VeMap: Indoor Road Map Construction via Smartphone-based Vehicle Tracking

Ruipeng Gao
Beijing Jiaotong University

Guojie Luo
Peking University

Fan Ye
Stony Brook University

***Abstract*—Since GPS signal is not applicable indoors, vehicle tracking has proven a hassle in underground parking structures. Recent solutions highly rely on floor map to constraint inertial sensors noises. In this paper, we propose *VeMap*, a road map construction system using only smartphones inside vehicles. It saves effort-intensive and time-consuming business negotiations with building operators, and expensive personnel cost to gather such data. It fuses multiple sensors to calibrate inertial noises, and uses Dynamic Time Warping to align multiple trajectories. We represent the floor plan with occupancy grid mapping, and explore a vision-mobile joint algorithm to extract its skeleton and form the road map. VeMap is tested in a $250m \times 90m$ parking structure, and it can be directly used for driving navigation to free parking spaces.**

I. Introduction

Vehicle tracking is the basis for many novel driving navigation applications and services. Thanks to a number of GPS systems and devices, outdoor vehicle tracking and navigation is widely deployed. However, whenever we drive into indoor environments such as underground parking structures where GPS signals can hardly penetrate, we lose vehicle location awareness.

Providing vehicle tracking capability indoors will navigate drivers to orient themselves relative to a large and unfamiliar environment. For example, it enables turn-by-turn instructions guiding drivers to free parking spaces, and record the final parking location to direct the driver back upon return. However, it is still far from straightforward. Majority work instrument the environment via sensor networks, which is not always feasible and costs expensive financial or human efforts.

In a recent work [1], we propose "VeTrack" that leverages smartphones in car to track vehicle driving. It utilizes different inertial sensors in smartphone, fuses them with the Particle Filter framework to represent vehicle states probabilistically, and harnesses constraints by indoor floor map and detected landmarks to robustly calibrate the vehicle location.

However, main limitation in those sensor networks and VeTrack is that they highly rely on indoor floor maps. Service providers have to conduct effort-intensive and time-consuming business negotiations with building owners to collect the floor map, which is deleterious to large scale coverage in short time. Besides, in order to reduce inertial sensor noises on smartphone, VeTrack even needs the road skeleton from floor map, which costs much more human efforts.

In this paper, we propose VeMap, which constructs the indoor road map from crowdsourced mobile traces. It also generates the road skeleton of constructed floor maps, which provide great conveniences to other indoor driving navigation services. Both the floor map construction and topology structure generation in VeMap are automatic, and it is a smartphone-only solution which gets rid of special hardware such as OBD tools.

VeMap design is based on the realization that computer vision and mobile techniques have complementary strengths. Mobile ones identify different landmarks (speed bumps and road turns), and describes the sketch of accessible areas; while vision ones can produce accurate geometric information and extract the skeleton. Thus we propose three methods to improve the accuracy of mobile measurements, and detect landmarks on each trace. Since VeMap inputs are crowdsourced with unpredictable noises, we also align multiple traces via Dynamic Time Warping. At last we use occupancy grid map to represent and generate the floor plan, and explore an automatic road skeleton extraction algorithm via vision and mobile techniques combination.

We make the following contributions in this work:

- We propose three methods to calibrate the vehicle trajectory from smartphone inertial sensors: angle calibration, dead-reckoning method and terminal points calibration. Vehicle trajectory accuracy is significantly improved.
- We formulate the multiple trajectory alignment problem as Dynamic Time Warping, and explore a dynamic programming algorithm to solve it.
- We represent and construct the indoor map via occupancy grid mapping, which is based on probabilistic formulations and robust to outliers; we also combine certain vision and mobile algorithms and automatically extract the road skeleton from constructed floor maps.
- We have conducted extensive evaluations in a $250m \times 90m$ indoor parking structure. We find that both the constructed floor map and road skeleton are accurate, and could be directly used by other indoor vehicle tracking and navigation applications.

II. Background on VeTrack

In this section, we describe how VeTrack [1] works and explain its limitations during deployment in reality.

A. Floor Plan Construction

Figure 1 shows basic VeTrack design. It leverages smartphone inertial data and floor maps of parking structures to track vehicles in indoor environments. Since inertial sensors on smartphone are noisy, we extract the skeleton of floor

978-1-5090-1328-9/16/$31.00 ©2016 IEEE

maps and project vehicle trajectories onto that skeleton, so as to reduce sensor noises. Then it utilizes a particle filter framework to track vehicles with that skeleton map. To deal with sensor noises and phone movements, it also leverages simple machine learning method, i.e. Support Vector Machine (SVM), to identify different kinds of landmark on the floor map (speed bumps and road turns), and uses them to calibrate vehicle locations in particle filter.

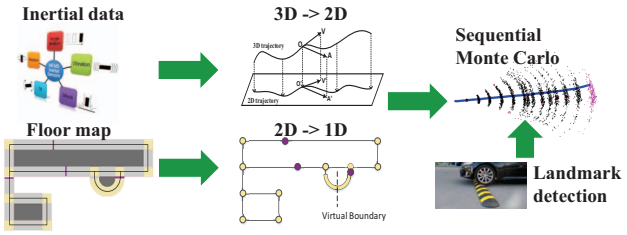


Fig. 1. Real time vehicle tracking procedures in VeTrack. It use road skeletons to simplify 3D tracking into 1D tracking, and leverages Particle Filter framework for robust tracking. It also detects landmark detection to calibrate vehicle locations.

We implement the VeTrack prototype on iOS, and conduct experiments in three different parking structures, with multiple vehicles and drivers. VeTrack can estimate the vehicle's real time location with error of $2 \sim 4$ parking spaces at 80-percentile.

B. Limitations in VeTrack

Though VeTrack achieves the accurate real time tracking errors in multiple environment, e.g. different drives, phone placement, vehicles, and parking structures, it relies on the securable floor maps of parking structures. However, the tracking service providers have to explore effort-intensive and time-consuming business negotiations with building operators, or hire dedicated personnel to collect the floor map. Neither is conducive to ubiquitous coverage of vehicle tracking in pervasive parking structures. Thus an automatic floor maps construction system is valuable in pushing indoor vehicle tracking and navigation applications.

Additionally, even with the floor map, service providers must spend extra human efforts to transform it into a road skeleton map. This impedes the large scale coverage of indoor vehicle tracking in short time.

In this paper, we propose VeMap, which provides accurate floor maps of indoor parking structures via mobile crowdsensing, and automatically extract their road skeletons which VeTrack could deploy on. We organize the paper as follows: in Section III, we calibrate different inertial sensors on smartphones to improve the accuracy of mobile trajectories; in Section IV, we detect different landmarks and leverage Dynamic Time Warping algorithm to align different mobile trajectories, and generate the floor map with occupancy grid mapping; we also propose an automatic skeleton extraction algorithm in Section V. After comparison to related work (Section VI), we conclude the paper in Section VII.

III. Inertial Data Co-Calibration

Inertial sensors on smartphones (e.g., accelerometer, gyroscope, compass), can be used to generate mobile trajectories for vehicles in indoor environments. However, due to potentially large sensor noises and error accumulation, those trajectories have extreme errors that could be hardly used directly, as Figure 2(b) shows. From the simple dead-reckoning method via gyroscope and double integrated accelerations, both tracking distance and angle have extreme errors, and even a road is missing.

In this section, we aim to provide accurate mobile trajectories via inertial sensor co-calibration. We first provide a angle calibration algorithm to improve the orientation accuracy, then we analyze different dead-reckoning method for tracking without a floor map. We also leverage terminal points to further calibrate whole trajectory. The final result of that example trajectory is shown in Figure 2(d).

A. Angle calibration

In inertial data based vehicle tracking, angles are used to update driving directions. The angle data could be obtained either from compass, or from gyroscope sensor. However, we observe that neither sensor provides accurate angle data: compass suffers frequently from interferences by surrounding electromagnetic objects, while gyroscope is accurate in short time but has long-time drifts. Figure 3(a)3(b) show their results for a 2-minute driving. We omit the errors when driving in road turns, since angle ground truth are difficult to measure during that period.

Although there are some work in mobile computing calibrate angles from sensor fusion ([2], [3]), they focus on walking scenarios which have unique walking patterns, and they leverage a short time window to provide real time angle estimations. Even though, angle calibration is still challenging in smartphone tracking.

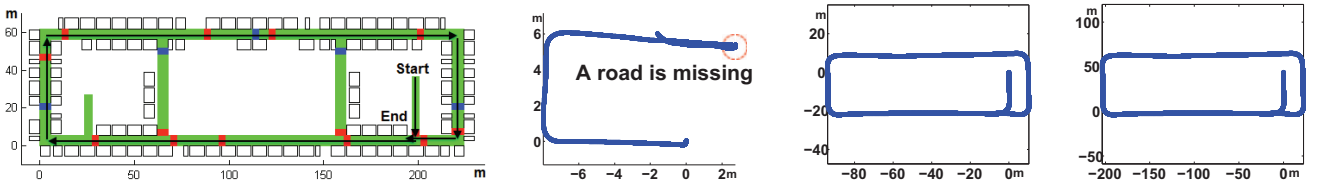
Observing that gyroscope sensor is accurate in short time and have linear drift [3], we aim to calculate that constant drift from other sensor, so as to track users with the calibrated gyroscope. Our intuition comes from that when vehicle leaves and enters the parking structure, its location and orientation is unique and can be easily obtained from GPS signal and open map library, i.e. at the time when GPS signal appear/disappear.

Figure 3(a)3(b) show the angle errors after calibration. We observe that there are no drifts for the calibrated angle, with maximum errors around 4°.

B. Dead-reckoning method

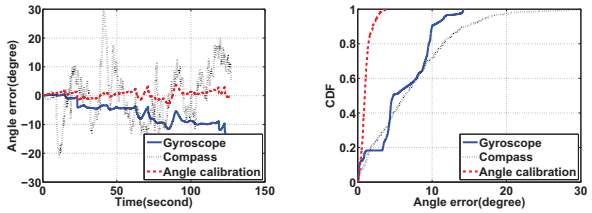
We aim to construct the indoor floor map via mobile crowdsensing, thus there are no available indoor maps for vehicle tracking at this stage. We could only use dead-reckoning methods to generate mobile trajectories.

Naive method: double integration on accelerations. Since smartphones are equipped with accelerometer sensors to measure the 3-axis acceleration in smartphone's coordinate system (shown in Figure 4(a)), a simple tracking method is to apply double integration on accelerations. In order to test the performance of different phone placement, we design a mould to



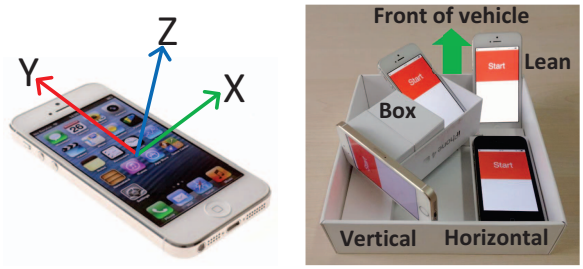
(a) Ground truth (b) Dead-reckoning (c) Calibrated angle and constant velocity (d) Terminal point calibration

Fig. 2. An example trajectory in indoor parking structure: (a) ground truth; (b) dead-reckoning method via accelerometer and gyroscope; (c) constant velocity model with calibrated angle, and the trace shape is correct ; and (d) after terminal points calibration, both trace size and global coordinates are correct.



(a) Real time angle errors (b) CDF of angle errors

Fig. 3. Angle errors in different methods.



(a) 3-axis of smartphone (b) Mould of 4 iPhones

Fig. 4. Coordinate system in smartphones and a mould with 4 iPhones.

hold 4 iPhones with 4 different poses: horizontal, lean, vertical and box (Figure 4(b)).

We leverage the gravity API in iOS to project 3-axis accelerations onto floor plane, and double integrate it for tracking with the calibrated angle data. Figure 2(b) shows how it performs for the example trajectory. We observe that in this method, there are enormous errors during double integration, which cause the loss of even one road!

Constant velocity model: We conduct experiments to evaluate how the vehicle speed changes when driving in parking structures. We leverage the 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. To align inertial data and video collected from different devices temporally, we first synchronize the time clock among all devices, and all data are recorded with a timestamp.

We observe that the real time vehicle speed and landmark events. We leverage the same detection method in VeTrack to identify speed bumps and road turns via inertial sensors. We observe that the vehicle has an approximately constant velocity when driving on flat and straight road, and its velocity decreases obviously when passing landmarks. We test the real time driving velocity for 20 mobile trajectories among multiple cars, drivers and phones, and find the similar situation except different constant velocity values on each scenario.

Based on this observation, we make the assumption of constant velocity out of landmark events, and assign δ as the constant velocity for each car, which could be estimated during outdoor low-speed driving. Figure 2(c) shows the trajectory from constant velocity model and calibrated angles, with similar shape as the ground truth and has no road loss. We further calibrate that constant velocity δ via terminal points in parking structures.

C. Terminal points calibration

Terminal points are the entrances/exits of indoor parking structures, which could be easily detected via GPS signal disappearance/appearance. We aim to leverage the GPS signal to obtain the global locations of terminal points, and use them to calibrate the scale of mobile trajectory. Note that though each mobile trajectory always has only one terminal point, e.g. entering the parking structure and stop their car, or launch their car from parking space and drive outside, we could identify the parking state and merge these two mobile trajectory into a complete one: starting from the entrance and ending from the exit.

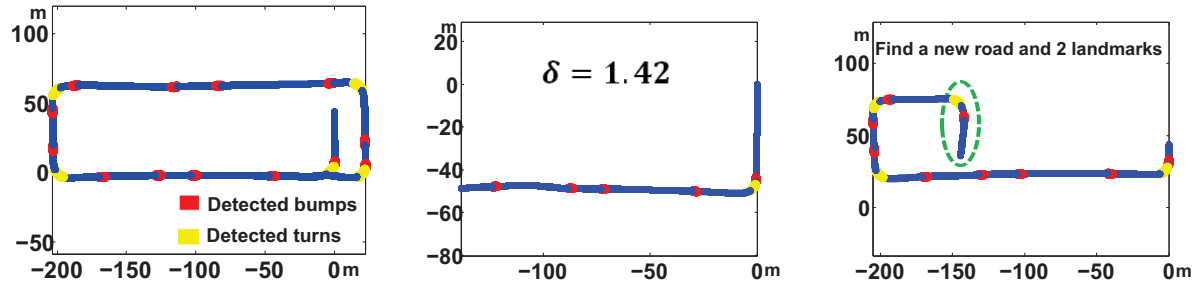
Denote X_{in} and X_{out} for 2D global coordinates of entrance/exit from GPS signal, and denote L_{trace} as the length of mobile trajectory between its two endpoints, we compute the scale factor as $s = \frac{|X_{out} - X_{in}|}{L_{trace}}$, and transform the mobile trajectory with 2D global coordinates.

The final result is shown in Figure 2(d), which shows both shape and size are accurate for the example trajectory.

IV. Trajectory Alignment

Even after we calibrate mobile trajectory for each vehicle, the velocity and angle accuracy differ a lot for multiple vehicles, thus their trace are always not coincident, and using their data as-is will cause severe errors in mapping. In this section, we aim to align multiple vehicle trajectories with the same scale and position. Our intuition is to identify different landmarks on each trajectory, and use them to align multiple trajectories.

First we leverage the same landmark detection process in VeTrack, and identify two kinds of landmarks: speed bumps and road turns. Bumps cause acceleration fluctuations in the



(a) Detected landmarks (b) Alignment for size calibration (c) Alignment for finds new roads

Fig. 5. Trajectory alignment via landmarks: (a) detected landmarks on example trajectory; (b) use alignment to calibrate the size of a new trace; (c) use alignment to find new road segments from new a trace.

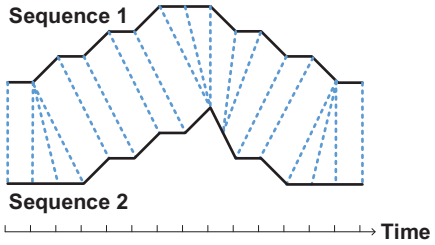


Fig. 6. A warping between two time sequences.

gravity-axis when a vehicle passes over, and turns are defined as durations in which a vehicle continuously changes its driving direction. We adopt the same features for each landmark, and explore a common machine learning algorithms, Support Vector Machine (SVM) for automatic detection. Thus we could mark different landmark events with time periods onto each trajectory.

Next we align multiple trajectories based on their landmark events. The alignment is not trivial since trajectory routes are not the same, and vehicle velocity also differs that even continuous landmark events might have different time periods for different vehicles. We leverage a mature voice recognition technique, Dynamic Time Warping (DTW) [4], to measure the similarity between two trajectories.

The trajectory is a time sequence, while two trajectories with the same route might have different length of time duration due to different velocity. Thus traditional Euclidean distance could not assess the distance/similarity between two time sequences. DTW extends and shortens time sequences, thus compute their similarity, as Figure 6 shows. Besides, the distance summation between similarity points, is denoted as Warp Path Distance.

We formulate the multiple trajectory alignment problem as DTW. Denote X and Y as two time sequences, and their lengths are $|X|$ and $|Y|$. The Warp Path is defined as $W = (w_1, w_2, ..., w_k)$, where $w_k = (i, j)$ represents the time point i in X and j in Y. In order to contain all time points in two sequences, a complete warp path W must begin with $w_1 = (1,1)$ and end $w_k = (|X|, |Y|)$.

Besides, since X and Y are time sequences, $w(i, j)$ should monotonically increase for both i and j, i.e.

$$w_k = (i, j), w_{k+1} = (i', j'), i \leq i' \leq i+1, j \leq j' \leq j+1 \quad (1)$$

The final warp path obtains the shortest distance, i.e.

$$D(i, j) = dist(i, j) + \min\{D(i-1, j), D(i, j-1), D(i-1, j-1)\} \quad (2)$$

where D is the cost matrix, and $D(i, j)$ represents warp path distance for two time sequences with length of i and j.

We leverage Dynamic Programming (DP) algorithm to solve the DTW problem, and align multiple trajectories with different landmarks. Figure 5(b)5(c) shows two usages of trajectory alignment: 1) calibrate the vehicle velocity of a new trace, and 2) find new road segments and assign new landmarks.

V. Automatic Mapping and Skeletonisation

In this section, we represent the floor map via occupancy grid map [5], which is a dominant paradigm for environment modeling in mobile robotics. Then we construct the map via crowdsourced vehicle trajectories. We also propose a vision-mobile joint algorithm to automatically extract the road skeleton.

A. Occupancy grid map

As Figure 7 shows, the occupancy grid map represents the environments by fine-grained grid cells with a variable representing the probability that the cell is accessible. We assign Gaussian distribution for each trajectory, and use them to compute the accessible probability for each cell.

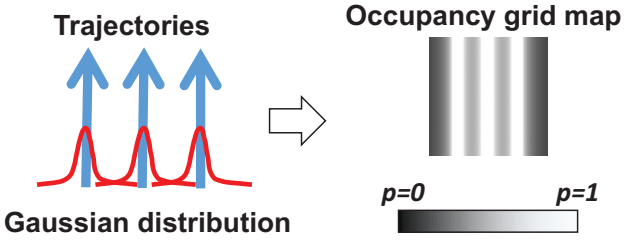


Fig. 7. Occupancy grid mapping via trajectories.

Next we use an automatic threshold based binaryzation technique to determine whether each cell is accessible, thus creating a binary map. To further improve the result, we implement a smoothing algorithm based on α-shape [6], which is a generalization of the concept of convex hull. The final floor map of that parking structure is shown in Figure 8.

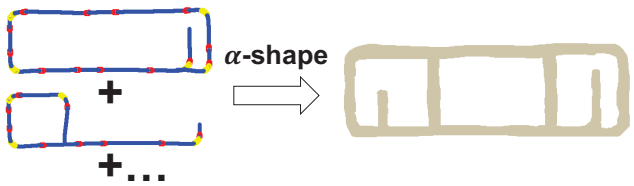


Fig. 8. Final floor map after α-shape smoothing.

B. Topology structure

In indoor vehicle tracking applications, e.g. VeTrack, they leverage the topology structure from floor map to constraint inertial sensor noises. Besides, the topology structure also provides significant convenience when navigating vehicles to available parking spaces in large parking lots. In these applications, service providers always measure landmark coordinates with a tape, and manually generate the topology structure offline, which cost extensive human efforts.

First we formulate the topology structure of indoor environments as a labeled undirected graph $G = \{N, E\}$, which represents a region in a compact form where nodes stand for landmarks and there is an edge between two nodes if the corresponding points are related by routes. Especially, there are three kypes of nodes for a topology structure, $N = \{S, B, T\}$, including: the position S for an entry or exit of the parking structure, B for each speed bump, and there are several road turns T on the routes. Those nodes represent specific landmark events when a vehicle goes through. Edges E are line segments that describe the actual routes in indoor environments between two nodes.

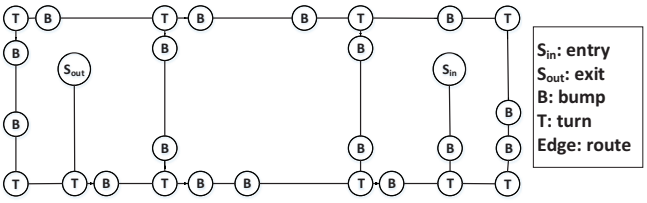


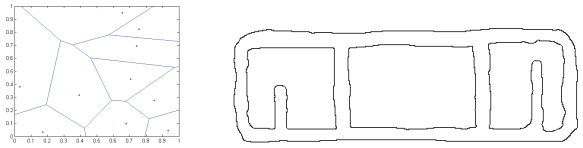
Fig. 9. Topology structure representation of the example parking structure.

One typical topology structure of the example parking structure is shown on Figure 9. We observe different types of nodes for landmarks, e.g. 17 speed bumps B, 10 road turns T, 1 entry S_{in} and 1 exit S_{out}.

Next we explore an automatic topology structure extraction algorithm. It leverages the reconstructed floor map to automatically generate the topology structure, and includes three steps:

Step 1: skeletonisation. We leverage a geometry algorithm, Voronoi diagram, to automatically generate the skeleton via reconstructed floor maps. Voronoi diagram partitions a plane into regions according to the distances to given points, as shown in Figure 10(a). Hence given the floor map, if the density of its boundary points goes to infinity, the Voronoi diagram converges to the skeleton.

We use Canny edge detection to identify the route boundary (shown in Figure 10(b)), and deploy Voronoi diagram to extract the skeleton (shown in Figure 11(a)). Then we project those detected landmarks onto the skeleton based on their shortest Euclidean distances. (Figure 11(b)).



(a) Voronoi diagram (b) Canny edge detection

Fig. 10. Preliminary for Skeletonisation: (a) Illustration of Voronoi diagram; (b) boundary from Canny edge detection.

Step 2: line-fitting. we use a line-fitting method to generate edges of the topology structure (Figure 11(c)). From each node we formulate a line to cover the skeleton, until it reaches another node. In case the accumulated errors between the line segment and skeleton are larger than a threshold, we end the line segment with a turn node, and try to find another line from that turn.

Thus we extract the topology structure automatically. Next we evaluate its performance for different indoor parking structures.

Evaluation. In order to deploy VeMap in constructing indoor floor map and road skeleton, we place 4 iPhones into the mould and put the mould on front console in the car, and collect 20 traces in a $250m \times 90m$ parking structure.

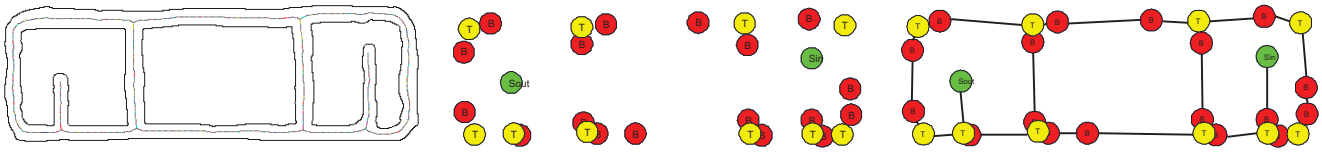
Next we evaluate the location accuracy of nodes. In Figure 12 we observe that the 80-percentile location errors of three kinds of landmarks are $0.8m$, $1.4m$ and $3.2m$, while road turns have the maximum errors since vehicle always turns near the road boundary.

At last we evaluate the orientation errors of generated edges. Figure 13 depicts the 80-percentile orientation errors are 4°, 7° respectively for two kinds of roads. The large errors are due to that vehicles always changes their directions at road turn regions.

VI. Related Work

Floor plan construction. Indoor floor plan construction is a new topic in mobile computing with limited pieces of work. Among them, CrowdInside [6] leverages inertial data from accelerometer, gyroscope and compass to reconstruct users' mobile trajectories, and uses "anchor points" with unique sensing data to correct accumulated errors. Such "anchor points" are also used for user localization (e.g., Unloc [7]). Walkie-Markie [8] leverages locations where the trend of WiFi signal strength reverses direction as anchor points, which are found to be more stable than signatures themselves. Jigsaw [9] and Sextant [10] combine mobile and vision techniques for site survey, and provide accurate floor plans in shopping malls. These work always rely on quantities of inertial sensor and WiFi signatures, and detailed data-gathering guidelines are also expected.

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 [11]



(a) Skeletonisation (b) Landmark projection (c) Line-fitting

Fig. 11. Topology structure generation process:(a) shows the skeleton from Voronoi diagram; (b) projects landmarks onto the skeleton, and generate different kinds of nodes; (c) uses line-fitting upon the skeleton and generate edges.

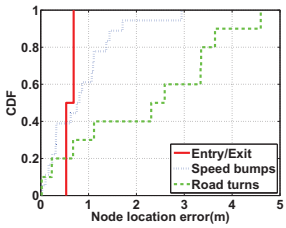


Fig. 12. Edge length errors.

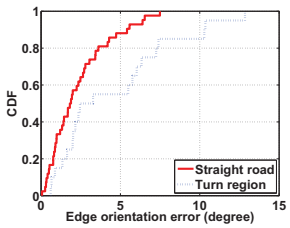


Fig. 13. Edge orientation errors.

and CarSafe [12]); inspect the road anomaly or conditions (e.g., Pothole Patrol [13]); and detect traffic accidents (Nericell [14] and WreckWatch [15]). The vehicle speed is a critical input in many such applications. While it is easy to calculate the speed using GPS outdoors [16], 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 [17], and tackle a multi-target tracking problem in indoor applications via sensor networks [18]. VeMap uses inertial data only, thus it works without any RF signal or extra sensor instrumentation.

Skeletonization. Skeletonization (i.e., skeleton extraction from a digital binary picture) provides region-based shape features. It is a common preprocessing operation in pattern recognition and raster-to-vector conversion for SLAM. There are three major skeletonization techniques: detecting ridges in the distance map of boundary points [19], calculating Voronoi diagram via the boundary points [20], and the thinning method via layer by layer erosion. However, we observe that those algorithms depend on parameters setting and always suffer from incorrect "branches" for skeletons, and those branches are difficult to remove by simple geometry methods.

VII. Conclusion

In this paper, we focus on the limitations in current indoor vehicle tracking systems: lacking of floor map and topology structure in underground parking structures. We propose a smartphone-only solution, and explore three methods to fuse multiple inertial sensors and calibrate the mobile trajectory. We also formulate the multiple trajectory alignment problem with Dynamic Time Warping, and use dynamic programming to solve it. At last we leverage occupancy grid map to represent and generate the floor map from crowdsourced smartphone users, and explore a vision-mobile joint algorithm to extract the road skeletons for other applications to use. VeMap enables service providers to efficiently construct floor plans at large scale from mobile users, thus avoiding the intensive efforts and time needed in business negotiations or environment surveys. Results in a large parking structure demonstrate that VeMap can produce reasonably complete and accurate locations/orientations of both floor map and road skeleton, and could be directly used by other indoor vehicle tracking applications.

References

[1] M. Zhao, T. Ye, R. Gao, F. Ye, Y. Wang, and G. Luo, "Vetrack: Real time vehicle tracking in uninstrumented indoor environments," in *ACM SenSys*, 2015.

[2] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Mobicom*, 2012.

[3] P. Zhou, M. Li, and G. Shen, "Use it free: Instantly knowing your phone attitude," in *ACM MobiCom*, 2014.

[4] "Dynamic Time Warping." [Online]. Available: Available at http://www.isi.edu/nsnam/ns/

[5] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots*, vol. 15, no. 2, pp. 111–127, 2003.

[6] M. Alzantot and M. Youssef, "Crowdinside: Automatic construction of indoor floorplans," in *SIGSPATIAL*, 2012.

[7] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *MobiSys*, 2012.

[8] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-markie: Indoor pathway mapping made easy," in *NSDI*, 2013.

[9] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li, "Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing," in *ACM MobiCom*, 2014.

[10] Y. Tian, R. Gao, K. Bian, F. Ye, T. Wang, Y. Wang, and X. Li, "Towards ubiquitous indoor localization service leveraging environmental physical features," in *IEEE INFOCOM*, 2014.

[11] J. Lindqvist and J. Hong, "Undistracted driving: A mobile phone that doesn't distract," in *ACM MobiSys*, 2011.

[12] 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 *ACM MobiSys*, 2013.

[13] 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 *ACM MobiSys*, 2008.

[14] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Using mobile smartphones for rich monitoring of road and traffic conditions," in *ACM SenSys*, 2008.

[15] J. White, C. Thompson, H. Turner, B. Dougherty, and D. C. Schmidt, "Wreckwatch: Automatic traffic accident detection and notification with smartphones," *Mob. Netw. Appl.*, vol. 16, no. 3, pp. 285–303, June 2011.

[16] 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 *ACM MobiSys*, 2008.

[17] 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 *ACM Ubicomp*, 2010.

[18] D. Ciuonzo, A. Buonanno, M. D'Urso, and F. Palmieri, "Distributed classification of multiple moving targets with binary wireless sensor networks," in *IEEE International Conference on Information Fusion (FUSION)*, 2011.

[19] Green Chris, "Improved alpha-tested magnification for vector textures and special effects," in *ACM SIGGRAPH*, 2007.

[20] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf, "Chapter 7: Voronoi diagrams," in *Computational Geometry (2nd revised ed.)*, 2000.