Precise Vehicle Positioning for Indoor Navigation via OpenXC

Yusuf Turk, Baturay Ozcan and Sezer Gören
Department of Computer Engineering, Yeditepe University, Kayışdağı, 34755, Istanbul, Turkey

Keywords: OpenXC, Indoor Positioning, Connected Car, Indoor LBS, Intelligent Parking Systems.

Abstract: We propose a method for vehicle positioning for indoor locations like parking garages. Our method does not require other external positioning systems such as GPS. Instead, we determine the vehicle position from the vehicle data obtained from an OpenXC dongle attached to the OBD-II interface of the vehicle. An accompanying smartphone application which connects with the dongle via Bluetooth is developed. This application calculates the position of the car and applies the algorithms proposed in this paper to the vehicle data received from the interface. The route of the vehicle is then constructed and displayed on the smartphone screen. As a future work, an assistant application will be developed that guides the driver to the spot where the car was parked before.

1 INTRODUCTION

Finding parking spaces and locating the parked vehicles in huge parking lots with multiple levels such as large shopping malls, airports, and office buildings is a common problem. Although traditional solutions such as coloring and numbering the areas worked until now, drivers still have to locate the car without any assistance. Also, assisting the driver with a single route to the location of the parked car is a more time efficient solution.

Navigation systems in vehicles are becoming a part of our daily life. Whether it is offered as an embedded system included by the car manufacturer or an external system using a smartphone, navigation systems are more accessible than ever. Most of the navigation systems use GPS which relies on positioning values retrieved from various satellites.

This paper focuses on offering a route assistance solution for the cases where GPS is not available such as an indoor parking garage multiple levels below the ground. Our solution is based on OpenXC platform (OpenXC Platform, 2011). OpenXC vehicle interface is composed of a microcontroller with two external connections – one to the CAN bus via the OBD-II port, and one to the host device via USB or Bluetooth. It passively listens for a subset of CAN messages, performs required unit conversion or factoring and outputs a generic version to the USB interface. Many vehicle output parameters such as vehicle speed, steering wheel angle, door status, wind shield status, etc. are available and sent through Bluetooth and received by a custom smartphone application developed on iOS or Android operating system. OpenXC API enables the development of custom applications using the information retrieved from the vehicle. The adapter and the smartphone communicate through Bluetooth. In this paper, we developed a smartphone application which uses the real-time values retrieved from OpenXC interface in order to draw and save the route of the vehicle on the indoor map of the parking garage and offers precise vehicle positioning for indoor navigation. In the future, our smartphone application can be updated to guide the driver to the already parked car using the saved route.

2 RELATED WORK

OpenXC is commonly used in vehicle systems along with complementing tools and software. Android application developed using OpenXC API can retrieve real-time values from the vehicle to compare them with threshold values (Suresh and Nirmalrani, 2014). Similarly, vehicle condition and sensor values can be integrated with a network such as WiMAX in order to create a networked Vehicle Sensing and Control (VSC) system (Wang et al., 2016). Aside from using OpenXC, vehicle data can also be obtained by using in-vehicle eDSMS along with an Android application (Akiyama et al., 2015).

Turk, Y., Ozcan, B. and Gören, S.
Precise Vehicle Positioning for Indoor Navigation via OpenXC.
DOI: 10.5220/0006770104400445
In *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2018)*, pages 440-445
ISBN: 978-989-758-293-6

Regarding the indoor parking problem, many solutions have been proposed before. While some of them uses built-in car sensors, others use external devices to determine the position of the vehicle. Wagner et al. (2010) proposes a method using Dead Reckoning (DR) and commercially available maps from companies or from building plans. The method utilizes a topological map matching algorithm that combines geometrical information and the topology of the road network. Bojja et al. (2013) takes advantage of low cost gyroscope and odometer sensor. Sensor data is combined with the 3D map of the building to obtain the indoor position. Razak et al. (2015) tried to navigate the driver to an appropriate parking position using QR codes placed in the parking space. Android application reads the QR code and navigates the driver to a parking spot using the map stored in the database. Vehicle movement is calculated using a step counter. Kumar et al. (2016) mounted cameras throughout an indoor parking lot to identify vehicle movement. The data was used to calculate vehicle position by using convolutional neural network with Deep Learning.

Indoor positioning and parking services also use smartphone sensors to determine the position of the vehicle. Some studies combined smartphone sensor data with RSSI data of WLAN and the Bluetooth signal data. Liu et al. (2012) proposed a cost and energy efficient solution. Wireless Local Area Network (WLAN) signals and the built-in smartphone sensor data combined to find the precise location of the vehicle inside an indoor parking space. In addition to that, another study by Liu et al. (2012) also used WLAN signals and smartphone sensors. An indoor positioning engine is developed for the positioning. Motion information is measured using smartphone sensors. The solution can be used for real-time navigation and location tracking. Similarly, Bluetooth signals strengths can also be used for location fingerprinting which is combined with relative positioning in a study by Wilfinger et al. (2016). Bluetooth Low Energy (BLE) is offered as a cost-effective solution. Khare et al. (2017) stated that the BLE and IEEE 802.15.4a compliant Ultra-wideband (UWB) sensor data can be processed in real-time to develop an indoor localization system. Unfortunately, UWB sensors are not in current handheld devices yet.

Research by Li et al. (2016) is focused on car searching system that will help users to find their cars in a large parking lot. QR codes are distributed over the parking lot and a smartphone application calculates the path to the vehicle using the shortest path algorithm.

Internet of Things (IoT) based applications and middleware systems are also used for parking solutions for both current use cases and smart cities. Ji et al. (2014) stated that car parking services must be intelligent for IoT based smart cities. They proposed a cloud based system offering users a car parking service experience. In an another IoT application by Bagula et al. (2015), RFID readers are distributed in a parking lot to help billing and provide smart parking. Similarly, Caballero-Gil et al. (2016) used Near Field Communication (NFC) tags for their indoor positioning system.

iBeacon is another technology which offers a low energy solution compared to GPS and WLAN. Zou et al. (2016) used iBeacons to provide positioning and navigation for vehicles using both indoor and outdoor spaces.

Indoor location positioning and navigation is also used for autonomous car parking systems. A study by Correa et al. (2017) proposes a cooperative vehicular positioning network using the Vehicular Sensor Networks (VSN).

3 PROPOSED METHOD

OpenXC supports various number of measurements of a car such as engine speed, steering wheel angle, fuel consumption, accelerator pedal position, ignition status, etc. The data retrieved using the OpenXC API as JSON formatted messages in a class called Vehicle Messages. In our design, we make use of three of the measurement data received through OpenXC vehicle interface: i. steering wheel angle, ii. odometer, and iii. ignition status. Steering wheel angle provides the information about the angle of the steering wheel in degrees in a range from -600 to +600. When the steering wheel is turned to the right, positive values are read in the message and negative when to left. Odometer gives the data of the distance that the vehicle travels as a unit of kilometer between the values 0 to 16777214 with about 0.2-meter resolution. Ignition status returns as a Boolean value indicating one of the 4 states of the ignition: off, accessory, run, and start.

In our implementation, we have three functions: i. CheckTurn (Figure 1), ii. UpdateTurn (Figure 2), iii. UpdateDirection (Figure 3). CheckTurn function checks whether the car has turned or not. If so, the route is updated. At the beginning, we consider the ignition status of the vehicle. If it is off and the car is in the parking area, we know that the driver has parked his/her car in a parking spot. Therefore, we update the route and return.

Algorithm 1 Check if the car has turned and update the map if so

```
function CHECKTURN(wheelAngle)
    if ignitionStatus = OFF && startRouting = true then
        routingOver←true
        update odometer values
        calculate distance
        update coordinates
        draw the line
        return
    end if
    if wheelAngle > threshold || wheelAngle < −threshold then
        if startRouting = false then
            return
        end if
        if turnStart = true then
            turnContinue←true
        else
            turnStart←true
        end if
    else
        if turnContinue = true then
            updateMap(wheelAngle)
        end if
        turnStart←false
        turnContinue←false
    end if
end function
```

Figure 1: Algorithm for CheckTurn Function.

In order to detect a turn, we set a threshold value for the steering wheel angle. If the current angle of the steering wheel exceeds the threshold we assume turn is started or continues when it has already started. Exceeding the threshold means that the angle can be over the positive threshold or below the negative threshold. When this is the case, we check the *startRouting* value which indicates if the car has entered the parking area or not. If false, just return. Otherwise, if the vehicle has already started to turn, we set *turnContinue* value as true. If not, that means rotation is just started, then we set *turnStart* value as true. If the threshold value is not exceeded but the car was in turning state, we predict that the rotation is over and update the map. To do this, we need the other two functions: updating the coordinates with turn points and updating the direction after the turn. The threshold value can be adjusted to detect the different types of curves. For example, a U-turn is detected when there are multiple and complete steering wheel rotations.

UpdateCoord (Figure 2) is a function which we save the coordinates of the turning points. It takes the parameter *distance* (value in the unit of meters showing the distance between the previous turn point and the current one) and updates x and y coordinates of the current turn point. For example, if the direction

Algorithm 2 Update the coordinates with turn points

```
function UPDATECOORD(distance)
    if direction = north then
        nextXcoord←lastXcoord
        nextYcoord←lastYcoord − distance
    else if direction = east then
        nextXcoord←lastXcoord + distance
        nextYcoord←lastYcoord
    else if direction = south then
        nextXcoord←lastXcoord
        nextYcoord←lastYcoord + distance
    else
        nextXcoord←lastXcoord − distance
        nextYcoord←lastYcoord
    end if
end function
```

Figure 2: Algorithm for UpdateCoord Function.

of the vehicle was north before the rotation, x coordinate does not change but y coordinate is decreased.

In UpdateDirection function (Figure 3), we set the direction of the vehicle after the turn which is a critical issue. Our solution runs in a way that we check two values together: steering wheel angle and the direction of the vehicle before the turn. First, we find to which direction the vehicle is rotated. If the value of the angle is positive, we predict that it is rotated to the right. Otherwise, to the left. Then, we take the previous direction of the vehicle under

consideration. For example, if the car is rotated to the left and the previous direction is east, it is clear that the driver was going to the east and then he/she turned to the left, so the new direction of the car is now north.

4 EVALUATIONS

We evaluated the system using an OpenXC adapter attached to a car that supports the adapter and a smartphone with Android operating system. The car was driven to a small parking garage with the parking plan given in Figure 4. In the figure, the color white represents the roads and the grey rectangles represent the parking spaces. The entrance of the parking lot is the road at the southern part of the map. The map is drawn proportionally.

The car was driven inside the indoor parking lot until an empty parking spot is found. The OpenXC adapter used in tests does not support GPS and all the location services including the GPS was disabled on the smartphone. The smartphone application was run when the car entered the parking lot. Although a static map is used in our system, it can be replaced according to the last known location retrieved by the smartphone.

Figure 5 shows the car movements captured by the application in real time. A new route is drawn on top of the map whenever a new turn is detected. Route is smoothened by the application in order to create an easy to understand style for the map. Parking sign at the bottom indicates the entrance of the indoor parking lot. The car icon indicates the location of the car and it is updated in real time. When the car is parked, the car icon is moved inside the correct parking spot. Smoothing the route created according to the data from the OpenXC allowed the application

Algorithm 3 Update the direction after turn

```
function UPDATEDIRECTION(wheelAngle)
    if wheelAngle > 0 then                    ▷ turned to right
        if direction = north then
            direction←east
        else if direction = east then
            direction←south
        else if direction = south then
            direction←west
        else
            direction←north
        end if
    else                                      ▷ turned to left
        if direction = north then
            direction←west
        else if direction = east then
            direction←north
        else if direction = south then
            direction←east
        else
            direction←south
        end if
    end if
end function
```

Figure 3: Algorithm for UpdateDirection Function.

to draw straight lines from corner to corner. Since the most parking spaces in office buildings, shopping malls, and airports consists of proportionally designed rectangular parking spots, it will be sufficient for most of the use cases. While we were doing the tests, we always observed that the vehicle and parking lot sizes are detected with an error of less than a half meter. In addition, we also observed that the route was successfully drawn without crossing the area of the parking space.

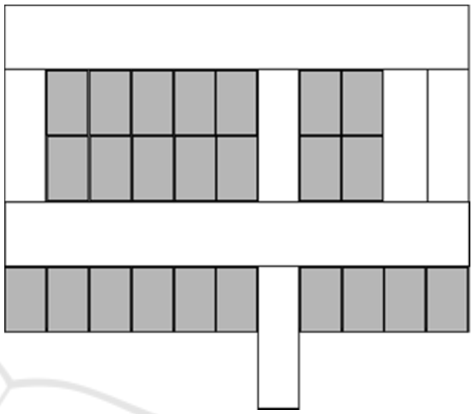

Figure 4: Blueprint of the car park.

The steering wheel angle range from -600 to 600 where positive values indicate that wheel is turning to the right and negative values indicate that the wheel is turning to the left. If the driver turns left, it does not necessarily mean that the steering wheel angle values will always be negative. This is because the steering action can consist of steering to the right for a certain angle, followed by a large angle steering to the left. Among all the in-car tests, our algorithm calculated the steering angles and turn direction correctly. The application saves the location of the parking spot when the ignition is switched off.

5 CONCLUSIONS

In this paper, we proposed a vehicle positioning method for the indoor navigation without the help of external positioning systems, but with OpenXC vehicle data. The indoor route of the vehicle is built and saved by our smartphone application. As a future work, driver assistance to the last parked location of the vehicle will be added as a new feature to our application. In addition, detection of the story level in multi-story parking garages is another future work. We expect that smart parking solutions and assistants like our approach will be more common in the following years.

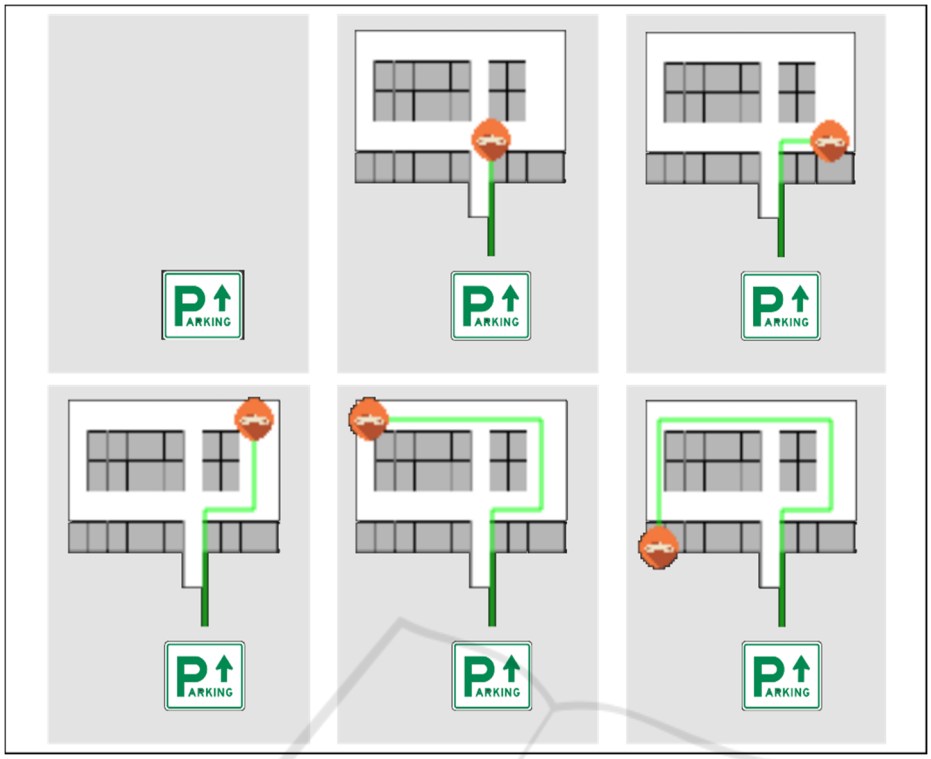

Figure 5: Car Movements in Real Time.

REFERENCES

Openxcplatform.com, 2011. OpenXC Platform. [online] Available at: www.openxcplatform.com/ [Accessed 3 November 2017].

V. Suresh and V. Nirmalrani. (2014). "Android based vehicle diagnostics and early fault estimation system". In: *International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC). Chennai, 2014, pp. 417-421*.

Y. Wang *et al*. (2016). "VLnsight: enabling open innovation in networked vehicle sensing and control". In IEEE Network, vol. 30, no. 4, pp. 34-44, July-August 2016.

S. Akiyama, Y. Nakamoto, A. Yamaguchi, K. Sato and H. Takada. (2015). "Vehicle Embedded Data Stream Processing Platform for Android Devices". In: *International Journal of Advanced Computer Science and Applications (IJACSA), 6(2)*.

J. Wagner, C. Isert, A. Purschwitz and A. Kistner. (2010) "Improved vehicle positioning for indoor navigation in parking garages through commercially available maps". In: *2010 International Conference on Indoor Positioning and Indoor Navigation*, Zurich, pp. 1-8.

J. Bojja, M. Kirkko-Jaakkola, J. Collin and J. Takala. (2013). "Indoor 3D navigation and positioning of vehicles in multi-story parking garages". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, 2013, pp. 2548-2552.

S. F. A. Razak, C. L. Liew, C. P. Lee and K. M. Lim. (2015). "Interactive android-based indoor parking lot vehicle locator using QR-code". In: *2015 IEEE Student Conference on Research and Development (SCOReD)*, Kuala Lumpur, 2015, pp. 261-265.

A. K. T. R. Kumar, B. Schäufele, D. Becker, O. Sawade and I. Radusch. (2016). "Indoor localization of vehicles using Deep Learning". In: *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Coimbra, 2016, pp. 1-6.

J. Liu, R. Chen, Y. Chen, L. Pei, L. Chen (2012). "iParking: An Intelligent Indoor Location-Based Smartphone Parking Service". In: Sensors 2012, 12, 14612-14629.

J. Liu, R. Chen, L. Pei, R. Guinness, H.A. Kuusniemi, (2012). "Hybrid Smartphone Indoor Positioning Solution for Mobile LBS". In: *Sensors 2012, 12*, 17208-17233.

R. Wilfinger, T. Moder, M. Wieser and B. Grosswindhager. (2016). "Indoor position determination using location fingerprinting and vehicle sensor data". In: *2016 European Navigation Conference (ENC), Helsinki, 2016, pp. 1-9.*

S. P. Khare, J. Sallai, A. Dubey and A. Gokhale. (2017). "Short Paper: Towards Low-Cost Indoor Localization Using Edge Computing Resources" *2017 IEEE 20th*

International Symposium on Real-Time Distributed Computing (ISORC), Toronto, ON, 2017, pp. 28-31.

J. Li, Y. An, R. Fei and H. Wang. (2016). "Smartphone based car-searching system for large parking lot" *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA), Hefei, 2016, pp. 1994-1998.*

Z. Ji, I. Ganchev, M. O'Droma, L. Zhao, X.A. Zhang. (2014). "Cloud-Based Car Parking Middleware for IoT-Based Smart Cities: Design and Implementation". *Sensors 2014, 14,* 22372-22393.

A. Bagula, A, L. Castelli, M. Zennaro. (2015). "On the Design of Smart Parking Networks in the Smart Cities: An Optimal Sensor Placement Model*". Sensors 2015, 15*, 15443-15467.

C. Caballero-Gil, P. Caballero-Gil, J. Molina-Gil. (2016). "Cellular Automata-Based Application for Driver Assistance in Indoor Parking Areas". *Sensors 2016, 16, 1921.*

H. Zou, H. Jiang, Y. Luo, J. Zhu, X. Lu, L. Xie,. (2016). "BlueDetect: An iBeacon-Enabled Scheme for Accurate and Energy-Efficient Indoor-Outdoor Detection and Seamless Location-Based Service". *Sensors 2016, 16, 268.*

A. Correa, G. Boquet, A. Morell, J. Lopez Vicario. (2017). "Autonomous Car Parking System through a Cooperative Vehicular Positioning Network". *Sensors 2017, 17, 848.*