Bumping: A Bump-Aided Inertial Navigation Method for Indoor Vehicles Using Smartphones

Guang Tan, *Member, IEEE*, Mingming Lu, *Member, IEEE*, Fangsheng Jiang, Kongyang Chen, Xiaoxia Huang, *Member, IEEE*, and Jie Wu, *Fellow, IEEE*

Abstract—Equipped with accelerometers and gyroscopes, modern smartphones provide an appealing approach to *infrastructure-free* navigation for vehicles in indoor environments (for example parking garages). However, a smartphone-based inertial navigation system (INS) faces two serious problems. First, it is subject to errors that accumulate over time rather quickly, which may grow to a level that renders the navigation meaningless. Second, without human input or external references, the smartphone can hardly infer its initial position/velocity, which is the basis for distance calculation, since all that a smartphone can learn is its acceleration. This raises a practical concern, as users often need to start indoor navigation precisely when they are uncertain of their current whereabouts. In this paper, we present *Bumping*, a Bump-Aided Inertial Navigation method that significantly alleviates the above two problems. At the core of this method is a *Bump Matching* algorithm, which exploits the position information of the readily available speed bumps to provide useful references for the INS. The proposed method is easy to implement, requires no infrastructures, and incurs nearly zero extra energy. We conducted real experiments in tree parking garages of different environmental characteristics. The Bumping method produces an average position error of 4-5 m in these scenarios, improving the accuracy by up to 87.1 percent, compared to the basic inertial navigation method.

Index Terms—Indoor localization, inertial navigation, smartphones, vehicles

1 Introduction

Vehicle navigation has become a commodity service with high availability, low cost, and reasonable accuracy. In contrast to the success of outdoor navigation, wall-mounted signs continue to be the primary reference for vehicle navigation in indoor environments such as parking garages, where GPS does not work. This has attracted intense interests among both industry and academia to develop indoor positioning technologies that enable seamless outdoor-to-indoor navigation. However, existing indoor positioning techniques with meter-level or higher accuracy are mostly cost-prohibitive to deploy, either because of the high cost of the signal transmitters (e.g., Pseudolite, Ultra Wideband), or because of the signal's short ranges (e.g., Bluetooth, Infrared, Ultrasound, RFID), whose scaling can be very costly. The WiFi RSSI based positioning technique provides average accuracy to a few meters, but requires extensive off-line fingerprinting, which adds to the overall cost of the solution.

Recently, smartphones have gained enormous popularity as not only a communication tool, but also as a platform hosting a variety of services. The accelerometers and gyroscopes/compasses equipped on many of these phones open an opportunity for *infrastructure-free* positioning in indoor environments. Given an initial position and velocity, the phone can calculate its moved distance by double integrating the acceleration readings based on the law of inertia, and can produce a position relative to the start point. Since its advent in the 1940s, inertial navigation systems (INS) have been widely used in both military and civil applications.

The main drawback of an INS is that it suffers from accumulative errors due to *integration drift*: small errors in the measurement of acceleration are integrated into increasingly larger errors in velocity, which cause still greater errors in distance and therefore the final position. The fact that the errors grow quadratically with time poses great challenges to applications requiring high accuracy. Although a high-quality INS can keep the error within a relatively small range (e.g., 1100 m in position and less than one degree in orientation per hour [5]), it is not clear whether the low-end inertial sensors on smartphones can sustain a satisfactory level of accuracy for vehicles in minutes of time, as often needed in an indoor environment.

To answer the question, we conducted an empirical study on the inertial positioning performance of various smartphones. It is found that smartphone INS has surprisingly low accuracy, generating errors up to 100 m within a minute, at a rate super-linear with time. This makes the indoor navigation almost meaningless. Another practical problem is that an INS always assumes a known initial position/velocity, which is the basis of distance/angle

- *G. Tan, F. Jiang, K. Chen, and X. Huang are with SIAT, Chinese Academy of Sciences. E-mail: {guang.tan, fs.jiang, ky.chen, xx.huangg}@siat.ac.cn.*
- *M. Lu is with Central South University. E-mail: ming.lu@gmail.com.*
- *J. Wu is with Temple University. E-mail: jiewu@temple.edu.*

Manuscript received 27 Nov. 2012; revised 21 June 2013; accepted 27 June 2013. Date of publication 4 Aug. 2013; date of current version 13 June 2014. Recommended for acceptance by D. Xuan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.194

1045-9219 © 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

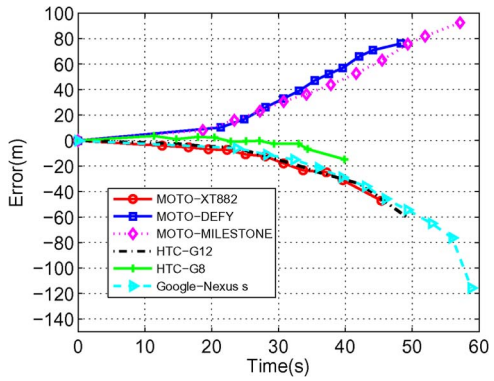


Fig. 1. INS errors of six smartphones during a straight course of 350 m.

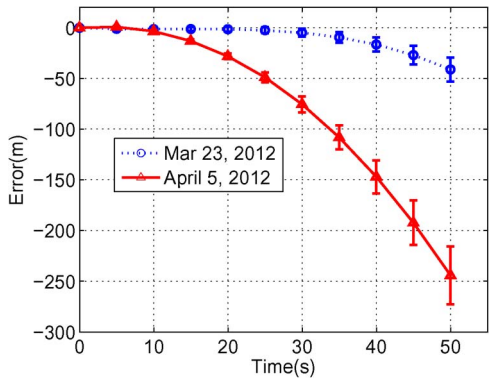


Fig. 2. Mean/standard deviation of error increasing with time, by a Google-Nexus S smartphone INS.

calculation. This is not always available, because the user may start using the INS only when the current location is unknown. Though the navigation application can ask the user to move to a nearest distinguishing spot (e.g., the entrance), a much more friendly solution is to discover the vehicle's current location automatically for the user.

In this paper, we present a novel method, named *Bumping*, that dramatically improves positioning accuracy and partially solves the automatic startup positioning problem, without using any extra infrastructure. The key idea is to exploit the position information of speed bumps that are readily available in parking garages. The bumps can be detected by accelerometers with a high success rate ($\geq$ 95 percent in our experiments), so as to provide important opportunities for the INS to correct its position and velocity. This way, the integration drift is limited within two successive bumps, instead of growing infinitely. Specifically, we make the following contributions:

1. an effective bump detection method based on the Gaussian mixture model;
2. a Hidden Markov Model (HMM)-based *Bump Matching* algorithm that accurately matches detection events to the right bumps, taking full consideration of false positives/negatives;
3. a set of methods that estimate the vehicle's restart speed and compensate for the accelerometer errors;
4. experimental evaluation of the navigation method in three parking garages.

Note that the evaluation results of the third garage is presented in the supplementary document.

For the tree parking garages we experimented with, the basic inertial navigation system produces average position errors of 39.52 m, 36.35 m, 14.30 m, while the Bumping method reduces the errors to 4.79 ms, 4.24 m, and 4.84 m, respectively. The achieved accuracy is very encouraging, because 4-5 m is merely the width of two parking lots in a row. Moreover, for an arbitrary drive without assuming a known initial position, Bumping can correctly start up the navigation 80 percent of the time by traversing only 3 bumps; the probability of startup at a wrong place is below 4 percent.

The outline of the paper is as follows. Section 2 describes a number of experiments that motivate our study. Sections 3 through 5 describe the bump detection, bump matching, and positioning methods. Section 6 presents the evaluation results and Section 7 concludes this paper. The overview of the proposed navigation system and the related works are provided in the supplementary document.

2 Motivation

This section shows a set of experimental results that serve as the motivation of our study.

2.1 Error Characteristics of Smartphone INS

The first set of experiments were conducted to show the basic error characteristics of a smartphone INS. We used six Android smartphones of different brands, namely Moto-XT882, Moto-Defy, Moto-Milestone, HTC-G12, HTC-G8, and Google-Nexus S, to calculate the moved distance of a car driven in a straight course of about 350 m. To obtain the ground truth of the distance traveled, we sampled the trajectory by manually pressing a button on the phone screen every time the car passed a lamppost. The distances between the lampposts were measured off-line. The *position errors* of the sample points on the INS trajectory are then obtained using the lampposts as reference points. Fig. 1 shows the position errors of the the smartphones as a function of time in a drive. It can be seen that the errors all grow super-linearly with time.

The errors of the inertial sensors include those caused by constant bias, thermo-mechanical white noise, temperature effects, calibration errors, and bias instability. For inertial sensors on the smartphone, the uncorrected bias errors and white noise are typically the primary error sources [10]. The bias largely determines the mean error, while the white noise has a zero mean and mainly impacts on the error variance. If the bias is not very small, then the error will build up very quickly. For instance, the Moto-DEFY smartphone generates an error as high as 30 m in only 30 s. An additional observation from Fig. 1 is that although the vehicle was driven at different speeds (as indicated by the different ending times of the curves), there is no clear correlation between error growth rate and driving speed. The multiple error factors and this observation suggest that it is highly difficult to accurately predict the error in dynamic environments without external references.

Fig. 2 plots the position errors of the Google-Nexus S smartphone, along with standard deviation, averaged over 14 runs for the same course as above. The two lines

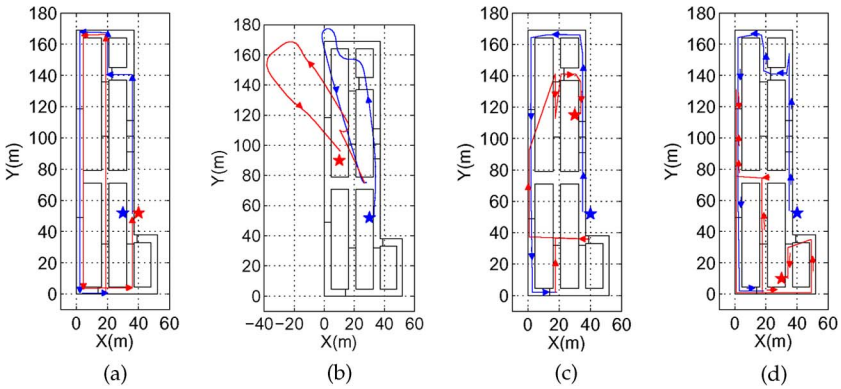


Fig. 3. Trajectories produced by a Google-Nexus S smartphone in a parking garage. Blue/red stars represent the start/end points of the trajectories. (a) Original trajectory. (b) INS trajectory. (c) INS trajectory with nearest-segment (NS) map matching. (d) INS trajectory with HMM map matching.

represent the statistics of two days. Since the original errors are sampled on particular distances, we obtain the errors of particular points in time by polynomial interpolation. From the figure, one can make three observations: 1) the mean errors in both cases exhibit a super-linear trend; 2) the trends are quite different for the two days, reflecting the inertial sensors' sensitivity to external factors such as temperature and driving speed; and 3) The standard deviations widen as time goes on, which can be partly explained by the white noise of the accelerometer that causes the standard deviation of errors to grow at a rate proportional to $t^{3/2}$, where t is the time span of navigation [10].

2.2 Smartphone INS in an Indoor Environment

To see how a smartphone-based INS works in an indoor environment, we conducted experiments in a parking garage (see Fig. 3a) using the Google-Nexus S smartphone, which is equipped with both an accelerometer and a gyroscope. The car was driven along a multi-round path, starting at the blue star symbol and ending at the red star symbol. The total drive took 175 s; the first part (with blue line) took 85 s, and the second (in red) took 90 s. The calculated trajectory is shown in Fig. 3b. It can be seen that the inertial sensors seriously underestimate the true acceleration, producing a much shrunken trace. At the end of the first part of drive, for instance, the position error reaches 70 m. To make things worse, the angular error by the gyroscope causes a wide disorientation of the trajectory.

Next, we examine how much map information can help with navigation. We use a basic map-matching algorithm, known as the *Nearest-Segment (NS) Matching* [4], to pull those off-track parts of the trajectory back to the roads. This algorithm simply projects a calculated position onto the closest road segment. Fig. 3c shows the matching result. From the result, we can see that the algorithm breaks the trajectory and creates random jumps, which are obviously wrong. We also use a more advanced map-matching algorithm [6] based on the HMM model. Compared to the basic NS matching, this algorithm also considers the transition probabilities between road segments, and uses a dynamic programming algorithm to optimally align the trajectory with road segments. Fig. 3d shows the result. Still, we can see some off-course trajectory segments, due to the large error accumulated during the drive.

3 Bump Detection

Bump detection is a process to identify bumps on the road surface. As a salient feature, a bump can cause vibration to a vehicle passing over, with such an intensity that it can be sensed by the accelerometers by checking the z-axis acceleration [1]. Our bump detection process outputs a *detection event* to an upper layer inertial navigation application when it believes it has detected a bump. A detection event occurs when the front wheels of a vehicle hit a bump. The bump will also be hit by the rear wheels, causing a *minor detection event*.

We use a Gaussian Mixture Model (GMM) [7] to detect bumps. The main difference of this model from previous approaches [1] is that the GMM model can adaptively learn the difference between bump-incurred signals and background signals from the road; thus, we do not need to find an *absolute* threshold of signal strength to differentiate bump signals from background ones.

The z-axis acceleration signal from a smooth road surface can be approximated by a Gaussian distribution. Its mean μ and variance σ^2 can be dynamically adjusted with sufficient and consistent evidence from the environment. In our context, the space of smooth roads is far more than that of bumps. Hence, a signal X will be regarded as a bump signal if its strength deviates significantly from the mean of smooth-road signal strength. More specifically, X is a bump signal if the absolute difference between X and μ is larger than the standard deviation multiplied by a predefined threshold V_{th}, that is

$$|X - \mu| > V_{th} \times \sigma.$$

Otherwise, X will be regarded as smooth-road signal, and the mean and variance will be updated as follows:

$$\mu' = (1 - \lambda(X))\mu + \lambda(X)X \tag{1}$$

$$\sigma' = \sqrt{(1 - \lambda(X))\sigma^2 + \lambda(X)(X - \mu)^2} \tag{2}$$

where $\lambda(\cdot)$ is the probability density function extracted from the past samples.

The above solution is called the single Gaussian model, which roughly approximates the background signal distribution. For improved accuracy, we have found that a better

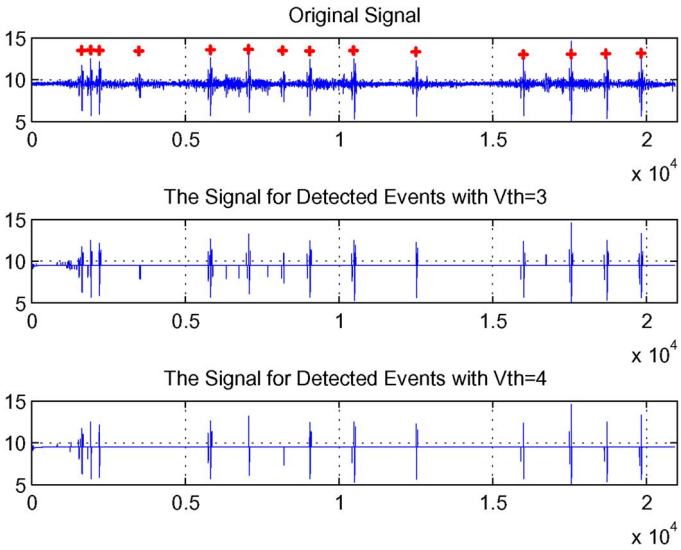


Fig. 4. A stream of acceleration signals and events detected by the GMM model with different V_{th}.

approximation is provided by GMM [7], which uses K Gaussian distributions to model background signals. The complexity stems from the fact that a signal from the accelerometers is the outcome of a combination of various forces and errors, which independently follow their own Gaussian distributions. The GMM model assigns a weight ω_k to each of the K Gaussian distribution, provided $\sum_k \omega_k = 1$. Each new evidence (new signal from a smooth road surface) causes each ω_k to be updated as follows

$$\omega'_k = (1-\alpha)\omega_k + \alpha F_k$$

where α is the learning rate, and F_k is 1 for the matched Gaussian model and 0 for the remaining models. In fact, F_k reflects whether the current signal matches model k, and ω_k is effectively the probability that previously-sensed signals match model k, with an exponential window on historic values. The weight of the matched model will increase, and the remaining will decrease at the rate of $\frac{1}{\alpha}$. Of course, the updated weights should be normalized accordingly. For more details of GMM, the reader is referred to [7].

The mean and standard deviation of the matched model are updated according to equations (1) and (2), and those of the unmatched models remain the same. In our implementation, the initial values of w_k, μ, and σ^2 are set to 0, 0, and 45, respectively. Fig. 4 shows a stream of acceleration signals and detection events by the GMM model. A threshold $V_{th} = 4$ is found to strike a nice balance between false negative rate and false positive rate.

4 Bump Matching Algorithm

In this section, we introduce the hidden Markov Model, and then describe how to model our particular problem. The modeling will use a *bump graph* that represents the connectivity of the bumps as well as distances between them. Next, we try to map a detection event to a bump using this HMM model, hoping to find out which bump the vehicle is currently located at during the drive.

4.1 HMM Preliminary

An HMM is a Markov process with a set of unobserved (hidden) states and observable states. Transitions between hidden states are governed by a set of *transition probabilities*, while an observable state is generated from the hidden states with an *emission probability distribution*. Given some input, an HMM traverses its states to produce its output: the observable state emitted at each hidden state. While one can observe the output (sequence of observable states), the sequence of hidden states is unknown. The target is to find out the sequence of hidden states that the process has gone through.

In the bump matching context, each hidden state corresponds to a bump, and each observable state corresponds to a bump detection event. Here, the transition probability from bump i to bump j is the probability of a vehicle finding its first detected bump to be j after passing bump i. The emission probability of a detection event is the probability of the vehicle having indeed moved the calculated distance from the last matched bump, conditioned on the last detection event being matched to that bump. In this context, our problem then is to determine the most probable bump given a recent detection event. For higher accuracy, we want to determine the most probable sequence of bumps for a sequence of recent detection events.

4.2 Bump Graph

The bump graph is a directed and weighted multi-graph, where a node represents a bump, and a directed edge represents a direct path (i.e., without intermediate bumps) between two bumps, with a weight equal to the length of that path. Two nodes have multiple connecting edges when there exist multiple direct paths between them. If a bump is deployed on a two-way road, then it is represented as two nodes on the bump graph. Fig. 5b shows an example of bump graph abstracted from the road topology in Fig. 5a. The nodes are labeled $1, 2, \ldots, n$. The kth edge between two nodes i and j has weight d_{ij}^k.

4.3 HMM for Bump Detection

We now explain the transition probabilities, which determine the possible sequences of bumps that have been passed and successfully detected, and the emission probabilities, which determine the most likely bump for a particular detection event. Assume that the probability of false positives between nodes i and j is $\alpha = \min(1, d_{ij} \cdot \alpha')$, where α' is the false positive probability for a unit distance, and the probability of false negative (missing a bump) is β.

4.3.1 Transition Probabilities

Fig. 5c shows a state transition diagram of the states. The diagram is also a multi-graph whose nodes are the same as those in the bump graph, and whose edges ϵ_{ij}^k can be of two types: single-hop edge and multi-hop edge. A single-hop edge corresponds to an edge in the bump graph, while a multi-hop edge corresponds to a viable path of at least two hops on the bump graph. Two nodes may have multiple connecting paths on the bump graph; these paths form multiple edges of the same direction between the two nodes.

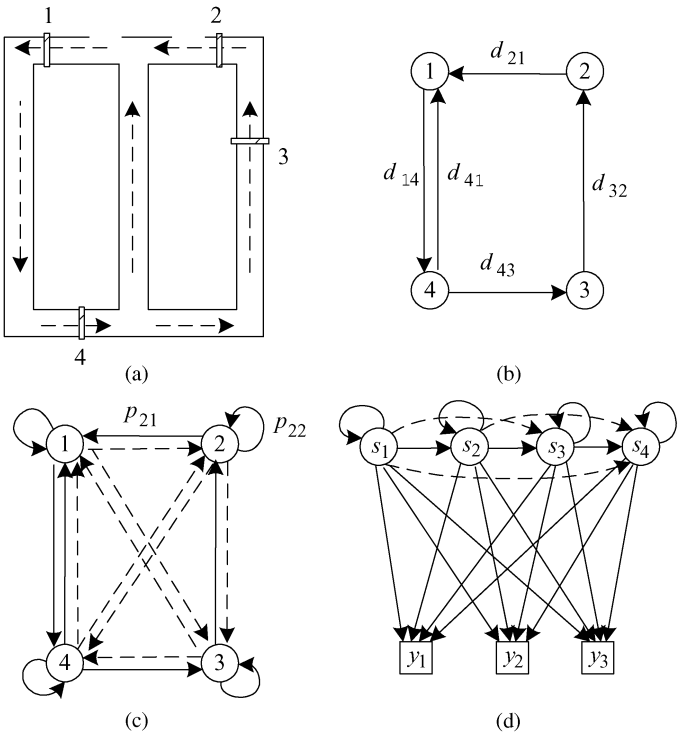


Fig. 5. (a) A garage map with four bumps (small shaded blocks). (b) The bump graph, where a node corresponds to a bump. (c) Diagram of bump state transition, where an arc represents a single-hop (solid line) or multi-hop path (dashed line). (d) HMM state diagram with emission probabilities. In the diagram, s_i's represent the hidden states (i.e., bumps) and y_i's the observable states (detection events). The purpose is to infer the sequence of bumps traversed from the sequence of detection events.

Now we assign transition probability to the edge ϵ_{ij}^k. For a single-hop edge, its transition probability is

$$p_{ij}^k = \left(1 - d_{ij}^k \alpha'\right) \cdot (1 - \beta),$$

for a multi-hop edge, we assume that its corresponding path on the bump graph has length D_{ij}^k and traverses h_{ij}^k intermediate bumps. Then

$$p_{ij}^k = (1 - D_{ij}^k \cdot \alpha') \cdot \beta^{h_{ij}^k} \cdot (1 - \beta).$$

Every node in the state diagram also has a self loop representing the residual probability, that is

$$p_{ii} = 1 - \sum_{j \neq i} \sum_k p_{ij}^k.$$

4.3.2 Emission Probabilities

Fig. 5d shows an example of HMM state diagram corresponding to the bump graph, where s_i represents a hidden state and y_i represents an observable state. Upon a bump detection event, the INS needs to determine which bump it has encountered. Let bump i be the last matched bump before the event occurs, and d_{acc} the moved distance from bump i, calculated from the acceleration measurement. Then, the emission probability is $p_{ij}^k \times p_{acc}$, where p_{acc} is the probability of the error being the difference between the calculated and true distances. In practice, the latter is the error probability density function integrated over a unit distance centered at the error value.

Typically, the errors are primarily caused by two factors: uncorrected bias and white noise [10]. The former is subject to bias instability, so for a particular time span t of navigation, the produced error can be modeled by a Gaussian distribution whose mean depends on t; the latter has a mean equal to zero, and can also be modeled with a Gaussian distribution whose variance depends on t [10]. Because of the independence of these two factors, we model the aggregate error using a Gaussian distribution $N(\mu_{ins}(t), \sigma_{ins}^2(t))$, where $\mu_{ins}(t)$ and $\sigma_{ins}^2(t)$ are the unknown mean and variance functions. These two functions can be estimated with a bump-aided error sampling and interpolation method (to be detailed in Section 5). In the beginning, we simply set $\mu_{ins}(t) = 0$ and $\sigma_{ins}^2(t) = 0$. Due to the limited number of bumps the vehicle encounters during navigation, the estimated $\mu_{ins}(t)$ and $\sigma_{ins}^2(t)$ may be very rough. Fortunately, our experimental study shows that these functions do not have to be precise to allow accurate bump matching, because of the very high success rate of bump detection that dominates the emission probability. This means that when a detection event occurs, it is very likely that the vehicle has indeed hit a bump, so the matching can be done with fairly high confidence, even without further judgment from the error probability of the INS.

Finally, we use the angular information of the gyroscope to refine the emission probability. Our navigation system samples the gyroscope and maintains a current angle A_{cur}, assuming a zero initial angle. Let the difference between the heading directions at B_{last} and B_{cur} be A_{true}. Then the final emission probability is multiplied by an additional factor $1 - |A_{cur} - A_{true}|/\pi$.

4.4 Online Bump Matching

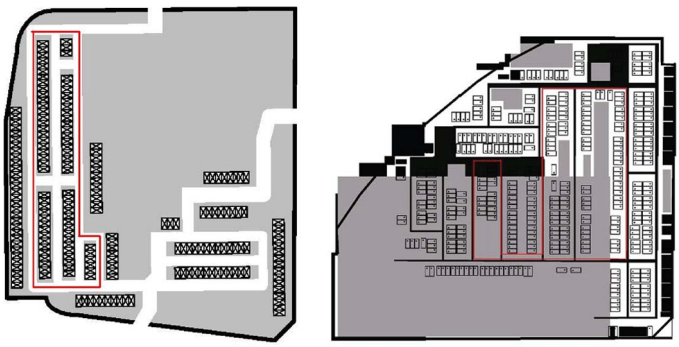
Given a detection event, it has an emission probability for every bump. The naive way of determining the current bump is to pick the bump with the largest emission probability, and discard all other possibilities. This near-sighted strategy can be improved by retaining all (or the top few) emission probabilities, and then considering a *joint match* for a sequence of most recent detection events. Generally, we consider a sliding window of recent detection events, and solve for the best joint match, using the Viterbi dynamic programming algorithm [9].

5 Positioning

The bump matching algorithm allows the INS to reset its current position, called a *restart position*, upon detection of a bump. In this section we describe how to calculate position based on the restart position. We call the navigation process between two successive detection events a *round*. The beginning time of a round is defined as the time when the vehicle's front wheels are determined to be right on top of a bump.

5.1 Restart Speed Estimation

In additional to the restart position, we also need to estimate the *restart speed*. The reason for using a new estimate instead of the integration result from historic data is the same as using a restart position, that is, to get rid of the accumulated errors.



(a) (b)

Fig. 6. Maps of the parking garages. The areas enclosed by red lines are tested areas. (a) Coastal City, 200 m × 180 m. (b) Central City, 160 m × 160 m.

Assume that the front and rear wheels get on top of a bump at times t_f and t_r, respectively, and that the vehicle has a speed v_f and v_r at the two points in time, respectively. Further assume the *wheelbase*, namely the distance between the centers of front and rear wheels, is known in advance. Based on v_r, we double integrate the acceleration readings for the time interval (t_f, t_r). The result should be approximately equal to the wheelbase. With this equation, we can obtain an estimate of v_r. Based on this estimate, we then integrate the accelerations for (t_f, t_r) to obtain an estimate for v_f, which is the desired startup speed.

5.1.1 Wheelbase determination

Wheelbase is a key parameter for some common vehicle size class systems, in which vehicles of the same class have a small difference in wheelbases. For example, in the U.S. system today, Mid-size cars usually have wheelbase between 2.68 and 2.79 m [8]. For simplicity, we can use the median wheelbase for a vehicle class. To obtain the wheelbase of a particular vehicle, the user can choose to enter/choose the vehicle's class, with which the vehicle's wheelbase can be approximately determined. Alternatively, if an input from the user is not available, we use the default value, 2.70 m. The error in the wheelbase estimate will result in a position error linear with time. In the supplementary document, we show that this error impacts on positioning accuracy moderately, but the Bumping method still performs far better than schemes without startup speed estimation, due to its avoidance of historic errors from previous rounds.

5.2 Positioning During Travel

With restart position and speed determined, the distance moved can be calculated by double integrating the acceleration measurements. To navigate correctly, we also use the gyroscope on the smartphone to infer the moving direction. The distance and direction together produce a meaningful trajectory, which is further fitted into a map topology using a map matching algorithm. We have tested the positioning system in combination with two map matching algorithms: the NS matching algorithm, and the HMM-based matching algorithm. The latter takes into account the transition constraints between road segments, so as to produce better results. This will be demonstrated in Section 6.

5.2.1 Error compensation

The matched bumps also provide an opportunity to predict the accelerometer's errors. Assume the latest two matched bumps have a distance of x, and the calculated distance between the two corresponding detection events is $\widehat{x}$, then we obtain an *error sample* $\langle t, err \rangle$, where t is the time span of distance calculation and $err = \widehat{x} - x$. These samples are saved to help us estimate the accelerometer's mean error and error variance for an arbitrary t via polynomial interpolation/extrapolation. The estimated errors will thus be compensated in real time by the navigation process in the next round. These error samples are also used to define the error probability density function $N(\mu_{ins}(t), \sigma^2_{ins}(t))$ required by the emission probabilities in Section 4.

5.3 Automatic Startup Positioning

INSs invariably assume a known initial position and speed, and this is the reason for its being used along with other systems such as GPS and Wifi [4], rather than as an independent system. In our context, however, the initialization condition may not be satisfied. For example, in a parking garage, the need of navigation often arises only when the user loses direction. The fact that the user is unaware of his/her current locations makes it infeasible for the INS to get an input from the user, therefore the navigation will simply be unable to start.

The bumps provide a possible solution for the problem of automatic startup positioning. Given a sequence of M recent detection events, we can obtain the joint emission probability p_i of such a sequence being matched to a particular sequence S_i of bumps $\langle B_1, B_2, \ldots, B_M \rangle$. We say that the current detection event is matched to the bump B_M with probability p_i, with respect to S_i. Considering all possible S_i that contains B_M as its last element, we have the probability of the current detection event being matched to B_M as

$$B_M\text{'s matching probability} = \sum_{\mathrm{i}} \mathrm{p_i}.$$

We pick the two bumps with the highest matching probabilities, say p^1_{match} and p^2_{match} $(p^1_{match} \geq p^2_{match})$, as the *candidate startup bumps*. As the vehicle moves along, the candidate startup bumps and their matching probabilities are updated. Only when p^1_{match}/p^2_{match}, called the *distinguishing ratio*, exceeds a predefined threshold η (e.g., 1.5), called the *distinguishing threshold*, the candidate startup bump with the larger matching probability is taken as the current bump, and its position is taken as the vehicle's current position. At this time the navigation starts up. The startup delay and positioning accuracy depend on M and η. This is demonstrated in Section 6.

6 Evaluation

We chose tree parking garages as experimental scenarios, where the evaluation results of the third parking garage refer to the supplementary document. Fig. 6 shows the sketch maps of these garages, and Table 1 lists the specifics of experimental settings. In each scenario, a Volkswagen Jetta was driven along planned routes. A Google Nexus S

TABLE 1
Experimental Scenarios

Scenario (garage)	Drive length	Num. Repetitions	Num. bumps	Num. events
Coastal City	760m	8	11	14
Central City	494m	4	9	9

smartphone fixed on the car measured the motion/angular accelerations with a sampling rate of 50 Hz. Our program calculated positions based on these data. For the HMM used in the bump matching algorithm, the false negative/positive rates of bump detection are set to be 5 percent and 0 respectively, according to the statistics of our empirical data. We consider seven schemes:

1. *InertialOnly*, in which the positioning is solely based on the inertia law;
2. *Inertial+NS*, which combines InertialOnly with the NS map matching algorithm [4];
3. *Inertial+HMM*, which combines InertialOnly with the HMM map matching algorithm [6];
4. *Bump*, which combines InertialOnly with our bump matching algorithm (i.e., restart position/speed estimation), without bias correction;
5. *Bump+NS*, which combines the Bump scheme with the NS map matching algorithm;
6. *Bump+HMM*, which combines Bump with the HMM map matching algorithm;
7. *BumpFull*, which combines Bump+HMM with error compensation.

For a navigation process, we sample its position at each detection event, and obtain the absolute difference between the calculated position (before bump matching) and the true bump's position, where the association is determined by manual annotation. These differences give us a set of error samples of the whole navigation process. In our experiments, the detection events are identically matched to the same sequence of bumps, so we are able to obtain an average position error for each event over multiple drives.

6.1 Positioning Accuracy

Figs. 7 and 8 show the trajectories of the smartphone INS in the two scenarios. Taking the Coastal City case as an example, we see that the InertialOnly scheme totally misses the track, especially in the second part (red line). This is because of the negative error bias of the inertial sensors in the phone. The bias can be more clearly seen in Fig. 2, where the INS produces a negative error of more than 80 m within just one minute of driving. It is thus no surprise that in the Coastal City scenario, the INS produces a trajectory of a much smaller size, with an average position error of 39.52 m. The map matching algorithms can help with the first part of the drive, but fails to work for the second part, in which the error is too large to permit a reasonable guess about the vehicle's true positions, even with full information of the map.

With the aid from the bumps, the situation becomes very different. Due to the high success rate of bump detection

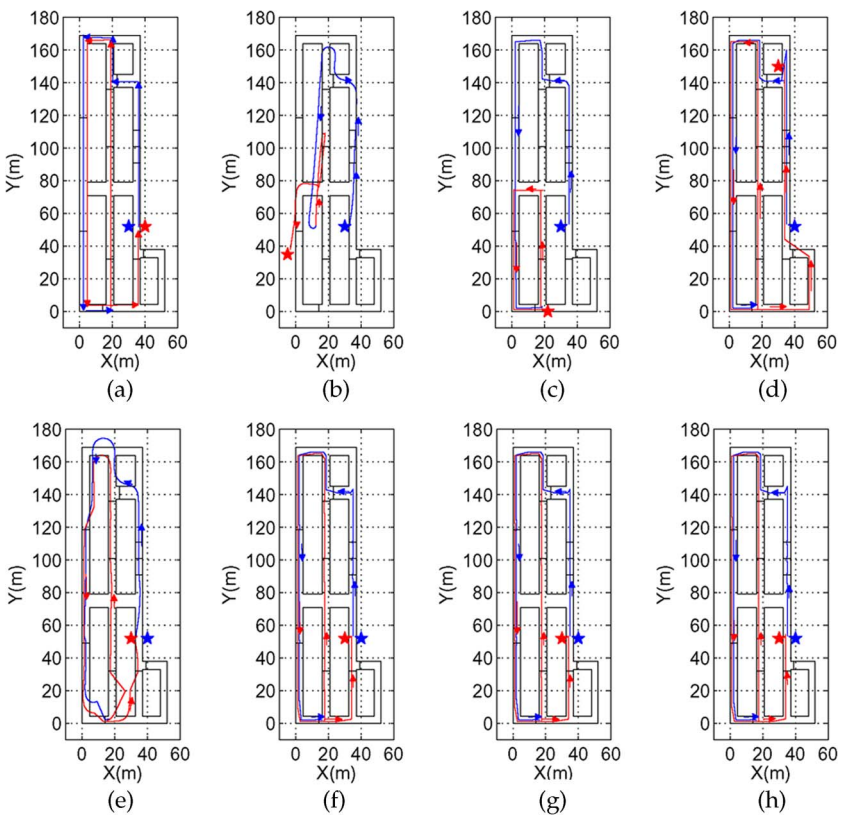


Fig. 7. Trajectories of a drive in the Coastal City garage produced by a Google-Nexus S smartphone. (a) Original. (b) InertialOnly. (c) Inertial+NS. (d) Inertial+HMM. (e) Bump. (f) Bump+NS. (g) Bump+HMM. (h) BumpFull.

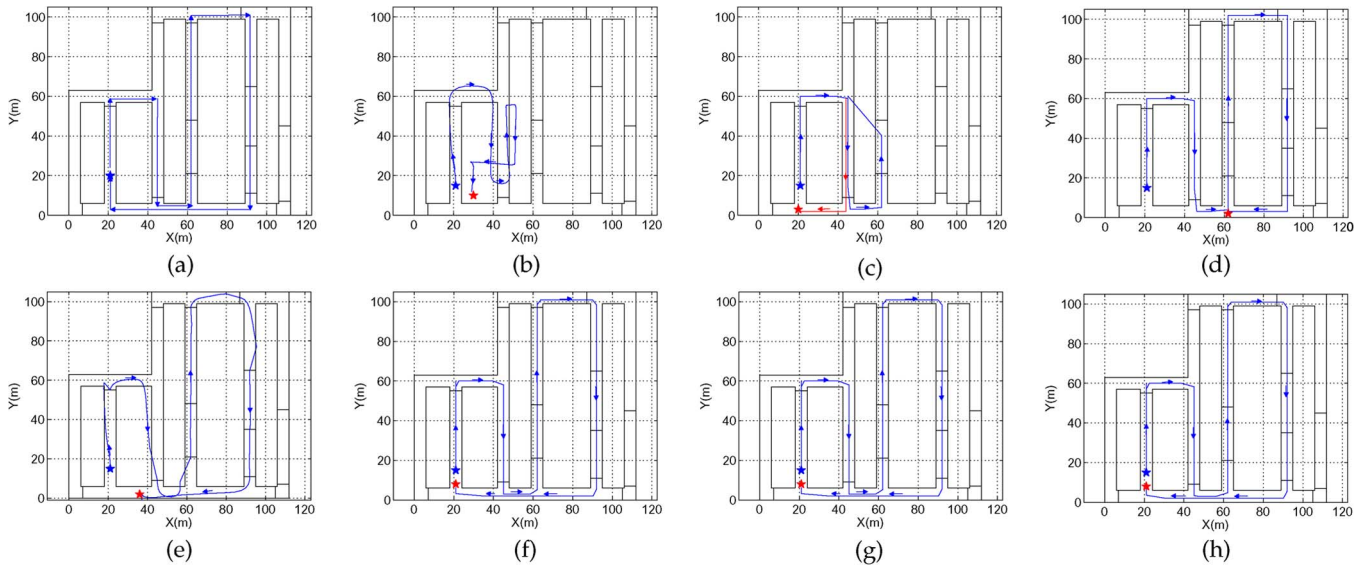


Fig. 8. Trajectories of a drive in the Central City garage produced by a Google-Nexus S smartphone. (a) Original. (b) InertialOnly. (c) Inertial+NS. (d) Inertial+HMM. (e) Bump. (f) Bump+NS. (g) Bump+HMM. (h) BumpFull.

($\geq$ 95 percent), the bump matching algorithm performs very well in resetting the navigation process, and allows the INS to restart its calculation. This can be clearly seen, for example, in Fig. 7e, where the bumps at $Y \approx 120$ m and $X \approx 0$ m successfully correct the position errors. The Bumping method, in effect, limits the accumulation of errors within successive bumps. The map matching algorithms further refine the trajectories, making them align better with the road topology. Little difference is observed between the two map matching algorithms.

Fig. 9 shows the average position error of each detection event (before bump matching) over multiple drives along the same route. In the Coastal City case (Fig. 9a), a total of 8 drives were performed along a route. Roughly speaking, the position error of the InertialOnly scheme tends to grow with time, reaching 39.52 m on average, and 85.20 at maximum. In contrast, the position error of the various bump-aided methods remains constantly below 11 m. In general, the BumpFull method works best among the various bump-aided schemes, producing an overall average error of 4.79 m and a maximum 12.52 m. The positioning accuracy is very encouraging, since the average error is merely the width of two parking lots in a row.

6.2 Startup Positioning

Automatic startup position estimation tries to match a window, called a *matching window*, of M recent detection events to a sequence of M bumps. For every bump, there is a matching probability p_{match} corresponding to the latest detection event. A metric, distinguishing ratio, is maintained as the ratio of the largest matching probability to the second largest one. When this ratio exceeds a threshold η, the bump with the largest matching probability is taken as current bump and its position as current position. There are three cases for the matching outcome: 1) *Correct match*, which means that the matched bump is correct; 2) *Mismatch*, which means that the matched bump is wrong; and 3) *Matching failure*, which means that the distinguishing ratio is no greater than η.

To obtain the overall performance of this method, and in particular, the impact of the window size M and the distinguishing threshold η, we drove the car along some random routes in the Coastal City and Central City garages. For every detection event, we backtrack M detection events in the reverse direction of the route to

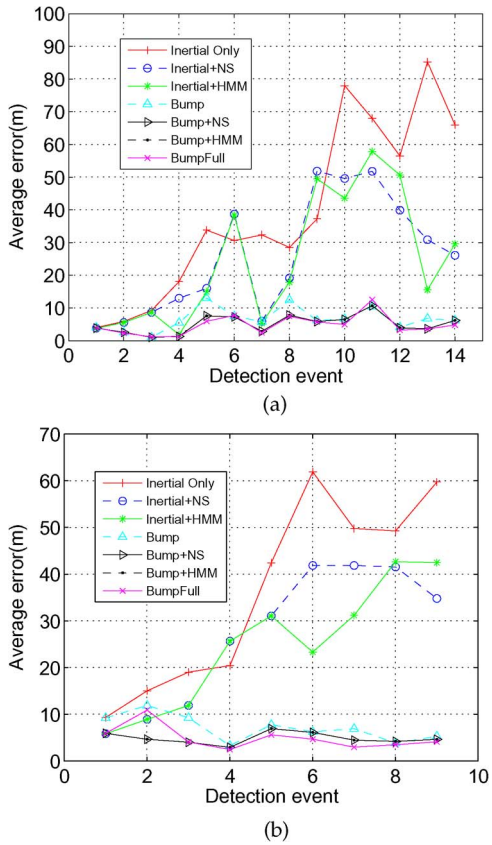


Fig. 9. Average position errors over multiple drives along planned routes. The overall average position errors (for all detection events) are 4.79 m and 4.24 m respectively. (a) Coastal City. (b) Central City.

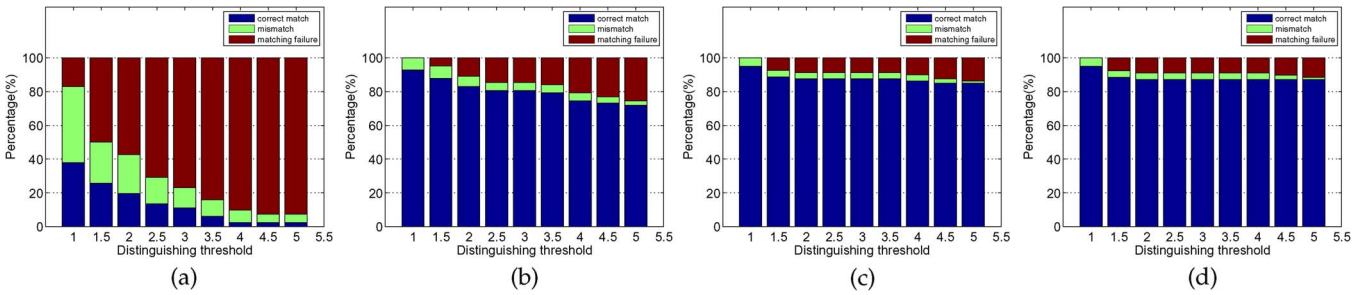


Fig. 10. Startup matching results for varying matching window size M and distinguishing threshold η. (a) $M = 2$. (b) $M = 3$. (c) $M = 4$. (d) $M = 5$.

create an event window. This can be viewed as if the user started navigation at the time of the earliest detection event in that window. We want to show what is the user's chance of a successful startup positioning, supposing the user launches the INS from a random bump and uses at most M detection events (which reflects the delay for the navigation system to start working).

A total of 78 such event windows are created, representing 78 random starting paths. Fig. 10 shows the correct match rate, mismatch rate, and matching failure rate for varying M and η. From the figure, we can make two observations: 1) as M grows, the correct match rate increases, and both of the other two rates decrease. This is no surprise, since a longer event window makes a stronger fingerprint that allows more accurate matching. From $M = 4$ to $M = 5$, the difference is too small to be noticeable, suggesting the the matching window size may not need to be very large. 2) For a particular M, a small η means more tolerance to matching ambiguities, hence a lower matching failure rate, but a higher mismatch rate. As η grows, the matching algorithm becomes more demanding, and so causes an increased matching failure rate and a smaller mismatch rate. For the sake of maximum user experience, the mismatch rate should be minimized. Therefore, for the two parking garages, $M = 4$ and a medium to high η appear to be the best choices.

6.3 Impact of Bump Detection Errors

Though our empirical data show a very low false negative rate (FN $\leq$ 5 percent) and false positive rate (FP $= 0\%$) for bump detection, we want to see how robust the bump matching algorithm is to higher FN/FP rates. We simulate various FN/FP rates by randomly removing detection events from, or inserting fake detection events into the events stream, and compare the positioning accuracy. The maximum rates simulated are 30 percent, which are very high for realistic indoor environments.

Fig. 11 gives the average position errors of the Coastal City and Central City scenarios under different combinations of FN/FP rates. We can see an obvious impact of these rates on position accuracy. In general, the higher the rates, the larger the average errors. However, even for FN $= 20\%$ and FP $= 30\%$ (or vice versa), the errors are still much smaller than the Inertial+HMM method, which is the best among the schemes without using bumps. For the Bumping method, though the missed or mistaken bump detections cause misalignment of the vehicle's position with the bumps, the HMM model shows its strong ability to avoid or rectify mismatches. Fig. 11c shows an example in the Coastal City scenario, in which the bump detection algorithm misses two bumps (enclosed by dashed red boxes) and mistakenly detects a bump that does not exist (light green box). For the two missed detection events, the distance and angular information of the inertial sensors plays a critical role in suppressing the possibility of mismatch. For example, for the first missed detection (top red box), a naive bump matching scheme may match the next detection event to the bump in the red box when the vehicle is really on the next one, resulting in an error equal to the length of the U-shaped road segment between the real and envisioned bumps (about 80 m). This error may persist for the rest of the navigation. With HMM matching, this mismatch can be avoided, because the emission probability considers the probabilities of both distance error and angular error. The same happens to the second missed bump, shown in the lower red box.

For the mistaken detection event (green box), the bump matching algorithm faces two possibilities: either the vehicle has indeed moved for the distance between the two real bumps before and after the event spot, or a false positive occurs. The algorithm finds that the former case is more probable, thus, wrongly matches the detection event to the next real bump on the route. Fig. 11d shows the position error at each detection event. As can be seen, the missed detections do not affect the error because they have both been taken care of by the HMM model; in contrast, the seventh event is a mistaken detection, and so is matched to the next real bump on the route, resulting in a big error. This error, however, is gradually corrected in subsequent events, especially after the route makes a U-turn.

7 Conclusion

In this paper, we show that the widely deployed speed bumps in parking garages provide opportunities to improve the accuracy of the INS. The proposed Bumping method first detects the bumps using a Gaussian mixture model, then maps the detection events to the right bumps with high accuracy, using a Hidden Markov Model. Field experiment results in three parking garages show that the proposed method significantly improves the positioning accuracy compared to the basic inertial navigation method, and is able to work without a known initial position most of the time. In the future, we will consider the possibility of installation of extra speed bumps in improving positioning accuracy, taking into account of issues of both navigation experience and safety.

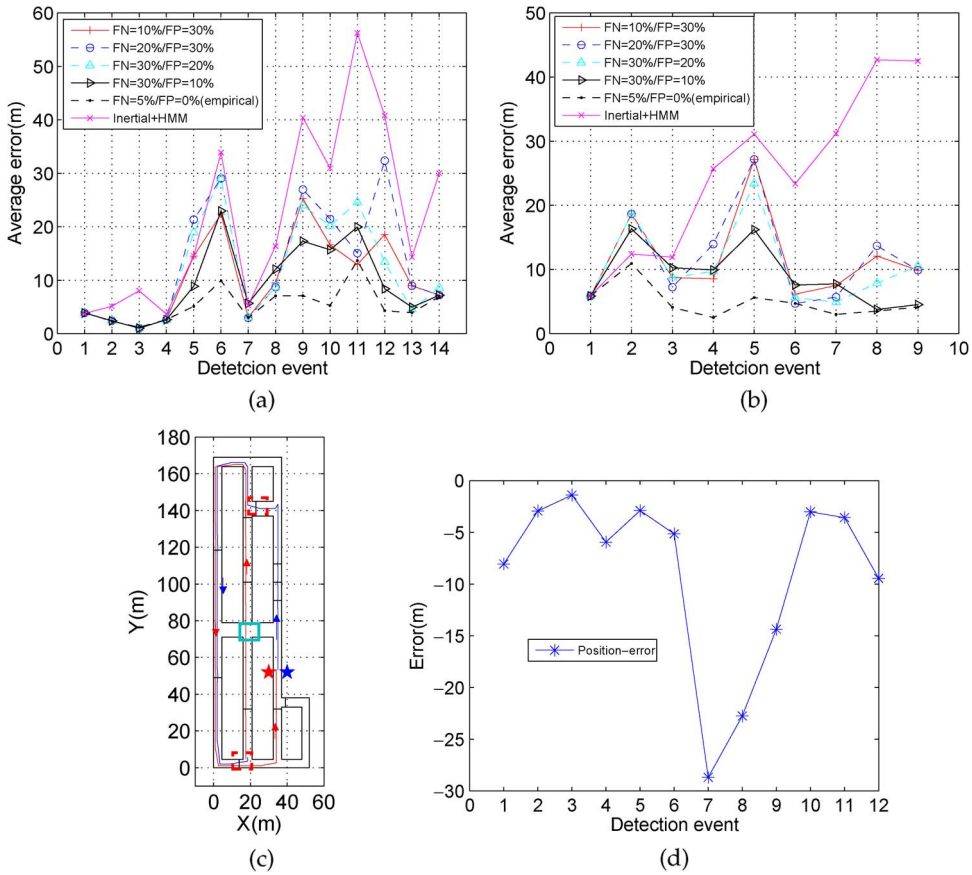


Fig. 11. Impact of bump detection accuracy on positioning accuracy. (a) Position error in Coastal City. (b) Position error in Central City. (c) An example of false negatives/positives. (d) Position error for each detection event in (c). FN refers to false negative (missed detection) rate, and FP refers to false positive (mistaken detection) rate. The positioning scheme is BumpFull, unless otherwise stated.

ACKNOWLEDGMENT

G. Tan's work was supported by the National Science Foundation of China (NSFC) under Grant 61103243, Youth Innovation Promotion Association, Chinese Academy of Sciences, the Ministry of Science and Technology 863 Key Project No. 2011AA010500, Shenzhen Overseas High-level Talents Innovation and Entrepreneurship Funds KQC201109050097A, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry). M. Lu's work was supported in part by the NSFC under grant 60903222.

REFERENCES

[1] 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 *Proc. ACM MobiSys*, 2008, pp. 29-39.

[2] D. Gusenbauer, C. Isert, and J. Kröche, ''Self-Contained Indoor Positioning on Off-the-Shelf Mobile Devices,'' in *Proc. Int. Conf. IPIN*, 2010, pp. 1-9.

[3] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson, ''Cooperative Transit Tracking Using GPS-Enabled Smart-Phones,'' in *Proc. ACM SenSys*, 2010, pp. 85-98.

[4] A. Thiagarajan, L. Sivalingam, K. LaCurts, S. Toledo, J. Eriksson, S. Madden, and H. Balakrishnan, ''VTrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones,'' in *Proc. ACM SenSys*, 2009, pp. 85-98.

[5] Inertial Navigation Systems. [Online]. Available: http://en.wikipedia.org/wiki/Inertial_navigation_system

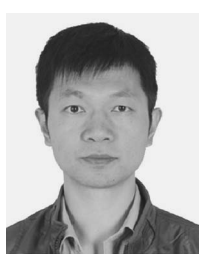
[6] P. Newson and J. Krumm, ''Hidden Markov Map Matching Through Noise and Sparseness,'' in *Proc. 17th ACM SIGSPATIAL GIS*, pp. 336-343.

[7] C. Stauffer and W.E.L. Crimson, ''Adaptive Background Mixture Models for Real-Time Tracking,'' in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 1999, pp. 2246-2252.

[8] Vehicle Size Classes: From a Minicompact to a Full Size Car. [Online]. Available: http://www.syl.com/travel/vehiclesizeclassesfromaminicompacttoafullsizecar.html

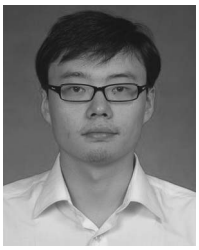
[9] A.J. Viterbi, ''Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm,'' *IEEE Trans. Information Theory*, vol. 13, no. 2, pp. 260-269, Apr. 1967.

[10] O.J. Woodman, ''An Introduction to Inertial Navigation,'' Comput. Lab., Univ. Cambridge, Tech. Rep. UCAM-CL-TR-696, 2007.

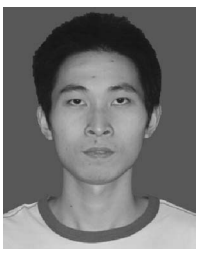


Guang Tan received the PhD degree in computer science from the University of Warwick, U.K., in 2007. He is currently an Associate Professor at Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences, China, where he works on the design of distributed systems and networks. From 2007 to 2010 he was a postdoctoral researcher at INRIA-Rennes, France. He has published more than 30 research articles in the areas of peer-to-peer computing, wireless sensor networks, and mobile computing. His research is sponsored by National Science Foundation of China and Chinese Academy of Sciences. He is a member ACM, CCF, and IEEE.

Mingming Lu received the PhD in computer science from Florida Atlantic University, USA in 2008 is an Associate Professor of Central South University (CSU). Before he joined CSU, he worked for Citrix System Inc. His research areas include opportunistic sensing, compressive sensing, routing algorithm, network coding, localization, and wireless/mobile sensor networks, etc. He has published over 30 scholarly journal and conference papers. He was a TPC member of IEEE INFOCOM 2011 and reviewers of numerous international journals and conference such as TPDS, TMC, ICNP, MobiHoc etc. He is a member of the IEEE.



Fangsheng Jiang received the BS and ME degrees from Harbin Institute of Technology, China in 2009 and 2012, respectively. He is currently a software engineer at Baidu, Inc. China. He works in the areas of mobile computing and wireless networks.



Kongyang Chen received the BE degree in electronic information science and technology in 2008, the M.E. degree in electronic science and technology in 2011, both in Central South University. He is currently a Research Associate in Shenzhen Institutes of Advanced Technology, Chinese Academy of Science, China. His research interests include various topics in the application and protocols of wireless networks. His current research focuses on mobile sensing and GPS optimization.

Xiaoxia Huang received the BE and ME degrees in electrical engineering from the Huazhong University of Science and Technology, China, in 2000 and 2002, respectively, and the PhD degree in electrical and computer engineering from the University of Florida in 2007. Currently, she is a Associate Professor at the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. Her research interests include wireless sensor networks, wireless communications, and mobile computing. She is a member of the IEEE.

Jie Wu is Chair and Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University. He was a Program Director at the National Science Foundation and Distinguished Professor at Florida Atlantic University. His research interests include wireless networks, mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. His publications include over 600 papers in scholarly journals, conference proceedings, and books. He has served on several editorial boards, including IEEE TRANSACTIONS ON COMPUTERS and Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, and IEEE DCOSS 2009 and was the program co-chair for IEEE INFOCOM 2011. He served as general chair for IEEE ICDCS 2013. He was an IEEE Computer Society Distinguished Visitor and the chair for the IEEE Technical Committee on Distributed Processing (TCDP). Currently, Dr. Wu is an ACM Distinguished Speaker and a Fellow of the IEEE. He is the recipient of 2011 CCF Overseas Outstanding Achievement Award.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.