Vehicular Route Identification Using Mobile Devices Integrated Sensors

Luca Bedogni*, Luciano Bononi*
*Department of Computer Science and Engineering, University of Bologna, Italy
{luca.bedogni4,luciano.bononi}@unibo.it

***Abstract*—Location based services are commonly used by several mobile applications and services, to provide content related to the area in which the user is located. This enables services such as navigation, particularly useful for vehicular applications, though possibly exposing private information about the user, which has to explicitly grant the location permission. However, smartphone have also many other sensors off the shelf, which currently do not require any permission to be used, and may be leveraged to track the users movements, hence the location, thus raising potentially serious privacy issues. In this paper we present a study which shows that by analyzing data obtained through the accelerometer and the magnetometer, it is possible to achieve less than 50 meters of localization accuracy even for long journeys, and 95% of accuracy on the road identification.**

I. Introduction

Nowadays, location based services are popular in different domains, as they offer the possibility to contextualize the application in use, hence making it more related to the current user context. Modern smartphones can utilize the GPS, which is often used as the standard solution to localize the device, given its precision and wide availability outdoors. Due to the possible private data which can be extracted from raw GPS measurements [12], currently Android applications need to explicitly ask for the permission to use the GPS. This is to prevent applications which may acquire the GPS data for malicious purposes, for instance understanding whenever the user is not at home. However, modern smartphones also carry several other integrated sensors, such as accelerometers, gyroscopes, magnetometers and barometers, to name few, which may be used to understand the context in which the device is located. Closer to the topic of this study is the possibility for a mobile device to use data obtained through the internal sensors, which currently do not require any explicit permission on Android devices, to track the user positions whilst driving. Clearly this would constitute a serious privacy issue, as users may have their private information exploited by malicious applications without being aware of it.

It has already been shown that through smartphone integrated sensors it is possible to understand movement patterns, which can be mapped to different activities performed by the user [4]. Focusing on the vehicular scenario, accelerometers can be used to understand accelerations, which correlated with time provide space and magnetometers indicate the direction the smartphone is pointing at. By building specific models which leverage the sensor dynamics, it is possible to build the so-called Dead Reckoning Inertial navigation, which by analyzing sensor measurements can give the direction and space traveled by a device, though they have been mostly used for pedestrian tracking and indoor localization [13]. However, simply relying on Dead Reckoning inertial navigation systems can lead to inaccurate estimates, as they can carry cumulative errors due to inaccuracy in the measurement. Therefore, inertial systems are typically corrected by sporadic GPS readings, which can reduce the number and entity of the error [1], eventually providing a more accurate localization.

In this paper, we present a study aimed at understanding the potentials of such as system and subsequently the possible privacy issues which may arise from the use of inertial sensors to understand tracks traveled by users whilst driving a car. By knowing the initial position of the user, we exploit data from the accelerometer to understand the distance traveled and from the magnetometer to derive turn events. All the used sensors currently require no permission on Android devices, hence any installed application can register to receive value updates from them. As we previously stated, we show that only relying on dead reckoning systems may and up in possible high errors, as inaccuracies sum up from the beginning to the end of the journey.

Thus, to improve the tracking accuracy, we also use real map data obtained through OpenStreetmap[1], which contains the road definition with lengths and turn angles. The general idea of our proposal is to obtain lengths and angles of road traveled, and then match them on a real road network. This empowers our proposed system to correct Dead reckoning errors at the end of each road segment, hence reducing the problem of accumulating errors throughout the entire journey.

Our performance evaluation is twofold: at first, we compare the results of our system against classical Dead Reckoning systems and GPS with real data obtained from a custom Android application developed within the scope of this study. The aim is to match the final point in which the vehicle stopped, by recognizing subsequent intermediate roads. Moreover, we also perform a simulative study on several cities worldwide, to understand the recognition boundaries of this systems depending on the size and area in which the tracking has to be performed. We show that in most cities worldwide it is possible to uniquely identify paths with at least 10 turns turns, even in big cities with many roads. Moreover, it exists a rather evident difference in performing such recognition

[1]http://www.OpenStreetmap.org

in the U.S. or in Europe, as the latter have more diverse road deployments, which make the recognition easier as road angles and lengths differ more compared to the former, which resembles more a Manhattan grid deployment. In practice, for Europe as little as 4 turns uniquely identify paths regardless of the city size, while for the U.S. it takes at least ten turns to achieve the same result.

The rest of this paper is organized as follows: Section II presents related work from literature; Section III details our proposal and outlines the algorithm used; Section III-C details the implementation of our proposal; Section IV presents results from the performance evaluation, and Section V concludes this work.

II. Related Work

In this section we present related work from literature, concerning the use of mobile devices integrate sensors for navigation, and the recognition of activities exploiting sensor data, commonly known as Transportation Mode Detection.

The need to navigate and route people and objects without GPS is certainly an interesting and useful problem to be addressed. For instance, indoor navigation systems could not rely on the GPS, as it is highly inaccurate indoors [16], and instead have to use inertial sensors [11], augmented reality [7] [21], or wifi fingerprinting [16], to name some possible techniques.

Transportation Mode Detection is an active research topic, being its information useful for a multitude of different services, ranging from context aware computing to personalized apps usage [3] [4] [8] [20]. Although methodologies differs from each other, they can be broadly classified in two groups: GPS-based and Sensor-based. While the former relies on the speed obtained from GPS data to infer the transportation mode, the latter uses inertial sensors which are typically classified using machine learning techniques. The GPS based systems offer a good accuracy for discriminating between motorized modes and pedestrian modes [20], but offer lower performance for classifying a more vast set of transportation modes [4]. Moreover, they also consume more battery than using inertial sensors, which is also something to account for in mobile devices.

Dead Reckoning Systems (DRS) exploit inertial sensor data to obtain a set of movements, which are usually based on the acceleration, from which it is possible to obtain the route [14]. Similar systems have also been applied to GPS enabled vehicles in roads, mainly to improve the accuracy of the system [22], in scenarios in which the GPS offers scarce accuracy in vehicular systems, typically due to urban canyons and tall buildings. Multiple sensors can also be aggregated together, to achieve a higher localization accuracy [15].

Few works can be found that merge the information of DRS and road data, to match the movements recognized by the DRS to specific roads. Most of them typically use the information from inertial sensors to update other data, such as WIFI maps [9], or to construct indoor maps [23]. In [2] the authors present a system which is able to perform indoor localization with good precision, given that it is provided the map of the indoor location. It is worth to note the work of [17], where the authors integrates OpenStreetmap information with DRS to obtain indoor navigation. Closer to this work is [19], which shows that with real data of magnetometers it is possible to match specific roads, hence journeys. In addition to [19] we also add the accelerometer into our systems, which helps in understanding the length of the road segment traveled. Moreover, we also present a simulation study which show the constraints in which such system can operate in real world cities.

Data from OpenStreetmap has been already used in a multitude of different domains, which spans from the generation of real vehicular traces [5], to vehicle navigation [18]. In general, the data offers plenty of details, although some areas are recorded with more precision than others compared to the real road deployment [10].

III. Model

In this section we describe the model we developed within the scope of this study. Our proposal builds on raw data obtained through smartphone integrated sensors, precisely the accelerometer and the magnetometer, used to infer the distance traveled and the turn events, respectively.

In particular, we compute a segmented trace, cut at every turn, leveraging data from inertial sensors. Eventually, we match the segmented trace to a real road deployment, with the aim to reduce the overall error, as matching intermediate segments would compensate for errors of the dead reckoning system. Specifically our algorithm jointly uses information obtained through accelerometer and magnetometer, and binds them with data obtained through OpenStreetmap. We proceed in different steps, which we summarize as:

- Step 1: *Segments Identification*, in which we identify the number of road segments by monitoring the magnetometer data
- Step 2: *Dead Reckoning System (DRS)*, through which we compute the length of each traveled segment thanks to the accelerometer measurements
- Step 3: *Segments assignment to OpenStreetmap data*, in which we build the possible path that has been traveled by the device searching for similar road segments in the real road deployment downloaded from OpenStreetmap

In the following sections we present each of these steps in detail.

A. Segments identification

At first, we need to identify the different segments, based on the orientation, obtained from the raw measurements read from the magnetometer. In other words, in the first step of our algorithm we need to compute the total number of roads that have been traveled and their orientation.

We define the measurement $\Omega(t)$ at time t as:

$$\Omega(t) =< \mathcal{A}(t), \mathcal{M}(t) >, \tag{1}$$

where $\mathcal{A}(t)$ is the tuple containing the 3 axis measurements of the accelerometer at time t and $\mathcal{M}(t)$ is the tuple of the 3 measurements of the magnetometer.

In this step, we only use the $\mathcal{M}(t)$ to obtain the turns performed by the device. More in detail, we compute the moving variance over the last N measurements of the magnetometer. To do it, we compute at first the moving mean at time t as:

$$\mu_t = \frac{\sum_{i=t-N}^{t} \mathcal{M}(i)}{N}, \tag{2}$$

and the corresponding moving variance at time t as

$$\sigma_t^2 = \frac{\sum_{i=t-N}^{t} \mathcal{M}(i)^2 - \mu_t^2}{N-1} \tag{3}$$

We then define a binary hypothesis $\mathcal{H}(t)$ to σ_t^2 defined as:

$$\mathcal{H}(t) = \begin{cases} 0 & \sigma^2(t) \geq \epsilon \\ 1 & \sigma^2(t) < \epsilon \end{cases}$$

with threshold ϵ equal to

$$\epsilon = \frac{\sum_{i=1}^{N} \mu_i}{N}, \tag{4}$$

which is basically the mean of the moving means computed through Equation 2.

At each time t, $\mathcal{H}(t)$ is then either a 1, identifying that the car was not turning, or a 0 which instead reflects a turn. In other words, a turn starts when we have a transition $1 \rightarrow 0$ and ends when we observe $0 \rightarrow 1$.

We can now easily build the vector of orientations $\mathcal{T}$ as

$$\mathcal{T} = \mathcal{M}(t) \cdot (1 - \mathcal{H}(t)), \tag{5}$$

which is then a sequence of 0 and angles, referring to the orientation of the vehicle at time t.

B. Dead Reckoning System

In this step, we describe the Dead Reckoning System which we leverage to obtain the length of each road segment.

At first, since the accelerometer reports the measurements on the 3 axis, and in order to be independent from the orientation of the device, we compute the magnitude $\dot{\mathcal{A}}(t)$ of the measurements of the accelerometer at time t as:

$$\dot{\mathcal{A}}(t) = \sqrt{\mathcal{A}_x^2(t) + \mathcal{A}_y^2(t) + \mathcal{A}_z^2(t)} \tag{6}$$

Using the magnitude is a popular solution for scenarios like the one we studied in this paper, and it is widely used in literature for similar tasks [4] [6], since it allows to abstract from the specific orientation of the device.

We then define the vector Λ as

$$\Lambda(t) = \mathcal{H}(t) \cdot \dot{\mathcal{A}}(t), \tag{7}$$

built with all the accelerometer values pertaining to straight segments. That is, we do not consider the lengths while the vehicle is turning, thus we only compute it for straight segments, as all the accelerometer values would be multiplied by 0.

We then define the route $\mathbb{R}$ as

$$\mathbb{R} = \{\mathcal{D}, \mathcal{O}\}, \tag{8}$$

where $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{N-1}\}$ is the list of computed distances of all the road segments, and $\mathcal{O} = \{\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{N-1}\}$ is the list of orientations of the next segment at each turn, where the generic $\mathcal{O}_i$ is the average of the orientation values of segment $i+1$.

We can now compute the total distance traveled over each road segment. Let $\mathcal{D}_i$ be the computed length of the i-th segment by picking the non-zero sequences in $\Lambda(t)$. Then, we can compute the total distance traveled $\mathcal{D}_i$ of segment i as

$$\mathcal{D}_i = (t - t_{i-1}) \cdot \mathcal{V}_i^0 + \int_{t_{i-1}}^{t_i} \dot{\mathcal{A}}(t) dt \tag{9}$$

Basically, we compute the length of segment $\mathcal{D}_i$ by calculating the total acceleration within the segment, summed to the initial speed when entering the segment. In fact, if the total acceleration over segment i is 0, then the distance traveled is simply computed by the entering speed V_i^0 multiplied by the travel duration within edge i.

Eventually, $\mathbb{R}$ completely defines the whole journey of the vehicle in terms of road lengths and turns, which will be used to match the measured values to real road segments, as we will describe in the following section.

C. OpenStreetmap Segment Matching

In this Section we describe the algorithm we developed to match sensor data to road segments starting from $\mathbb{R}$, by leveraging OpenStreetmap data, which we call PRIME (Probabilistic Recursive Inertial Matching Edges).

We start by defining a graph $G = (V, E)$, in which V are the vertices and E are the arcs.

The general idea of the algorithm is straightforward: our aim is to match each segment $\mathcal{D}_i \in \mathbb{R}$ with arcs in E, and all the vertices V with turns $\mathcal{O}_i \in \mathbb{R}$. More in detail, starting from vertex V_0 we look for all the outer arcs, and select that which matches with the angle $\mathcal{O}_0$ and length $\mathcal{D}_0$. We then repeat the exact same procedure in V_1, by selecting the arc which matches the angle $\mathcal{O}_1$ and the arc with length $\mathcal{D}_1$. However, data from the sensors and that extracted from OpenStreetmap may carry errors, which make the exact matching of arcs and vertices unpractical. Hence, we extend further the edge selection phase to account for noise error, eventually considering more road segments. We then define probabilities to select those that better match the computed lengths and orientations we got in the previous steps.

At first, it is highly challenging to correctly estimate the length traveled from the device. This is because the data from the accelerometer may be inaccurate due to measurements frequency and noise, and hence the computed distance may be slightly different than that actually traveled by the vehicle. Hence, we introduce a threshold η which estimates our admissible error. Basically, we assume that the length of each segment $\mathcal{D}_i$ should instead be intended as $\mathcal{D}_i \pm \eta$. More formally we change each $\mathcal{D}_i$ into $\bar{\mathcal{D}}_i$ as following:

$$\bar{\mathcal{D}}_i = [\mathcal{D}_i - \eta, \mathcal{D}_i + \eta] \tag{10}$$

Another possible issue comes from the inaccuracy in the azimuth reading from the magnetometer, which might exclude some of the arcs even if small differences are found. Therefore, we apply a similar technique to the one presented before, in which we add a threshold θ which accounts for possible measurement errors of the magnetometer. Hence, when the algorithm checks whether the real road segment K might have been traveled and therefore should be added to the list of segments to be evaluated, we check if $K \in \bar{O}_j$, where $\bar{O}_j$ is defined as

$$\bar{O}_j = [\mathcal{O}_j - \theta, \mathcal{O}_j + \theta]. \tag{11}$$

We now build a tree of possibilities $\mathbb{T}$, starting from the initial vertex. We then add all the possible arcs for which the length is between $\bar{D}_0$. We then repeat the same methodology at the end of the i-th arc, by considering the arcs which have a turning angle $\in \bar{O}_i$. We then add those which have a length $\in \bar{D}_i$, and we repeat the same for all the elements in $\mathbb{R}$. Clearly, using high values of η and θ would include more possible paths in $\mathbb{T}$, while using low values may possibly exclude the right path, due to measurement errors and noise. Obviously, some of the paths of the tree may not have all the segments of the journey, as some arcs may not be found starting from a vertex, thus leading to a wrong path without the number of road segments in $\mathbb{R}$.

Then, we define a series of probabilities towards identifying the most probable path among all the possible ones. We define $p_i^{\mathcal{D}}$ as the probability for the i-th arc as:

$$p_i^{\mathcal{D}} = 1 - \frac{|\mathcal{D}_i - E_i|}{E_i}, \tag{12}$$

where E_i is the real length of the i-th road segment. In case of a high error, $p_i^{\mathcal{D}}$ would be close to 0, while in case of a small one it will be close to 1. We divide by E_i to weigh the errors depending on the length of the arc, as the same absolute error weighs less on a longer arc than in a shorter one.

We perform a similar operation to assess the correctness of the orientation of the edge with $p_i^{\mathcal{O}}$, defined as

$$p_i^{\mathcal{O}} = 1 - \frac{|\mathcal{S}_i - V_i|}{V_i}, \tag{13}$$

where in this case V_i is the azimuth of the i-th vertex.

We then compute the total probability $\mathbb{P}(\mathcal{J})$ for the whole journey $\mathcal{J}$ as

$$\mathbb{P}(\mathcal{J}) = \prod_{i \in \mathcal{J}} p_i^{\mathcal{D}} \cdot p_i^{\mathcal{O}}. \tag{14}$$

Finally, we select the path $\mathcal{J}$ whose $\mathbb{P}(\mathcal{J})k$ is the highest one, thus ending the algorithm.

IV. Performance Evaluation

In this section we provide performance evaluation of our proposal. We separate our analysis in two parts: the first is devoted to analyze the effectiveness of our system in inferring user location routes, which we analyzed with real data obtained through a custom made Android application. We present this study in Section IV-A. We then move to understanding how different paths and city deployments may affect the practical possibility to recognize paths, based on roads heterogeneity and city size, which is presented in Section IV-B.

A. System evaluation

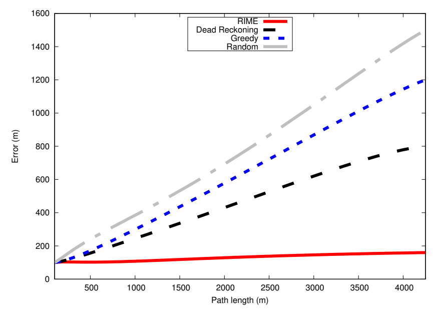
The first analysis we perform is devoted to understand the effectiveness and accuracy of the proposed system in terms of understanding the path traveled by the car, hence the final point. To provide a more comprehensive analysis, we compare our proposed PRIME algorithm with 3 other different proposals, which are:

- *Dead reckoning*: this algorithm only uses the data computed in Section III-A and Section III-B. In other words, it does not match the edges on real map data, but only uses inertial sensors to calculate a trajectory from an initial point to an end point.
- *Random*: this algorithm matches segments on real map data, but does this randomly at each turn. Basically, it understands when user turns, but takes a random edge, hence mimicking a system with a corrupted magnetometer.
- *Greedy*: this proposal uses both the dead reckoning system and the real road matching. However, instead of building a tree of possible routes, it always select the best matching segment at each turn, which is the one which deviates less from the measurement. In case of errors, it may lead to wrong decisions which may end up in huge estimation errors.

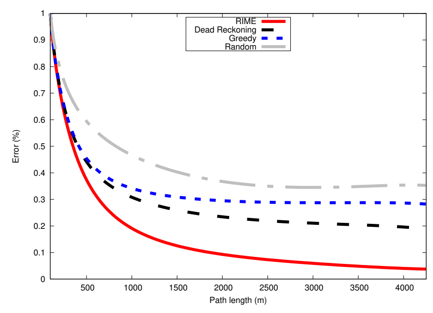
Our analysis is performed on real data. To gather it, we developed an Android application which records the measurements from the accelerometer and the magnetometer installed in the smartphone. During a single measurement run, we do not variate the position of the device, which is instead varied among different measurement runs. Data is sampled at 10 Hz, and we gathered more than 5 hours of samples, with a total of 50 unique paths.

In Figure 1 we present the result of the study, which have been obtained by running the algorithm on all the paths we have recorded, with variable length from 100 meters to 5000 meters.

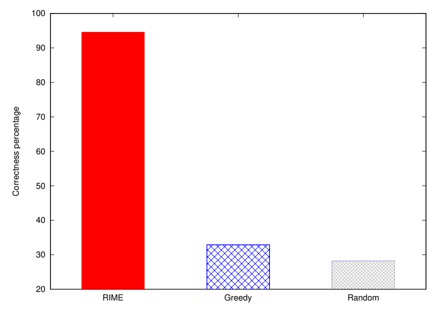
In particular, Figure 1(a) shows the error achieved by the four different algorithms between the correct final position and the computed one. That is, the euclidean distance between the real ending point and the computed one. As it can be seen, PRIME performs the best, while all the others offer lower performance. Surprisingly, the Greedy and the Random version offer worse performance compared to the Dead Reckoning.



(a) Absolute Error



(b) Relative Error



(c) Road Correctness

Fig. 1. Performance evaluation of the proposed solution.

This aspect lies in the imprecision of the measurements, which are instead better compensated by PRIME, which explores a wider set of possible edges. In particular, it is interesting to note that for PRIME the error does not increase too much as the total length of the path increases. For all the other algorithms, longer paths exhibit higher errors. This is explained by the fact that PRIME constrains the path to be on roads, and thanks to the fact that it explores a set of possible edges, eventually selecting the most probable path, it ends up in achieving a stable performance regardless of the length of the path. It is important to note that PRIME may select a wrong segment within the path. However, as we select the path with the best overall probability, this may be compensated by subsequent segments which instead are well recognized. This is also evident from the fact that even for short paths less than 500 meters long, PRIME achieves an error comparable with other proposals. However, it remains constant, while for the other proposals it scales up, hence indicating further errors during the path, which are instead not observed with PRIME.

Figure 1(b) shows a similar analysis, where however the studied metric is the ratio of error over the total length of the path. Here it is possible to see that all proposals follow a similar path, but PRIME achieves again the best results, since its error is somewhat constant regardless of the length of the path traveled, something which is not true for the other three proposals. However, all proposal seem to suffer from higher errors from shorter paths, something which was also possible to see in Figure 1(a).

In systems like the one studied in this paper, the focus may be not precisely on the ending point, but maybe on the area. Hence, we evaluate in Figure 1(c) whether the final computed road corresponds with the real one. This studies the scenario in which we are not interested to understand the precise point, but we narrow the ending position to a single road. In this analysis we do not show the Dead Reckoning proposal, as it does not computes path on roads, but follows the measurements of the sensors which can lead in places without roads. PRIME achieves again the best results, being able to recognize the final road almost in the 95 % of the cases, while other proposals fall shortly of 35 %. Again, this happens as PRIME can compensate the errors in the middle of the path better than other algorithms.

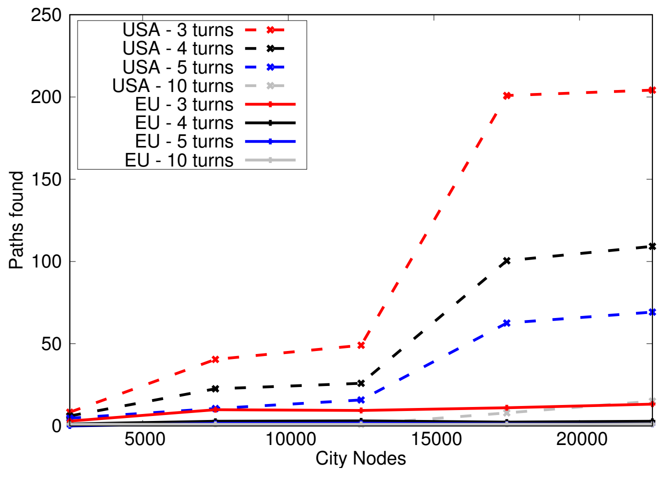


Fig. 2. Analysis on different city size in Europe and USA.

B. City wide analysis

The second analysis we performed is aimed at understanding how much real city roads, and different city deployments, impact the possibility to recognize paths traveled by cars based solely on the recognition through the accelerometer and the magnetometer. We select different city sizes in the U.S.A. and Europe, which exhibit vastly different road deployments, with the U.S.A. which tend to have a more Manhattan grid like scenario, hence with turns which are similar between them in terms of angle, and Europe, in which cities are more diversified, hence with roads which have turns angles which differ more.

For each city, we randomly select a path and simulate as there was a vehicle diving on it, which then records sensor measurements. We then look how distinctive such measurements are in the same city, looking for similar paths within the same area. Clearly smaller cities have less possibilities, hence it is easier to find unique paths, while bigger cities have more roads, hence possibly more paths. Due to space constraints, we perform the tests only with $\eta = \theta = 25$. Obviously increasing these values would result in a higher number of identified paths, though some would be far less similar to the real one, while setting them to a lower value would restrict the analysis too much, without considering noise which is present in similar measurements. A complete analysis on the importance of η and θ is left as future study.

Figure 2 shows the results of our analysis, which we performed for paths of 3, 4, 5 and 10 turns in total, and for different cities in the U.S.A. and Europe. We plot the total number of paths found over the city nodes, which are the crossroads at which a vehicle may turn. In total, we have analyzed more than 700.000 unique paths in 60 cities, grouped into 5 different sizes. As it is straightforward to note, a higher number of turns reduces the number of possible paths with similar values. The U.S.A. are far less identifiable, as roads tend to have more similar angles of 90 degrees, which make paths more similar among each other. Concerning the case for Europe, we achieve a low number, hence high possibility of identifying the path only with 3 turns. Harder to see on the chart are the Europe lines with more than 3 turns, which achieve results close to 1, thus meaning that in all the city deployment it was possible to find only 1 path with the characteristics of the simulated one. In general it is fair to say that globally, paths in the U.S.A. are far less identifiable, and in general only a higher number of turns enables the identification of them. Different the case for Europe, where cities with similar sizes compared to the U.S.A. scenario offer far less similar paths.

V. Conclusion

In this paper we have presented a study towards identifying driving paths of users by exploiting data obtained through mobile device integrated sensors. In case of malicious installed applications on mobile devices, sensors can be accessed without requiring explicit permissions to the users, which then may not be aware of data leaks. We have then shown how these data may be exploited to track driving paths of users, by using popular sensors such as the accelerometer and magnetometer, now available in many off the shelf devices. Results from field tests have shown the accuracy of the system, which outperforms other proposals we compared against. We have also performed a simulation study on several cities in the U.S. and Europe, showing that with less than 10 turns it is possible to almost uniquely identify a singular path, even in large cities.

Future works on this topic are towards reducing the turn angle and road segment length, which would enable to improve the recognition of the system.

References

[1] Mahmoud Abd Rabbou and Ahmed El-Rabbany. Tightly coupled integration of GPS precise point positioning and MEMS-based inertial systems. *GPS Solutions*, 19(4):601–609, oct 2015.

[2] J. C. Aguilar Herrera, A. Hinkenjann, P. G. Ploger, and J. Maiero. Robust indoor localization using optimal fusion filter for sensors and map layout information. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 1–8. IEEE, oct 2013.

[3] Media A Ayu, Teddy Mantoro, Ahmad Faridi A Matin, and Saeed S O Basamh. Recognizing User Activity Based on Accelerometer Data from a Mobile Phone. pages 617–621, 2011.

[4] Luca Bedogni, Marco Di Felice, and Luciano Bononi. Context-aware Android applications through transportation mode detection techniques. *Wireless Communications and Mobile Computing*, 16(16):2523–2541, nov 2016.

[5] Luca Bedogni, Marco Gramaglia, Andrea Vesco, Marco Fiore, Jérôme Härri, and Francesco Ferrero. The Bologna ringway dataset: Improving road network conversion in SUMO and validating urban mobility via navigation services. *IEEE Transactions on Vehicular Technology*, 64(12):5464–5476, 2015.

[6] Armir Bujari, Bogdan Licar, and Claudio Enrico Palazzi. Movement Pattern Recognition through Smartphone's Accelerometer. In *Proc IEEE DENVECT*, 2012.

[7] Ibrahim Arda Cankaya, Arif Koyun, Tuncay Yigit, and Asim Sinan Yuksel. Mobile indoor navigation system in iOS platform using augmented reality. In *2015 9th International Conference on Application of Information and Communication Technologies (AICT)*, pages 281–284. IEEE, oct 2015.

[8] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Human activity recognition from accelerometer data using a wearable device. *Pattern Recognition and Image Analysis*, 6669 LNCS:289–296, 2011.

[9] Qiang Chang, Samuel Van de Velde, Weiping Wang, Qun Li, Hongtao Hou, and Steendam Heidi. Wi-Fi Fingerprint Positioning Updated by Pedestrian Dead Reckoning for Mobile Phone Indoor Localization. pages 729–739. 2015.

[10] Hongchao Fan, Alexander Zipf, Qing Fu, and Pascal Neis. Quality assessment for building footprints data on OpenStreetMap. *International Journal of Geographical Information Science*, 28(4):700–719, apr 2014.

[11] Carl Fischer, Kavitha Muthukrishnan, Mike Hazas, and Hans Gellersen. Ultrasound-aided pedestrian dead reckoning for indoor navigation. In *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments - MELT '08*, page 31, New York, New York, USA, 2008. ACM Press.

[12] Philippe Golle and Kurt Partridge. On the anonymity of home/work location pairs. In Hideyuki Tokuda, Michael Beigl, Adrian Friday, A. J. Bernheim Brush, and Yoshito Tobe, editors, *Pervasive Computing*, pages 390–397, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[13] A K M Mahtab Hossain and Wee-Seng Soh. A survey of calibration-free indoor positioning systems. *Computer Communications*, 66:1–13, 2015.

[14] A.R. Jimenez, F. Seco, C. Prieto, and J. Guevara. A comparison of Pedestrian Dead-Reckoning algorithms using a low-cost MEMS IMU. In *2009 IEEE International Symposium on Intelligent Signal Processing*, pages 37–42. IEEE, aug 2009.

[15] Wonho Kang and Youngnam Han. SmartPDR: Smartphone-Based Pedestrian Dead Reckoning for Indoor Localization. *IEEE Sensors Journal*, 15(5):2906–2916, may 2015.

[16] H. Leppäkoski, J. Collin, and J. Takala. Pedestrian Navigation Based on Inertial Sensors, Indoor Map, and WLAN Signals. *Journal of Signal Processing Systems*, 71(3):287–296, jun 2013.

[17] Jo Agila Bitsch Link, Paul Smith, Nicolai Viol, and Klaus Wehrle. FootPath: Accurate map-based indoor navigation using smartphones. In *2011 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–8. IEEE, sep 2011.

[18] Dennis Luxen and Christian Vetter. Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '11*, page 513, 2011.

[19] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir. The perils of user tracking using zero-permission mobile apps. *IEEE Security Privacy*, 15(2):32–41, March 2017.

[20] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, 6(2):1–27, 2010.

[21] Umair Rehman and Shi Cao. Augmented-Reality-Based Indoor Navigation: A Comparative Analysis of Handheld Devices Versus Google Glass. *IEEE Transactions on Human-Machine Systems*, pages 1–12, 2016.

[22] Ahmed Abdel Wahab, Ahmed Khattab, and Yasmine A. Fahmy. Two-way TOA with limited dead reckoning for GPS-free vehicle localization using single RSU. In *2013 13th International Conference on ITS Telecommunications (ITST)*, pages 244–249. IEEE, nov 2013.

[23] Xiuming Zhang, Yunye Jin, Hwee-Xian Tan, and Wee-Seng Soh. CIMLoc: A crowdsourcing indoor digital map construction system for localization. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6. IEEE, apr 2014.