Smartphone-Based Sensor Fusion for Improved Vehicular Navigation

Oliver Walter, Joerg Schmalenstroeer, Andreas Engler, Reinhold Haeb-Umbach
Department of Communications Engineering
University of Paderborn, Germany
{walter, schmalen, haeb}@nt.uni-paderborn.de

***Abstract*—In this paper we present a system for car navigation by fusing sensor data on an Android smartphone. The key idea is to use both the internal sensors of the smartphone (e.g., gyroscope) and sensor data from the car (e.g., speed information) to support navigation via GPS. To this end we employ a CAN-Bus-to-Bluetooth adapter to establish a wireless connection between the smartphone and the CAN-Bus of the car. On the smartphone a strapdown algorithm and an error-state Kalman filter are used to fuse the different sensor data streams. The experimental results show that the system is able to maintain higher positioning accuracy during GPS dropouts, thus improving the availability and reliability, compared to GPS-only solutions.**

***Index Terms*—Smartphone, navigation, sensor fusion**

I. Introduction

In the past decade mobile phones have undergone a tremendous paradigm change. At first only intended for human-to-human communication purposes, the mobile phones evolved towards "smartphones" offering services and techniques for different applications. The technical break-through became possible by new human-machine interfaces (e.g., touchscreens, speech technologies), high and affordable data rates of mobile networks, and by new powerful and simultaneously energy efficient microprocessors. Modern smartphones support a multitude of wireless communication protocols, be it for cellular communications (e.g., UMTS, LTE), for short range, high data rate connections (BT, WLAN) or for very short range contactless data exchange, e.g., for payment services (NFC).

One of the first add-ons not related to communication was the integration of GPS (Global positioning system) receivers, thus enabling positioning and navigation services for the user. Smartphone-based positioning has found widespread use both for indoor positioning [1] and as an alternative to built-in navigation system for cars. Their main advantages compared to dedicated devices for car navigation are reduced costs, as no extra hardware needs to be purchased, and increased flexibility, because the device can be used both for on-board and off-board navigation.

However, a well-known deficiency of GPS-only navigation is that no positioning information is available during GPS dropouts, which may occur, e.g., in narrow street canyons, tunnels or parking garages. Higher availability of precise positioning information is therefore obtained by built-in car navigation systems which utilize inertial measurement units (IMUs) and other information (e.g., car speed information, vector maps, map matching) to predict the position of the car based on the last position information before the GPS dropout. This is particularly important for applications where the positioning information is used for electronic toll collection, as, e.g., in the German truck toll collecting system [6].

In the following we propose an approach which aims at achieving the same high precision positioning as built-in car navigation systems, however without additional hardware and installation costs. To this end we realized a sensor fusion algorithm on a smartphone which combines the GPS data with angular velocity information from the gyroscope sensors of the smartphone and, further, with the car speed information obtained from the vehicle's CAN-Bus via a wireless CAN-Bus-to-Bluetooth (CAN-BT) adapter. Sensor fusion is carried out by an error-state Kalman filter, whose complexity has been reduced such that it operates well below real-time.

Currently, many car manufacturers pursue approaches to integrate smartphones in cars to realize telematics or multimedia services, such as Mercedes Benz' *MBrace* [3], Ford's *Sync* [2], Toyota's *Entune* [4] or BMW's *ConnectedDrive* [5]. These proprietary systems allow access to the home entertainment system from within the car, or provide car information to the user (e.g., battery charge status of electric/hybrid cars). To the best of our knowledge, however, no such system realizes a sensor fusion algorithm as proposed here which aims at achieving positioning accuracy and availability that could be used for such sensitive applications as tolling services.

In section II we give a brief overview of the proposed system, explaining the overall system architecture and its components. More details about the components and the underlying algorithms for the sensor data fusion can be found in section III. Section IV shows some sample experimental results obtained from real world data, while section V shows some results of simulations carried out using real world data. Finally, in section VI we draw some conclusions and give an outlook.

II. System Overview

The key component of the proposed positioning system is an Android [7] smartphone, which hosts the applications for navigation and sensor fusion. It has multiple hardware components of which we mainly use the GPS receiver, the gyroscope and the Bluetooth receiver (see Fig. 1).

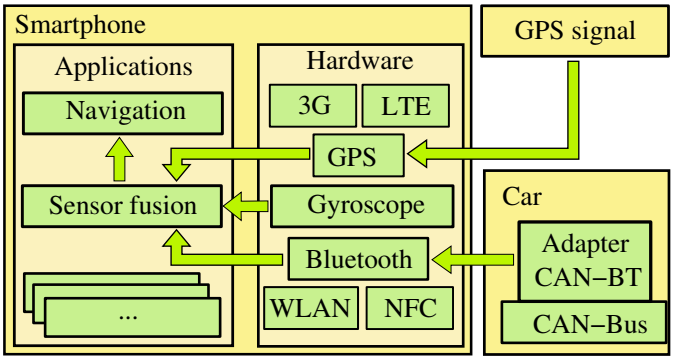

Fig. 1. System overview showing hardware and software components

A. Global Positioning System

The built-in GPS receiver delivers position information in the NMEA 0183 string format. GPS information is the main source of positioning information. In case of GPS signal loss, the last GPS position information available before the dropout is used to initialize the IMU-based positioning.

B. Gyroscope

The gyroscope delivers information about the angular velocities of the smartphone in three orthogonal directions. However, the measurements of the inexpensive smartphone sensor suffer from bias errors and sensor data drifts such that a post filtering is required.

C. CAN-Bus-to-Bluetooth Adapter

A CAN-BT adapter is plugged to the car CAN-Bus, which continuously reads the CAN-Bus data and sends them via Bluetooth to the smartphone. In the work presented here we only employ the car speed information read from the CAN-Bus. Beforehand, a negotiation procedure for pairing the smartphone and the CAN-BT adapter has to be performed. A pairing procedure via NFC is planed for future versions of the system to improve the usability. So far, tests in different cars from various manufacturers showed no compatibility problems.

D. Data Rates

The GPS receiver offers position information at a rate of 1 positioning sample per second. Gyroscope data is available at an average data rate of 21 values per second, while the CAN-Bus data are transmitted via Bluetooth at a rate of 2 speed values per second.

The sensor fusion application harmonizes the different data rates of the sensors by increasing the sampling rates of the low rate data streams using sample and hold blocks. Computing position estimates at this high rate has the advantage, that improved positioning precision in-between GPS values is available compared to just holding the GPS output until the next GPS position estimate becomes available.

III. Sensor Fusion Algorithm

In Fig. 2 the block diagram of the sensor fusion is depicted. The colors of the computing blocks and connections indicate the processing/data rates. The strapdown algorithm operates at the high data rate of the gyroscope (blue color). We run the Kalman filter with the GPS data rate (orange color) and use sample and hold (S&H) blocks to harmonize its data stream with the strapdown processing block.

The main idea of the sensor fusion algorithm is to combine the absolute value of the velocity v_{CAR} of the car provided by the CAN-BT adapter with the direction of movement information $\hat{\psi}_{\mathrm{CAR}}$ of the car provided by the smartphone to compute an estimate of the change in position of the car. $\hat{\psi}_{\mathrm{CAR}}$ is obtained either by fusing direction information of GPS and IMU using an error-state Kalman filter or by the output of the IMU alone in case of GPS signal loss.

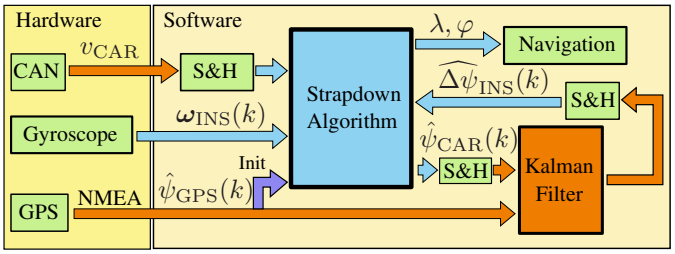

Fig. 2. Block diagram of the sensor fusion approach

In the following we are going to briefly outline the strapdown algorithm. The three-dimensional angular velocity vector $\boldsymbol{\omega}_{\mathrm{INS}}$ provided by the gyroscope sensors of the smartphone is used in the strapdown inertial navigation algorithm to calculate an estimate of the Euler angles roll ϕ, pitch θ and yaw ψ, which describe the orientation of the smartphone in 3-dimensional space [8], [10]. Let

$$\boldsymbol{r}(k) = \begin{pmatrix} \cos\left(\frac{\|\boldsymbol{\omega}_{\mathrm{INS}}(k)\| \cdot T}{2}\right) \\ \frac{\boldsymbol{\omega}_{\mathrm{INS}}(k)}{\|\boldsymbol{\omega}_{\mathrm{INS}}(k)\|} \sin\left(\frac{\|\boldsymbol{\omega}_{\mathrm{INS}}(k)\| \cdot T}{2}\right) \end{pmatrix} \tag{1}$$

denote the quaternion formed from the angular velocity vector measured by the gyroscope, where k denotes the discrete time index and T the sampling interval of the gyroscope. The strapdown inertial navigation algorithm iteratively calculates the current orientation $\boldsymbol{q}(k)$ based on the orientation in the last time step and current sensor data via

$$\boldsymbol{q}(k) = \boldsymbol{q}(k-1) \bullet \boldsymbol{r}(k), \tag{2}$$

where the operand $\bullet$ denotes quaternion multiplication. The calculation of the orientation as a quaternion was chosen to guarantee stability and avoid a gimbal-lock which could occur when directly using Euler angles instead. The estimate of the vector of Euler angles $\hat{\boldsymbol{\Psi}}_{\mathrm{INS}} = \left(\hat{\phi}_{\mathrm{INS}}, \hat{\theta}_{\mathrm{INS}}, \hat{\psi}_{\mathrm{INS}}\right)^{\mathrm{T}}$ is then obtained by transformation of the quaternion

$$\hat{\boldsymbol{\Psi}}_{\mathrm{INS}}(k) = f_{\mathrm{q2e}}\left(\boldsymbol{q}(k)\right), \tag{3}$$

where the function f_{q2e} denotes the transformation of the orientation stored as a quaternion to Euler angles via the direction cosine matrix [10]. The resulting value of the yaw angle $\hat{\psi}_{\mathrm{INS}}(k)$ is then taken as the estimate of the course of the car if no GPS is available:

$$\hat{\psi}_{\mathrm{CAR}}(k) := \hat{\psi}_{\mathrm{INS}}(k). \tag{4}$$

Equating the direction of movement of the car with the yaw angle of the smartphone, irrespective of the orientation of the phone in 3-dimensional space is allowable, because we can freely rotate around the local vertical, as long as we are not interested in the remaining angles $\hat{\phi}_{\mathrm{INS}}$ and $\hat{\theta}_{\mathrm{INS}}$ relative to the car.

By combing the direction of movement with the absolute value of the velocity of the car, v_{CAR}, read out from the CAN-BT device, we arrive at an estimate for the change in position. Assuming a movement on a 2-dimensional plane, where $\hat{\psi}_{\mathrm{CAR}}$ is the angle of the direction of movement of the vehicle relative to the north direction, counting from the north direction to the east direction, we can estimate the velocity of the vehicle in the north v_{n} and the east direction v_{e} using the following equations:

$$v_{\mathrm{n}}(k) = v_{\mathrm{CAR}}(k) \cdot \cos\left(\hat{\psi}_{\mathrm{CAR}}(k)\right) \tag{5}$$

$$v_{\mathrm{e}}(k) = v_{\mathrm{CAR}}(k) \cdot \sin\left(\hat{\psi}_{\mathrm{CAR}}(k)\right) \tag{6}$$

The car position is then stored in GPS coordinates as latitude φ and longitude λ, which can be calculated using the following equations, where $R = 6371 \cdot 10^3\mathrm{m}$ is the mean earth radius:

$$\varphi(k) = \varphi(k-1) + \frac{T \cdot v_{\mathrm{n}}(k)}{R} \tag{7}$$

$$\lambda(k) = \lambda(k-1) + \frac{T \cdot v_{\mathrm{e}}(k)}{R\cos\left(\varphi(k)\right)}. \tag{8}$$

Due to the integration of sensor data over time and the iterative calculation of the orientation using the strapdown inertial navigation algorithm, sensor errors such as additive noise and sensor bias will result in an error in the estimated orientation which increases over time. This is compensated by fusing the INS positioning information with GPS information in case the GPS signal is available. This sensor fusion is done using an error-state Kalman filter (ESKF), as proposed, e.g., in [9]. The ESKF estimates errors in the navigation solution rather than the navigation solution itself. This has various advantages, among which the reduced complexity due to the lower data rate is most important for a realization on a smartphone, which should eat up as little battery power as possible. Note that the ESKF operates at the lower GPS data rate rather than at the high gyroscope data rate, see Fig. 2.

To even further reduce the complexity we assume a scalar error model:

$$\hat{\psi}_{\mathrm{INS}}(k) = \psi(k) + \Delta\psi_{\mathrm{INS}}(k) \tag{9}$$

$$\Delta\psi_{\mathrm{INS}}(k) = \Delta\psi_{\mathrm{INS}}(k-1) + w(k), \tag{10}$$

where $\psi(k)$ denotes the true direction of movement of the car at the time index k. The estimation error in direction is denoted by $\Delta\psi_{\mathrm{INS}}(k)$ and modeled as a random walk process, where $w(k) \sim \mathcal{N}\left(0, \sigma_w^2\right)$ is white Gaussian noise modeling the change in the orientation error at every time step.

The error in the orientation, $\Delta\psi_{\mathrm{INS}}(k)$, is estimated using the error-state Kalman filter. To obtain a measurement of this error we use the GPS measurement

$$\hat{\psi}_{\mathrm{GPS}}(k) = \psi(k) + v(k) \tag{11}$$

of the course over ground (cog), where $v(k) \sim \mathcal{N}\left(0, \sigma_v^2\right)$ is the error in the GPS measurement which is modeled as white Gaussian noise. A measurement $\Delta\tilde{\psi}(k)$ for the error in orientation is then given by

$$\begin{aligned}\Delta\tilde{\psi}(k) &= \hat{\psi}_{\mathrm{INS}}(k) - \hat{\psi}_{\mathrm{GPS}}(k) \\ &= \hat{\psi}_{\mathrm{INS}}(k) - (\psi(k) + v(k)) \\ &= \Delta\psi_{\mathrm{INS}}(k) - v(k).\end{aligned} \tag{12}$$

This is the measurement equation of the ESKF, while the state equation is given by eq. (10) above.

The estimate $\widehat{\Delta\psi}_{\mathrm{INS}}(k)$ for the orientation error and its a-priori error variance $p(k|k-1)$ can be calculated by the following equations, where $g(k)$ denotes the Kalman gain:

$$g(k) = \frac{p(k|k-1)}{p(k|k-1) + \sigma_v^2} \tag{13}$$

$$\widehat{\Delta\psi}_{\mathrm{INS}}(k) = g(k)\Delta\tilde{\psi}(k) \tag{14}$$

$$p(k+1|k) = p(k|k-1)\left(1 - g(k)\right) + \sigma_w^2. \tag{15}$$

Note that no state update is required, since the filter's state vector is reset to zero after each iteration and correction of the direction [8].

After the estimation of the error we use equation (9) to correct the orientation estimate:

$$\hat{\psi}^{+}_{\mathrm{INS}}(k) = \hat{\psi}_{\mathrm{INS}}(k) - \widehat{\Delta\psi}_{\mathrm{INS}}(k). \tag{16}$$

This is the improved estimate of the car course, $\hat{\psi}_{\mathrm{CAR}}(k) := \hat{\psi}^{+}_{\mathrm{INS}}(k)$, which can be computed if GPS is available. It replaces the estimate of eq. (4) and is used in eq. (5) and eq. (6) to determine the velocity and, subsequently, the position of the car. It is further used in the next time step for the orientation calculation in the strapdown inertial navigation algorithm. Because of this feedback of the error to the orientation calculation the filtering approach is also called a feedback error-state Kalman filter.

The remaining elements of the Euler angles vector $\hat{\mathbf{\Psi}}_{\mathrm{INS}}$ can be supported by a similar algorithm as described above. In the case of the roll angle $\hat{\phi}_{\mathrm{INS}}$ and the pitch angle $\hat{\theta}_{\mathrm{INS}}$ one could use the measurements of the accelerometers and the local gravity estimate to calculate the corresponding angles and then form a error-state model similar to eq. (10).

IV. Experimental Results

In this section we describe three test runs in a car to demonstrate the benefits of our approach in a real world scenario, before carrying out a more quantitative evaluation in the subsequent section. Each of the test runs demonstrates

different properties of the proposed system. In the first experiment the car position is estimated assuming all sensor data to be available. Next, absence of GPS position information is assumed, while GPS direction information is still used to correct the IMU's direction estimates. Third, neither GPS position nor direction information is used to track the car.

Fig. 3 displays the results of the first test run. Here, we periodically used the GPS position information (red circles) to re-initialize the position estimate of the strapdown algorithm. In between two GPS estimates, the position of the car was predicted using the corrected direction estimate, eq. (16), and the car's velocity according to eqs. (5) - (8) (blue curve). Note that the sharp right turn could be better predicted by this sensor fusion than by holding the GPS estimates until the next GPS position estimate becomes available. This mode will be used when GPS measurements are available to increase the position accuracy in between two measurements.

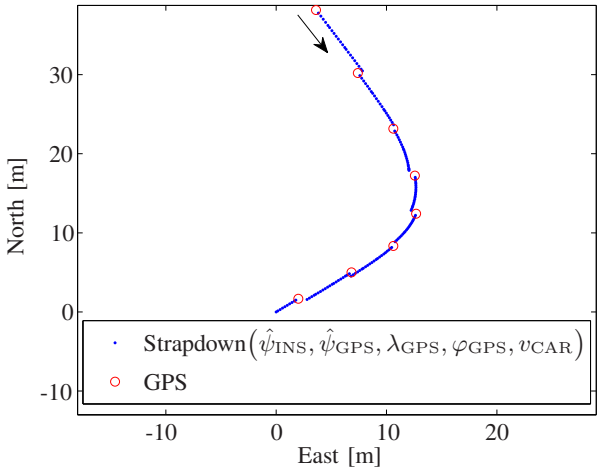

Fig. 3. Position estimates of sensor fusion (blue curve) and GPS alone (red circles) when driving a right curve (arrow denotes driving direction)

In a second test run, depicted in Fig. 4, we drove around a block of houses in clockwise direction. During this experiment the GPS position estimate was used once to initialize the first position. The arrow indicates this position and the driving direction. From there on, the GPS position information was dropped and only the GPS direction information was used to calculate the measurement input of the ESKF. This setup was chosen to demonstrate the usefulness of the velocity information of the car's speedometer when combined with the direction estimate. We again used the strapdown algorithm in combination with the Kalman filter to estimate the position of the car (blue curve). The red circles indicate the GPS estimate of the car position, which are included in the figure to serve as a kind of ground truth.

In the third test run, depicted in Fig. 5, we drove around the same block of houses as in the previous test run. This time we also dropped the use of the GPS direction estimate, thus simulating a true GPS dropout. The GPS position and direction estimate was used once at the beginning of the trajectory to

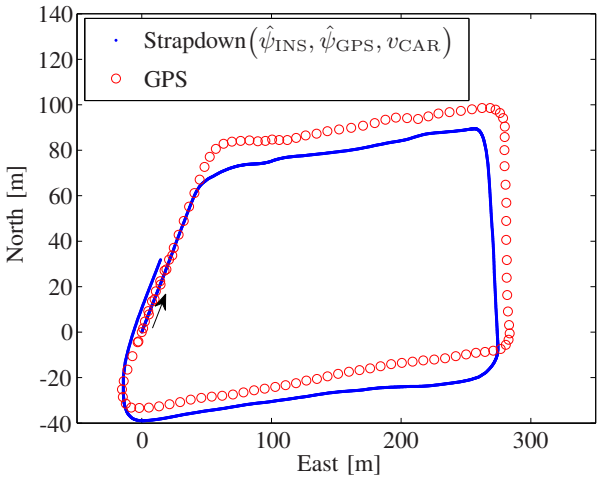

Fig. 4. Position estimates of sensor fusion without GPS position information (blue curve) and the GPS position estimate (red circles) as an indication of the ground truth

initialize the strapdown algorithm. For the rest of the trajectory we used the strapdown algorithm to predict the position using the uncorrected direction estimate of the IMU and the car's speed and no GPS information whatsoever. This experiment clearly shows the benefit of our algorithm to track the car in case of a GPS dropout. During the first and second turn we were able to maintain track of the direction and position change, while the third turn was not estimated quite as well due to error accumulation of the uncorrected IMU in the course of time.

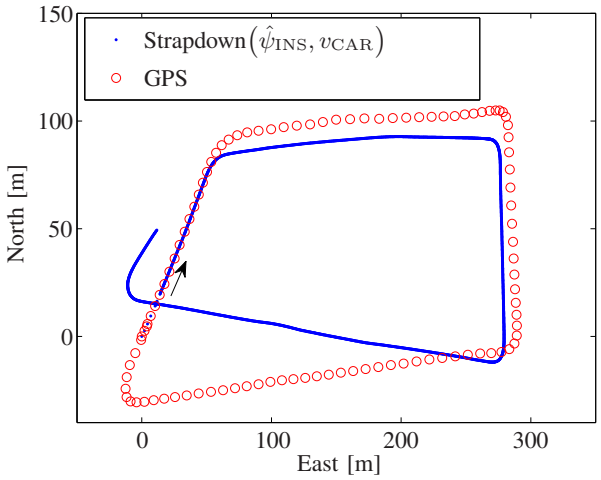

Fig. 5. Position estimates solely based on IMU and car speed data simulating a GPS dropout (blue curve). GPS position estimate (red circles) as an indication of the ground truth

V. Simulation Results

To quantitatively evaluate the performance of our algorithm we performed some simulations. For the simulations we used

recorded field data and simulated GPS dropouts in circular regions around randomly chosen positions $\boldsymbol{p}_{\text{drop}}$ with a given radius r. This way of simulating a GPS dropout accounts for the fact that a GPS signal loss is usually spatially localized, i.e., it occurs in the vicinity of obstructions like buildings or tunnels. As no other ground truth was available, we again employed the GPS position estimates and linearly interpolated between two estimates to get the vector $\boldsymbol{p}_{\text{GPS}}(k)$ as the true values of the position. Clearly, a GPS dropout for a given GPS "ground truth" measurement $\boldsymbol{p}_{\text{GPS}}(k)$ is assumed if the condition

$$\|\boldsymbol{p}_{\text{GPS}}(k) - \boldsymbol{p}_{\text{drop}}\| < r \tag{17}$$

is met, else there will be no dropout.

During a GPS dropout our algorithm uses the uncorrected direction estimate of the IMU and the car's speed to compute the position estimate, denoted by $\boldsymbol{p}_{\text{INS}}(k)$. As a baseline, against which we compared the performance of our algorithm, we kept the last known GPS course and speed before the dropout constant and used them to predict the following positions, denoted by $\boldsymbol{p}_{\text{hold}}(k)$.

As a performance measure we used the root mean square error (rmse) Δp between the positions $\boldsymbol{p}(k)$ estimated by the algorithms and the ground truth GPS positions:

$$\Delta p = \sqrt{\frac{1}{N}\sum_{k=1}^{N} \|\boldsymbol{p}(k) - \boldsymbol{p}_{\text{GPS}}(k)\|^2}, \tag{18}$$

where $\boldsymbol{p}(k)$ is to be replaced by either $\boldsymbol{p}_{\text{INS}}(k)$ or $\boldsymbol{p}_{\text{hold}}(k)$, and N is the number of position estimates during dropout.

For the simulations we used measurements recorded during a test drive of approximately 6.5km length and 10 minutes duration. The recorded data consisted of approximately 600 GPS, 13000 gyroscope and 1200 speed measurements. During the simulations each of the 600 GPS measurements was chosen alternately as dropout center $\boldsymbol{p}_{\text{drop}}$, and the dropout radius was modified from 50 to 600 meters in steps of 50 meters.

TABLE I
RMS OF PROPOSED ALGORITHM (Δp_{INS}) AND BASELINE ALGORITHM (Δp_{hold}) DURING DROPOUT AND ERROR AT THE END OF A DROPOUT. ALL VALUES IN METERS.

r [m]	Δp_{INS}	Δp_{hold}	$\Delta p_{\text{INS}}(N)$	$\Delta p_{\text{hold}}(N)$
50	7.0	31.4	10.8	52.1
100	12.7	59.5	22.3	113.2
150	21.3	91.9	39.6	179.6
200	31.9	128.8	63.8	259.6
250	51.4	165.3	103.9	333.8
300	73.3	198.0	152.5	393.6
350	96.1	233.1	211.6	449.5
400	129.0	275.1	289.8	521.6
450	160.4	316.4	385.2	596.1
500	203.4	351.5	492.8	652.9
550	253.7	397.2	608.1	732.4
600	306.9	452.9	707.7	817.1

Table I shows the resulting rmse values for different dropout radii. Additionally the rmse values $\Delta p_{\text{INS}}(N)$ and $\Delta p_{\text{hold}}(N)$ for the last position estimates during the dropouts are given as an indication of the maximum error which presumably occurs at the end of a dropout. The positioning error during a GPS dropout using our algorithm is much smaller compared to a position estimate based on the last valid GPS course and speed value before dropout. Further, the error during and at the end of a dropout is almost always smaller than the radius of the dropout region when using our algorithm. It is not surprising that the performance of the proposed algorithm decreases with increasing dropout radius r. This is due to the error accumulation of the uncorrected IMU in the course of time.

The figures 6 and 7 show two examples of a GPS dropout with a radius of 100 meters. In Figure 6 a left turn was taken during a dropout. Our algorithm (blue curve) is capable of following the GPS (ground truth) trajectory as indicated by the red circles. On the other hand, holding the GPS course and speed (red curve) results in a straight line and a big difference to the ground truth track, since there is no information available about the change in direction. This scenario could, e.g., occur for a left turn at an intersection. Using our algorithm a navigation system will be able to correctly navigate to the destination while in the case of holding the GPS velocity and course it could assume to be on the wrong road and start giving false direction information.

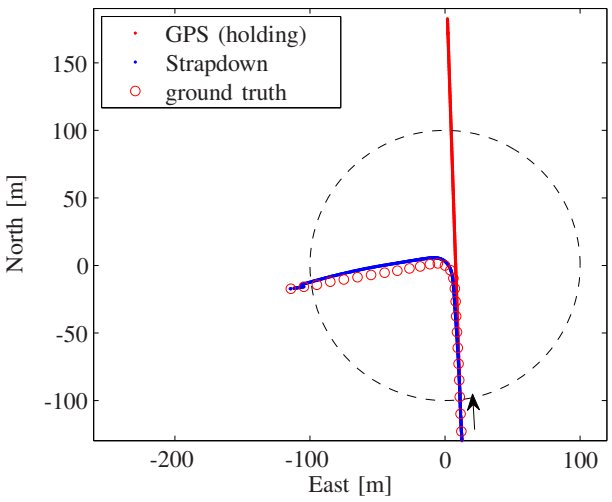

Fig. 6. Position estimates solely based on IMU and car speed data (blue curve) and by holding course and GPS speed (red curve) simulating a GPS dropout wile taking a turn. GPS position estimate (red circles) as an indication of the ground truth. Region of dropout is the interior of black dashed circle

Figure 7 shows a GPS dropout while driving an almost straight line. Our algorithm follows the GPS track and finishes close to the last ground truth position. Holding the GPS course seems to be a good idea while driving a straight line, but holding the GPS speed still results in an increasing error in position. The reason for this is that acceleration and deceleration of the car remain unnoticed during the GPS dropout.

The simulation results show the advantages of our algorithm

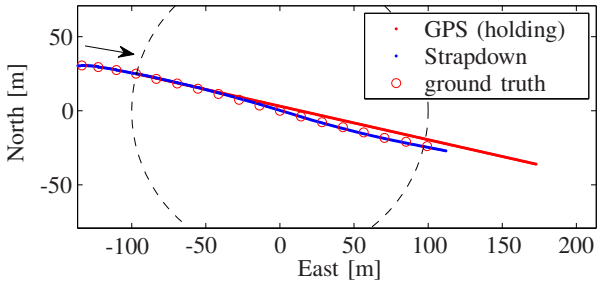

Fig. 7. Position estimates solely based on IMU and car speed data (blue curve) and by holding course and GPS speed (red curve) simulating a GPS dropout wile driving straight. GPS position estimate (red circles) as an indication of the ground truth. Region of dropout is interior of black dashed circle

as it keeps track of the change in velocity by directly using the measurements provided by the speedometer of the car and the change in course by using the smartphone's gyroscope to estimate course and position.

VI. Conclusions

We have presented a smartphone based sensor fusion algorithm which combines GPS information with the smartphone's internal gyroscope and the car's speedometer data to track the position of a moving car. Sensor fusion is carried out using a computationally inexpensive error-state Kalman filter.

Test drives and simulations have been carried out to demonstrate the usefulness of the sensor fusion, in particular in the case of GPS dropouts. The experiments showed that the proposed algorithm achieves significantly improved positioning accuracy compared to predicting the position on the basis of the most recent GPS course and speed estimate before the GPS signal loss.

The developed approach combines the advantages of smartphone-only positioning and positioning by built-in vehicle navigation systems: It preserves the cost effectiveness and flexibility of smartphone based navigation systems, while at the same time achieving a positioning accuracy and availability towards (expensive) IMU stabilized navigation systems of cars.

In future work we will extend the sensor fusion to include information of a digital map to further increase the accuracy and allow navigation through larger GPS dropout regions. Also, the possibility of using more complex error models and filter algorithms will be investigated.

References

[1] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy, Precise Indoor Localization Using Smart Phones, in *Proc. of ACM International Conference on Multimedia*, Florence, Italy, pp. 787-790.
[2] Ford, Sync Technology, http://www.ford.com/technology/sync/, 2011.
[3] Mercedes-Benz MBrace, http://www.mbusa.com/mercedes/mbrace/convenience, 2011
[4] Toyota Entune, http://www.toyota.com/entune/, 2011
[5] BMW ConnectedDrive, http://www.bmw.com/com/en/insights /technology/connecteddrive/2010/index.html, 2011
[6] Toll collect GmbH, http://www.toll-collect.de, 2011.
[7] Android homepage, http://http://www.android.com, 2011.
[8] J. Wendel. Integrierte Navigationssysteme, Oldenburg, 2007.
[9] S. Roumeliotis, G. Sukhatme, and G. Bekey, Circumventing Dynamic Modeling: Evaluation of the Error-State Kalman Filter applied to Mobile Robot Localization, in *Proc. IEEE International Conference on Robotics and Automation*, Detroit, May 1999.
[10] M. Titterton, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, Inc., 2001.
[11] Bevermeier, M., Walter, O., Peschke S. and Haeb-Umbach, R., Barometric Height Estimation Combined with Map-Matching in a Loosely-Coupled Kalman-Filter, in *Proc. 7th Workshop on Positioning, Navigation and Communication 2010 (WPNC'10)*, Dresden, Germany, March 11-12, 2010