VeTrack: Real Time Vehicle Tracking in Uninstrumented Indoor Environments

Mingmin Zhao[1], Tao Ye[1], Ruipeng Gao[1], Fan Ye[2], Yizhou Wang[1], Guojie Luo[1]
[1]EECS School, Peking University, Beijing 100871, China
[2]ECE Department, Stony Brook University, Stony Brook, NY 11794, USA
{zhaomingmin, pkuyetao, gaoruipeng, Yizhou.Wang, gluo}@pku.edu.cn, fan.ye@stonybrook.edu

ABSTRACT

Although location awareness and turn-by-turn instructions are prevalent outdoors due to GPS, we are back into the darkness in uninstrumented indoor environments such as underground parking structures. We get confused, disoriented when driving in these mazes, and frequently forget where we parked, ending up circling back and forth upon return. In this paper, we propose VeTrack, a smartphone-only system that tracks the vehicle's location in real time using the phone's inertial sensors. It does not require any environment instrumentation or cloud backend. It uses a novel "shadow" tracing method to accurately estimate the vehicle's trajectories despite arbitrary phone/vehicle poses and frequent disturbances. We develop algorithms in a Sequential Monte Carlo framework to represent vehicle states probabilistically, and harness constraints by the garage map and detected landmarks to robustly infer the vehicle location. We also find landmark (e.g., speed bumps, turns) recognition methods reliable against noises, disturbances from bumpy rides and even hand-held movements. We implement a highly efficient prototype and conduct extensive experiments in multiple parking structures of different sizes and structures, with multiple vehicles and drivers. We find that VeTrack can estimate the vehicle's real time location with almost negligible latency, with error of $2 \sim 4$ parking spaces at 80-percentile.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; C.3 [**Special-Purpose and Application-based Systems**]: Real-time and embedded systems

Keywords

Vehicle real time tracking; indoor environments

1. INTRODUCTION

Thanks to decades of efforts in GPS systems and devices, drivers know their locations at any time outdoors. The location awareness enables drivers to make proper decisions and gives them a sense of "control." However, whenever we drive into indoor environments such as underground parking garages, or multi-level parking structures where GPS signals can hardly penetrate, we lose this location awareness. Not only do we get confused, disoriented in maze-like structures, frequently we do not even remember where we park the car, ending up circling back and forth searching for the vehicle.

Providing real time vehicle tracking capability indoors will satisfy the fundamental and constant cognitive needs of drivers to orient themselves relative to a large and unfamiliar environment. Knowing where they are generates a sense of control and induces calmness psychologically, both greatly enhancing the driving experience. In smart parking systems where free parking space information is available, real time tracking will enable turn-by-turn instructions guiding drivers to those spaces, or at least areas where more spaces are likely available. The final parking location recorded can also be used to direct the driver back upon return, avoiding any back and forth search.

However, real time vehicle tracking indoors is far from straightforward. First, mainstream indoor localization technology leverages RF signals such as WiFi [7, 45] and cellular [28], which can be sparse, intermittent or simply non-existent in many uninstrumented environments. Instrumenting the environment [4, 5] unfortunately is not always feasible: the acquisition, installation and maintenance of sensors require significant time, financial costs and human efforts; simply wiring legacy environments can be a major undertaking. The lack of radio signals also means lack of Internet connectivity: no cloud service is reachable and all sensing/computing have to happen locally.

In this paper, we propose VeTrack, a real time vehicle tracking system that utilizes inertial sensors in the smartphone to provide accurate vehicle location. It does not rely on GPS/RF signals, or any additional sensors instrumenting the environment. All sensing and computation occur in the phone and no cloud backend is needed. A driver simply starts the VeTrack application before entering a parking structure, then VeTrack will track the vehicle movements, estimate and display its location in a garage map in real time, and record the final parking location, which can be used by the driver later to find the vehicle.

Such an inertial and phone-only solution entails a series of non-trivial challenges. First, many different scenarios exist for the phone *pose* (i.e., relative orientation between its coordinate system to that of the vehicle), which is needed to transform phone movements into vehicle movements. The phone may be placed in arbitrary positions - lying flat on a

SenSys'15, November 1–4, 2015, Seoul, South Korea..
© 2015 ACM. ISBN 978-1-4503-3631-4/15/11 ...$15.00.
DOI: http://dx.doi.org/10.1145/2809695.2809726.

surface, slanted into a cup holder. The vehicle may drive on a non-horizontal, sloped surface; it may not go straight up or down the slope (e.g., slanted parking spaces). Furthermore, unpredictable human or road condition disturbances (e.g., moved together with the driver's pants' pockets, or picked up from a cupholder; speed bumps or jerky driving jolting the phone) may change the phone pose frequently. Despite all these different scenarios and disturbances, the phone's pose must be reliably and quickly estimated.

Second, due to the lack of periodic acceleration patterns like a person's walking [23, 26, 33], the traveling distance of a vehicle cannot be easily estimated. Although landmarks (e.g., speed bumps, turns) causing unique inertial data patterns can calibrate the location [39], distinguishing such patterns from other movements robustly (e.g., driver picking up and then laying down the phone), and recognizing them reliably despite different parking structures, vehicles and drivers, remain open questions.

Finally, we have to balance the conflict between tracking accuracy and latency. Delaying the location determination allows more time for computation and sensing, thus higher tracking accuracy. However, this delay inevitably increases tracking latency, which adversely impacts real time performance and user experience. How to develop efficient tracking algorithms to achieve both reasonable accuracy and acceptable latency, while using resources only on the phone, is another great challenge.

VeTrack consists of several components to deal with the above challenges to achieve accurate, real time tracking. First, we propose a novel *"shadow trajectory"* tracing method that greatly simplifies phone pose estimation and vehicle movements computation. It can handle slopes and slanted driving on slopes; it is highly robust to inevitable noises, and can quickly re-estimate the pose after each disturbance. We devise robust landmark detection algorithms that can reliably distinguish landmarks from disturbances (e.g., drivers picking up the phone) causing seemingly similar inertial patterns. Based on the vehicle movements and detected landmarks, we develop a highly robust yet efficient probabilistic framework to track a vehicle's location.

In summary, we make the following contributions:

- We develop a novel robust and efficient "shadow trajectory" tracing method. Unlike existing methods [17, 40, 46] that track the 3-axis relative angles between the phone and vehicle, it only tracks a single heading direction difference. To the best of our knowledge, it is the first that can handle slopes and slanted driving on slopes, and re-estimates a changed pose almost instantaneously.

- We design states and algorithms in a Sequential Monte Carlo framework that leverages constraints from garage maps and detected landmarks to reliably infer a vehicle's location. It uses probability distributions to represent a vehicle's states. We further propose a one-dimensional *skeleton road* model to reduce the vehicle state complexity, and a prediction-rollback mechanism to cut down tracking latency, both by one order of magnitude to enable real time tracking.

- We propose robust landmark detection algorithms to recognize commonly encountered landmarks. They can reliably distinguish true landmarks from disturbances that exhibit similar inertial data patterns.

- We implement a prototype and conduct extensive experiments with different parking structures, vehicles and drivers. We find that it can track the vehicle in real time against even disturbances such as drivers picking up the phone. It has almost negligible tracking latency, 10° pose and $2 \sim 4$ parking spaces' location errors at 80-percentile, which are sufficient for most real time driving and parked vehicle finding.

Next, we give a brief overview (Section 2), describe the shadow trajectory tracing (Section 3), Sequential Monte Carlo algorithm design and the simplified road skeleton model (Section 4), landmark detection algorithms and prediction-rollback (Section 5). We report evaluation (Section 6), review related work (Section 7). After a discussion of limitations (Section 8), we conclude the paper.

2. DESIGN OVERVIEW

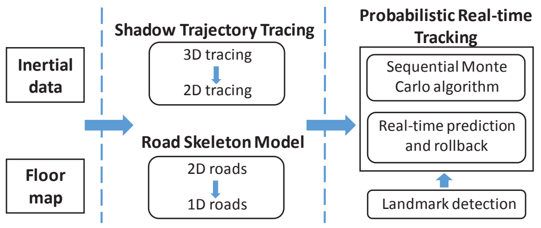

Inertial data
Floor map
Shadow Trajectory Tracing
3D tracing
2D tracing
Road Skeleton Model
2D roads
1D roads
Probabilistic Real-time Tracking
Sequential Monte Carlo algorithm
Real-time prediction and rollback
Landmark detection

Figure 1: Shadow trajectory tracing simplifies 3D vehicle tracing into 2D shadow tracing while road skeleton model further reduces 2D tracing into 1D. VeTrack represents vehicle states probabilistically and uses Sequential Monte Carlo framework for robust tracking. It uses landmark detection to calibrate vehicle states and prediction/rollback for minimum latency.

VeTrack utilizes smartphone inertial data and garage floor maps (assumed already available). The two components of shadow trajectory tracing and skeleton road model simplify the problem, while the probabilistic framework utilizes landmark detection results and prediction/rollback mechanism for robust and real time tracking (Figure 1). Shadow trajectory tracing tracks the shadow of the vehicle on 2D plane instead of the vehicle in 3D space; the road skeleton model abstracts 2D strip roads into 1D line segments to remove inconsequential details while keeping the basic shape and topology.

To deal with noises and disturbances in data, VeTrack explicitly represents the states of vehicles (e.g., locations) with probabilities and we develop algorithms in Sequential Monte Carlo framework for robust tracking. They leverage landmark detection results to help calibrate the vehicle locations to where such landmarks exist, and the prediction/rollback mechanism to generate instantaneous landmark recognition results without waiting for completely passing of landmarks.

3. TRAJECTORY TRACING

3.1 Conventional Approach

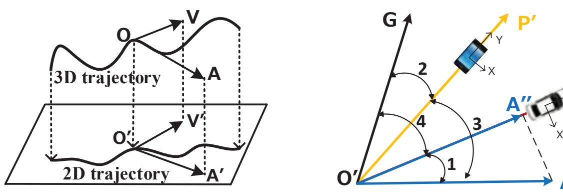
The standard approach to infer a vehicle's current location $\overrightarrow{x}(t)$ is double integration of acceleration: $\overrightarrow{x}(t) = \iint \overrightarrow{a}(t)dt$. This requires the vehicle's acceleration in the global coordinate system G be estimated, and is usually done in three steps in existing work [17, 40, 46].

Assume the 3 axes of the vehicle's coordinate system are X^V, Y^V and Z^V. First the gravity direction is obtained using mobile OS APIs [1] that use low-pass Butterworth filters to remove high frequency components caused by rotation and translation movements [47]. It is assumed to be the direction of Z^V in the phone's coordinate system (i.e., vehicles moving on level ground).

Next the gravity direction component is deducted to obtain the acceleration on the horizontal plane. The direction of maximum acceleration (caused by vehicle accelerating or decelerating) is estimated as Y^V (i.e., forward direction). Finally, X^V is determined as the cross product of Y^V and Z^V using the right-hand rule. The X^V, Y^V and Z^V directions in the phone's coordinate system give a transformation matrix that converts the phone's acceleration into that of the vehicle.

However, during investigation we find several limitations: First, when a vehicle is on a slope (straight up/down or slanted), the direction of gravity is no longer the Z-axis of the vehicle. Second, accelerometers are highly noisy and susceptible to various disturbances from driving dynamics and road conditions. Thus the direction of the maximum horizontal acceleration may not always be the Y-axis. In experiments we find that it has around 40^o errors at 80-percentile (Section 6.2). Finally, to reliably detect the direction of maximum horizontal acceleration, a changed phone pose must remain the same at least 4s [46], which may be impossible when frequent disturbances exist.

3.2 Shadow Trajectory Tracing



(a) (b)

Figure 2: (a) Intuition: points O and O' are the positions of the vehicle and its shadow. $\overrightarrow{OV}$ and $\overrightarrow{OA}$ are the velocity and acceleration of vehicle in the 3D space. V' and A' are the projection of V and A onto the 2D ground. (b) Illustration of the method to estimate $\angle 1$ from $\angle 2$, $\angle 3$ and $\angle 4$.

To overcome the above limitations, we propose a "shadow" trajectory tracing method that traces the movement of the vehicle's *shadow* projected onto the 2D horizontal plane (Figure 2(a)). Points O and O' represent the positions of the vehicle and its shadow. $\overrightarrow{OV}$ and $\overrightarrow{OA}$ are the velocity and acceleration of the vehicle in 3D space. V' and A' are the projection of V and A onto the 2D ground. It can be shown easily that $\overrightarrow{O'V'}$ and $\overrightarrow{O'A'}$ are the velocity and acceleration of the shadow. This is simply because the projection eliminates the vertical direction component but preserves those on the horizontal plane, thus the shadow and vehicle have the same horizontal acceleration, and thus the same 2D plane velocity and coordinates.

We use another method to trace the shadow location. Instead of direct double integrating on the original acceleration vector ($\overrightarrow{\boldsymbol{x}}(t) = \iint \overrightarrow{\boldsymbol{a}}(t)dt$), we use the moving direction of the shadow (unit length vector $\overrightarrow{\boldsymbol{T}}(t)$) and its speed amplitude $s(t)$: $\overrightarrow{\boldsymbol{x}}(t) = \int \overrightarrow{\boldsymbol{T}}(t) \cdot s(t)dt$, where $s(t)$ can be computed as $\int a(t)dt$, integration of the acceleration amplitude along moving direction. Although there are still two integrations, the impact of vertical direction noises is eliminated due to the projection, and the moving direction $\overrightarrow{\boldsymbol{T}}(t)$ can be measured reliably by gyroscope and maps (i.e., forward/backward along pathways only). Thus our method achieves better accuracy and robustness than the conventional approach.

We need to estimate three variables in this method (Figure 2(b)): 1) the shadow's moving direction $\overrightarrow{O'V'}$ (i.e., $\overrightarrow{\boldsymbol{T}}(t)$) in the global coordinate system. 2) the horizontal (i.e., shadow's) acceleration $\overrightarrow{O'A'}$. 3) angle $\angle V'O'A'$ ($\angle 1$), the angle between the horizontal acceleration vector and vehicle's shadow's heading (i.e., moving) direction; this is used to project the shadow's acceleration along the vehicle moving direction $\overrightarrow{O'V'}$ to get tangential acceleration amplitude $|\overrightarrow{O'A''}|$(i.e., $s(t)$).

Next we explain how to estimate them in three steps.

1) When the vehicle is driving straight, the shadow's moving direction is approximated by the direction of the road, which can be obtained from the garage map and the current location estimation. When the vehicle is turning around a corner, VeTrack accumulates the gyroscope's "yaw" (around gravity direction) to modify the heading direction until the vehicle goes straight again. We develop robust algorithms to distinguish straight driving from turning and disturbances (Section 5).

2) From existing mobile OS APIs [1], the gravity direction can be detected. We deduct the gravity direction component from the phone's acceleration vector to obtain the horizontal acceleration vector $\overrightarrow{O'A'}$.

3) Figure 2(b) illustrates how to calculate $\angle 1$ ($\angle V'O'A'$): $\angle 1 = \angle 2 + \angle 3 - \angle 4$ (i.e.,$\angle V'O'A' = \angle GO'P' + \angle P'O'A' - \angle GO'V'$). $\overrightarrow{O'G}, \overrightarrow{O'P'}, \overrightarrow{O'V'}$ are the Y-axes of the global, phone's shadow's and vehicle's shadow's coordinate system. 3.1) $\angle 2$ is the phone's shadow's heading direction in the global coordinate system. Its relative changes can be obtained reliably from the gyroscope's "yaw", and we use a distribution around the compass' reading upon entering the garage to initialize it. Because the Sequential Monte Carlo framework can calibrate and quickly reduce the error (Section 4), an accurate initial direction is not necessary. 3.2) $\angle 3$ is essentially the horizontal acceleration direction in the phone's shadow's coordinate system, which is already obtained in step 2). 3.3) $\angle 4$ is the vehicle's shadow's moving direction in the global coordinate system, already obtained in step 1).

Shadow trajectory tracing and 3D tracing are theoretically equivalent when there are no noises. However, shadow tracing needs much less variables and is subject to less noises. 1) Shadow tracing does not need to track variables in the vertical dimension (e.g., altitude, angle, speed and acceleration). All of them are subject to noises and require more complexity to estimate. 2) On the horizontal plane, the moving direction can be estimated accurately based on the prior knowledge of road directions (Section 4.4). The distance is computed using the acceleration amplitude along the moving direction. Thus inertial noises perpendicular to the moving direction do not impact the distance estimation. 3) Shadow tracing uses gyroscopes to estimate pose, while conventional 3D tracing uses accelerometers that are more susceptible to external disturbances. Therefore, shadow tracing is much less complex, subject to less noises, and thus achieves better accuracy and higher robustness.

During experiments, we find that: our shadow tracing method can handle arbitrary phone and vehicle poses and the vehicle can go straight up/down or slanted on a slope. It has much smaller errors ($5 \sim 10^\circ$ at 80-percentile) and better robustness. It also re-estimates a changed phone pose almost instantaneously because gyroscopes have little latency; thus it can handle frequent disturbances.

4. REAL TIME TRACKING

4.1 Intuition

The basic idea to locate the vehicle is to leverage two types of constraints imposed by the map, namely paths and landmarks. Given a trajectory estimated from inertial data (Figure 3), there are only a few paths on the map that can accommodate the trajectory. Each detected landmark (e.g., a speed bump or turn) can pinpoint the vehicle to a few possible locations. Jointly considering the two constraints can further reduce the uncertainty and limit the possible placement of the trajectory, thus revealing the vehicle location. We will first describe the tracking design here, then landmark detection in Section 5.

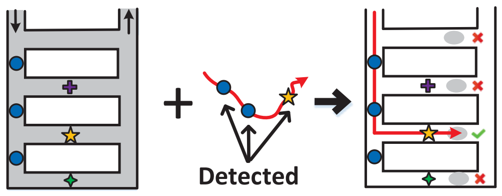


Figure 3: Using both map constraints and detected landmarks can narrow down the possible placement of the trajectory more quickly.

To achieve robust and real time tracking, we need to address a dual challenge. First, the inertial data have significant noises and disturbances. Smartphones do not possess speedometer or odometer to directly measure the velocity or distance; they are obtained from acceleration integration, which is known to generate cubic error accumulation [39]. External disturbances (e.g., hand-held movements or road conditions) causing sudden and drastic changes may not be completely separated from vehicle movements. Together they make it impossible to obtain accurate trajectories from inertial data only. Second, the requirement of low latency tracking demands efficient algorithms that can run on resource-limited phones. We have to minimize computational complexity so no cloud backend is needed.

To achieve robustness, we use probability distributions to explicitly represent vehicle states (e.g., location and speed) and the Sequential Monte Carlo (SMC) method to maintain the states. This is inspired by probabilistic robotics [35]: instead of a single "best guess", the probability distributions cover the whole space of possible hypotheses about vehicle locations, and use evidences from sensing data to validate the likelihoods of these hypotheses. This results in much better robustness to noises in data. To achieve efficiency, we use a 1D *"skeleton road"* model that abstracts paths into one dimensional line segments. We find this dramatically reduces the size of vehicle states. Thus the number of hypotheses is cut by almost one order of magnitude, which is critical to achieve real time tracking on resource limited phones. Next we will describe the road skeleton model and the detailed SMC design.

4.2 Road Skeleton Model

The road skeleton model greatly simplifies the representation of garage maps. It abstracts away inconsequential details and keeps only the essential aspects important to tracking. Thus it helps reduce computational overheads in the probabilistic framework. We assume that garage maps are available (e.g., from operators), while how to construct them is beyond the scope of this paper.

Given a map of 3D multi-level parking structure, we represent each level by projecting its map onto a 2D horizontal plane perpendicular to the gravity direction. Thus the vehicle location can be represented by a number indicating the current level, and a 2D coordinate for its location on this level. To accommodate changes when a vehicle moves across adjacent levels, we introduce "virtual boundaries" in the middle of the ramp connecting two levels. As shown in Fig.4(b), a vehicle crossing the dash line of the virtual boundary between levels will be assigned a different level number. This kind of 2D representation suits the needs for shadow tracing while retaining the essential topology and shape for tracking.

Note that we call it 2D representation because the floor level remains unchanged and does not need detection most of the time. It is updated only when the vehicle crosses virtual boundaries between levels. Its estimation is also much simpler and easier than accurate 2D tracking, where most challenges exist.

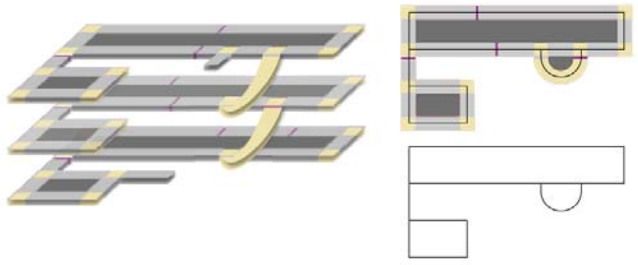


Figure 4: (a) shows the 3D floor plans of a multi-level parking structure. A vehicle enters the entrance on the floor B1, goes down to other two levels crossing the virtual boundaries. (b) shows the 2D projection of Floor B2 in (a). (c) shows the 1D road skeleton model of (b). Points on (c) represent landmarks, corresponding to bumps and corners in (a) and (b).

The key insight for the skeleton model is that the road width is not necessary for tracking vehicle locations. Since the paths are usually narrow enough for only one vehicle in each direction, the vehicle location has little freedom in the width direction. Thus we simplify the road representation with their medial axis, and roads become 1D line segments without any width (Fig.4(c)).

Compared to a straightforward 2D strip representation of roads, the skeleton model reduces the freedom of vehicle location by one dimension, thus greatly cutting down the state space size in the probabilistic framework and resulting in one order of magnitude less complexity.

4.3 Probabilistic Tracking Framework

The tracking problem is formulated as a Sequential Monte Carlo (SMC) problem, specifically, the particle filtering framework [21]. The vehicle states (e.g., location, speed) at time t are represented by a multi-dimensional random variable $s(t)$. Each hypothesis (with concrete values for each dimension of $s(t)$) is called a "particle" and a collection of J particles $\{s_t^{(j)}\}_{j=1}^J$ are used to represent the distribution of possible vehicle states at time t.

The framework operates on discrete time $\{1, ..., t-1, t, ...\}$ and repeat three steps for each time slot. Without loss of generality, we assume J particles $\{s_{t-1}^{(j)}\}_{j=1}^J$ already exist at $t-1$ and describe the progress from $t-1$ to t.

State update predicts the set of states $\{\hat{s}_t^{(j)}\}_{j=1}^J$ at time t based on two known inputs, the previous state $\{s_{t-1}^{(j)}\}_{j=1}^J$ and most recent movement m_t such as the speed, acceleration that govern the movement of the vehicle. For example, given the previous location and most recent speed, one can predict a vehicle's next location. To capture uncertainties in movement and previous states, a random noise is added to the estimated location. Thus J predictions $\{\hat{s}_t^{(j)}\}_{j=1}^J$ are generated.

Weight update uses measurements z_t made at time t to examine how much evidence exists for each prediction, so as to adjust the weights of particles $\{\hat{s}_t^{(j)}\}_{j=1}^J$. The likelihood $p(z_t|s_t)$, how likely the measurement z_t would happen given state s_t, is the evidence. A prediction $\hat{s}_t^{(j)}$ with a higher likelihood $p(z_t|s_t = \hat{s}_t^{(j)})$ will receive a proportionally higher weight $w_t^{(j)} = w_{t-1}^{(j)} p(z_t|s_t = \hat{s}_t^{(j)})$. Then all weights are normalized to ensure that $\{w_t^{(j)}\}_{j=1}^J$ sum to 1.

Resampling draws J particles from the current state prediction set $\{\hat{s}_t^{(j)}\}_{j=1}^J$ with probabilities proportional to their weights $\{w_t^{(j)}\}_{j=1}^J$, thus creating the new state set $\{s_t^{(j)}\}_{j=1}^J$ to replace the old set $\{s_{t-1}^{(j)}\}_{j=1}^J$. Then the next iteration starts.

Note that the above is only a framework. The critical task is the detailed design of particle states, update, resampling procedures. Thus we cannot simply copy what has been done in related work, and have to carefully design algorithms tailored to our specific problem.

4.4 Tracking Algorithms

4.4.1 State and Initialization

Our particle state is a collection of factors that can impact the vehicle tracking. Since the number of particles grows exponentially with the dimensionality of the state, we select most related factors to reduce the complexity while still preserving tracking accuracy. Our particle states include:

- level number k,
- position on 2D floor plane $X = (x, y)$,
- speed of the vehicle v,
- α/β, phone/vehicle shadows' 2D heading directions.

The first dimension k is introduced for multi-level structures. Position of the vehicle is represented as a 2D coordinate $X = (x, y)$ for convenience. In reality, due to the 1D skeleton road model, the position actually has only one degree of freedom. This greatly reduces the number of particles needed.

Initialization of particles: We use certain prior knowledge to initialize the particles' state. The last GPS location before entering the parking structure is used to infer the entrance, thus the level number k and 2D entrance location (x, y). The vehicle speed v is assumed to start from zero. The vehicle heading direction β is approximated by the direction of the entrance path segment, and the phone heading direction α is drawn from a distribution based on the compass reading before entering the garage. As shown later (Section 6), the phone's heading direction can be calibrated to within 15°, showing strong robustness against compass errors known to be non-trivial [37].

4.4.2 State Update

For a particle with state $(k_{t-1}, x_{t-1}, y_{t-1}, v_{t-1}, \alpha_{t-1}, \beta_{t-1})$, we create a prediction $(\hat{k_t}, \hat{x_t}, \hat{y_t}, \hat{v_t}, \hat{\alpha_t}, \hat{\beta_t})$ given movement $m_t = (a_x, a_y, \omega_z)$ where a_x, a_y and ω_z are X, Y-axis accelerations and Z-axis angular speed in the coordinate system of the phone's shadow.

First, $(\hat{x_t}, \hat{y_t})$ is updated as follow:

$$\hat{x_t} = x_{t-1} + v_{t-1}\Delta t \cdot \cos\beta_{t-1} + \epsilon_x, \tag{1}$$

$$\hat{y_t} = y_{t-1} + v_{t-1}\Delta t \cdot \sin\beta_{t-1} + \epsilon_y, \tag{2}$$

where ϵ_x, ϵ_y are Gaussian noises. If $(\hat{x_t}, \hat{y_t})$ is no longer on the skeleton, we project it back to the skeleton. Level number $\hat{k_t}$ is updated when a particle passes through a virtual boundary around the floor-connecting-ramp, otherwise $\hat{k_t} = k_{t-1}$.

Next, velocity v_t is updated as follow:

$$\hat{a_t} = a_y \cdot \cos\gamma_t - a_x \cdot \sin\gamma_t + \epsilon_a, \tag{3}$$

$$\hat{v_t} = v_{t-1} + a_t \cdot \Delta t + \epsilon_v, \tag{4}$$

where γ_t is the angle between the Y axes of the two shadows' coordinate systems and ϵ_a, ϵ_v are Gaussian noises.

Finally, α_t and β_t are updated as follows:

$$\hat{\alpha_t} = \alpha_{t-1} + \omega_z\Delta t + \epsilon_\alpha, \tag{5}$$

$$\hat{\beta_t} = \begin{cases} \beta_{t-1} + \omega_z\Delta t + \epsilon_\beta, & \text{if } turn = True; \\ \text{road direction at } (k_t, x_t, y_t), & \text{otherwise.} \end{cases} \tag{6}$$

where $\epsilon_\alpha, \epsilon_\beta$ are random Gaussian noises. The above allows the phone to change its angle α to accommodate occasional hand-held or jolting movements, while such movements will not alter the vehicle's angle β if the vehicle is known to travel straight.

4.4.3 Weight Update

Weight update uses detected landmarks and floor plan constraints to recalculate the "importance" of the current particle states. The basic idea is to penalize particles that behave inconsistently given the floor plan constraints. For example, since a vehicle cannot travel perpendicularly to path direction, a particle with velocity orthogonal to the road direction will be penalized. It will have much smaller weights and less likely to be drawn during resampling.

We compute the weight w_t as

$$w_t := w_{t-1} \prod_{i=0}^{2} w_{ti}, \tag{7}$$

Each w_{ti} is described as follows.

- **Constraints imposed by the map.** We define $w_{t0} = cos^2(\beta_t - \beta_{t-1})$. It is designed to penalize particles that have a drastic change in the vehicle heading direction, since during most of the time a vehicle does not make dramatic turns.
- **Detected landmarks.** When an i-th type landmark [1] is detected, w_{ti} of the current state is updated as $\mathcal{N}(D_i(x_t, y_t); 0, \sigma_i^2)$ where $D_i(x_t, y_t)$ is the distance to the closest landmark of the same type and σ_i^2 is a parameter controlling the scale of the distance. If no landmark is detected, $w_{ti} = 1$. This method penalizes the predicted states far away from detected landmarks.

Finally all weights are normalized so they sum up to 1.

4.4.4 *Resampling*

A replacement particle is selected from the predicted particle set $\{\hat{s}_t^{(j)}\}_{j=1}^{J}$ where each particle $\hat{s}_t^{(j)}$ has probability $w_t^{(j)}$ being selected. This is repeated J times and J particles are selected to form the new state set $\{s_t^{(j)}\}_{j=1}^{J}$. Then the next iteration starts.

5. LANDMARK DETECTION

A parking structure usually has a limited number of landmarks (e.g., speed bumps and turns), and their locations can be marked on a garage map. When a vehicle passes over a landmark, it causes distinctive inertial data patterns, which can be recognized to calibrate the vehicle's location.

However, realizing accurate and realtime landmark detection is not trivial because: 1) road conditions and hand movements impose disturbances on inertial sensor readings; and 2) to minimize delay, landmark recognition results are needed based on partial data before the vehicle completely passes a landmark. We present landmark detection algorithms robust to noises and hand movements, and a prediction and rollback mechanism for instantaneous landmark detection.

5.1 Types of Landmarks

Speed bumps generate jolts, hence acceleration fluctuations in the Z-axis when a vehicle passes over. Note that drainage trench covers, fire shutter bottom supports may also cause similar jolting patterns. We include them as "bumps" as well in the garage map.

Many factors can cause ambiguities in bump detection. For example, Figure 5 shows the acceleration signal along the Z-axis as a vehicle starts and passes over four bumps along a straight road. The first tremor in the left green box (around $10 \sim 17s$ marked with "J") is caused by the vehicle's starting acceleration. It lasts longer but with smaller magnitude compared to those caused by the bumps (in red boxes marked "$B1$"-"$B4$"). The tremor in the right green box (around 60s marked "M") is due to the user's action - holding the phone in hand, then uplifting the phone to check the location. They generate vertical acceleration that may be confused with those by bumps.

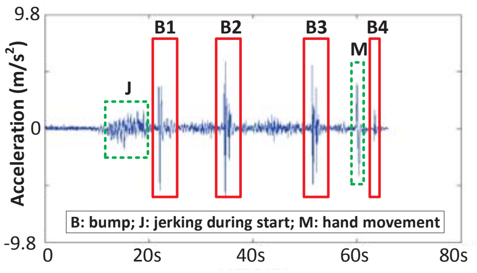
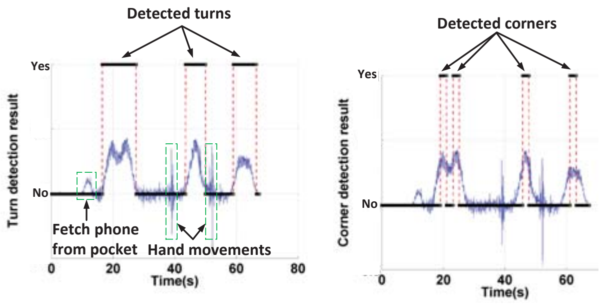


Figure 5: Acceleration along the Z-axis. There are starting acceleration (J), four bumps (B1-B4) and one hand movement (M) along the trajectory.



(a) Turn detection (b) Corner detection

Figure 6: Turn and corner detection. (a) Three turn periods are correctly detected, even there are several different hand movements. (b) 4 corners are correctly separated, even when only 3 turned are detected.

Turns are defined as durations in which a vehicle continuously changes its driving direction, usually around road corners. They can be detected from the gyroscope readings of angular velocities around the gravity direction (i.e., "yaw"). During turns a vehicle's direction differs from the road direction. Its direction changes in such periods are accumulated to track the vehicle's heading direction.

There exists work [46] using simple thresholding on turning angles to detect turns. However, we find they cannot reliably distinguish vehicle turns from hand movements (e.g., putting the phone on adjacent seat and picking it up to check the location).

Corners. A turn may span over an extended period, from its start to the end. The corner where two road segments join can be used to precisely calibrate the vehicle's location. The main challenge is consecutive turns: they might be detected as a single one, hence missing some corners. For example, in Figure 6(a), the first two turns may be detected as only one turn period.

We observe that when a vehicle passes at a corner, its angular velocity usually is at a local maxima, corresponding to the maximum turn of the steering wheel. To identify corners precisely, we use a sliding window to find local maxima of angular velocities within each turning period. Each local maxima is marked as a corner. Figure 6(b) shows that

[1]We use only bump and corner here because their locations are precise; turns are used in vehicle angle β update in Eqn 6.

the left most two consecutive corners within the same turn period are properly separated.

5.2 Feature and Classification Algorithm

We use machine learning techniques to recognize bumps and turns. Corners are detected within turns using the above local maxima searching. The critical issue is what features should be used. Although one may feed the raw signal directly to these algorithms, it is usually much more efficient to design succinct, distinctive features from raw signals.

For bumps, we divide acceleration along the Z-axis into 2-second windows sliding at 0.2s intervals. This window size is chosen empirically such that both front and rear wheels can cross the bump for complete bump-passing. For turns, we use gyroscope angular velocities around the vertical direction, and divide the signal the same way. We observe that smaller windows lead to drastic accuracy drop, while larger ones incurs more delay.

We observe that there are two kinds of common hand movements that may be confused with bumps or turns: 1) hold the phone in hand, and occasionally uplift it to check the location; 2) put the phone in pockets/nearby seat, pick up the phone to check the location and then lay it down. The first causes a jitter in Z-axis acceleration, and might be confused with bumps; the second also has Z-axis gyroscope changes, and might be confused with turns.

We have tried a number of different feature designs, both time-domain and frequency-domain, to help distinguish such ambiguities. We list five feature sets which are found to have considerable accuracy and low computation complexity (detailed performance in Section 6).

(1) **STAT35** (35 dimensions): we equally divide one window into 5 segments, and compute a 7-dimensional feature [29] from each segment, including the mean, maximum, second maximum, minimum, and second minimum absolute values, the standard deviation and the root-mean-square.

(2) **DSTAT35** (70 dimensions): In addition to STAT35, we also generate a "differential signal" (i.e., the difference between two consecutive readings) from the raw signal, and extract a similar 7-dimensional feature from each of its 5 segments.

(3) **FFT5** (5 dimensions): we do FFT on the raw signal in the whole window, and use the coefficients of the first five harmonics as a 5-dimensional feature.

(4) **S7FFT5** (35 dimensions): in addition to FFT5, we also extract the same 5 coefficients from each of two half-size windows, and four quarter-size windows. Thus we obtain 35 dimensions from 7 windows.

(5) **DFFT5** (10 dimensions): the first five FFT coefficients of both raw and differential signals.

We explore a few most common machine learning algorithms, Logistic Regression (LR) [9] and Support Vector Machine (SVM) [9]. After feature extraction, we manually label the data for training. We find that SVM has higher accuracy with slight more complexity than LR, while both can run fast enough on the phone. So we finally decide to use SVM in experiments. We find it has bump and turn detection accuracies (percentage of correctly labeled samples) around 93% (details in Section 6.2).

We have also tried some threshold-based methods on temporal [14] and frequency domain [22] features, but find it is impossible to set universally effective thresholds, and the frequency power densities by hand movements can be very similar to those of landmarks. Thus they are not sufficiently robust.

5.3 Prediction and Rollback

The reliability of landmark detection depends on the "completeness" of the signal. If the window covers the full duration of passing a landmark, more numbers of distinctive features can be extracted, and the detection would be more reliable. In reality, this may not always be possible. The landmark detection is repeated at certain intervals, but many intervals may not be precisely aligned with complete landmark-passing durations. One naive solution is to wait until the passing has completed. Thus more features can be extracted for reliable detection. However, this inevitably increases tracking latency and causes jitters in location estimation and display, adversely impacting user experience.

We use a simple prediction technique to make decisions based on data from such partial durations. To identify whether a car is passing a landmark at time t, assume that the signal spanning from $t - \tau$ to $t + \tau$ covering the full 2τ landmark-passing duration is needed for best results. At any time t, we use data in window $[t - 2\tau, t]$ to do the detection. The results are used by the real time tracking component to estimate the vehicle location. At time $t + \tau$, the data of full landmark-passing duration become available. We classify data in $[t - \tau, t + \tau]$ and verify if the prediction made at t is correct. Nothing needs to be done if it is. If it was wrong, we rollback all the states in the tracking component to t, and repeat the computation with the correct detection to re-estimate the location.

This simple technique is based on the observation that most of the time the vehicle is driving straight and landmarks are rare events. Thus the prediction remains correct most of the time (i.e., during straight driving), and mistakes/rollbacks happen only occasionally (i.e., when a landmark is encountered). From our experiments, rollbacks happen in a small fraction ($\sim 10\%$) of the time. Thus we ensure low latency most of the time because there is no waiting, while preserving detection accuracy through occasional rollback, which incurs more computation but is found to have acceptable latency ($0.1 \sim 0.3$s) (Section 6).

6. PERFORMANCE EVALUATION

6.1 Methodology

We implement VeTrack on iOS 6/7/8 so it can run on iPhone 4/4s/5/5s/6. Our code contains a combination of C, C++ for algorithms and Objective C for sensor and GUI operations. A sensor data collector sends continuous data to landmark detectors to produce detection results. Then the real time tracking component uses such output to estimate the vehicle's location, which is displayed on the screen. The initialization (e.g, loading map data) takes less than 0.5 second. Sensors are sampled at 50Hz and the particle states are evolved at the same intervals ($20ms$). Since each landmark lasts for many $20ms$-intervals, the detectors classify the landmark state once every 10 samples (i.e., every 0.2 second), which reduces computing burden.

We conduct experiments in three underground parking lots: a $250m \times 90m$ one in an office building, a $180m \times 50m$ one in a university campus and a 3-level $120m \times 80m$ one in a shopping mall (floor plans shown in Figure 7). There are 298, 79, 423 parking spots, 19, 12, 10 bumps, 10, 11, 27 turns and 4, 2, 6 slopes, respectively.

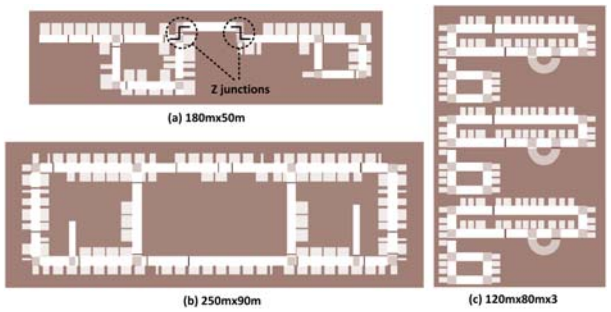


Figure 7: Floor maps of three underground parking lots: (a) university campus: $180m \times 50m$ with 79 parking spots, 12 bumps and 11 turns. (b) office building: $250m \times 90m$ with 298 parking spots, 19 bumps and 10 turns. (c) shopping mall: 3-level $120m \times 80m$ with 423 parking spots, 10 bumps and 27 turns. The chosen parking spots and entrance are marked for each lot.

For each parking lot, we collect 20 separate trajectories each starting from the entrance to one of the parking spots (shown in Figure 7) for inertial sensor data at different poses. The average driving time for trajectories is 2∼3 minutes, and the longest one 4.5 minutes. Exemplar trajectories to five test spots are illustrated in Figure 8.

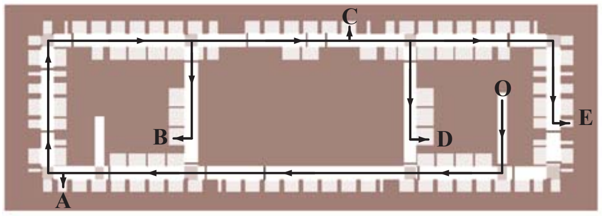
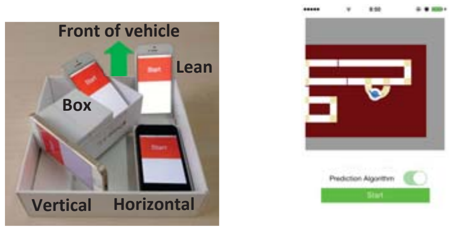


Figure 8: Driving trajectories and test spots. Each trajectory begins at the entrance O and ends at one of the test spots (A to E).

For all three lots, we use a mould to hold 4 iPhones with 4 different poses: horizontal, lean, vertical and box (Figure 9(a)). To further test the performance and robustness of our system, we use 4 more iPhones for the challenging 3-level parking lot with one in driver's right pants' pocket, one in a bag on a seat and two held in hand. The one in pocket is subject to continuous gas/brake pedal actions by the driver, while the one in bag to vehicle movements. Once in a while, one hand-held phone is picked up and put down on the user's thigh, causing Z-axis accelerations similar to those by bumps; the other is picked up from and laid down to adjacent seat, causing Z-axis angular changes similar as those by turns. These 8 poses hopefully cover all common driving scenarios. The UI of VeTrack is shown in Figure 9(b).

We use video to obtain the ground truth of vehicle location over time. During the course of driving, one person holds an iPad parallel to the vehicle's heading direction to record videos from the passenger window. After driving, we manually examine the video frame by frame to find when the vehicle passed distinctive objects (e.g., pillars) with known locations on the map. Such frames have those objects in the middle of the image, thus the error is bounded by 0.5 vehicle length and usually much better.



(a) (b)

Figure 9: Mould and VeTrack UI.

To align inertial data and video collected from different devices temporally, we first synchronize the time on all the iPhones and iPad. Then different people holding different devices will start the data collecting/recording applications at the same time. These operations establish the correspondence of data in the time series of different devices.

6.2 Evaluation of Individual Components

Landmark classification accuracy. To train landmark detectors and test their performance, we use recorded videos to find encountered landmarks and label their locations on the whole trajectory. Then we use sliding windows to generate labeled segments of sensor readings. Note that disturbances caused by hand movements are labeled as non-bump and non-turn because they should not be confused with bumps or turns. In total we generate 14739 segments for bump detector and 57962 segments for turn detector.

We evaluate classification accuracy (percentage of test samples correctly classified) of six different sets of features (described in Section 5). We randomly choose 50% of the whole dataset to train the SVM classifier and others to test the performance. We repeat it 20 times and report the average performance in Table 1. It shows that they all have high accuracy around 90%. We decide to use DFFT5 with relatively high accuracies (93.0% and 92.5% for bump and turn) and low complexity in further evaluation.

Table 1: Accuracies of different feature sets.

	dimension	bump	turn
STAT35	35	92.7%	92.8%
DSTAT35	70	92.6%	93.4%
FFT5	5	91.8%	92.2%
S7FFT5	35	92.5%	92.6%
DFFT5 (chosen)	10	93.0%	92.5%

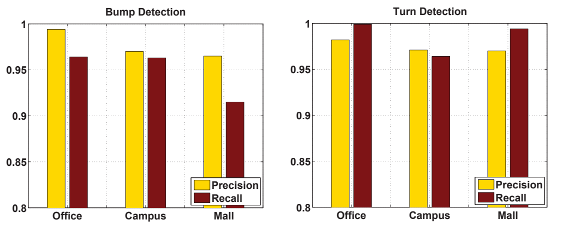
We repeat the test across different garages: using the data from one as training and another as testing. In reality, we can only obtain data from a limited number of garages for training, at least initially. Thus this test critically examines whether high accuracies are possible for vast numbers of unknown garages. Table 2 shows the cross-test accuracies of bump and turn detection, respectively. Each row represents training data and column test data. We observe that the accuracies are around, and some well above 90%. This encouraging evidence shows that it is very possible to retain the accuracy when training data are available from only limited numbers of garages.

Precision and recall of landmark detection.

After training landmark detectors, we further test their precision (ratio of true detections among all detected land-

Table 2: Cross-test of bump/turn detection (%)

train/test	office	campus	mall
office	95.5/93.6	91.9/95.6	90.1/90.3
campus	93.7/94.1	93.9/96.3	88.5/90.8
mall	94.1/92.3	90.6/94.6	91.5/91.0



(a) (b)

Figure 10: Precision and recall of bump and turn detection in three different garages.

marks) and recall (ratio of true detections to groundtruth number of such landmarks) over whole traces. They tell how likely the detector makes mistakes (high precision means less chances for mistakes), and how close all groundtruth ones are detected (high recall means more real ones are detected). An ideal detector would have 100% precision and recall.

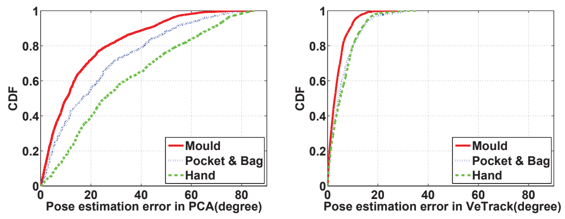
The precision and recall of bump and turn detection are shown in Figure 10. Both prediction and recall of bump detection are over 91% and those of turn are over 96%. Turn detection has better performance because it uses features from more reliable gyroscope data. We also find that poses in the mould has the best performance because they have the least disturbances; those in pocket and bag are better than in those in hand because they do not experience disturbances from hand movements.

Accuracy of shadow tracing. The performance of trajectory tracing highly depends on the accuracy of phone pose estimation (relative orientation between the phone and vehicle's coordinate systems). We compare its accuracy in 3D and 2D tracing methods. Similar to other work [17, 40], we use principle component analysis (PCA) in 3D method to find the maximum acceleration direction as Y-axis. To obtain the ground truth, we fix a phone to the vehicle and align its axes to those of the vehicle. The error is defined as the angle between the estimated and ground truth Y-axis of the phone. For fair comparison, we project the 3D pose to horizontal plane before calculating its error.

The CDFs of errors (Figure 11) show that: 1) Our 2D method is more accurate, with the 90-percentile error at $10 \sim 15^o$ while that of the 3D method is around $50^o \sim 70^o$, which in reality would make accurate tracking impossible. 2) The 2D method is more robust to disturbances in unstable poses such as pocket/bag and hand-held, whereas the 3D method has much larger errors for the latter two. This shows that our shadow tracing is indeed much more practical for real driving conditions. In addition, we find that the PCA needs a window of 4s for unchanged pose, while the 2D method is almost instantaneous.

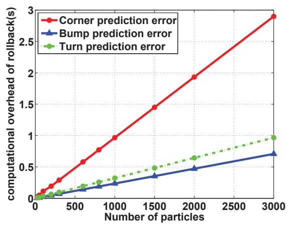
6.3 Realtime Tracking Latency

Realtime tracking latency is the time the tracking component needs to finish computing the location after obtaining sensor data at t. When there are prediction mistakes, it also includes latencies for detecting mistakes, rollback and



(a) (b)

Figure 11: CDFs of pose estimation error: (a) 3D method. (b) Our 2D method.



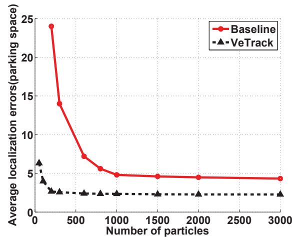


Figure 12: Latency by different rollback types and numbers of particles.

Figure 13: Tracking error by numbers of particles.

re-computing of the current location. This is measured on iPhone 4s, a relatively slower model. As shown in Table 3, landmark detection for bump, turn and corner each cost ~ 0.2ms. In almost $\sim 90\%$ of time where predictions are correct, one round of tracking is computed within 1.7ms. The 2.3ms computing finishes within the 20ms particle state update interval, causing no real time delay. For about $\sim$ 10% of time, recovering for bump, turn and corner errors (each $\sim$3%) take 64ms, 47ms and 193ms. The worst case is less than 0.2s, hardly noticeable to human users.

Table 3: Realtime Tracking Latency.

	bump	turn	corner
landmark detection	0.21ms	0.22ms	0.22ms
90% realtime tracking	1.7ms		
10% rollback	47ms	64ms	193ms

Figure 12 shows the latency as a function of number of particles, each curve for one different type of wrong predictions resulting in rollback. All curves grows linearly, simply because of the linear overhead to update more particles. Note that the difference between latencies of different curves is caused by different sizes of rollback windows (1s, 1s and 3s for bump, turn and corner detection errors, respectively). Although bump and turn detection have the same rollback window sizes, recovering turn errors has slightly higher computation overhead. In experiments we find that $100 \sim 200$ particles can already achieve high accuracy, which incurs only $0.05 \sim 0.2s$) latency. Such disruptions are minimal and not always perceptible by users.

6.4 Tracking Location Errors

Parking location errors. The final parking location is important because drivers use it to find the vehicle upon return. We use the number of parking spaces between the real and estimated locations as the metric, since the search

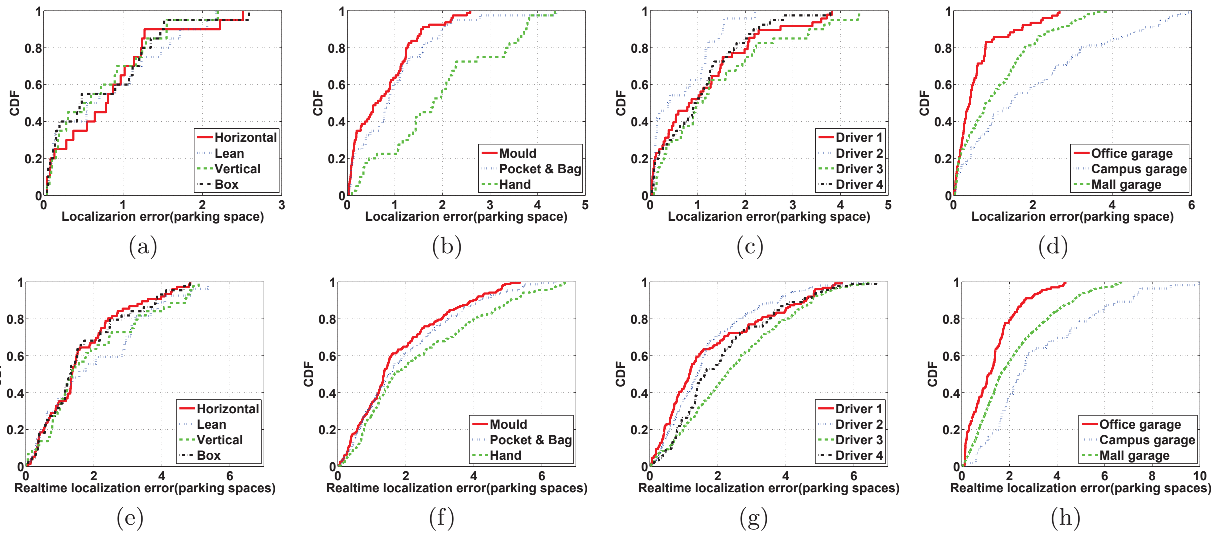


Figure 14: Final parking location errors (1st row) and realtime tracking location errors (2nd row). (a)(e) 4 phones in the mould. (b)(f) in mould, pocket&bag, and hands. (c)(g) different drivers. (d)(h) different garages.

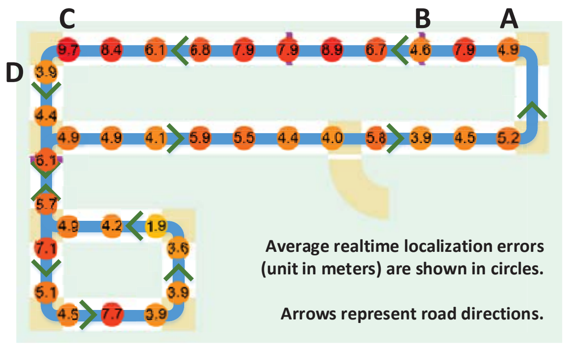


Figure 15: Average realtime tracking errors on different garage locations.

effort depends more on how many vehicles to examine, not the absolute distance.

In order to compare all 8 poses, we show the results in the mall garage. Figure 14(a) shows the 4 phones in the mould. They have relatively small errors: all four poses have similar performance, with 90-percentile error less than 2 parking spaces. The maximum error is less than 3 parking spaces, which is sufficient for remote keys to trigger a honk to locate the car.

Figure 14(b) shows results for different pose categories. Poses in the mould category achieve the best performance, i.e., ~ 2 parking spaces at 90-percentile, with maximum error of 3 parking spaces. Those in the pocket or bag endure small disturbances thus achieve performance that is a little worse than mould category, i.e., ~ 2 parking spaces at 90-percentile, with maximum error of 4 parking spaces. Those in hand have largest errors, i.e., ~ 4 parking spaces at 90-percentile, with maximum error of 5 parking spaces. These larger errors are due to hand disturbances causing more incorrect landmark detections.

We also evaluate the impact of different drivers. Figure 14(c) shows those of two taxi drivers (1 and 3) driving cabs and two volunteer drivers (2 and 4) driving their own cars. The results do not differ too much among different drivers; all have $1.5 \sim 3$ parking space errors at 80-percentile, while the maximum error of 5 parking spaces is from a taxi driver who drives very aggressively, which causes more incorrect detections.

Finally we evaluate impact of the type of parking garage. The 90-percentile errors are around 2,3 and 5 parking spaces, respectively, and maximum errors are 3, 5 and 6 parking spaces, respectively. The difference is caused by different structures. The office garage has best results because it has regular shapes (7(b)) and smooth paved surfaces which minimize road disturbances. The campus garage is the worst because of its irregular shape (7(a)), especially the "Z"-junction of two consecutive turns where many drives take a shortcut instead of two 90-degree turns.

Real time location error. We present the CDFs of real time tracking error in the second row of Figure 14, arranged the same as the first row. The trends are similar in general, but real time errors are generally $50 \sim 100\%$ larger than corresponding parking errors. For example, Figure 14(e) shows all 4 poses in the mould have 90-percentile error around 4 parking spaces. The maximum error is ~ 5 parking spaces. While those in Figure 14(a) are 2 and 3 parking spaces. Figure 14(f) shows that poses in the mould have the least errors, while those in hand have largest errors, the same trend as Figure 14(b) while all errors are about 60% larger than those in Figure 14(b). Figure 14(g) and (h) are similar as well.

This is because: 1) For final parking location we penalize particles still having non-negligible speeds after the vehicle has stopped. Thus remaining particles are those that have correctly "guessed" the vehicle states. 2) Real time errors include many locations in the middle of long straight driving, where no landmarks are available for calibration. Such locations tend to have larger errors. 3) The vehicle location has much larger uncertainty at the beginning. Thus relatively greater errors are included in real time results. But final location is usually after multiple calibrations, thus better accuracy.

Spatial distribution of real time tracking errors on a garage map with 3 bumps and 8 turns is shown in Figure 15. Each circle has a number, the error averaged over different traces and poses for that location. We observe that in general the error grows on straight paths, and is reduced after encountering landmarks (e.g., from 4.9m after a corner A, growing to 7.9m then reduced to 4.6m after a bump B; 9.7m at C before a corner to $3.9m$ at D).

The number of particles also impact tracking accuracy. We compare VeTrack with a straightforward baseline that uses 3D tracing and 2D road strips, without two critical components of 2D tracing and 1D roads, Figure 13 shows results for the mall. VeTrack's average localization error quickly converges to ~ 2.5 parking spaces when there are 200 particles (the office and campus garages need only $100 \sim 150$ particles). More particles do not further decrease the error because they are still subject to landmark detection mistakes. The baseline needs about 1000 particles to stabilize, and it is around 5 parking spaces. This shows that VeTrack needs about one order of magnitude less particles, thus ensuring efficient computing for real time tracking on the phone; it also has better accuracy because of the two critical components.

7. RELATED WORK

Phone pose estimation. Existing work [17, 24, 40] estimates the 3D pose of the phone. The latest one, A^3 [47], detects high confidence compass and accelerometer measurements to calibrate accumulative gyroscope errors. The typical approach [40] in vehicular applications is to use the gravity direction as the Z-axis of the vehicle, assuming it is on level ground; gyroscope is used to determine whether the vehicle is driving straight; and the direction of maximum acceleration is assumed to be the Y-axis of the vehicle. As explained in Section 3, it cannot handle vehicles on a slope, and the direction of maximum acceleration may not be vehicle forwarding direction. The estimation also requires long time of unchanged pose, unsuitable under frequent disturbances.

Landmark detection. Distinctive data patterns in different sensing modalities of smartphones have been exploited for purposes including indoor localization [6, 32, 39]. Similarly, VeTrack detects distinctive inertial sensor patterns by road conditions (e.g., bumps and turns) to calibrate the location estimation. Its algorithms are designed specifically for robustness against noises and disturbances on inertial data from indoor driving.

Robotic localization. SLAM is a popular technique for a robot to acquire a map of its environment while simultaneously localizing itself in this map [25]. It has been adapted for indoor localization leveraging WiFi signals [15]. Unlike SLAM, VeTrack assumes the map is available and determines the vehicle location. This is equivalent to robot localization which finds the pose of a robot relative to a given map [35].

Earlier work on robot localization uses Kalman Filters [38] and Markov localization [35]. More recent Sequential Monte Carlo (SMC) [16, 36] uses a set of samples (i.e., "particles") drawn from a probability distribution to represent the position. The samples evolve over time based on the action model and measurements [35].

VeTrack uses the same SMC approach. But SMC is only a framework: what states are needed to model the dynamics of the physical system, what algorithms are needed to update such states, are all problem dependent. Also smartphone inertial data have significant noises and disturbances; they do not have high precision sensors such as laser rangers, high definition cameras, wheels that can provide accurate measurements for robots. Thus the state/weight initialization and update have to be carefully designed to produce reasonable results despite low quality data.

Dead-reckoning. Dead reckoning is a well explored approach that estimates the future location of a moving object (e.g., ground vehicle, robot or aircraft [3, 8, 20]) based on its past position and speed. Compared with them, VeTrack does not have special, high precision sensors (e.g., odometer in robotics or radar [27] for ground vehicles), while the required accuracy is much higher than that of aviation.

Dead reckoning has been used for indoor localization using smartphones equipped with multiple inertial sensors [12, 30]. Its main problem is fast error accumulation due to inertial data noises and a lot of work has attempted to mitigate the accumulation. Foot-mounted sensors have been shown effective in reducing the error [31, 42]. Smartphones are more difficult because their poses are unknown and can change. Some outdoor localization work (e.g., CompAcc [13]) employs periodic GPS measurements to recalibrate the location. UnLoc [39] replaces GPS with virtual indoor landmarks with unique sensor data patterns for calibration.

To prevent the error accumulation, VeTrack simultaneously harnesses constraints imposed by the map and environment landmarks. Landmark locations most likely remain unchanged for months or even years. The 2D pose estimation handles unknown and possibly changing phone poses. Their output provide calibration opportunities in the SMC framework to minimize error accumulation.

Estimation of vehicle states. There have been many research efforts using smartphones' embedded sensors to monitor the states of vehicles (e.g. dangerous driving alert [23], car speaker [43] and CarSafe [44]); inspect the road anomaly or conditions (e.g., Pothole Patrol [14]); and detect traffic accidents (Nericell [24] and WreckWatch [41]). The vehicle speed is a critical input in many such applications. While it is easy to calculate the speed using GPS outdoors [19, 34], the signal can be weak or even unavailable for indoor parking lots. Some alternative solutions leverage the phone's signal strength to estimate the vehicle speed [10, 11]. VeTrack uses inertial data only, thus it works without any RF signal or extra sensor instrumentation.

There have been autonomous parking system prototypes where a vehicle can sense the environment, detect available spots and finally park itself (e.g., CoCar [2]). Such systems usually leverage high precision cameras, laser rangers and advanced computer vision techniques, which are not feasible within the much constrained hardware of commodity smartphones.

8. DISCUSSION

Uncertainty in starting locations and speed. During the initialization of particles, we assume the location is at entrance and the speed zero. However, when the VeTrack app starts, the car may have already entered the garage. Thus the start of the trace may not be the entrance and the velocity not zero. This assumption can be easily removed by sampling from an empirical distribution to set the initial location and velocity. We do need more particles to represent the distribution, but the constraints imposed by paths and landmarks will quickly narrow down the possible locations of the vehicle. In one experiment we tried a speed distribution between $0 - 10m/s$ and 3000 particles. We find that after $2 - 3$ landmarks the location estimation quickly converges and the system can still achieve high accuracy. Basically when enough landmarks exist, the uncertainty can still be reduced and vehicle location determined.

Unsupervised feature learning. We design certain features manually for machine learning techniques. However, those features might not be the optimal under all circumstances; more efficient features specific to different environment may exist. We plan to investigate unsupervised feature learning [18] to automatically devise features for better performance and generality.

Stationary or reverse conditions. There are certain movement patterns not yet considered in this work: when the parking lot is congested, the vehicle may exhibit stop-and-go movements with long stationary periods; drivers may back up the vehicle into parking spaces. We plan to add stationary and reverse detection algorithms so that these conditions can be properly recognized and handled within the real time tracking component.

Simultaneous disturbances and landmarks. In our experiments disturbances such as hand-held movements happen only during straight driving but not bumps or turns. This is a reasonable assumption for a single driver that is most likely focusing on driving through these landmarks, and will not have the extra cognitive bandwidth to pick up the phone and look. However, if the phone is held by a passenger, hand-held movements could happen any time, even with small probabilities. We will investigate what impact this may have on VeTrack and further improve the robustness of our detection algorithms.

Other types of landmarks. In VeTrack, we mainly use bumps and turns as landmarks to recalibrate the location of vehicles. However, other types of landmarks (e.g., spots with RF signals or special magnetic fields) could also be incorporated. Whenever available, they provide additional constraints and calibration opportunities that limit the possible locations of the vehicle and would further improve the accuracy. It's just unfortunate that in the three underground parking structures we experimented, no RF signal were available. This is because the functional areas of those buildings (e.g., office, malls) are usually above ground. Thus no WiFi APs are deployed underground. We acknowledge although this is not uncommon in big Asian cities due to high population density, in other parts of the world (e.g., North America) above-ground parking structures can take advantage of WiFi signals wherever available.

Generalization of techniques. The real time tracking problem itself is quite specific. However the shadow trajectory tracing and 2D phone pose estimation techniques are essential to model and infer vehicle dynamics where arbitrary phone pose or special driving conditions (e.g., slopes or even slanted driving on slopes) exist. Thus the benefits are general to phone-vehicular applications, not just underground navigation.

Landmark mapping. Garage maps may not necessarily have landmark locations measured and marked. In reality we did not find this too much an overhead. Within half an hour we were able to manually measure their locations and mark them on maps. Because this is a one-time effort with long-lasting benefits, we believe it is worthwhile.

More varieties in different factors. The performance of VeTrack may depend on many factors, including the structures of parking lots, drivers, vehicle and phone make/models. We intentionally choose parking structures of three different building types (office, campus and mall), with several drivers and vehicle make/models. For example, the one in the mall has circular connecting ramps which are common for multi-level parking structures. We find that the yaw reading from gyroscope can reliably tell the turned angle and thus the location on such cylinder ramps.

Although we have tried our best to extend these factors, they are not exhaustive due to our limited resources and manpower. Nevertheless, we believe the most challenging part is the accurate 2D tracking of vehicle location, which we have designed multiple techniques and addressed extensively. We plan to conduct more comprehensive experiments in more types of parking structures, drivers, vehicle and phone make/models to identify potential limits and improve the design.

9. CONCLUSIONS

In this paper we describe VeTrack which tracks a vehicle's location in real time and records its final parking location. It does not depend on GPS or WiFi signals which may be unavailable, or additional sensors to instrument the indoor environment. VeTrack uses only inertial data, and all sensing/computing happen locally on the phone. It uses a novel shadow trajectory tracing method to convert smartphone movements to vehicle ones. It also detects landmarks such as speed bumps and turns robustly. A probabilistic framework estimates its location under constraints from detected landmarks and garage maps. It also utilizes a 1D skeleton road model to greatly reduce the computing complexity.

Prototype experiments in three parking structures and with several drivers, vehicle make/models have shown that VeTrack can track the vehicle location around a few parking spaces, with negligible latency most of the time. Thus it provides critical indoor location for universal location awareness of drivers. Currently VeTrack still has quite some limitations, such as manual feature design, simultaneous disturbances as discussed previously. We plan to further investigate along these directions to make it mature and practical in the real world.

Acknowledgments

We thank the anonymous reviewers for their constructive critique, and our shepherd Marco Gruteser for his valuable feedback and advice, all of which have helped us greatly improve this paper. We also thank Kaigui Bian for his input during early investigation of the work. This work is supported in part by NSF CNS-1513719, NSFC 61231010 and Beijing Municipal NSF 4142022.

10. REFERENCES

[1] Apple Developer Center. https://developer.apple.com/.

[2] CoCar: Autonomous Parking in an Underground Parking Garage. https://www.youtube.com/watch?v=G4XYMbtH758.

[3] Inertial navigation system, Wikipedia. https://en.wikipedia.org/wiki/Inertial_navigation_system.

[4] Parking sensors mesh network. http://www.streetline.com/parking-analytics/parking-sensors-mesh-network/.

[5] SFpark. http://sfpark.org/how-it-works/the-sensors/.

[6] 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.

[7] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE INFOCOM*, 2000.

[8] D. M. Bevly and B. Parkinson. Cascaded kalman filters for accurate estimation of multiple biases, dead-reckoning navigation, and full state feedback control of ground vehicles. *Control Systems Technology, IEEE Transactions on*, 15(2):199–208, 2007.

[9] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

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

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

[17] H. Han, J. Yu, H. Zhu, Y. Chen, J. Yang, Y. Zhu, G. Xue, and M. Li. Senspeed: Sensing driving conditions to estimate vehicle speed in urban environments.

[18] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[19] 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.

[20] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for an autonomous mobile robot. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 384–389 vol.1, May 1990.

[21] K. Kanazawa, D. Koller, and S. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 346–351. Morgan Kaufmann Publishers Inc., 1995.

[22] K. Li, M. Lu, F. Lu, Q. Lv, L. Shang, and D. Maksimovic. Personalized driving behavior monitoring and analysis for emerging hybrid vehicles. *Pervasive Computing*, 2012.

[23] 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.

[24] 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.

[25] 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.

[26] 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 &#38; Networking*, MobiCom '13, pages 75–86, New York, NY, USA, 2013. ACM.

[27] D. H. Nguyen, J. H. Kay, B. J. Orchard, and R. H. Whiting. Classification and tracking of moving ground vehicles. *Lincoln Laboratory Journal*, 13(2):275–308, 2002.

[28] V. Otsason, A. Varshavsky, A. LaMarca, and E. De Lara. Accurate gsm indoor localization. In *UbiComp 2005: Ubiquitous Computing*, pages 141–158. Springer, 2005.

[29] S. Preece, J. Goulermas, L. Kenney, and D. Howard. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data.

[30] 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.

[31] 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.

[32] 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.

[33] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 85–98, New York, NY, USA, 2010. ACM.

[34] 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.

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

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

[37] Y. Tian, R. Gao, K. Bian, F. Ye, T. Wang, Y. Wang, and X. Li. Towards ubiquitous indoor localization service leveraging environmental physical features. 2014.

[38] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE, 2000.

[39] 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.

[40] 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*, pages 41–54. ACM, 2013.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] M. Youssef and A. Agrawala. The horus wlan location determination system. In *ACM MobiSys*, 2005.

[46] M. Zhao, R. Gao, J. Zhu, T. Ye, F. Ye, Y. Wang, K. Bian, G. Luo, and M. Zhang. Veloc: finding your car in the parking lot. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pages 346–347. ACM, 2014.

[47] P. Zhou, M. Li, and G. Shen. Use it free: Instantly knowing your phone attitude. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 605–616. ACM, 2014.