Techniques for improving localization applications running on low-cost IoT devices

Evelina Forno*, Simone Moio§, Michael Schenatti§, Enrico Macii‡ and Gianvito Urgese‡
*DAUIN Politecnico di Torino, 10138, Torino (TO), ITALY. Email: evelina.forno@polito.it
†Tierra S.p.A., 33020, Leiní (TO), ITALY. Email: smoio@tierratelematics.com
‡DIST Politecnico di Torino, 10138, Torino (TO), ITALY. Email: gianvito.urgese@polito.it

***Abstract*—Nowadays, localization features are widespread in low-cost and low-power IoT applications such as bike-sharing, off-road vehicle fleet management, and theft prevention of smart devices. For such use cases, since the item to be tracked is inexpensive, older or power-constrained (e.g. battery-powered vehicles), localization features are realized by the installation of low-cost and low-power devices. In this paper, we describe a set of low-computational power techniques, targeting low-cost IoT devices, to process GPS and INS data for accomplishing specific and accurate localization and tracking tasks. The methods here proposed address the calibration of low-cost INS comprised of accelerometer and gyroscope without the aid of external sensors, correction of GPS drift when the target position is static, and the minimization of localization error at device boot. The performances of the proposed methods are then evaluated on several datasets acquired on the field and representing real use-case scenarios.**

***Index Terms*—Localization, Tracker, Inertial navigation system, GPS/INS Integration, Micro electro mechanical systems (MEMS), global positioning system (GPS), Automotive, IoT, IMU**

I. Introduction

Localization methods have long been employed in vehicle navigation to provide efficient navigation solutions, with systems greatly varying in precision and cost, depending on the target application (civilian vs. military, etc.). In addition to this, localization-based services are emerging as an attractive aspect of the IoT field, with applications related to environmental monitoring [1] as well as product operations and tracking in life cycle management [2], traffic tracking [3], optimise usage of battery in drone controlling [4], and sport monitoring [5].

Localization and tracking methods usually depend on the use case. For instance, indoor localization can be performed by monitoring the radio signal strength of known beacons utilizing Bluetooth [6], while outdoors applications can use Wi-Fi beacons [7] or the LoRa network [8]. Mobile phones and other GSM-enabled devices can be tracked by triangulating the position of local cell towers [9]. Robots and self-driving vehicles can take advantage of visual systems such as cameras [10] and LiDAR [11]. Localization and tracking of mobile targets in an open field generally relies on the integration of information from the Global Positioning System (GPS) with the output of local sensors, in order to improve the accuracy and the robustness of the result. The sensors used are usually part of an Inertial Measurement Unit (IMU) including devices able to measure motion (accelerometer) and rotation (gyroscope) in the local reference frame.

The spread of low-cost localization systems has been greatly facilitated by the introduction of Microelectromechanical Systems (MEMS)-based technology for IMU modules. This made affordable the mass production of lightweight, miniaturized and inexpensive integrated devices that can be easily included in the design of new appliances or strapped to existing equipment. However, the convenience of a MEMS-based IMU comes at a great cost of accuracy, as the sensors are plagued by stochastic errors that can worsen in adverse environmental conditions, as well as systematic non-linearities [12].

This paper proposes techniques for calibration and performance improvement of low-budget GPU/INS integration systems. In section II, we give an overview of existing GPS/INS integration and augmentation methods and their applicability to low-cost devices. In section III, we illustrate the proposed techniques and explain our method for collecting experimental datasets in the field. In section IV, we demonstrate the results of the proposed techniques using selected datasets, and give an estimation of the computational costs. Finally, section V contains our comments and conclusions.

II. Background

GPS augmentation techniques include algorithms that integrate knowledge from local maps [13] [14], neural network-aided fusion [15], and the integration of other data such as the Position Dilution of Precision (PDOP), a figure produced by the GPS that indicates the degree of confidence for the given solution [16]. However, by far the most widespread and successful technique is GPS/INS integration by Kalman Filter.

The traditional Kalman Filter [17] is an optimal filter for linear systems, where noise is independent, random, and additive. However, in the case of land navigation and localization, the abundance of environmental noise sources as well as nonlinearities inherent in the sensor equipment lead to suboptimal results when using a linear filter [12]. For this reason, traditionally, the most used fusion filter in the field is the Extended Kalman Filter (EKF), based on nonlinear error models. However, one important drawback of the EKF is that if the sensor errors violate the principle of local linearity, the filter quickly diverges [18], especially when dealing with low-cost MEMS [19]. Moreover, the EKF is computationally intensive since it requires calculation of Jacobian matrices [20], which makes its implementation prohibitive on devices based on low-computational power microcontrollers.

The operating principle of any Kalman Filter is the probabilistic integration of data from independent (complementary) sensors; the error sources for the GPS and INS are completely decoupled and therefore can cancel each other out. The presence of many independent sensors can vastly improve the performance of such a system; however, most low-to-mid-budget devices do not include more than an inexpensive MEMS-based accelerometer and, at most, a gyroscope. In this paper we seek to optimize the use of such limited resources, by reducing IMU errors without the aid of external sensors and introducing methods to use IMU information to improve the GPS solution with a low computational effort.

One important correction for the INS is alignment to the navigation frame and mounting-angle detection. Static calibration of the INS sensor axes to the gravity plane can be easily attained using only an accelerometer [21]. However, when it comes to in-motion alignment and yaw detection, more sophisticated techniques are necessary. Many such techniques are designed for navigation-grade IMUs [22] , which are vastly more precise than inexpensive MEMS and operate at higher frequencies than low-cost, low-power computing systems can handle. Techniques suitable for MEMS often require additional sensors such as a magnetometer [5], an odometer [23], a barometer [24], or a GPS calculating the velocity vector [25].

III. Methods

In this section, we will investigate operations that can be performed before INS/GPS fusion to aid the fusion filter and improve localization results, while maintaining a low computational cost. In section III-A, we present the block diagram of a possible configuration of preprocessing steps, and we describe in detail the operation of each algorithmic block. In section III-B, we explain the reasoning, planning and technique behind the acquisition of field data, for the purpose of designing and testing solutions to specific issues encountered during real-life usage of the proposed system.

A. Low-Effort Techniques

The full model of the GPS/INS integration model is described in Fig. 1. The colored blocks (identified by numbers) represent the preprocessing steps that can reduce the computational effort of the following integration steps, and are described in the following subsections.

1) Saving the last known position: This is a very simple measure that, nevertheless, brings important benefits. Fundamentally, the most recent position of the target is periodically saved to memory. When the device is powered on, the last known position is loaded from memory; since most localization devices can be woken up by the accelerometer, it is reasonable to assume that the position of the target has not significantly changed since power down. This allows us to have a reliable starting position even in the event that the GPS takes a long time to startup and produce a quality fix; we also set a minimum acceptable quality for GPS solutions, which we evaluate by checking the PDOP field of the GPS output.

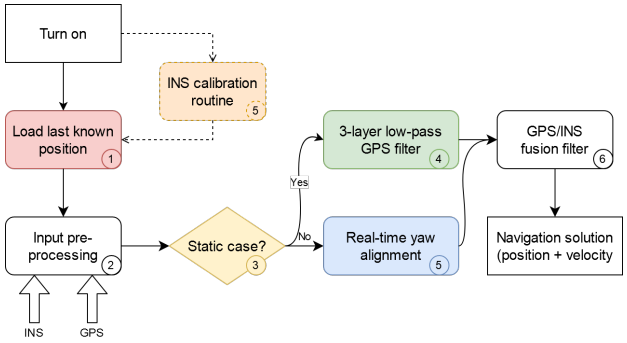

Fig. 1. Block diagram of low-effort techniques in a localization system.

2) Input preprocessing: This block is a cascaded Kalman Filter which follows part of the implementation described in [5]. It takes as input the raw, time-sequenced GPS and INS data and uses separate filters to generates the rotation matrix R_b^n from the sensors' reference frame to the world coordinates (North, East, Down). A simplified diagram of this block is provided in Fig. 2.

The Tilt Kalman Filter integrates the accelerometer and gyroscope outputs to extract the roll and pitch angles through evaluation of the gravity vector.

The Yaw Kalman Filter is dedicated to the calculation of the yaw angle; in the original work, this was attained by integrating information from a gyroscope and a magnetometer, however, in the low-cost boards that are within our scope of interest, a magnetometer is often not available. The only other source of orientation information available on our board is the heading angle measured by the GPS when the vehicle is moving, and we use this value as the integration factor for the gyroscope output. However, since the gyroscope measures rotations in the sensor's body frame, and the GPS measures heading in the navigation frame, one crucial piece of information is missing: the vertical-axis angle of rotation from the sensor frame to the vehicle frame. We will see in section III-A5 how this data is calculated by module (5).

Finally, the Gyro Bias Kalman Filter iteratively computes the bias correction to be applied to the gyroscope output.

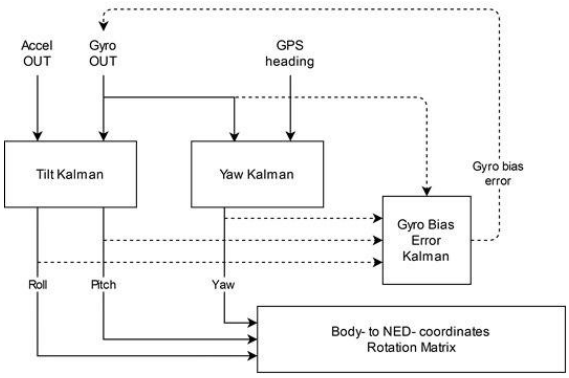

Fig. 2. Simplified diagram of the Input preprocessing block (2).

3) Static case determination: In order to refine the processing of the data received from the sensors, the system needs to identify whether the vehicle is moving or static. This particular feature can be useful to drive the activation of other analysis procedures, as in the case proposed in Fig. 1.

Due to fluctuations in the position produced by the GPS, it is more reliable to make this check by testing the INS output. Using the rotation matrix obtained by block ② and the accelerometer output in the sensor frame y_A^b, we obtain the measured acceleration in the navigation frame by Eq. 1:

$$y_A^n = R_b^n \times y_A^b \tag{1}$$

The speed over ground reading is acquired from the GPS and optionally filtered (e.g. by a moving average) to minimize drift errors. When this value drops below a certain threshold, we check the norm of the North and East elements of y_A^n, i.e., the portion of the accelerator vector lying in the horizontal plane. If this acceleration is smaller than a threshold θ_A, we assume to be in the static case. For the experiments in this paper, we set $\theta_A = 0.125\ \mathrm{m/s^2}$.

4) 3-layer low-pass GPS filter: The GPS output is more reliable when the vehicle is moving. In the event that the vehicle remains in the same location for a long time, the GPS position can begin to drift wildly. This is undesirable behavior, in the sense that it can give false positives when trying to determine whether the vehicle has exited a virtual geofence.

In order to make up for this problem we have inserted a 3-tier low-pass filter for the GPS position. This filter is activated only when the static case is detected in block ③. At this point, the GPS positions begin to be saved into a buffer.

When the 1st-level buffer fills up, the valid position is taken as the average of all positions in the buffer, and saved into the 2nd-level circular buffer. In this way, the 2nd-level circular buffer is filled with averages over subsequent time periods of the positions in the 1st buffer; likewise, once the 2nd buffer fills up, the 3rd level circular buffer is filled with the averages of the positions in the 2nd buffer. The latest average of the highest-level buffer available is taken at any time as the new GPS measurement; the device effectively goes into a sort of "sleep mode", where position updates progressively slow down as long as the vehicle stands still. The system, however, continues to work at the usual frequency; as soon as the horizontal acceleration grows greater than θ_A, the algorithm immediately returns to the normal update regime.

5) Yaw angle alignment: The cascaded Kalman Filter in block ② is able to extract the rotation matrix R_b^n in real time for any INS device that is equipped with an accelerometer and gyroscope. However, it is not able to extract the mounting angle on the yaw/Z axis, because this angle cannot be deduced from the gravity vector. This angle is necessary, for example, to reconcile the data recorded by the INS (in the sensor frame) with the speed-over-ground and heading angle measurements recorded by the GPS (relative to the vehicle movement in the navigation frame), by completing the transformation from the sensor frame to the vehicle frame. In the original work [5] this problem is solved by including a magnetometer; however, many low-cost IMUs are equipped only with an accelerometer and gyroscope. We then set out to find a way to overcome this limitation, using a software approach that exploits the non-holonomic constraints [26] on the axes of movement for land vehicles, that is, assuming that the acceleration component on the Y-axis of a vehicle moving on a straight path is zero.

First of all, we need to identify periods of time when the vehicle is moving in a straight path. To this end, any time the vehicle is not in the static case, the direction of movement (i.e., the heading angle α measured by the GPS in the navigation frame) is saved into a buffer, together with the accelerometer output y_A^b. In cases where the vehicle is subject to significant engine vibrations, the accelerometer input may need to be filtered before this step, e.g. using Fast Fourier Transform.

Whenever the buffer is full, we test the variance of α over the buffer; if it is smaller than a chosen threshold θ_α, we have identified a straight path. The approximate mounting error is then found using the mean of the acceleration data in the buffer, $\bar{y}_A^b$, minus the gravity vector, using Eqs. 2 and 3:

$$[a_x a_y a_z]_T = (R_b^n \times \bar{y}_A^b) - g_n \tag{2}$$

$$\epsilon_\alpha = \arctan\left(\frac{a_y}{a_x}\right) \tag{3}$$

This method can update the mounting angle in real time as the system is running, at any time when a straight path is encountered. Alternately, in cases where the INS is firmly fixed to the vehicle frame, this estimation may be executed as an optional, one-off calibration routine, in which case the error may be saved to memory, saving subsequent computations.

B. Datasets acquired on the field

In order to develop and validate the system, data must be acquired from the sensors during real-time tests in various environments [25] [27] [28]. In fact, by using simulated data we would not be able to properly evaluate the impact of sensor errors and non-idealities; the use of data gathered from the real world is crucial for the estimation of the applied methods, since it allows us to obtain a GPS data stream that's coherent with INS data describing the same route. For this motivation, all the analyses proposed in this paper are performed on datasets physically acquired on the field.

To that end, we developed a firmware which performs the dump of raw sensor data into formatted files. For the INS gyroscope and accelerometer data, a CSV-formatted file is produced directly, containing the timestamped outputs of both sensors. For the GPS data, the program dumps the stream of NMEA strings into a text file while interspersing it with timestamps. The text files form the devices are post-processed by a Python parser that produces a CSV file integrating the time-stamped data from both sensors into a single timeline.

Dataset acquisition should be planned in order to obtain a collection of scenarios that are both consistent with real-life usage and presenting features that allow for the identification of well-defined critical cases and that can be used as benchmarks aimed at evaluating the performance of specific

modules of a system, both in isolation and in concert with other modules. A few examples of such scenarios include:

- Routes where the IMU is mounted with its sensors axes misaligned with respect to the vehicle's reference system, which can be used to evaluate self-alignment of the IMU with respect to the vehicle's coordinate frame.
- Highly dynamic routes including many variations in the inclination and orientation of the vehicle; these datasets can be used to evaluate real-time self-alignment of the IMU with respect to the navigation coordinate frame.
- Keeping the device turned on in a static position for a long time; the drift of GPS position error while a vehicle is static can be evaluated, as well as any correction measures.
- Routes that include sources of GPS errors (such as obstructions, cloudy weather, or electromagnetic noise) or outright GPS outages (such as closed buildings or tunnels). A GPS outage can also be simulated *a posteriori* by deleting selected portions of the GPS output, while modelling environmental errors is more complex. Using such datasets, dead reckoning and correction capabilities of a fusion system can be tested.
- Datasets recorded starting from a cold boot of the GPS system, which show the time necessary for the first valid fix to be produced and can be used to design and test measures to make up for this systematic outage, such as loading previous positions from memory and PDOP-based corrections.

IV. Results

In this section, we illustrate the details of three experimental datasets (A, B and C), acquired in the field and selected with the purpose of highlighting the effects of the processing techniques described in this paper. The results of the input preprocessing methods applied to each dataset are presented in sections IV-A, IV-B and IV-C, while IV-D contains an estimation of the computational cost of each proposed block.

In dataset A, the device was placed in a courtyard and left in the same position for about 90 minutes. The location presents some non-idealities for visibility and multipath errors as the sky is partially obscured by trees and buildings. This dataset is useful for evaluating the shift of the position and the average error of the accelerometer when the board is in a steady condition. We use this dataset to validate the effectiveness of module ④ in converging to the correct steady point while the GPS position drifts for a long time.

In dataset B, the device was mounted on the deck of an electric scooter which was driven following a racing track around a rugby field. The scooter's deck runs parallel to the ground. Multiple laps were recorded at an average speed of 17.6 km/h. The path followed is the most external white line marking the track boundary; this gives us a reference on the ground. The device is mounted with its X axis pointed at a rough 90° clockwise angle with respect to the head of the scooter, and the dataset is a sequence of straight paths alternated with long U turns. This dataset is designed to confirm the ability of module ⑤ to correctly identify straight stretches, find the mounting angle of the sensor to correct the rotation matrix, and preserve this correction as laps repeat.

In dataset C, the device was turned on in a courtyard and carried in a bag while walking a lap around a parking lot, finally returning to the starting point. The dataset presents a long time for GPS position to converge to a quality fix. This dataset is useful for estimating the time of position fixing, and is used to validate the use of module ① for the reduction of the error during board start-up, in the position fixing phase.

All datasets were acquired using a Tierra WL11 board, mounting a Cortex M3 processor and 32 MB of RAM. The board integrates a Telit Jupiter SE868-v3 positioning system and an STM LSM6DS3 inertial module. The GPS position updates are acquired at a frequency of 1 Hz. The IMU was sampled at a frequency of 4 Hz for datasets A and C, and 40 Hz for dataset B.

A. 3-layer low-pass GPS filter

In Fig. 3, the "cloud" of green points represents the positions produced in dataset A over a period of 90 minutes, while the red dot represents the real position occupied by the device during the test period. This position is particularly critical; the GPS position did not immediately converge when the device exited the house, and there are multipath and satellite visibility issues due to buildings and trees.

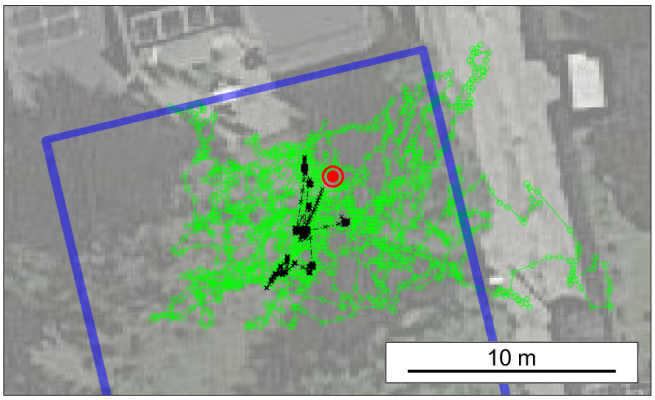

Fig. 3. Effects of the 3-layer low-pass GPS filter on dataset A.

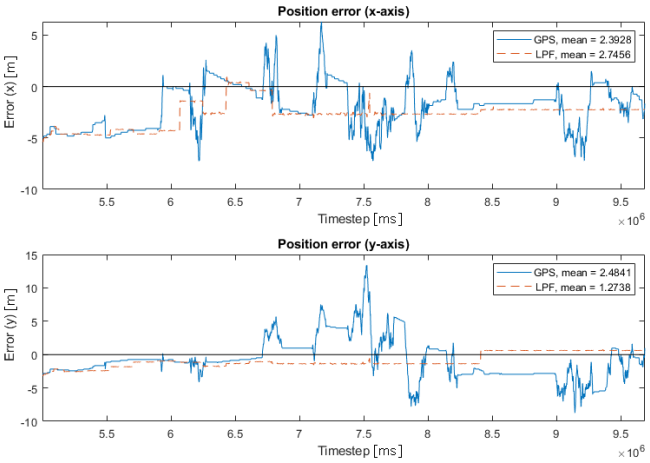

Fig. 4. Position error estimate before (GPS) and after (LPF) application of the 3-layer low-pass position filter.

The black points plot the fusion of the accelerometer data with the low-pass filtered GPS positions, produced by the algorithm in block ④, after the static case was identified by block ③. The resulting collection of points shows much less variation in time and never exits the courtyard enclosure (marked in the figure by the blue line).

Fig. 4 illustrates the position error with respect to the ground truth (the red dot in Fig. 3) during the time of the experiment. The X axis corresponds to the west-east direction and the Y axis to the south-north direction. The mean error for each solution is reported in the figure legend; while the X-axis magnitude of the error is on average greater for the filtered position, it is notable that the position remains stable for a long time, and peaks are averted. At the same time, the overall error on the Y-axis is reduced.

B. Real-time yaw alignment

The yaw alignment problem reduces to finding the rotation matrix to apply to accelerometer data in order to align it to the vehicle body frame; that is, assuming that the alignment of the Y and Z axes can be deduced from the gravity vector, the angle to be found is that of the X axis of the accelerometer with respect to the axis of motion of the vehicle. In dataset B, the device is mounted with an angle of about 90° clockwise between the X axis and the head of the scooter.

Fig. 5.A shows the effect of calibration on this dataset. When the device is powered on, the X axis is assumed to be pointing to the head of the vehicle. As soon as the vehicle encounters a straight path long enough to satisfy calibration requirements, the mounting angle ϵ_α is updated. Effectively, we expect that after calibration the X axis should be modeled as pointing to the right hand side of the vehicle.

In Fig. 5.B we can see how the calibration is handled as the device is running. The vehicle is running laps counter-clockwise around the field; the colored line represents the path followed by the vehicle (as the union of subsequent GPS positions) and the black arrow represents the direction of the X axis in the rotation matrix R_b^n. At the end of the first straight portion in the track (running from north to south), the alignment buffer is filled; the rotation angle is updated using the method described in III-A5. At the next position update, the direction of the sensor-frame X axis is correctly identified as being to the right hand side of the direction of movement ($\epsilon_\alpha = 82.08°$). Fig. 5.C shows how the mounting angle correction is preserved in subsequent laps.

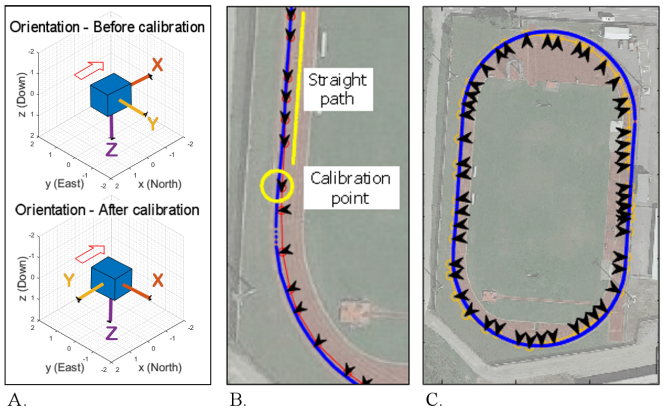

Fig. 5. A: Rotation applied to the sensor model at startup and after calibration. B, C: Effect of real-time yaw calibration on dataset B.

C. Use of last known position at start-up

In Fig. 6, we demonstrate the effectiveness of module ① on dataset C. The algorithm is set up so as to refuse GPS updates with PDOP > 10. The red line represents the sequence of GPS positions. The first ones received are very distant from the actual starting position; they are refused for the high associated PDOP. The algorithm outputs the last known position as long as the INS does not detect any movement. Once a reliable (PDOP < 10) GPS fix arrives, it is accepted and the position is integrated into the navigation solution.

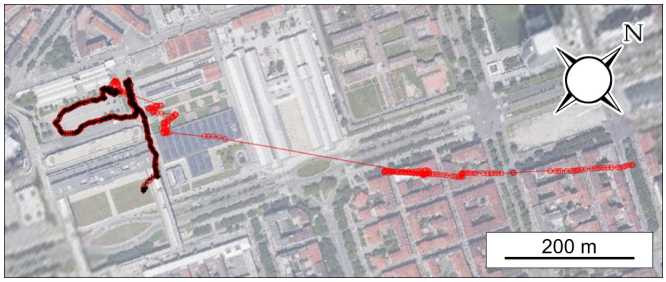

Fig. 6. Effect of excluding high-PDOP fixes at GPS start-up on dataset C.

D. Evaluation of computational cost

As a study of the feasibility of the algorithms on a low-cost system, we have carried out an estimation of the computational cost of each module. In table I two different measures of the computational effort for each module are given. The FLOP count for each module is a big-O estimation based on the properties of Kalman Filter algorithms [29]. In this estimation, both sums and multiplications are considered FLOPs.

The count of sums and multiplications for the Kalman Filter portions of the system is also provided. These figures are an analytic estimation, based on the filter parameters (the state vector size n, the measurement vector size m and the command vector size p) and assuming a Cholesky implementation of the matrix inversion [30].

We can observe that certain blocks, such as ① and ④, have an extremely low computational effort and therefore can be easily integrated in any system at a negligible cost. Block ⑤, while being more complex, still consists of a very limited number of operations. Block ②, containing 3 Kalman Filters, requires a greater computational effort; however, it is interesting to put this cost into perspective by comparing it with that of a typical GPS/INS fusion filter (block ⑥). The fusion filter we examined in this estimation is the linear Kalman Filter described in [5]; this filter operates on 9 state variables versus the 3 used by filters in block ②, thus presenting a computational cost that's more than $10\times$ greater than each of the aforementioned filters. We can then assume

TABLE I
COMPUTATIONAL EFFORT ESTIMATION

Block	*Module*	*Portion*	*FLOPS [29]*	*KF params [n,m,p]*	*sum [30]*	*mul [30]*
①	Load position	Memory access	4			
②	Tilt KF	KF Predict	120	[3,3,3]	54	63
		KF Predict	120		54	63
		KF Update	183		159	207
		Total	**423**		**267**	**333**
	Yaw KF	KF Predict	120	[3,3,3]	54	63
		KF Predict	120		54	63
		KF Update	183		159	207
		Total	**423**		**267**	**333**
	Gyro Bias KF	KF Predict	120	[3,3,3]	54	63
		KF Update	183		159	207
		Total	**303**		**213**	**270**
④	Static Filter	Mean of buffer (worst case)	40			
⑤	Yaw alignment	Mean of buffer	50			
		Other Processing	64			
		Rotm conversion	20			
		Total	**134**			
⑥	Fusion KF	KF Predict	3078	[9,5,3]	1476	1566
		KF Update	2165		2493	2869
		Total	**5243**		**3969**	**4435**

that for a system that's been designed to accommodate a fusion filter, the addition of the cascaded KF-based preprocessing would require an acceptable amount of computational effort.

V. CONCLUSIONS

We have presented a collection of methods for the improvement of performance of INS/GPS systems that have a low computational effort and are targeted at low-cost systems. These methods focus on specific critical cases, identified by datasets we acquired in the field, that make these methods suitable for a variety of applications that foresee the use of low-budget, strapdown devices. For instance, block ① can be used to identify whether a vehicle has been moved while the device was turned off, while block ④ can exclude false movements when the vehicle is static; these methods can be applied to create anti-theft measures for vehicles and bike sharing systems, or to compensate for excessive position drift on slow moving vehicles. The dynamic INS calibration provided by blocks ② and ⑤ can be useful in applications for wearable IoT devices or for localization devices that are not fixed to the vehicle body. In all such cases, the proposed techniques are designed to have a minimal impact on the reduced computational pool in low-cost, low-power devices, leaving most of the resources free to execute other tasks.

REFERENCES

[1] Joseph L Awange. *Environmental monitoring using GNSS: Global navigation satellite systems*. Springer Science & Business Media, 2012.

[2] Gianvito Urgese et al. "An Engineering Process model for managing a digitalised life-cycle of products in the Industry 4.0". In: *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2020, pp. 1–6.

[3] Fekher Khelifi et al. "A survey of localization systems in internet of things". In: *Mobile Networks and Applications* 24.3 (2019), pp. 761–785.

[4] Donkyu Baek et al. "Battery-aware operation range estimation for terrestrial and aerial electric vehicles". In: *IEEE Transactions on Vehicular Technology* 68.6 (2019), pp. 5471–5482.

[5] Shaghayegh Zihajehzadeh et al. "A cascaded Kalman filter-based GPS/MEMS-IMU integration for sports applications". In: *Measurement* 73 (2015), pp. 200–210.

[6] Pavel Kriz, Filip Maly, and Tomas Kozel. "Improving indoor localization using bluetooth low energy beacons". In: *Mobile Information Systems* 2016 (2016).

[7] Anthony LaMarca et al. "Place lab: Device positioning using radio beacons in the wild". In: *International Conference on Pervasive Computing*. Springer. 2005, pp. 116–133.

[8] D. Croce et al. "Performance of LoRa for Bike-Sharing Systems". In: *2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*. 2019, pp. 1–6.

[9] I Smith et al. "Are GSM phones THE solution for localization?" In: *Seventh IEEE Workshop on Mobile Computing Systems & Applications (WMCSA'06 Supplement)*. IEEE. 2005, pp. 34–42.

[10] Yihong Wu, Fulin Tang, and Heping Li. *Image Based Camera Localization: an Overview*. 2016. arXiv: 1610.03660 [cs.CV].

[11] Z. Su et al. "Global localization of a mobile robot using lidar and visual features". In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2017, pp. 2377–2383.

[12] Neda Navidi et al. "A new technique for integrating MEMS-based low-cost IMU and GPS in vehicular navigation". In: *Journal of Sensors* 2016 (2016).

[13] Nguyen Quang Vinh. "INS/GPS integration system using street return algorithm and compass sensor". In: *Procedia Computer Science* 103 (2017), pp. 475–482.

[14] Yi-Hsuan Lee and Kai-Wei Chiang. "The performance analysis of a 3D map embedded INS/GPS fusion algorithm for seamless vehicular navigation in elevated highway environments". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 39 (2012), B1.

[15] D. Li, X. Jia, and J. Zhao. "A Novel Hybrid Fusion Algorithm for Low-Cost GPS/INS Integrated Navigation System During GPS Outages". In: *IEEE Access* 8 (2020), pp. 53984–53996.

[16] Qingsheng Kong, Shenglei Xu, and Sang-Sun Lee. "using PDOP to estimate Kalman filters measurement noise covariance for GPS Positioning". In: 2012.

[17] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[18] J. L. Crassidis. "Sigma-point Kalman filtering for integrated GPS and inertial navigation". In: *IEEE Transactions on Aerospace and Electronic Systems* 42.2 (2006), pp. 750–756.

[19] Y. Li and X. Xu. "The Application of EKF and UKF to the SINS/GPS Integrated Navigation Systems". In: *2010 2nd International Conference on Information Engineering and Computer Science*. 2010, pp. 1–5.

[20] Simon J. Julier and Jeffrey K. Uhlmann. "New extension of the Kalman filter to nonlinear systems". In: *Defense, Security, and Sensing*. 1997.

[21] Christopher J Fisher. "Using an accelerometer for inclination sensing". In: *AN-1057, Application note, Analog Devices* (2010), pp. 1–8.

[22] Qiangwen Fu et al. "Autonomous in-motion alignment for land vehicle strapdown inertial navigation system without the aid of external sensors". In: *The Journal of Navigation* 71.6 (2018), pp. 1312–1328.

[23] Y. Wu. "Versatile land navigation using inertial sensors and odometry: Self-calibration, in-motion alignment and positioning". In: *2014 DGON Inertial Sensors and Systems (ISS)*. 2014, pp. 1–19.

[24] Zhang Lei, Shu Rong, and Wang Jianyu. "Initial alignment for SINS based on low-cost IMU". In: *Advances in Modeling and Simulation Guest Editor: Jinrong Zhu* 6.6 (2011), p. 1080.

[25] Burak H Kaygısız and Bekir Şen. "In-motion alignment of a low-cost GPS/INS under large heading error". In: *The Journal of Navigation* 68.2 (2015), pp. 355–366.

[26] Gamini Dissanayake et al. "The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications". In: *IEEE transactions on robotics and automation* 17.5 (2001), pp. 731–747.

[27] Yiqing Yao et al. "A hybrid fusion algorithm for GPS/INS integration during GPS outages". In: *Measurement* 103 (2017), pp. 42–51.

[28] Seung-hee Han and Jinling Wang. "A Novel Initial Alignment Method for GPS/SINS Integration with Large Initial Heading Error". In: 2009.

[29] Aurélien Valade et al. "A study about Kalman filters applied to embedded sensors". In: *Sensors* 17.12 (2017), p. 2810.

[30] C. Ingemarsson and O. Gustafsson. "On fixed-point implementation of symmetric matrix inversion". In: *2015 European Conference on Circuit Theory and Design (ECCTD)*. 2015, pp. 1–4.