coverage with a strong demand for localization capability. This paper describes a system and associated algorithms to address the indoor vehicle localization problem without the installation of additional infrastructure. In this paper, we propose VeLoc, which utilizes the sensor data of smartphones in the vehicle together with the floor map of the parking structure to track the vehicle in real time. VeLoc simultaneously harnesses constraints imposed by the map and environment sensing. All these cues are codified into a novel augmented particle filtering framework to estimate the position of the vehicle. Experimental results show that VeLoc performs well when even the initial position and the initial heading direction of the vehicle are completely unknown.

Keywords: vehicle localization; inertial tracking; virtual landmarks; mobile crowdsensing

1. Introduction

With the rapid growth of the number of personal vehicles and the increase in urban parking space demands, numerous large parking facilities have appeared, most of which are built underground. A driver may spend several minutes finding an available parking space in busy cities. What is worse, the driver is likely to forget the exact position of his car and may find it difficult to search for it in dim and maze-like parking lots. There is need to develop an easy-to-use system without specialized equipment to assist these drivers.

Localizing a vehicle in parking structures is a challenging problem because of the special environment most parking structures have. Although most vehicles are equipped with vehicle-mounted GPS and the mobile devices of the driver also have GPS, the indoor environment (especially underground) makes them impractical to use. The radio frequency (RF) fingerprinting of WiFi signals, which has been a popular approach to indoor localization [1–4], does not work here due to the poor WiFi coverage in parking lots.

The smartphone is an ideal device to address this problem. If the result of localization is recorded in the smartphone, the driver can use it to find his car when he comes back. In addition, providing drivers with a map of the parking lot and the real-time position of the vehicle can aid them to search for unoccupied parking places, especially in an unfamiliar environment .

In this paper, a system called VeLoc is developed to address the indoor vehicle localization problem. We develop a mobile sensor computing system to help users find their vehicles in parking lots. To use this system and drive safely, the user does not make a phone call but puts the mobile on the platform at the front of the vehicle. While the user is driving around, VeLoc leverages a three-axis accelerometer and a three-axis gyroscope present in the smartphone to track the vehicle and obtain a real-time position. When the vehicle finally stops, our system records the parking position on the map for the user, who may need it to find the vehicle later.

The only external input that VeLoc depends upon is a map of the indoor space of interest, which is necessary not only for the purpose of tracking but also to show the localization result to the users. Unlike common tracking methods, VeLoc does not need the initial state of the vehicle as an input since it is best to minimize the amount of explicit input or other actions from users. Shown in Section 5, VeLoc is robust and performs well even when both the initial position and the initial heading direction are unknown. If other information can be provided for the initial state of the vehicle (for example, in nearly all cases vehicles need to stop at one of the entrances of the parking lots), our system will converge to the true state of the vehicle more quickly.

There are three key ideas behind the automatic estimation of vehicles' locations in parking lots. We introduce these ideas first, and then describe how they are codified together to address the vehicle localization problem.

(1) *Knowing the starting location, the motion trajectory of a device can, in principle, be computed by integrating inertial sensor measurements over time.* This is called urban dead reckoning. Due to noise in the mobile sensors and the error that occurs during integration, the dead-reckoned trajectories are accurate at the beginning, but diverge from the truth quickly over time. Recalibration is necessary to prevent the accumulation of errors. The observation that smartphones inside vehicles provide simpler motion than for walking people shows possibility to use dead reckoning in the vehicle localization problem.

(2) *The constraints imposed by the map and the regularity of vehicles' motion filter out the infeasible locations over time and converge on the true location.* Some examples of constraints are that the vehicle cannot move through a wall, or that vehicles in other areas cannot be marked on the map. This information can not only be used as cues for recalibration in dead reckoning but also by itself allows for the localization of a vehicle from inertial sensor measurements. For instance, assuming we have no idea where the vehicle is, inertial sensor measurements will provide a trajectory when the vehicle is moving. While the above information does not, by itself, reveal location, it could when viewed together with a map since there may be only one path on the map that could accommodate the trajectory observed. Thus, at the conclusion of the driving, we can infer that the vehicle's final location and then trace back to infer the starting location.

(3) *Inertial sensor measurements show particular patterns when the vehicle is turning, coming to a slope, or traveling through "road anomalies", including speed bumps and potholes as well as other rough road conditions [5].* (We only use particular patterns on inertial sensor measurements to define several landmarks while the indoor environment is rich with ambient signals, like sound, light, magnetic field, temperature, WiFi, 3G, etc. SurroundSense [6] extends this idea and builds a map using several features.) We can define several kinds of landmarks as the positions at which inertial sensors inside vehicles provide measurements with particular patterns. Once those landmarks can be detected automatically, this kind of information can be used to recalibrate dead reckoning. The density of landmarks can be guaranteed since different building structures force vehicles to behave in specific ways and parking lots provide enough "road anomalies", especially speed bumps.

To combine the above ideas into VeLoc, we represent the uncertainty on the state of vehicles explicitly as probability distributions. We incorporate the uncertainty arising from sensing and the information provided by the map or detected landmarks into a novel Bayes Filtering framework, specifically, the *augmented particle filtering* framework. While particle filtering in the context of localization typically uses particles only to represent the uncertainty in location, we design novel particles that also incorporate the uncertainty in terms of other aspects such as the direction of the smartphone and the velocity of the vehicle. To use landmark information for recalibration, we use pattern recognition and machine learning techniques to find particular patterns and a implement real-time classification algorithm in VeLoc to detect predefined landmarks automatically. The result of detection is used as input to augmented particle filtering algorithm.

The Bayes filtering algorithm possesses two essential steps: the time update and measurement update. Similar to dead reckoning, VeLoc estimates the state of every particle based on the previous state at the time update step. During the measurement update, VeLoc utilizes all the information, as measurements consist of inertial sensor measurements, constraints imposed by the map, and the detected landmark, together with conditional distributions of measurements to estimate the probability of every particle.

Our main contributions may be summarized as follows:

- This is the first system (to our knowledge) to address the real need for localizing vehicles in parking structures where GPS do not work and WiFi signals are not available.
- We identify an opportunity to simultaneously harness constraints imposed by the map and environment sensing for localization. Our approach does not require calibration nor the installation of additional infrastructure.
- VeLoc overcomes the error accumulation problem. We show experimentally that VeLoc achieves a localization error of 3∼5 m in an underground parking structure.

The rest of the paper is organized as follows. Section 2 presents the background and related work, followed by system architecture in Section 3. Design details are presented in Section 4, while results is discussed in Section 5. Section 6 concludes this paper and discusses future work.

2. Background and Related Work

VeLoc draws on prior work in multiple areas, including dead reckoning, robotic navigation, and road surface monitoring.

2.1. Dead Reckoning

Dead reckoning using inertial sensors is a well-studied research area. Typical sensors used in such domains are of high-quality but expensive. Utilizing noisy smartphone sensors in indoor environments [7] is attractive since consumer mobile devices such as smartphones are increasingly being equipped with sensors such as accelerometers, gyroscopes, magnetometers, and barometers. However, directly applying the idea to human-scale environments is non-trivial since several factors cause fluctuations in acceleration, resulting in erroneous displacements.

Many methods have attempted to address the accumulation of error. Foot-mounted sensors are able to reduce the error [8,9], but the accumulation of error still remains and this method is not in accordance with the typical placement of devices such as smartphones. Outdoor localization schemes like CompAcc [10] have triggered periodic GPS measurements to recalibrate the user's location. UnLoc [11] presents an option to replace the GPS with virtual indoor landmarks that are created using existing sensing modalities. In the area of robotics, methods based on map of the place of interest implement high-accuracy localization [12,13]. Mobile computing uses this idea to address the indoor localization problem [14], and characterizations of the smartphone's inertial sensors are well investigated for vehicle tracking [15–17].

To prevent the error from being accumulated, VeLoc simultaneously harnesses constraints imposed by the map and environment sensing. The only external input that VeLoc depends on is a map of the indoor space of interest. Since the map of a place does not change for several months or years, no calibration is needed in VeLoc. In addition, the need for special-purpose hardware and infrastructure is avoided to make VeLoc more practical for real use.

2.2. Robotic Navigation

A highly popular and successful technique in robotics, called SLAM, allows the robot to acquire a map of its environment while simultaneously localizing itself relative to this map [18]. Recently, WiFi-SLAM [19] proposed the usage of the WiFi signal for SLAM. Unlike SLAM, VeLoc assumes the availability of a map and the problem to be addressed is mobile robot localization. Mobile robot

localization, which is often called position estimation or position tracking, is the problem of determining the pose of a robot relative to a given map of the environment [20].

The early work on mobile robot localization problem used Kalman filters, which are 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 [21,22]. 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 [20].

However, robot localization typically depends on using explicit environment sensors, such as laser range finders and cameras. Moreover, the rotation of the robot wheels offers a precise computation of displacement. Unlike robot localization, VeLoc uses smartphone sensors to compute the displacement and direction of vehicles, and landmarks are essentially virtual landmarks, as described next.

2.3. Virtual Landmarks

In the mobile robot localization problem, robots equipped with sensors such as laser-based ranging and cameras are assumed to be exploring the space of interest. The space is assumed to have landmarks, which are typically artificially inserted (e.g., barcodes pasted on walls or a particular pattern painted on the ceiling). However, in the area of indoor localization, smartphones, not robots, are carried around in the space of interest. We cannot depend on additional sensors since these are either not present in typical consumer devices (e.g., laser ranging) or even if present are not amenable to use (e.g., smartphone cameras). Thus we need resort to virtual landmarks which are essentially ambient signatures or recognized activities.

Ambient signatures and activity recognition have been utilized [6,23], and UnLoc [11] is believed to be the first to apply virtual landmarks for dead reckoning. P^2 [5] uses mobile sensor measurements to detect road anomalies for the purpose of road surface monitoring. VeTrack [24] and Jigsaw [25] identify visual landmarks and construct maps of shopping malls with parking structures. 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 static vehicle and a moving one can be viewed as a measurement of the velocity of the vehicle.

3. System Architecture

In this section, we describe the software architecture of VeLoc and Figure 1 depicts a pictorial overview of the architecture. There are two key components in VeLoc: the pattern detector (PD) and the augmented particle filter (APF). The PD uses pattern recognition techniques to analyze the inertial sensor measurements. The result of the PD consists of detected landmarks whether the vehicle is moving or not. The output of the PD together with the map and inertial sensor measurements is used at the measurement update step of the APF. Note that raw accelerometer measurements are replaced with linear acceleration, which excludes the effect of the Earth's gravity. The APF performs time update and measurement update recursively to estimate the state of the vehicle over time.

The Pattern Detector: The PD uses the accelerometer and gyroscope data to perform two key functions. The first function reliably determines whether the vehicle is moving. The fundamental observation behind this is that if the vehicle is moving there exists a high frequency vibration in the output of accelerometer regardless of the way the vehicle moves. Second, the PD views measurements as signals over time and analyzes subsection of those signals for a specified time duration to determine whether a landmark is encountered. Landmarks used in VeLoc includes turnings and road anomalies.

The Augmented Particle Filter: The key function of the APF is to track the probability distribution of a vehicle's state as it moves. Since sensors on a smartphone only provide measurements for acceleration but not velocity, a natural computation method is to double-integrate acceleration as $\int \int a(t)dtdt$, where the first integration computes speed and the second computes displacement. Unfortunately, several factors cause fluctuations in acceleration, resulting in erroneous displacements. One solution to address this problem is to view velocity as one dimension of the state and simultaneously estimate velocity with the location and heading direction. VeLoc maintains a four-dimensional joint probability distribution function in the form of a particle filter and estimates all these values as the vehicle moves on the floor.

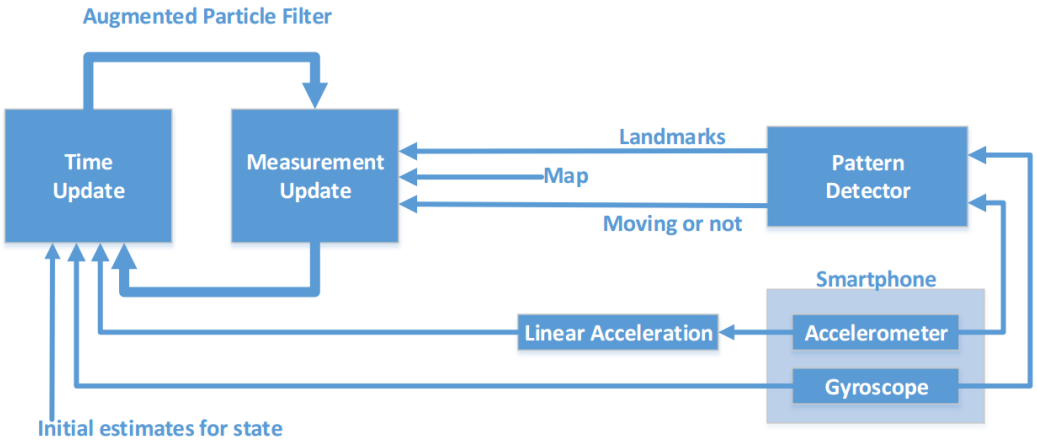
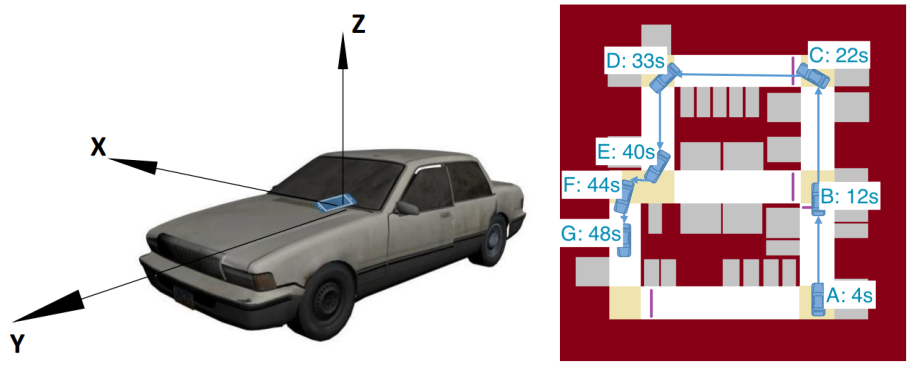


Figure 1. Indoor vehicle localization architecture.

4. Design Details

This section describes the algorithms and engineering details underlying VeLoc.

To use VeLoc, users need to put their smartphones on the platform at the front of the vehicle. Also, the smartphone needs to be facing towards the front. Thus, we can define the coordinate system of the vehicle in the same way as the smartphone. Figure 2a shows how the coordinate system of the smartphone and the coordinate system of the vehicle are related.



(**a**) (**b**)

Figure 2. The vehicle coordinate system and the map of an example vehicle route. Seven key points are demonstrated with time stamps. The vehicle starts at A and stops at G. The timer starts before the vehicle starts and its time is recorded at every key point. (**a**) Vehicle coordinate system; (**b**) Map of the example scenario shown with the route.

4.1. Pattern Detection

The intuition behind our algorithm is that both anomalous road conditions and whether the vehicle is moving are reflected in features of the acceleration data. For the sake of clarity, we will illustrate this part with an example scenario. Figure 2b shows the map of an underground parking lot with the route drawn on it. Seven key points, i.e., A to G, are demonstrated with time stamps. Figure 3 shows the sensor data while the vehicle is moving along the route.

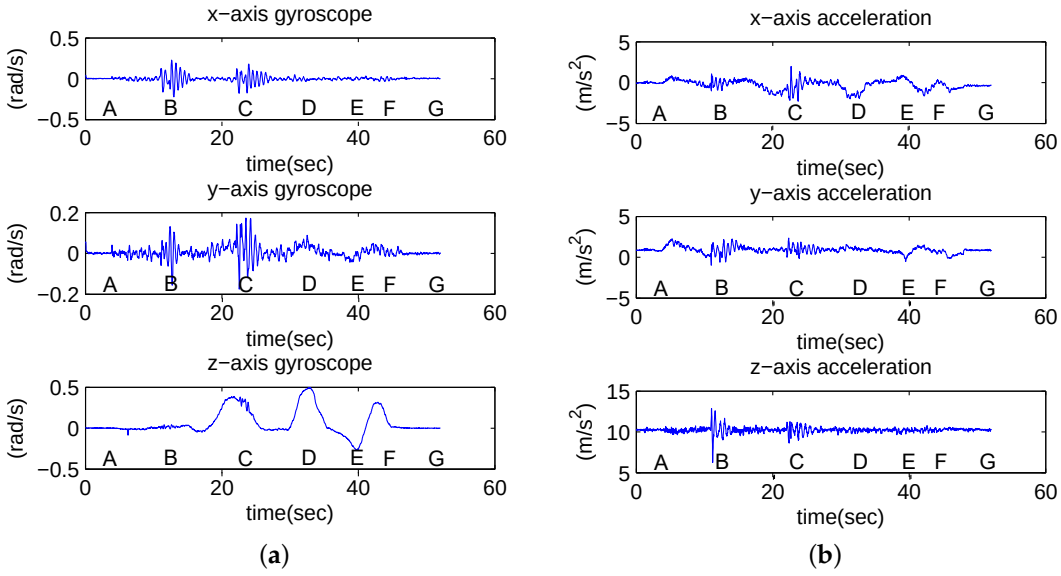


Figure 3. Sensor data recorded during the example drive. (**a**) Gyroscope; (**b**) Acceleration.

We will discuss the features used in different tasks first and then use a machine learning method to optimize all the parameters in our model.

4.1.1. Moving Detection

We can distinguish the static state from a state of movement with constant speed, in principle, if the floor is absolutely flat. However, in practice, the floor of the parking lot cannot be that flat at all, which causes vibration in sensor data while the car is moving. Based on this intuition, VeLoc calculates the following feature of a signal $s(t)$:

$$E(s,t) = \int_{t-\delta}^{t} s(x)^2 dx \tag{1}$$

and the discrete version (the first item) with standardization (the second item):

$$E(s,t) = \sum_{i=t-k+1}^{t} s(i)^2 - \frac{1}{k}\{\sum_{i=t-k+1}^{t} s(i)\}^2 \tag{2}$$

where both δ and k describe the window size. In our implementation, $k = 100$, corresponding to $\delta = 2s$ since sensor data is collected every $0.02s$ in VeLoc (the default sampling rate for inertial sensors in iOS). The feature used in moving detection is defined as:

$$F_1(t) := w_{11}E(acc_x,t) + w_{12}E(acc_y,t) + w_{13}E(acc_z,t) \tag{3}$$

where w_{11}, w_{12}, w_{13} are weights summing up to 1.

The vehicle is regarded as static at time t if and only if the following inequality is satisfied:

$$F_1(t) < thres_1 \tag{4}$$

where $thres_1$ is a threshold.

During the example drive, the feature F_1 calculated for moving detection is shown in Figure 4a. Note that only two key points, A(4s) and G(48s), show relatively small values.

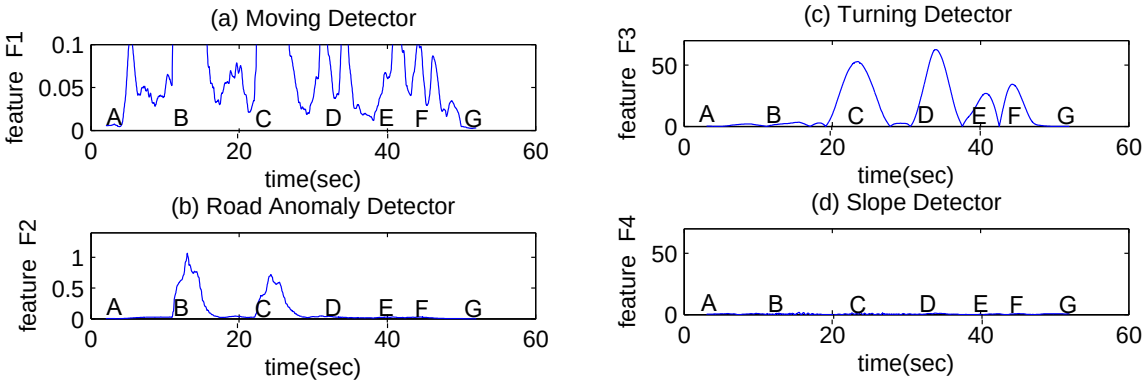


Figure 4. Features calculated for different detectors.

4.1.2. Road Anomaly Detection

When a vehicle encounters speed bumps, potholes, or other rough road conditions, there are high-energy events in both acceleration signals and gyroscope signals. Acceleration along the z-axis best characterizes this anomaly since vibrating up and down changes the acceleration along this direction. Gyroscope signals along the x- and y-axes can also be used to detect road anomalies because these signals are always close to zero unless there is vibration caused by a road anomaly.

We use the definition of feature E in Section 4.1.1 to define the feature for road anomaly detection as follows:

$$F_2(t) := w_{21}E(acc_z, t) + w_{22}\frac{E(gyr_x, t)}{b} + w_{23}\frac{E(gyr_y, t)}{b} \tag{5}$$

where w_{11}, w_{12}, w_{13} are weights summing up to 1. b is used to balance the difference in scale between two kinds of data. In VeLoc, we set $b = (0.1)^2 = 0.01$.

The road anomaly is detected at time t if and only if the following inequality is satisfied:

$$F_2(t) > thres_2 \tag{6}$$

where $thres_2$ is a threshold.

During the example drive, the feature F_2 calculated for road anomaly detection is shown in Figure 4b. Note there are two responses corresponding to two speed bumps shown in the Figure 3, and the activities occur after key points B (12s) and C (22s). The magnitude of the response also depends on the velocity of the vehicle. At high speeds, even small road anomalies can create high peak acceleration readings. This can be used to explain why the second speed bump causes a lower peak in the F_2 feature since the vehicle turns earlier and the velocity is relatively low.

4.1.3. Turning Detection and Slope Detection

Both turnings and slopes can be detected by gyroscope signals since rotational velocities are measured. Turnings result in rotational velocity along the z-axis, while slopes result in rotational velocity along the x-axis. Based on this intuition, VeLoc calculates the following feature of a signal $s(t)$:

$$C(s,t) = |\int_{t-\delta}^{t} s(x)dx| \tag{7}$$

with discrete version:

$$C(s,t) = |\sum_{i=t-k+1}^{t} s(i)| \tag{8}$$

where both δ and k describe the window size. In our implementation, $k = 150$, corresponding to $\delta = 3s$ since it may take longer time to travel through a turn.

A turn is detected at time t if and only if the following inequality is satisfied:

$$F_3(t) := C(gyro_z, t) > thres_3 \tag{9}$$

where $thres_3$ is a threshold.

A slope is detected at time t if and only if the following inequality is satisfied:

$$F_4(t) := C(gyro_x, t) > thres_4 \tag{10}$$

where $thres_1$ is a threshold.

During the example drive, the feature F_3 calculated for turn detection is shown in Figure 4c while feature F_4 calculated for slope detection is shown in Figure 4d. Note there are four responses corresponding to four turnings shown in the Figure 2. The vehicle turns around key points C (22s), D (33s), E (40s), F (44s). Since there is no slope, feature F_4 always has a very small response.

The magnitude of the response depends on the degree of the angle that the vehicle turns. For instance, the vehicle does not perform a full 90-degree turn at the two consecutive turnings at E, F, which correspond to two lower peaks.

4.1.4. Training The Detectors

At time t, VeLoc learns to assign a label to indicate the state of the vehicle. The classifier shown in Figure 5 is composed of pattern detectors described above. The labels can take the five different values shown in Table 1.

The detectors described above are controlled through tuning parameters **thres** = $[thres_1, thres_2, thres_3, thres_4]$ and $\mathbf{w} = [w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, w_{23}]$, which control the thresholds and weights for predicting labels.

Table 1. Meanings and conditions of labels.

Labels	Meanings	Conditions
IM	Static	Moving detector returns 0
RA	Road anomaly	Anomaly detector returns 1
TU	Turnings	Turning detector returns 1
SL	Slopes	Slope detector returns 1
MN	Moving normally	Else

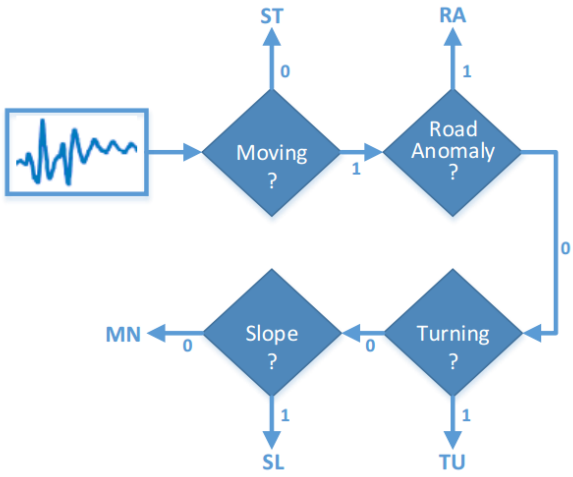


Figure 5. Classifier composed of four pattern detectors.

For each set of parameter values **thres** and **w**, we compute loss function *loss*(**thres**,**w**) as follows:

$$loss(\mathbf{thres}, \mathbf{w}) = \sum_{t \in \mathrm{T}} \frac{\delta_{\mathbf{thres},\mathbf{w}}(t)}{|\mathrm{T}|} \tag{11}$$

where

$$\delta_{\mathbf{thres},\mathbf{w}}(t) = \begin{cases} 0, & \text{predicted value equals labeled value;} \\ 1, & \text{else.} \end{cases} \tag{12}$$

The final parameter set is chosen to minimize the loss function:

$$\begin{aligned} \underset{\mathbf{thres},\mathbf{w}}{\text{minimize}} \quad & loss(\mathbf{thres}, \mathbf{w}) \\ \text{subject to} \quad & w_{11} + w_{12} + w_{13} = 1, \\ & w_{21} + w_{22} + w_{23} = 1. \end{aligned} \tag{13}$$

The values of these parameters are computed using a combined exhaustive search method. Specifically, we manually set the reasonable searching range and step for each parameter after careful observations of the corresponding sensory data.

4.2. Augmented Particle Filter

In this section we describe how VeLoc uses the augmented particle filter (APF) to track the vehicles' paths as they move in a parking lot. We will first introduce the APF algorithm and then describe the time update and the measurement update, which are the two essential steps in all Bayes filter-based algorithms.

The particle filter is an alternative nonparametric implementation of the Bayes filter. The key idea of the particle filter is to represent a distribution by a set of samples drawn from this distribution. In particle filters, the samples of a posterior distribution are called *particles* and are denoted as:

$$\mathcal{X}_t := \boldsymbol{x}_t^{[1]}, \boldsymbol{x}_t^{[2]}, ..., \boldsymbol{x}_t^{[M]}$$

Each particle $\boldsymbol{x}_t^{[m]}$(with $1 \leq m \leq M$) is a concrete instantiation of the state at time t, that is, a hypothesis as to what the true world state may be at time t. Here M denotes the number of particles in the particle set $\mathcal{X}_t$.

Augmented particles are novel particles that also incorporate the uncertainty in other aspects such as the direction of the smartphone and the velocity of the vehicle.

Just like all other Bayes filter algorithms, the APF algorithm constructs the distribution at time t recursively from the distribution one time step earlier. Since beliefs are represented by sets of particles, this means that the APF constructs the particle set $\mathcal{X}_t$ recursively from the set $\mathcal{X}_{t-1}$.

The following is the pseudocode for APF algorithm (Algorithm 1) in VeLoc, where $\boldsymbol{u}_t$ and $\boldsymbol{z}_t$ are control and measurement, respectively, at time t, and $\sim$ denotes that a variable is subject to a certain probability distribution.

Algorithm 1 *Augmented_Particle_Filter*$(\mathcal{X}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t)$.

1: $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
2: **for** $m = 1$ to M **do**
3: sample $\boldsymbol{x}_t^{[m]} \sim p(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}^{[m]})$
4: $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t \cup \{\boldsymbol{x}_t^{[m]}\}$
5: **end for**
6: **for** $m = 1$ to M **do**
7: $w_t^{[m]} = p(\boldsymbol{z}_t | \boldsymbol{x}_t^{[m]})$
8: **end for**
9: **for** $m = 1$ to M **do**
10: draw $\boldsymbol{x}_t^{[i]}$ from $\overline{\mathcal{X}}_t$ with probability $\propto w_t^{[i]}$
11: $\mathcal{X}_t = \mathcal{X}_t \cup \{\boldsymbol{x}_t^{[i]}\}$
12: **end for**
13: **return** $\mathcal{X}_t$

Time update, which is also known as *control update*, is shown in Lines 2 through 5, while *measurement update* is shown in Lines 6 through 12.

During the time update, the APF generates a hypothetical state $\boldsymbol{x}_t^{[m]}$ for time t based on the particle $\boldsymbol{x}_{t-1}^{[m]}$ and the control $\boldsymbol{u}_t$. The resulting sample is indexed by m, indicating that it is generated from the m-th particle in $\mathcal{X}_{t-1}$. This step involves sampling from the transition distribution $p(\boldsymbol{x}_t | \boldsymbol{u}_t, \boldsymbol{x}_{t-1}^{[m]})$ which is not always possible for arbitrary distributions.

During the measurement update, the APF first calculates for each particle $\boldsymbol{x}_t^{[m]}$ the so-called importance weights, denoted $w_t^{[m]}$. Importance weights are used to incorporate the measurement $\boldsymbol{z}_t$ into the particle set. The importance weight, thus, is the probability of the measurement $\boldsymbol{z}_t$ under the particle $\boldsymbol{x}_t^{[m]}$, that is, $w_t^{[m]} \propto p(\boldsymbol{z}_t | \boldsymbol{x}_t^{[m]})$. The real "trick" of the particle filter algorithm occurs in Lines 9 through 12, which implement what is known as *resampling* or *importance resampling*. The algorithm draws with M replacement particles from the temporary set $\overline{\mathcal{X}}_t$. The probability of drawing each particle is given by its importance weight. By incorporating the importance weights into the resampling process, the distribution of the particles changes over time.

In the rest of this section, we describe the transition distribution, importance weights, and resampling method in VeLoc.

4.2.1. Transition Distribution

To define the transition distribution in VeLoc, we need first introduce the state of a particle and the control. The state of a particle is a four-dimensional vector defined as follows:

$$\langle x, y, \theta, v \rangle$$

where x, y are position of the vehicle, θ is the heading direction of the vehicle, and v is the velocity along the y-axis of the vehicle shown in Figure 6. The control is defined as follows:

$$\langle w, a \rangle$$

where w is the rotational velocity along the z-axis of the vehicle and a is the acceleration along the y-axis of the vehicle shown in Figure 6.

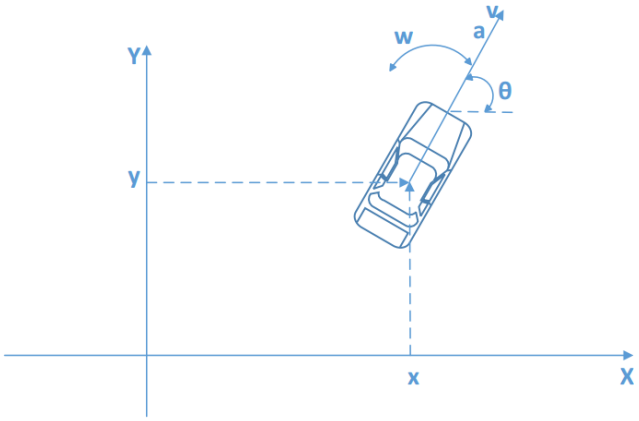


Figure 6. State and control.

We use the four-dimensional Gaussian distribution

$$\mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) = \frac{1}{4\pi^2}\frac{1}{|\boldsymbol{\Sigma}_t|^{\frac{1}{2}}}e^{-\frac{1}{2}(\boldsymbol{x}_t-\boldsymbol{\mu}_t)^T\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{x}_t-\boldsymbol{\mu}_t)} \tag{14}$$

to represent the transition distribution in VeLoc.

$\boldsymbol{\mu}_t$ is defined as follows:

$$\boldsymbol{\mu_t} = \boldsymbol{G}(\boldsymbol{x}_{t-1}) + \boldsymbol{H}(\boldsymbol{z}_t) \tag{15}$$

where

$$\boldsymbol{G}(\boldsymbol{x}_{t-1}) = \boldsymbol{G}\left(\begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \\ v_{t-1} \end{pmatrix}\right) = \begin{pmatrix} x_{t-1} + v_{t-1}\Delta t \cdot \cos\theta_{t-1} \\ y_{t-1} + v_{t-1}\Delta t \cdot \sin\theta_{t-1} \\ \theta_{t-1} \\ v_{t-1} \end{pmatrix} \tag{16}$$

and

$$\boldsymbol{H}(\boldsymbol{z}_t) = \boldsymbol{H}\left(\begin{pmatrix} w_t \\ a_t \end{pmatrix}\right) = \begin{pmatrix} 0 \\ 0 \\ w_t\Delta t \\ a_t\Delta t \end{pmatrix} \tag{17}$$

$\boldsymbol{\Sigma}_t$ is defined as follows:

$$\boldsymbol{\Sigma}_t = \begin{pmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & 0 & \sigma_v^2 \end{pmatrix} \tag{18}$$

where all σ^2 describe measurement noise.

4.2.2. Importance Weights

We define the measurement z_t in VeLoc as follows:

$$\langle road\ anomaly, turning, slope, moving, reachability \rangle$$

Every element can take 0 or 1 so that $z_t \in \{0,1\}^5$. The first three elements are outputs of PD, indicating whether a landmark is encountered. The element *moving* is also one output of PD, indicating whether the vehicle is moving. The last element *reachability* indicates whether the vehicle can reach the current position and obviously it is always 1.

To calculate the probability $p(z_t|\boldsymbol{x}_t^{[m]})$, we assume that elements of z_t are independent so that:

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t^{[m]}) = \prod_{i=1}^{5} p(z_{ti}|\boldsymbol{x}_t^{[m]}) \tag{19}$$

As described before, $w_t^{[m]} \propto p(\boldsymbol{z}_t|\boldsymbol{x}_t^{[m]})$. We can also calculate $w_t^{[m]}$ as:

$$w_t^{[m]} := \prod_{i=1}^{5} w_{ti}^{[m]} \tag{20}$$

where for $1 \leq i \leq 5$

$$w_{ti}^{[m]} \propto p(z_{ti}|\boldsymbol{x}_t^{[m]}) \tag{21}$$

In the rest of this part we will introduce the way to calculate $w_{ti}^{[m]}$ for $1 \leq i \leq 5$.

$w_{t1}^{[m]}$ for *road anomaly*: Since all the road anomalies are noted on the map, we can calculate for every position (x, y) the closest distance to any road anomalies $dist_{RA}(x, y)$. $w_{t1}^{[m]}$ is defined as follows:

$$w_{t1}^{[m]} = \begin{cases} 1, & road\ anomaly = 0; \\ \mathcal{N}(dist_{RA}(x_{t1}^{[m]}, x_{t2}^{[m]})|0, \sigma_{RA}^2), & road\ anomaly = 1. \end{cases} \tag{22}$$

when a road anomaly is detected, a particle which is closer to a road anomaly noted on the map will have a bigger importance weight.

Similarly, $w_{t2}^{[m]}$ and $w_{t2}^{[m]}$ can be defined as follows:

$$w_{t2}^{[m]} = \begin{cases} 1, & turning = 0; \\ \mathcal{N}(dist_{TU}(x_{t1}^{[m]}, x_{t2}^{[m]})|0, \sigma_{TU}^2), & turning = 1. \end{cases} \tag{23}$$

$$w_{t3}^{[m]} = \begin{cases} 1, & slope = 0; \\ \mathcal{N}(dist_{SL}(x_{t1}^{[m]}, x_{t2}^{[m]})|0, \sigma_{SL}^2), & slope = 1. \end{cases} \tag{24}$$

$w_{t4}^{[m]}$ for *moving*: When the vehicle is detected to be static, a particle with velocity close to 0 should have a large importance weight. Thus, we define $w_{t4}^{[m]}$ as follows:

$$w_{t4}^{[m]} = \begin{cases} \mathcal{N}(x_{t4}^{[m]}|0, \sigma_{MV}^2), & moving = 0; \\ 1, & moving = 1. \end{cases} \tag{25}$$

$w_{t5}^{[m]}$ for *reachability*: To use the constraints imposed by the map, we give a 0 importance weight to those particles whose positions are noted not able to reach on the map. Assume that $R(x, y)$ is the

accessible matrix provided by the map to indicate whether the position (x, y) is able to be reached. Thus, we define $w_{t5}^{[m]}$ as follows:

$$w_{t5}^{[m]} = \begin{cases} 0, \ R(x_{t1}^{[m]}, x_{t2}^{[m]}) = 0; \\ 1, \ R(x_{t1}^{[m]}, x_{t2}^{[m]}) = 1. \end{cases} \tag{26}$$

4.2.3. Resampling Method

VeLoc uses the low variance sampler [20] to fulfill the resampling task. It is worth mentioning that since resampling is most time-consuming part of the algorithm, VeLoc does not perform resampling after every update. VeLoc maintains for each particle an importance weight that is initialized by 1 after resampling and updated multiplicatively untill next resampling. In Veloc, resampling is performed every 10 updates.

5. Results

Methodology: we implement VeLoc on iOS, and our code includes C++ for algorithms and Objective C for sensor and GUI operations. During experiments, we use smartphones to collect motion sensor readings on a vehicle in an underground parking structure, and its size is 80 m $\times$ 90 m. We conduct 20 vehicle traces in the parking structure with four iPhones with different poses to simultaneously collect inertial sensor data during the drive. We design a platform to fix four iPhones with different poses as shown in Figure 7. The platform is placed flat inside the vehicle with a forward direction. To evaluate our system's robustness, we invited three volunteers to drive their own cars in one parking structure. The cars had been purchased at around 10,000, 20,000, and 30,000 dollars, respectively.

We first show results in Figure 8, which shows the positions of all the particles as time goes by. In this case, the initial state is informed as being somewhere near the entrance of the parking lot. As shown in Figure 8a, all the particles are initialized around the entrance. Figure 8b shows that particles diverge due to the noise in measurement. Figure 8c,d show how speed bumps could make the particles closer. Figure 8e,f show how turns could make the particles closer. Figure 8g shows all the particles after another turn. Figure 8h,i show the ability to estimate the velocity of the vehicle, as particles with too high or too low velocity will cause the particles to reach a wall noted on the map. Figure 8j,k show how particles pass through two consecutive turns. Figure 8l shows the final position of all the positions and the average position is shown with a red point. Compared with the route shown in Figure 2, VeLoc obtains a very accurate result in the localization problem.

Figure 9 shows the results if the initial position is unknown but the initial heading direction is somehow known (e.g, using compass and the regularity of heading directions of a vehicle in the parking lot). As shown in Figure 9a, particles are initialized everywhere in the parking lot. While the vehicle is moving, constraints imposed by the map filter out some particles as shown in Figure 9b. Figure 9c shows how particles converge quickly when a landmark is detected. Figure 9d,e show the effect of velocity estimation. Figure 9e looks similar to Figure 8g, meaning that VeLoc succeeds in estimating the state of the vehicle without the initial position. We can also trace back to determine the initial position of the vehicle, which is not given at the beginning. Figure 9f shows the final position of all the positions and the average position is shown with a red point. Compared with the route shown in Figure 2, VeLoc gets a very accurate result in the localization problem, without an initial position.

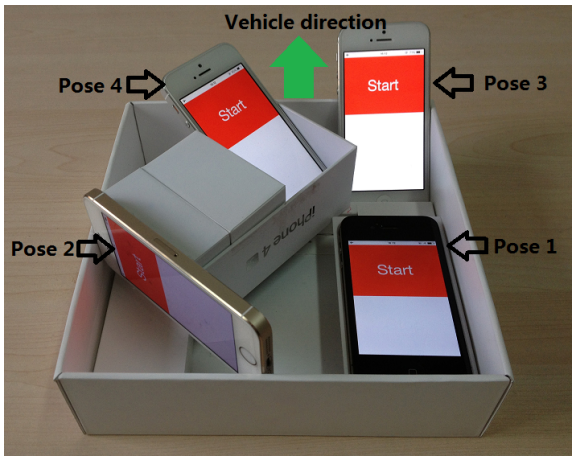


Figure 7. A platform is with four iPhones.

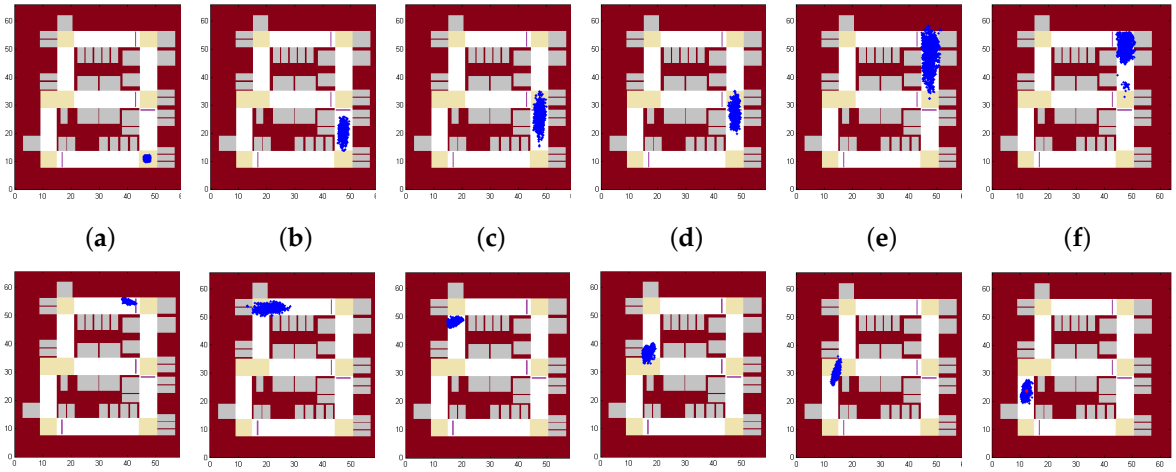


Figure 8. Particles over time.

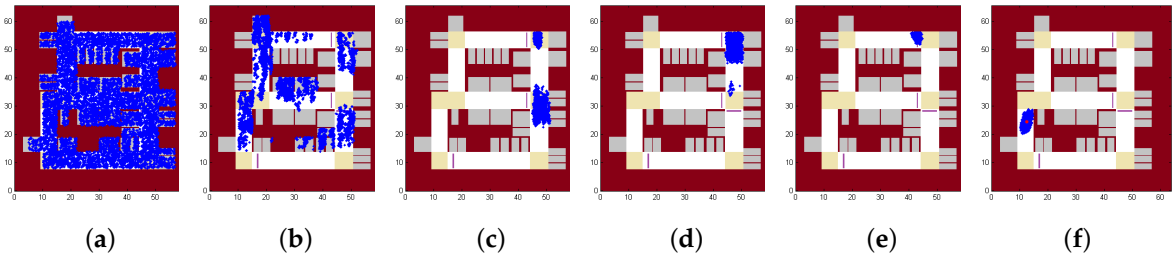
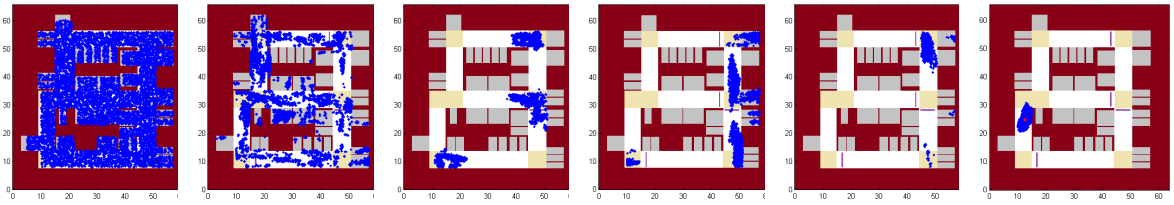


Figure 9. Particles over time without the initial position.

Figure 10 shows the results if both the initial position and initial heading direction are unknown. VeLoc is still able to estimate the true state of the vehicle but it may take a little longer to converge. As shown in Figure 10a, particles are initialized everywhere in the parking lot. Constraints imposed by the map filter out some particles but the appearance is still poor, as shown in Figure 10b. Figure 10c shows how particles converge quickly when a landmark is detected. However, compared with Figure 9c, particles converge in more places since the heading direction is unknown. Figure 10d shows that constraints imposed by map filter out some other particles. Figure 10e shows that particles converge after a second landmark detected. Figure 10f shows the final position of all the positions, and

the average position is shown with a red point. Compared with the route shown in Figure 2, VeLoc obtains a very accurate result in the localization problem when both the initial position and the initial heading direction are unknown.



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

Figure 10. Particles over time without the initial position and the initial heading direction.

Figure 11 shows the vehicle localization accuracy in different scenarios. The 90-percentile localization error is around 10 m for all four poses, and the maximum errors are about 30 m, as shown in Figure 11a. Additionally, as Figure 11b shows, different drivers with their own cars achieve a localization accuracy of around 10 m at the 90th percentile.

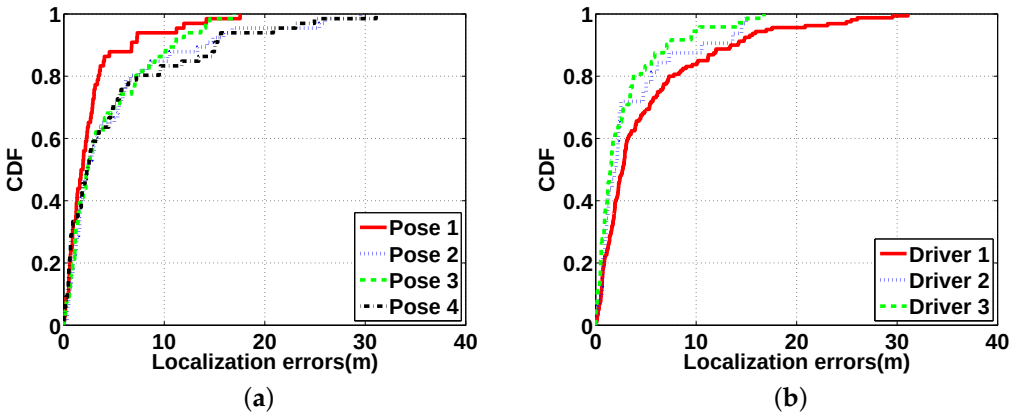


Figure 11. Vehicle localization errors in different scenarios: (**a**) four different poses; (**b**) three different cars and drivers in one parking structure.

6. Conclusions and Future Work

In this paper we describe how VeLoc 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 nor WiFi signals, which may be unavailable in environments such as underground parking lots. It does not require additional sensors to determine the environment. VeLoc first detects landmarks such as speed bumps, turns, and slopes, and combines them with the map information to estimate the vehicle's location using a probabilistic model. Experiments have shown that VeLoc can track the parking locations to within four parking spaces, which is sufficient for the driver to trigger a honk using the car key.

Currently VeLoc depends on accurate parking structure maps to reduce the uncertainty in 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 maps are available. This will further extend VeLoc's capabilities in the real world.

Author Contributions: R.G. conceived and designed the system and wrote the paper; F.H. implemented the system; T.L. performed the experiments.

Acknowledgments: This work is supported in part by the Fundamental Research Funds for the Central Universities 2016RC032 and NSFC 61702035.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bahl, P.; Padmanabhan, V. Radar: An in-building rf-based user location and tracking system. In Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel Aviv, Israel, 26–30 March 2000.
2. Chintalapudi, K.; Iyer, A.P.; Padmanabhan, V.N. Indoor localization without the pain. In Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking (MobiCom'10), Chicago, IL, USA, 20–24 September 2010.
3. Liu, H.; Gan, Y.; Yang, J.; Sidhom, S.; Wang, Y.; Chen, Y.; Ye, F. Push the limit of wifi based localization for smartphones. In Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom'12), Istanbul, Turkey, 22–26 August 2012.
4. Youssef, M.; Agrawala, A. The horus wlan location determination system. In Proceedings of the 3rd international conference on Mobile systems, applications, and services (MobiSys'05), Seattle, WA, USA, 6–8 June 2005.
5. Eriksson, J.; Girod, L.; Hull, B.; Newton, R.; Madden, S.; Balakrishnan, H. The pothole patrol: Using a mobile sensor network for road surface monitoring. In Proceedings of the The Sixth Annual International Conference on Mobile Systems, Applications and Services (MobiSys 2008), Breckenridge, CO, USA, 17–20 June 2008.
6. Azizyan, M.; Constandache, I.; Choudhury, R.R. Surroundsense: Mobile phone localization via ambience fingerprinting. In Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom '09), Beijing, China, 20–25 September 2009.
7. Constandache, I.; Bao, X.; Azizyan, M.; Choudhury, R.R. Did you see Bob?: Human localization using mobile phones. In Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking (MobiCom'10), Chicago, IL, USA, 20–24 September 2010.
8. Robertson, P.; Angermann, M.; Krach, B. Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors. In Proceedings of the 11th International Conference on Ubiquitous Computing (Ubicomp'09), Orlando, FL, USA, 30 September–3 October 2009.
9. Woodman, O.; Harle, R. Pedestrian localisation for indoor environments. In Proceedings of the 10th International Conference on Ubiquitous Computing (Ubicomp'08), Seoul, Korea, 21–24 September 2008.
10. Constandache, I.; Choudhury, R.; Rhee, I. Towards mobile phone localization without war-driving. In Proceedings of the IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010 .
11. Wang, H.; Sen, S.; Elgohary, A.; Farid, M.; Youssef, M.; Choudhury, R.R. No need to war-drive: Unsupervised indoor localization. In Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys'12), Low Wood Bay, Lake District, UK, 25–29 June 2012.
12. Levinson, J.; Montemerlo, M.; Thrun, S. Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems*; Citeseer: Atlanta, GA, USA, 2007.
13. Levinson, J.; Thrun, S. Robust vehicle localization in urban environments using probabilistic maps. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK, USA, 3–7 May 2010; pp. 4372–4378.
14. Rai, A.; Chintalapudi, K.K.; Padmanabhan, V.N.; Sen, R. Zee: Zero-effort crowdsourcing for indoor localization. In Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom'12), Istanbul, Turkey, 22–26 August 2012.
15. Antoniou, C.; Gikas, V.; Papathanasopoulou, V.; Danezis, C.; Panagopoulos, A.D.; Markou, I.; Efthymiou, D.; Yannis, G.; Perakis, H. Localization and Driving Behavior Classification with Smartphone Sensors in the Direct Absence of Global Navigation Satellite Systems. *Transp. Res. Rec. J. Transp. Res. Board* **2015**, *2489*, 66–76. [CrossRef]
16. Saeedi, S.; Moussa, A.; El-Sheimy, N. Context-aware personal navigation using embedded sensor fusion in smartphones. *Sensors* **2014**, *14*, 5742–5767. [CrossRef] [PubMed]

17. Gikas, V.; Perakis, H. Rigorous performance evaluation of smartphone GNSS/IMU sensors for ITS applications. *Sensors* **2016**, *8*, 1240. [CrossRef] [PubMed]
18. Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. Fastslam: A factored solution to the simultaneous localization and mapping problem. In Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, AB, Canada, 28 July–1 August 2002; pp. 593–598.
19. Ferris, B.; Fox, D.; Lawrence, N.D. Wifi-slam using gaussian process latent variable models. In Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI'07), Hyderabad, India, 6–12 January 2007; pp. 2480–2485.
20. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; MIT Press Cambridge: Cambridge, MA, USA, 2005; Volume 1.
21. Fox, D.; Burgard, W.; Dellaert, F.; Thrun, S. Monte carlo localization: Efficient position estimation for mobile robots. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99), Orlando, FL, USA, 18–22 July 1999; pp. 343–349.
22. Thrun, S.; Fox, D.; Burgard, W.; Dellaert, F. Robust monte carlo localization for mobile robots. *Artif. Intell.* **2001**, *128*, 99–141. [CrossRef]
23. Tarzia, S.P.; Dinda, P.A.; Dick, R.P.; Memik, G. Indoor localization without infrastructure using the acoustic background spectrum. In Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys'11), Bethesda, MD, USA, 28 June–1 July 2011.
24. Gao, R.; Zhao, M.; Ye, T.; Ye, F.; Wang, Y.; Luo, G. Smartphone-Based Real Time Vehicle Tracking in Indoor Parking Structures *IEEE Trans. Mob. Comput.* **2017**, *16*, 2023–2036.
25. Gao, R.; Zhao, M.; Ye, T.; Ye, F.; Wang, Y.; Bian, K.; Wang, T.; Li, X. Jigsaw: Indoor Floor Plan Reconstruction via Mobile Crowdsensing In Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom'14), Maui, HI, USA, 7–11 September 2014.